Nonlinear Acoustic Echo Cancellation based on a Sliding-Window Leaky Kernel Affine Projection Algorithm[‖]

Jose M. Gil-Cacho[*], Marco Signoretto[†], Toon van Waterschoot[‡], Marc Moonen[§], and Søren Holdt Jensen[¶]

[*]Corresponding author. KU Leuven, Department of Electrical Engineering-ESAT, SCD-SISTA/iMinds Future Health Department, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium, Tel. +32 16321856, Fax +32 16321970, E-mail pepe.gilcacho@esat.kuleuven.be

[†]KU Leuven, Department of Electrical Engineering-ESAT, SCD-SISTA/iMinds Future Health Department, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium, Tel. +32 16328657,
Fax +32 16321970, E-mail marco.signoretto@esat.kuleuven.be

[‡]KU Leuven, Department of Electrical Engineering-ESAT, SCD-SISTA/iMinds Future Health Department, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium, Tel. +32 16321927, Fax +32 16321970, E-mail toon.vanwaterschoot@esat.kuleuven.be

[§]KU Leuven, Department of Electrical Engineering-ESAT, SCD-SISTA/iMinds Future Health Department, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium, Tel. +32 16321060, Fax +32 16 321970, E-mail marc.moonen@esat.kuleuven.be

[¶]Department of Electronic Systems Aalborg University, Fredrik Bajers Vej 7, DK-9220 Aalborg, Denmark, Tel. +45 9940 8654, E-mail shj@es.aau.dk

[‖]EDICS: AUD-ECHO

Abstract

Acoustic echo cancellation (AEC) is used in speech communication systems where the existence of echoes degrades the speech intelligibility. Standard approaches to AEC rely on the assumption that the echo path to be identified can be modeled by a linear filter. However, some elements introduce nonlinear distortion and must be modeled as nonlinear systems. Several nonlinear models have been used with more or less success. The kernel affine projection algorithm (KAPA) has been successfully applied to many areas in signal processing but not yet to nonlinear AEC (NLAEC). The contribution of this paper is three-fold: (1) to apply KAPA to the NLAEC problem, (2) to develop a sliding-window leaky KAPA (SWL-KAPA) that is well suited for NLAEC applications, and (3) to propose a kernel function, consisting of a weighted sum of a linear and a Gaussian kernel. In our experiment set-up, the

This research work was carried out at the ESAT Laboratory of Katholieke Universiteit Leuven, in the frame of K.U.Leuven Research Council CoE EF/05/006 'Optimization in Engineering' (OPTEC) and PFV/10/002 (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P6/04 'Dynamical systems, control and optimization' (DYSCO) 2007-2011, Research Project FWO nr. G.0600.08 'Signal processing and network design for wireless acoustic sensor networks', EC-FP6 project 'Core Signal Processing Training Program' (SIGNAL) and was supported by a Postdoctoral Fellowship of the Research Foundation Flanders (FWO-Vlaanderen, T. van Waterschoot). The scientific responsibility is assumed by its authors.

proposed SWL-KAPA for NLAEC consistently outperforms the linear APA, resulting in up to 12 dB of improvement in ERLE at a computational cost that is only 4.6 times higher. Moreover, it is shown that the SWL-KAPA outperforms, by $4-6$ dB, a Volterra-based NLAEC, which itself has a much higher 413 times computational cost than the linear APA.

Index Terms

Kernel adaptive filters, nonlinear acoustic echo cancellation.

I. Introduction

Acoustic echo cancellation (AEC) is used in many speech communication systems where the existence of echoes degrades the speech intelligibility and listening comfort [1], [2]. These applications range from mobile or hands-free telephony to teleconferencing or voice over IP (VoIP) services, which are often integrated in smartphones, tablets, notebooks, laptops, etc., which are becoming standard devices in our every-day life. While mobile devices are becoming smaller, the demand for quality, performance, and special features in audio products is increasing. Moreover, the need for higher sound pressure levels, provided by smaller devices, presents a huge challenge to engineers who have to deal with (usually cheap) loudspeakers working close to saturation.

The general set-up for an AEC is depicted in Fig. 1. A far-end speech signal $u(i)$ is played back

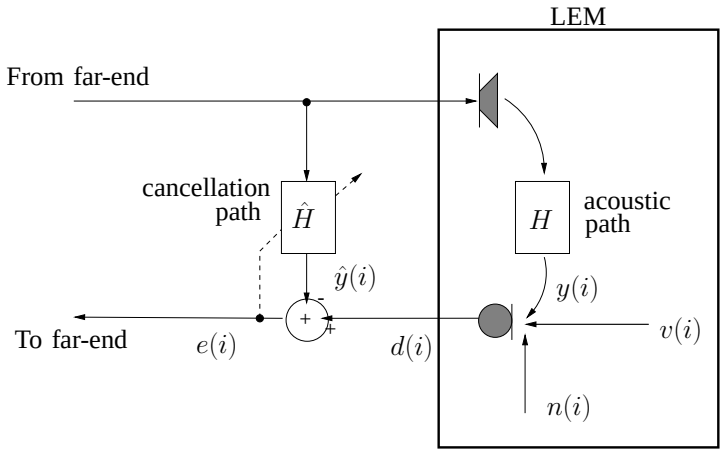


Fig. 1. Acoustic echo canceler set-up.

into an enclosure (i.e., the room) through a loudspeaker. In the room, there is a microphone to record

a near-end speech signal, which is to be transmitted back to the far-end side. An acoustic echo path H between the loudspeaker and the microphone exists, so that the microphone (i.e., desired) signal $d(i) = y(i)+v(i)+n(i)$ contains an undesired echo $y(i)$ plus the near-end speech $v(i)$ and the background noise $n(i)$. The echo signal $y(i)$ can be considered as the far-end signal $u(i)$ filtered by the echo path, i.e., the loudspeaker-enclosure-microphone (LEM) response. An acoustic echo canceler seeks to cancel the echo signal component $y(i)$ in the microphone signal $d(i)$, ideally leading to an *echo-free* error (or residual) signal $e(i)$, which is then transmitted to the far-end side. This is done by subtracting an estimate of the echo signal $\hat{y}(i)$ from the microphone signal, i.e., $e(i) = d(i) - \hat{y}(i) = y(i) - \hat{y}(i) + n(i)$, where a perfect double-talk detector (DTD) is assumed to be available so that periods when only the far-end speaker is active, i.e., $v(i) = 0, \forall i$, can be considered. Thus, the main objective of an echo canceler is to identify a model that represents a best fit to the echo path (i.e., similar to a system identification problem) or, in other words, a function $f(\cdot)$ that models the underlying mapping $y(i) = f\left[\mathbf{u}(i)\right]$, where $\mathbf{u}(i)$ is the input signal vector defined as $\mathbf{u}(i) = [u(i), u(i-1), ..., u(i-L+1)]^T \in \mathbb{R}^L$.

Typically, a linear finite-impulse-response (FIR) filter is assumed to provide a good model for the echo path, leading to a class of linear-adaptive-filter-based AEC systems. This assumption may be valid for some components in the LEM system. In fact, even in nonlinear AEC (NLAEC), the room (enclosure) and microphone responses may be modeled as linear systems. However, loudspeakers, as well as amplifiers, DACs, coders, etc., included in the LEM introduce nonlinear distortion and must be modeled as nonlinear systems [3]. In this paper, we will show simulation results obtained from a simulated nonlinear system in a controlled scenario, so as to avoid other problems related to real AEC applications (e.g., insufficient-order adaptive filters, or scenarios comprising non-stationary and colored background noise). If the mapping between $\mathbf{u}(i)$ and $y(i)$ is highly nonlinear, poor performance may be expected from any linear-adaptive-filter-based AEC. Several nonlinear models, intended to overcome the limitations of linear filters, have been used for NLAEC with more or less success.

Neural networks have been used for NLAEC [4], [5], but in that case the adaptation consists of solving a non-convex optimization problem with many local minima, potentially leading to suboptimal solutions. A proper initialization is then required to avoid local minima, which is already a big challenge in itself. Structured and block-oriented nonlinear models [6], [7] have also been studied but, besides the need for proper initialization, the modeling capabilities are then limited to the specified nonlinear structure. Linear-in-the-parameter models that require nonlinear expansions of the input vector are appealing from an adaptive filtering perspective since standard linear adaptive filtering techniques can be applied and global

optimality can be guaranteed [8], [9]. The main problem however is that many more filter coefficients are needed than in the linear case, resulting in a large computational complexity together with inherently slow convergence [10]-[13]. Volterra [14]-[16] and simplified Volterra [8] filters are common solutions belonging to the linear-in-the-parameters family, although typically only very low nonlinear orders are considered due to computational complexity constraints.

On the other hand, kernel adaptive algorithms [17]-[20] and on-line learning algorithms [21], [22] have been subjects of great attention due to their good performance in nonlinear signal processing applications. Kernel methods are developed based on the theory of reproducing kernel Hilbert spaces (RKHS) [23]; a nonlinear model is obtained via a reproducing kernel by implicitly mapping data into a high-dimensional feature space. If the adaptive filtering operations can be expressed by particular vector inner products, then it is possible to apply the so-called *kernel trick*. The power of this idea is that the model parameters of the underlying nonlinear mapping $y(i) = f\left[\mathbf{u}(i)\right]$ are calculated by applying *nonlinear* methods on the input data, corresponding to *linear* operations in the feature space. This enables formulations where convergence to a global optimum may be guaranteed. In particular, the kernel affine projection algorithm (KAPA) [17] has been successfully applied to nonlinear equalization, nonlinear system identification, and nonlinear noise cancellation, as well as to the prediction of nonlinear time series. Although in [24] we have already introduced the use of KAPA for NLAEC, its formal application is lacking so far.

The contribution of this paper is therefore three-fold: (1) to formally apply KAPA to the NLAEC problem for the first time and confirm its potential for this application, (2) to develop a sliding-window leaky KAPA (SWL-KAPA) that is best suited for NLAEC applications, and (3) to propose a kernel function, that is a weighted sum of a linear and a Gaussian kernel, which is efficient for acoustic applications. The motivation in designing this kernel is basically to separate the problem into linear and nonlinear sub-problems. The weights in the kernel also impose different forgetting mechanisms in the sliding window, which in turn translates to more flexible regularization.

The outline of the paper is as follows. In Section II, we present a review of the literature, which is instrumental for our purposes, namely, the affine projection algorithm (APA), kernel methods, and the kernel APA (KAPA). In Section III, the SWL-KAPA and the proposed kernel is described with explanation and justification of the choices made for NLAEC. In Section IV, computer simulation results are provided to verify the performance of the proposed algorithm, an overview of existing results in the NLAEC literature is given, and a computational complexity analysis of the competing algorithms is shown, in particular, the linear APA and the 3rd-order Hammerstein and Volterra filters. Finally, Section

V concludes the paper.

II. Linear and Kernel Affine Projection Algorithm

A. Linear APA

Standard approaches to AEC rely on the assumption that the echo path can be modeled by a linear FIR filter [1], [2], [25]-[27], in which the coefficients of the echo path are collected in the parameter vector $\mathbf{h} = [h_0, h_1, ..., h_{L-1}]^T \in \mathbb{R}^L$ with model order $L-1$. An adaptive filter is used to provide an estimate $\hat{\mathbf{h}} \in \mathbb{R}^L$ of $\mathbf{h}$ with sufficient order, such that the echo signal estimate at the microphone position is $\hat{y}(i) = \hat{\mathbf{h}}^T(i)\mathbf{u}(i)$ with $\mathbf{u}(i) = [u(i), u(i-1), \ldots, u(i-L+1)]^T$. Notice that, for now, we have also assumed that the echo path is time-invariant, $\mathbf{h}^T(i) = \mathbf{h}^T, \forall i$ over the estimation time window $[1, 2, \ldots, i]$ with $i \gg L$; we will consider time-varying echo paths further on. A least-squares regression data model may be constructed as

$$\mathbf{d}_{\text{LS}}(i) = \mathbf{U}_{\text{LS}}(i)\mathbf{h} + \mathbf{e}_{\text{LS}}(i) \tag{1}$$

where

$$\mathbf{d}_{\text{LS}}(i) = [d(i), d(i-1), ..., d(1)]^T \tag{2}$$

$$\mathbf{e}_{\text{LS}}(i) = [e(i), e(i-1), ..., e(1)]^T \tag{3}$$

$$\mathbf{U}_{\text{LS}}(i) = [\mathbf{u}(i), \mathbf{u}(i-1), ..., \mathbf{u}(1)]^T. \tag{4}$$

The error in (1) is minimized in a least-squares sense, giving

$$\hat{\mathbf{h}}(i) = \arg\min_{\hat{\mathbf{h}}(i)} \sum_{j=1}^{i} \left| d(j) - \hat{\mathbf{h}}^T(j)\mathbf{u}(j) \right|^2 \tag{5}$$

$$= \left[\mathbf{U}_{\text{LS}}^T(i)\mathbf{U}_{\text{LS}}(i)\right]^{-1} \mathbf{U}_{\text{LS}}^T(i)\mathbf{d}_{\text{LS}}(i) \tag{6}$$

where $\frac{1}{i}\mathbf{U}_{\text{LS}}^T(i)\mathbf{U}_{\text{LS}}(i) = \mathbf{R}_{uu}(i)$ is the sample autocorrelation matrix and $\frac{1}{i}\mathbf{U}_{\text{LS}}^T(i)\mathbf{d}_{\text{LS}}(i) = \mathbf{r}_{du}(i)$ is the sample cross-correlation vector. Nearly all linear adaptive filtering algorithms are related to the LS estimate, which is unique if the autocorrelation matrix is nonsingular. This condition is, however, not always met. Indeed a common problem in AEC applications is that the matrix $\mathbf{R}_{uu}(i)$ is ill-conditioned or even singular due to poor excitation [28]. This situation will generate large variance in the estimation of $\hat{\mathbf{h}}(i)$. Therefore, to improve data generalization (i.e., to prevent small changes in the data from producing large changes in the solution), the l_2-norm of the vector $\hat{\mathbf{h}}(i)$ is often used as a penalty term. This type of regression, commonly known as *ridge regression*, was introduced by Tikhonov and Arsenin [29] (and

formulated as a linear algebra problem in [30]) as a remedy for ill-posedness. Essentially, it establishes a trade-off between fitting the data and reducing the norm of the solution, i.e., the smoothness of the solution. Thus, the modified so called *regularized* LS criterion is formulated as

$$\hat{\mathbf{h}}(i) = \arg\min_{\hat{\mathbf{h}}(i)} \sum_{j=1}^{i} \left| d(j) - \hat{\mathbf{h}}^T(i)\mathbf{u}(j) \right|^2 + \lambda \left\| \hat{\mathbf{h}}(i) \right\|_2^2 \tag{7}$$

$$= \left[\mathbf{U}_{\text{LS}}^T(i)\mathbf{U}_{\text{LS}}(i) + \lambda \mathbf{I}_L \right]^{-1} \mathbf{U}_{\text{LS}}^T(i)\mathbf{d}_{\text{LS}}(i) \tag{8}$$

where $\mathbf{I}_L$ is the $L \times L$ identity matrix and λ is a regularization parameter that controls the smoothness of the solution. This regularized LS solution basically turns an ill-posed problem into a well-posed problem by adding a scaled identity matrix to $\mathbf{R}_{uu}(i)$.

Since the echo path may be time-varying due to changes in the room (e.g., people moving around, or changes in microphone or loudspeaker position) and/or the loudspeaker characteristics (e.g., temperature or acoustic load changes), the system must be equipped with a mechanism that adapts to changes in the echo path. Therefore, a recursive least-squares (RLS) algorithm may be derived from the LS estimate of the filter $\hat{\mathbf{h}}(i)$ [26]. Essentially the RLS algorithm updates the LS estimate every time a new pair of data $\{\mathbf{u}(i), d(i)\}$ is available [26]. The RLS algorithm includes an exponential weighting factor δ that "forgets" older data in order to improve its ability for tracking changes in the echo path. The *Tikhonov regularized* RLS, featuring both exponential weighting and regularization, was proposed in [31]. It was called the leaky RLS algorithm because of its similarity with the leaky least mean squares (LMS) algorithm [32], [33].

In general, RLS is computationally too expensive for AEC applications [1]; hence, computationally cheaper algorithms are called for. The most popular and extensively used algorithm is the normalized (NLMS) version of Widrow's LMS algorithm [34], which works with instantaneous estimates of the autocorrelation matrix and hence typically has slow convergence. On the other hand, the so-called affine projection algorithm (APA), originally proposed in [35], computes an estimate of $\mathbf{R}_{uu}(i)$ from the current and $P-1$ past input vectors $\mathbf{u}(j)$. APA obtains faster convergence rates compared to NLMS and better tracking capabilities compared to RLS, so it may be seen as a good compromise between NLMS and RLS in both performance and complexity. APA is, therefore, also adopted in AEC applications when faster convergence is necessary [36], [37].

In [38], APA is derived formally from the underdetermined RLS (URLS) family (whereas RLS is an overdetermined algorithm). The derivation in [38] adopts a deterministic framework, in contrast to the *classical* stochastic derivation [26], [27], [35]. In [28], optimally-regularized APA and NLMS algorithms

are derived as part of the URLS family. We use this framework to be consistent with the notation and derivations in this paper. The APA (and also NLMS) weight-update equation can be written as [39],

$$\hat{\mathbf{h}}(i) = \hat{\mathbf{h}}(i-1) + \mu \left[\hat{\mathbf{R}}_{uu}(i) + \lambda \mathbf{I}\right]^{-1} \left[\hat{\mathbf{r}}_{du} - \hat{\mathbf{R}}_{uu}(i)\hat{\mathbf{h}}(i-1)\right] \tag{9}$$

where μ is a fixed step size that is included as a relaxation mechanism to protect the recursion from divergence due to the use of noisy estimates $\hat{\mathbf{R}}_{uu}(i)$ and $\hat{\mathbf{r}}_{du}(i)$, and λ is a fixed regularization parameter. In contrast to NLMS, which updates the weight vector based only on the current input data $\{\mathbf{u}(i), d(i)\}$ (i.e., $P = 1$), APA updates the weight vector based on $\{\mathbf{u}(i) \dots \mathbf{u}(i-P+1), d(i) \dots d(i-P+1)\}$ (i.e., $P > 1$). Using moving-average estimates

$$\hat{\mathbf{R}}_{uu}(i) = \frac{1}{P}\mathbf{U}^T(i)\mathbf{U}(i),$$

and

$$\hat{\mathbf{r}}_{du}(i) = \frac{1}{P}\mathbf{U}^T(i)\mathbf{d}(i)$$

where $\mathbf{U}(i) = [\mathbf{u}(i), \mathbf{u}(i-1), ..., \mathbf{u}(i-P+1)]^T \in \mathbb{R}^{P\times L}$, $\mathbf{u}(i) = [u(i), \dots, u(i-L+1)]^T \in \mathbb{R}^L$, and $\mathbf{d}(i) = [d(i), d(i-1), ..., d(i-P+1)]^T \in \mathbb{R}^P$, the weight-update (9) becomes

$$\hat{\mathbf{h}}(i) = \hat{\mathbf{h}}(i-1) + \mu \left[\mathbf{U}^T(i)\mathbf{U}(i) + \lambda \mathbf{I}\right]^{-1} \mathbf{U}^T(i) \left[\mathbf{d}(i) - \mathbf{U}(i)\hat{\mathbf{h}}(i-1)\right]$$

By using the matrix inversion lemma

$$\left[\mathbf{U}^T(i)\mathbf{U}(i) + \lambda \mathbf{I}_L\right]^{-1} \mathbf{U}^T(i) = \mathbf{U}^T(i) \left[\mathbf{U}(i)\mathbf{U}^T(i) + \lambda \mathbf{I}_P\right]^{-1} \tag{10}$$

the basic APA recursion is finally given as

$$\mathbf{e}(i) = \mathbf{d}(i) - \mathbf{U}(i)\hat{\mathbf{h}}(i-1) \tag{11}$$

$$\hat{\mathbf{h}}(i) = \hat{\mathbf{h}}(i-1) + \mu\mathbf{U}^T(i) \left[\mathbf{U}(i)\mathbf{U}^T(i) + \lambda \mathbf{I}\right]^{-1} \mathbf{e}(i). \tag{12}$$

The *Tikhonov regularized*, or leaky, APA is the underdetermined version of leaky RLS and is written as

$$\mathbf{e}(i) = \mathbf{d}(i) - \mathbf{U}(i)\hat{\mathbf{h}}(i-1) \tag{13}$$

$$\hat{\mathbf{h}}(i) = (1 - \lambda\mu)\hat{\mathbf{h}}(i-1) + \mu\mathbf{U}^T(i) \left[\mathbf{U}(i)\mathbf{U}^T(i) + \lambda \mathbf{I}\right]^{-1} \mathbf{e}(i). \tag{14}$$

The presence of the leakage term in (14) is known to introduce filter coefficient bias, which degrades mean-squared error performance [26], [33]. However, in some applications, its use proves to be beneficial, in particular when using non-stationary, non-white signals like speech [40]-[42]. This is due to the poor excitation properties of this type of signal and the subsequent need for regularization [28], as explained

in the introduction. Here, the leaky APA will be the basis for developing a regularized kernel adaptive algorithm that moreover permits the use of a sliding-window, as will be explained in the following sections.

B. Kernel Methods

A kernel [23] is a continuous, symmetric, and positive-definite function $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ where $\mathbb{U}$ is the input space. A *feature* mapping from the input space to the feature space, $\boldsymbol{\varphi} : \mathbb{U} \to \mathbb{F}$, can be constructed such that for $\mathbf{x}$, $\mathbf{y} \in \mathbb{U}$

$$\kappa(\mathbf{x}, \mathbf{y}) = \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}(\mathbf{y}), \tag{15}$$

where $\mathbb{F}$ is referred to as the feature space and $\boldsymbol{\varphi}(\mathbf{x})$ is referred to as the feature vector. Conveniently, based on the theory of RKHS [23] and making use of the Mercer's condition [43], several algorithms may be constructed that apply the so-called kernel trick based on (15), which was originally proposed in [44]. We give an example using a simple kernel for which explicit computations of the mapping $\boldsymbol{\varphi}$ can be shown.

The polynomial kernel is defined as $\kappa(\mathbf{x}, \mathbf{y}) = \left(\mathbf{x}^T\mathbf{y} + c\right)^d$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^L$ and $c, d \in \mathbb{R}$. For example, if $d = 2$, $\mathbf{x} = [x_1, \ldots, x_L]^T$ and $\mathbf{y} = [y_1, \ldots, y_L]^T$, then

$$\begin{aligned} \kappa(\mathbf{x}, \mathbf{y}) &= \left(\sum_i x_i y_i + c\right)^2 \\ &= \sum_i \sum_j x_i x_j y_i y_j + \sum_i \left(\sqrt{2c}x_i\right)\left(\sqrt{2c}y_i\right) + c^2. \end{aligned} \tag{16}$$

From (16), we see that $\kappa(\mathbf{x}, \mathbf{y})$ represents the inner product in $\mathbb{F}$, i.e., $\kappa(\mathbf{x}, \mathbf{y}) = \left(\mathbf{x}^T\mathbf{y} + c\right)^2 = \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}(\mathbf{y})$, implicitly constructed by the mapping $\boldsymbol{\varphi}$ defined as

$$\begin{aligned} \boldsymbol{\varphi}(\mathbf{x}) &= \left[x_1x_1, ..., x_1x_L, ..., x_Lx_L, \sqrt{2c}x_1, ..., \sqrt{2c}x_L, c\right]^T \\ \boldsymbol{\varphi}(\mathbf{y}) &= \left[y_1y_1, ..., y_1y_L, ..., y_Ly_L, \sqrt{2c}y_1, ..., \sqrt{2c}y_L, c\right]^T. \end{aligned}$$

The feature mapping, in this simple example, takes the form

$$\boldsymbol{\varphi} : \mathbb{R}^L \to \mathbb{R}^{L^2+L+1}$$

The kernel trick is applied here to work implicitly in an $(L^2 + L + 1)$-dimensional space without having to transform the input data into this space. In fact, the mapping does not necessarily have to be known, since all computations can be done by evaluating the kernel function in the input space. This also holds

for any other polynomial order (i.e, $d > 2$) or for any other kernel function, e.g., the Gaussian kernel $\kappa(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\rho\|\mathbf{x}-\mathbf{y}\|_2^2}$. By using the Taylor's expansion of $\mathrm{e}^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$, we see that the Gaussian kernel has an infinite set of polynomial terms, each corresponding to a feature i.e., $\boldsymbol{\varphi} : \mathbb{R}^L \rightarrow \mathbb{R}^\infty$. This in particular provides powerful modeling capabilities to kernel adaptive algorithms when dealing with nonlinear mappings; the only requirement for using the kernel trick is that the algorithm in the feature space can be expressed in terms of inner products, which are then replaced by kernel operations in the input space. Furthermore, it is clear that the estimation complexity will be determined by the input dimension, independent of the order of the nonlinearity [45]. This is one of the most appealing characteristics of kernel methods, as opposed to nonlinear adaptive filters requiring explicit nonlinear expansions of the input data as, for instance, Volterra filters [46].

C. Leaky APA in the Feature Space

To apply the kernel trick, the input signal vector $\mathbf{u}(i) \in \mathbb{R}^L$ is transformed by means of the feature mapping $\boldsymbol{\varphi}$, into a high-dimensional (possibly infinite-dimensional) feature vector $\boldsymbol{\varphi}\left[\mathbf{u}(i)\right] \in \mathbb{R}^F$. The LS problem is defined in the feature space as,

$$\hat{\boldsymbol{\omega}}(i) = \arg\min_{\hat{\boldsymbol{\omega}}(i)} \sum_{j=1}^{i} \left| d(j) - \hat{\boldsymbol{\omega}}^T(i)\boldsymbol{\varphi}\left[\mathbf{u}(j)\right] \right|^2 . \tag{17}$$

Note that $\hat{\boldsymbol{\omega}} \in \mathbb{R}^F$ is a vector of estimated filter coefficients in the feature space $\mathbb{F}$ and therefore has the same dimension as $\boldsymbol{\varphi}\left[\mathbf{u}(i)\right]$. In the sequel, the simplified notation

$$\boldsymbol{\varphi}(i) = \boldsymbol{\varphi}\left[\mathbf{u}(i)\right] \quad \text{and} \quad \kappa(i,j) = \kappa\left[\mathbf{u}(i), \mathbf{u}(j)\right]$$

is introduced for compactness. The use of a high-dimensional feature space $\mathbb{F}$ provides kernel methods with a very high degree of flexibility in solving minimization problems [18]. However, this appealing characteristic may cause the solution to perfectly fit any given data set while not generalizing well to new incoming data. This so-called overfitting problem particularly occurs if the Gaussian kernel is used and no precautions are taken. In order to prevent overfitting, the minimization problem should be regularized, which is again achieved by using the l_2-norm of the solution as a penalty term [17], [18], [21]. By introducing a regularization term, the solution is indeed found to generalize better to new data points [22], [47], [48]. The regularized LS problem on the data $\{d(1), d(2), ...\}$ and $\{\boldsymbol{\varphi}(1), \boldsymbol{\varphi}(2), ...\}$ can be formulated in the feature space as

$$\hat{\boldsymbol{\omega}}(i) = \arg\min_{\hat{\boldsymbol{\omega}}(i)} \sum_{j=1}^{i} \left| d(j) - \hat{\boldsymbol{\omega}}^T(i)\boldsymbol{\varphi}(j) \right|^2 + \lambda \left\|\hat{\boldsymbol{\omega}}(i)\right\|_2^2 \tag{18}$$

where λ is the regularization parameter. Following the same reasoning as in the previous section, the leaky APA (14) can be formulated in the feature space to compute $\hat{\boldsymbol{\omega}}$ using the *primal* model representation as

$$\mathbf{e}(i) = \mathbf{d}(i) - \boldsymbol{\Phi}(i)\hat{\boldsymbol{\omega}}(i-1) \tag{19}$$

$$\hat{\boldsymbol{\omega}}(i) = (1-\lambda\mu)\hat{\boldsymbol{\omega}}(i-1) + \mu\boldsymbol{\Phi}^T(i)\mathbf{G}^{-1}(i)\mathbf{e}(i) \tag{20}$$

where $\mathbf{d}(i) = [d(i), d(i-1), \ldots, d(i-P+1)]^T$, $\mathbf{e}(i) = [e_1(i), e_2(i), \ldots, e_P(i)]^T$, $\boldsymbol{\Phi}(i) = [\boldsymbol{\varphi}(i), \boldsymbol{\varphi}(i-1), \ldots, \boldsymbol{\varphi}(i-P+1)]^T$, $\mathbf{G}(i) = \left[\boldsymbol{\Phi}(i)\boldsymbol{\Phi}^T(i) + \lambda\mathbf{I}\right] \in \mathbb{R}^{P\times P}$. As discussed before, $\boldsymbol{\varphi}^T(i)\hat{\boldsymbol{\omega}}(i) \in \mathbb{R}$ is a much more powerful model than the usual $\mathbf{u}^T(i)\hat{\mathbf{h}}(i) \in \mathbb{R}$ because of the transformation from $\mathbf{u}(i)$ to $\boldsymbol{\varphi}(i)$. However, in practice, $\hat{\boldsymbol{\omega}}(i)$ will never be computed explicitly. Instead, the *dual* model representation will be derived based on the Representer Theorem [49], where the filter coefficient vector $\hat{\boldsymbol{\omega}}(i)$ is expressed as an expansion of all transformed input data up to time i, which then permits application of the kernel trick as shown in the next section.

D. Leaky KAPA

It is straightforwardly shown, by repeated application of (20), that

$$\hat{\boldsymbol{\omega}}(i) = \sum_{j=1}^{i} a_j(i)\boldsymbol{\varphi}(j) \tag{21}$$

$$= \sum_{j=1}^{i-1} a_j(i-1)\boldsymbol{\varphi}(j) + \mu\sum_{k=1}^{P}\boldsymbol{\varphi}(i-k+1)\bar{e}_k(i) \tag{22}$$

where the expansion coefficients $a_j(i)$ are computed recursively as

$$a_j(i) = \begin{cases} \mu\bar{e}_{i-j+1}(i), & j = i \\ (1-\lambda\mu)a_j(i-1) + \mu\bar{e}_{i-j+1}(i), & j = i-1, ..., i-P+1 \\ (1-\lambda\mu)a_j(i-1), & j > P \end{cases} \tag{23}$$

and where $\bar{e}_{i-j+1}(i)$ is the prediction error normalized by the $P \times P$ Gram matrix $\mathbf{G}(i)$ as,

$$\bar{e}_{i-j+1}(i) = \sum_{k=1}^{P} G^{-1}_{i-j+1,k}(i)e_k(i) \tag{24}$$

with $G^{-1}_{j,k}(i)$ defined as the matrix element lying on the jth row and kth column of the inverse Gram matrix at time i, and with

$$e_k(i) = d(i-k+1) - \boldsymbol{\varphi}^T(i-k+1)\boldsymbol{\omega}(i-1). \tag{25}$$

The leaky KAPA [17] will include the expansion coefficients updating (23), but will never construct $\hat{\boldsymbol{\omega}}(i)$ explicitly. However, although (21) is never explicitly computed, it allows one to rearrange (19) and (20) based on the kernel trick, leading to the leaky KAPA. First, the output vector $\hat{\mathbf{y}}(i) = \boldsymbol{\Phi}(i)\hat{\boldsymbol{\omega}}(i-1)$ in (19) can be computed in terms of kernel evaluations as

$$\begin{aligned}\hat{\mathbf{y}}(i) &= \boldsymbol{\Phi}(i)\sum_{j=1}^{i-1} a_j(i-1)\boldsymbol{\varphi}(j) \\ &= \sum_{j=1}^{i-1} a_j(i-1)\left[\kappa(i,j),\ \kappa(i-1,j), ..., \kappa(i-P+1,j)\ \right]^T.\end{aligned}$$

Next, the error vector in (19), i.e., with elements given in (25), can be directly computed as

$$e_k(i) = d(i-k+1) - \sum_{j=1}^{i-1} a_j(i-1)\kappa(i-k+1,j). \tag{26}$$

The matrix $\mathbf{G}(i)$ can be computed based on

$$\begin{aligned}\boldsymbol{\Phi}(i)\boldsymbol{\Phi}^T(i) &= [\boldsymbol{\varphi}(i), ..., \boldsymbol{\varphi}(i-P+1)]^T[\boldsymbol{\varphi}(i), ..., \boldsymbol{\varphi}(i-P+1)] \\ &= \begin{bmatrix} \kappa(i,i) & \kappa(i,i-1) & \kappa(i,i-P+1) \\ \kappa(i-i,i) & \kappa(i-1,i-1) & \kappa(i-1,i-P+1) \\ \vdots & \ddots & \vdots \\ \kappa(i-P+1,i) & \dots & \kappa(i-P+1,i-P+1) \end{bmatrix}.\end{aligned} \tag{27}$$

In [17] and [19], it has been shown that, since KAPA is just APA in RKHS, the important properties of APA (convergence in mean, misadjustment, etc.) also hold for KAPA.

III. Sliding-Window Leaky Kernel Affine Projection Algorithm for Nonlinear Acoustic Echo Cancellation

A. Pruning by Use of a Sliding Window

From a practical implementation point of view, leaky KAPA has to have two memory buffers, one for the set of input vectors (usually called the *dictionary*), $\mathbf{U}_{\mathrm{D}}(i) = [\mathbf{u}(1), ..., \mathbf{u}(i)]$, needed to evaluate the kernel functions in (26) and (27), and another for the corresponding vector of expansion coefficients $\mathbf{a}(i)$ in (23). The equations defining leaky KAPA (23)-(27) start from an empty set $\mathbf{U}_{\mathrm{D}}(0)$ and gradually add new input vectors $\mathbf{u}(i)$ (usually called the *centers*) to this set. The size of the dictionary and the corresponding expansion coefficient vector grow linearly with time i. This results in an increasing memory and computational complexity requirement as time evolves. In the literature, there exist several strategies

to cope with this problem, based on essentially two criteria: a sparse dictionary criterion and a *pruning* criterion.

The first criterion is to optimally construct a sparse dictionary, i.e., whenever new input data is considered, a decision is made whether or not to add the new center $\mathbf{u}(i)$ to the dictionary. The well known novelty criterion for on-line sparsification, which was originally proposed for resource-allocation networks (RAN) in [50], is based on the Euclidean distance between the new center and the dictionary [17]. Approximate linear dependency (ALD) [51] is another criterion, which constructs the dictionary based on the linear dependency of the new input data and the dictionary. In [52], the concept of *surprise* has been proposed, which measures the information a new center can contribute to the learning system, based on Gaussian process theory. Also, in [53], sparseness has been achieved in an off-line configuration based on the solution vector of a least-squares support vector machine (LS-SVM). Coherence-based sparsification has been proposed in [54] and employed in [55] using multikernels. Also in [55], the so-called block soft-thresholding is employed in on-line sparsification. However, these strategies are not readily applicable to our problem; either because they are restricted to off-line applications or because, as in on-line sparsification, the sparse memory buffers are in fact not bounded and therefore still grow without memory limitation. This is specially so, if using non-stationary data [55].

A number of pruning criteria have also been developed that discard either redundant or least-significant information from the memory buffers. In [56], [57] a number of different criteria are presented to prune LS-SVMs. In [58] a criterion presented in [57] has been used to prune the kernel autocorrelation matrix in Kernel RLS (KRLS) based on selecting the data that incurs the least error after it is omitted. The calculation of the error and further data selection appears to be computationally expensive. A simpler strategy has been adopted for KRLS in [59] where a sliding window considers only the last pairs of data to form the kernel autocorrelation matrix. The problem with these strategies, in relation to KAPA, is that they involve operations on the kernel autocorrelation matrix, which carries much more information than our $P \times P$ matrix $\mathbf{G}(i)$. The use of these pruning strategies would therefore result in significant errors and poor performance when applied to KAPA.

On the other hand, in the NORMA algorithm [21], which is equivalent to a kernel version of leaky LMS, the problem is solved by pruning the kernel expansion coefficients. The idea is that, since at each time instant i the expansion coefficients are scaled by the leakage term $(1-\lambda\mu)$, which is less than 1, the oldest term can be dropped without incurring significant error. This pruning criterion essentially converts NORMA into a sliding-window kernel LMS (SW-KLMS) algorithm. Following the spirit of NORMA,

the leaky KAPA is converted here into a sliding-window leaky KAPA (SWL-KAPA). Thus, SWL-KAPA exploits the benefits of the leaky KAPA both in terms of regularization and in that the pruning criterion can be used. The implementation of the leaky KAPA used here differs from that of [17] in that the error signal in NLAEC applications is indeed computed explicitly, as this is the signal sent back to the far-end side.

B. Weighted Sum of Kernels

The choice of the kernel implicitly defines a RKHS with an appropriate inner product [19], [60]. Furthermore, the choice of the kernel also defines the type of nonlinearity that underlies the model used. The choice of the kernel is therefore vital in the development of different algorithms, and the rationale behind the choice may be multiple. For instance, one of the most commonly used kernels is the Gaussian kernel since its performance is superior to that of other kernels, like polynomial kernels. However in [45], polynomial kernels are the preferred choice since the obtained solutions can be directly transformed into their corresponding Wiener or Volterra representation. In NLAEC applications, the enclosure impulse response is usually of very high order, e.g., hundreds of taps in mobile communication systems and even thousands of taps in room acoustic applications. This in particular makes the single application of most of the kernels, e.g., Gaussian kernels, inefficient for real-time applications.

Kernels verifying Mercer's condition [48], which guarantees the existence of a Hilbert space and an inner product associated with the kernel considered, are of particular interest in on-line learning. Essentially, convergence of the training algorithm to a unique optimum is ensured. This type of kernel can be used to construct other families of kernels that also meet these conditions [22], [47]. Several techniques have been applied to finding the best-suited kernel for a given application. Most of the techniques are based on convex optimization of the kernel designing problem, so they are only suitable for off-line applications, e.g., [61]. On the other hand, in [55], a multikernel approach has been proposed to choose the best kernel parameters in an adaptive fashion. However, only one type of kernel (i.e., Gaussian), with different parameters, is used.

Here, a kernel that consists of a weighted sum of linear and Gaussian kernels is proposed for the NLAEC applications. The motivation for this kernel is to separate the problem into linear and nonlinear subproblems. The proposed kernel is defined as

$$\begin{aligned}\kappa_{\Sigma}(i,j) &= \alpha\kappa_{\mathrm{L}}(i,j) + \beta\kappa_{\mathrm{G}}(i,j) \\ &= \alpha\mathbf{u}^{T}(i)\mathbf{u}(j) + \beta\exp(-\rho\left\|\mathbf{u}(i)-\mathbf{u}(j)\right\|_2^2)\end{aligned} \tag{28}$$

where $0 \leq \alpha \leq 1$, $\beta = (1-\alpha)$, and ρ is the Gaussian kernel parameter controlling its bandwidth. The kernel κ_Σ is positive definite and thus a reproducing kernel in RKHS, which immediately follows from the property that if $\alpha, \beta \geq 0$, then $\kappa_\Sigma = \alpha\kappa_\mathrm{L} + \beta\kappa_\mathrm{G}$ is positive definite provided that $\kappa_\mathrm{L}, \kappa_\mathrm{G}$ are two arbitrary positive definite kernels [62]. Moreover, this kernel elegantly fits into the SWL-KAPA since the parameters α and β give yet another degree of flexibility in the regularization strategy by expanding the regularization term in (18) on the last D observations as

$$\begin{aligned}
\lambda \left\|\hat{\boldsymbol{\omega}}\right\|^2 = \lambda \hat{\boldsymbol{\omega}}^T \hat{\boldsymbol{\omega}} &= \lambda \left(\sum_{i=1}^{D} a_i(D) \boldsymbol{\varphi}\left[\mathbf{u}(i)\right] \right)^T \left(\sum_{j=1}^{D} a_j(D) \boldsymbol{\varphi}\left[\mathbf{u}(j)\right] \right) \\
&= \lambda \sum_{i=1}^{D}\sum_{j=1}^{D} a_i a_j \boldsymbol{\varphi}(i)^T \boldsymbol{\varphi}(j) = \lambda \sum_{i=1}^{D}\sum_{j=1}^{D} a_i a_j \kappa_\Sigma(i,j) \\
&= \lambda \sum_{i=1}^{D}\sum_{j=1}^{D} a_i a_j \left[\alpha\kappa_\mathrm{L}(i,j) + \beta\kappa_\mathrm{G}(i,j)\right] \\
&= \lambda_\mathrm{L} \sum_{i=1}^{D}\sum_{j=1}^{D} a_i a_j \kappa_\mathrm{L}(i,j) + \lambda_\mathrm{G} \sum_{i=1}^{D}\sum_{j=1}^{D} a_i a_j \kappa_\mathrm{G}(i,j).
\end{aligned}$$

The regularization can now be more favored in one kernel than in the other by varying the weights $\lambda_\mathrm{L} = \lambda\alpha$ and $\lambda_\mathrm{G} = \lambda\beta$.

C. Sliding-Window Leaky Kernel Affine Projection Algorithm

A complete SWL-KAPA description is provided in Algorithm 1. The following variables are adopted: P is the APA projection order, D is the length of the memory buffers, i.e., the sliding-window length, μ is the step size, λ is the regularization parameter, $\mathbf{u}(i) = [u(i), u(i-1), ..., u(i-L+1)]^T$ is the $L \times 1$ input (far-end) signal vector, $d(i)$ is the desired (microphone) signal, $\hat{\mathbf{y}}(i) = [\hat{y}_1(i), ..., \hat{y}_P(i)]^T$ is a $P \times 1$ output vector, $\mathbf{a}(i)$ is the $D \times 1$ expansion coefficients vector, the dictionary $\mathbf{U}_\mathrm{D}$ is a $L \times D$ matrix, $e_\mathrm{AEC}(i)$ is the NLAEC error (residual) signal that is sent to the far-end side, $\mathbf{z}$ is a $P \times 1$ auxiliary vector, $\mathbf{G}(i) = [\boldsymbol{\Phi}(i)\boldsymbol{\Phi}^T(i) + \lambda\mathbf{I}]$ is the matrix defined in (27), and $\mathbf{e}(i) = [e_1(i), ..., e_P(i)]^T$ is the $P \times 1$ error vector. Note that although the kernel function was previously defined for only two vectors, the kernel κ_Σ in Algorithm 1, line 4, takes inputs comprising an $L \times 1$ vector $\mathbf{u}(i)$, and an $L \times D$ matrix $\mathbf{U}_\mathrm{D}(i-1)$. The outcome of this operation is simply a $D \times 1$ vector whose elements are the kernel evaluations of the current input vector with the D previous input vectors that constitute the dictionary.

Algorithm 1: Sliding-Window Leaky KAPA

initialization;

D, P, μ, λ, $a_1(1) = \mu e_1(1) = \mu d(1)$, $\mathbf{z} = \mathbf{0}$, $\mathbf{U}_{\mathrm{D}}(1) = \mathbf{u}(1)$;

while $\{\mathbf{u}(i), d(i)\}$ available **do**
  for $i = 2 \rightarrow end$ **do**
    for $k = 1 \rightarrow P$ **do**
      $\hat{y}_k(i) = \mathbf{a}^T(i-1)\boldsymbol{\kappa}_{\Sigma}\left[\mathbf{u}(i-k+1), \mathbf{U}_{\mathrm{D}}(i-1)\right]$;
      $e_k(i) = d(i-k+1) - \hat{y}_k(i)$;
    end for
    $e_{\mathrm{AEC}}(i) = e_1(i)$;
    $\mathbf{z} = \begin{bmatrix} \mathbf{a}_{end-P+2:end}(i-1) \\ 0 \end{bmatrix} + \mu \mathbf{G}^{-1}(i)\mathbf{e}(i)$;
    $\mathbf{a}(i) = \begin{bmatrix} (1-\lambda\mu)\left[\mathbf{a}(i-1) + \mathbf{z}_{1:P-1}\right] \\ z_P \end{bmatrix}$;
    $\mathbf{U}_{\mathrm{D}}(i) = [\mathbf{U}_{\mathrm{D}}(i-1) \quad \mathbf{u}(i)]$;
    if length$\mathbf{a}(i) > D$ **then**
      $a_1(i) \leftarrow \emptyset$ (`Delete first element`)
      $\mathbf{U}_{\mathrm{D},1}(i) \leftarrow \emptyset$ (`Delete first column vector`)
    end if
  end for
end while

IV. Evaluation

A. Simulated Nonlinear System

The simulated nonlinear system is a Hammerstein-like system implemented using a 3rd-order Volterra nonlinearity (instead of a memoryless nonlinearity as commonly comprises a Hammerstein system):

$$
\begin{aligned}
y_{\text{lin}}(i) &= \sum_{j=0}^{L} h_j u(i-j) \\
y_{\text{volt}}(i) &= \sum_{j=0}^{L_{\text{v}}} \sum_{k=j}^{L_{\text{v}}} h_{2,(j,k)} u(i-j)u(i-k) \\
&\quad + \sum_{j=0}^{L_{\text{v}}} \sum_{k=j}^{L_{\text{v}}} \sum_{l=k}^{L_{\text{v}}} h_{3,(j,k,l)} u(i-j)u(i-k)u(i-l) \\
y_{\text{nl}}(i) &= \sum_{j=0}^{L} h_j y_{\text{volt}}(i-j) \\
d(i) &= y_{\text{lin}}(i) + \sigma\, y_{\text{nl}}(i) + n(i) = f\left[u(i)\right] + n(i)
\end{aligned}
$$

where $u(i)$ is the far-end signal, $d(i)$ is the microphone signal, $n(i)$ is the near-end noise signal, $y_{\text{lin}}(i)$ is the linear echo, $y_{\text{nl}}(i)$ is the nonlinear echo, $y_{\text{volt}}(i)$ is the output of a 3rd-order Volterra filter, $\mathbf{h} = [h_0, \ldots, h_{L-1}]^T$ is a 400-taps measured acoustic room impulse response (RIR), and σ controls the linear-to-nonlinear ratio (LNLR), i.e., $\text{LNLR} = 10\log_{10}\dfrac{\sum_{j=1}^{N} y_{\text{lin}}^2(j)}{\sum_{j=1}^{N} y_{\text{nl}}^2(j)}$ with N the total length of the signals. The order (i.e., memory) of the linear adaptive filter is $L-1 = 399$, while the memory for the Volterra kernels is $L_{\text{v}} - 1 = 79$. Kernel in this case refers to the nested summation of each nonlinearity. The simulated nonlinear system has a physical meaning since the nonlinearities are concentrated locally on the amplifier and loudspeaker, which are then filtered by the RIR. The coefficients in the 3rd-order Volterra kernels (i.e., h_2 and h_3) were obtained from an i.i.d. Gaussian random process. This choice makes the Volterra kernels fully populated which may generalize well for any type of loudspeaker [3], [63], [64]. Moreover, the 3rd-order nonlinearity demonstrates the validity of the SWL-KAPA using the kernel κ_Σ in dealing with high-order nonlinearities, without having to specify the order of the nonlinearity. By contrast, in adaptive Volterra filters, the order of the nonlinearity has to be explicitly set in advance and besides has a direct impact on the computational complexity.

B. Competing Nonlinear Adaptive Filters: Hammerstein and Volterra

The two nonlinear adaptive filters that will be compared to SWL-KAPA are defined by the input-output relationship

$$y_{\text{F}}(i) = \mathbf{h}_{\text{F}}^T \mathbf{u}_{\text{F}}(i) \tag{29}$$

where $\mathbf{h}_{\text{F}}$ is a vector containing L_{F} filter coefficients and $\mathbf{u}_{\text{F}}(i)$ is the corresponding vector whose L_{F} entries are formed with a nonlinear expansion of the input samples in $\mathbf{u}(i)$. This expression can be interpreted as an FIR filter with L_{F} coefficients in which the entries of $\mathbf{u}_{\text{F}}(i)$ are used in place of the samples $\mathbf{u}(i)$. If $\mathbf{u}_{\text{F}}(i)$ is chosen equal to $\mathbf{u}(i)$, this corresponds to a linear FIR filter with $L_{\text{F}} = L$ coefficients. It will be convenient to subdivide $\mathbf{u}_{\text{F}}(i)$ and $\mathbf{h}_{\text{F}}(i)$ into appropriate sub-vectors as

$$\mathbf{u}_{\text{F}}(i) = \left[\mathbf{u}^T(i),\ \mathbf{u}_2^T(i),\ \mathbf{u}_3^T(i)\right]^T, \tag{30}$$

$$\mathbf{h}_{\text{F}}(i) = \left[\mathbf{h}^T(i),\ \mathbf{h}_2^T(i),\ \mathbf{h}_3^T(i)\right]^T, \tag{31}$$

so that

$$y_{\text{F}}(i) = \mathbf{h}^T\mathbf{u}(i) + \mathbf{h}_2^T\mathbf{u}_2(i) + \mathbf{h}_3^T\mathbf{u}_3(i). \tag{32}$$

It is clear that the linear APA recursion in (11) and (12) can be applied directly using (29) and (30), when the dimensions are adjusted accordingly.

- Hammerstein filter

Due to its intrinsic simplicity, the Hammerstein filter has been studied and used for many applications including NLAEC [10]. The Hammerstein filter is formed with a memoryless nonlinearity followed by a linear filter. Since the nonlinearity is usually expressed as a polynomial function and the linear filter has a finite impulse response of L samples, it is considered as a simple member of the class of Volterra filters. The sub-vectors of $\mathbf{u}_{\mathrm{F}}(i)$ in (30) are defined by

$$\mathbf{u}_2(i) = [u^2(i), u^2(i-1),\ \cdots, u^2(i-L+1)]^T,$$
$$\mathbf{u}_3(i) = [u^3(i), u^3(i-1), \cdots, u^3(i-L+1)]^T.$$

The total number of filter coefficients is therefore $L_{\mathrm{H}} = 3L$.

- Volterra filter

Volterra filters are characterized by input-output relationships that result from two truncations of the discrete Volterra series [46], namely a memory truncation of the Volterra kernel orders ($L_2 - 1$ and $L_3 - 1$ in this case), and a nonlinearity order truncation (3 in this case) to limit the number of Volterra kernels. The Volterra kernels can be assumed to be symmetric and thus can be recast in the more compact triangular form

$$\mathbf{u}_2(i) = [u^2(i), u(i)u(i-1), ..., u^2(i-1),$$
$$u(i-1)u(i-2), ..., u^2(i-L_2+1)]^T$$

with dimension $L_2(L_2+1)/2 \times 1$ and

$$\mathbf{u}_3(i) = [u^3(i), u(i)u(i)u(i-1), ..., u(i)u(i-1)u(i-2),$$
$$..., u^3(i-L_3+1)]^T$$

with dimension $L_3(L_3+1)(L_3+2)/6 \times 1$. Thus, the total number of filter coefficients is $L_{\mathrm{V}} = L + L_2(L_2+1)/2 + L_3(L_3+1)(L_3+2)/6$

C. Simulation Results

The performance measure is the *echo return loss enhancement* (ERLE), which is given as

$$\mathrm{ERLE}(m) = 10\log_{10}\frac{\sum_{j=1}^{q} d^2[(m-1)q+j]}{\sum_{j=1}^{q} e^2[(m-1)q+j]} \quad m = 1, ... \tag{33}$$

and which can be seen as the achieved echo attenuation averaged over time frames of length q. Simulations are performed using speech signals (female speech sampled at 8 kHz) and i.i.d. background noise $n(i)$

with a SNR of 20 dB. Results are provided for two LNLR values 6 dB and 12 dB. Some parameters in every algorithm are the same, in order to make the comparison as fair as possible: $\mu = 1$, $P = 24$, $\rho = 1$, $D = 1000$, $\lambda = 10^{-2}$. The input dimension of the Gaussian and linear kernel is $L = 400$ and the linear APA filter order is $L - 1 = 399$. The linear (dark-solid) and nonlinear (light-solid) part of the echo are shown in the bottom of each figure as a reference for the instantaneous LNLR. In each scenario, the ERLE of four algorithms is compared over time. The algorithms are the SWL-KAPA, the 3rd-order Volterra and Hammerstein filters updated using APA, and the linear APA. The 3rd-order Volterra filter is computed using lower projection and memory orders, i.e., $P = 3$, $L_2 - 1 = 349$ and $L_3 - 1 = 349$. The reason for this adjustment in the Volterra filter is simply that it is too expensive to compute the 3rd-order Volterra using the same APA updating as in the other three algorithms; even a low APA projection order, $P = 3$, results in excessive memory usage and computation times.

Fig. 2 shows the ERLE of the four algorithms at different LNLR. As expected, in the presence of non-negligible nonlinear echo, the linear APA provides the worst performance, followed by the Hammerstein and Volterra filters. The Volterra filter either obtains the same results as the Hammerstein filter or it performs $5-6$ dB better. These differences become more clear in the case with 12 dB LNLR. Interestingly, SWL-KAPA generally outperforms any other algorithm for both 6 and 12-dB LNLR, obtaining an almost constant improvement over the other three algorithms. The improvement with respect to the linear APA is at some time instants 10 to 11 dB in the 12-dB LNLR case. In this comparison, the kernel weights are equal, $\alpha = \beta = 0.5$, and thus the freedom to choose different values is not exploited. To this end, Fig. 3 shows the ERLE obtained using SWL-KAPA for three different pairs of values for the kernel weights, $\{\alpha, \beta\} = \{0.15, 0.85\}, \{0.5, 0.5\}, \{0.85, 0.15\}$. It is clear that large ERLE improvements can be achieved by choosing different kernel weight values at different times, essentially depending on the instantaneous LNLR. Fig. 4 is similar to Fig. 2, but in this case the weights are variables, choosing the best option among the three at each time instant. It is clear that, again, SWL-KAPA generally outperforms any other algorithm, now with a general ERLE improvement with respect to Fig. 2. Particularly, around 0.35 s, SWL-KAPA goes from being outperformed by the Volterra filter, in Fig. 2, to reaching the same ERLE in Fig. 4. These observations show that a dynamic choice of the kernel weights can further improve the performance of SWL-KAPA. The proposed SWL-KAPA consistently outperforms the linear APA, resulting in 12-dB maximum improvement in ERLE at a computational cost that is only 4.6 times higher. Moreover, it is shown that the SWL-KAPA outperforms, by 4 to 6 dB, the Volterra-based NLAEC, which itself is 413 times more expensive than the linear APA. Moreover, the Volterra filter performance

during high instantaneous LNLR (e.g., between 0.5 s and 0.8 s in both cases) decreases considerably with respect to low LNLR, which is a known fact [16]; the linear APA performance obviously increases during instantaneous high LNLR. In contrast, SWL-KAPA maintains its superior performance constantly in both cases, during high or low instantaneous LNLR. This fact indeed constitutes another appealing characteristic of SWL-KAPA.

Although, comparing different methods appears to be cumbersome due to the fact that no simulation using a standard set of signals, echo paths, and, most importantly, nonlinear echo models, is found, we will now relate our results to those given in the NLAEC literature. In [11], a 2nd-order Volterra filter has been found to result in 4 to 5 dB improvement when using Gaussian noise as an input signal. In [12], a comparison has been presented among several nonlinear cascaded filters, namely Hammerstein, Wiener, Wiener-Hammerstein, and Bilinear. In a simulated echo path with Gaussian noise as input signal, up to 8.5 dB of improvement has been achieved. Compared to those results, SWL-KAPA achieves better ERLE performance, even with speech as the input signal. For speech input signals, up to 8 dB of ERLE improvement has been reported in [7] by using a memoryless (i.e., Hammerstein-like) preprocessor, however, this improvement has only been observed at specific time instants and in the almost noise-free case. In [10] an orthogonalized version of the polynomial Hammerstein filter (called "power filter") has been suggested to improve convergence. An improvement of 5 to 6 dB has been reported compared with a linear filter. SWL-KAPA still outperforms these latter examples even when they show a simpler model than a 3rd-order Volterra, and in the first case an almost noise-free environment. In [8], an improvement of 5 dB has been achieved using 2nd-order Volterra filters. In [13], a comparison has been presented among a linear filter, a time-domain Volterra filter, and the so-called multidelay filter (MDF), which applies partitioned block methods for fast convolution in the DFT domain. Results have shown around 5 dB of average ERLE improvement and up to 8 dB at specific times using the MDF as compared with the linear filter. Even though SWL-KAPA outperforms these computationally cheaper 2nd-order Volterra filters, by 4 dB in the latter case, it is necessary to stress the fact that SWL-KAPA can easily deal with 3rd-order Volterra nonlinearities. The only 3rd-order Volterra-based NLAEC is found in [15], where two models have been proposed, showing improvements of 5 and 2 dB, respectively. It is clear that SWL-KAPA outperforms these latter results and moreover at a much cheaper computational complexity. In [16], an interesting approach has been proposed with a convex combination of filters. The combination of linear and nonlinear filters achieves the best possible results, depending on the LNLR. Results have been given for a simulated 2nd-order Volterra nonlinear path with speech as input signal and in the noise-free case.

TABLE I

COMPUTATIONAL COMPLEXITY $L = 400$, $L_{\mathrm{H}} = 1200$ $P = 24$, $L_{\mathrm{V}} = 7269025$, $P_{\mathrm{V}} = 3$ AND $D = 1000$

APA update	Computation	Operations
Linear	$(P^2 + 2P)L + P^3 + P^2$	264000
Hammerstein 3rd order	$(P^2 + 2P)L_{\mathrm{H}} + P^3 + P^2$	763200
Volterra 3rd order	$(P_{\mathrm{V}}^2 + 2P_{\mathrm{V}})L_{\mathrm{V}} + P_{\mathrm{V}}^3 + P_{\mathrm{V}}^2$	109035411
SWL-KAPA	$(L + 2L)D + PD + P^3$	1237824

By varying the ratio between the linear and nonlinear contribution in the echo path, the performance of their proposed algorithm (which is a convex combination of a linear and a 2nd-order Volterra filter) and the standard adaptive 2nd-order Volterra filter is compared. Improvements of 4 dB and 13 dB are reported for medium and high LNLR compared to the linear filter. This is the only reported algorithm that achieves better performance, however it should be stressed that in [16] the explicit calculation of the 2nd-order Volterra kernel is necessary, hence increasing its computational complexity. Moreover, a simpler model for the nonlinear echo (i.e., a smaller quadratic Volterra kernel, without filtering by the RIR) is considered in a noise-free case.

D. Computational Complexity

The computational complexity, i.e., the number of operations per sample update, of the linear APA is $(P^2 + 2P)L + P^3 + P^2$. SWL-KAPA has the same matrix inversion with $\mathbf{G}(i)$, but on the other hand does not have the explicit calculations with the filter coefficient vector. Instead SWL-KAPA calculates the error signal based on kernel evaluations. The linear kernel evaluation requires L operations and the Gaussian kernel requires roughly $2L$ operations plus the exponential calculation. These operations are repeated D times per sample update. To calculate the echo estimate, SWL-KAPA dot-multiplies the kernel output with the vector of expansion coefficients for each output sample, summing up to PD operations altogether. Overall, the SWL-KAPA computational complexity is $(L + 2L)D + PD + P^3$.

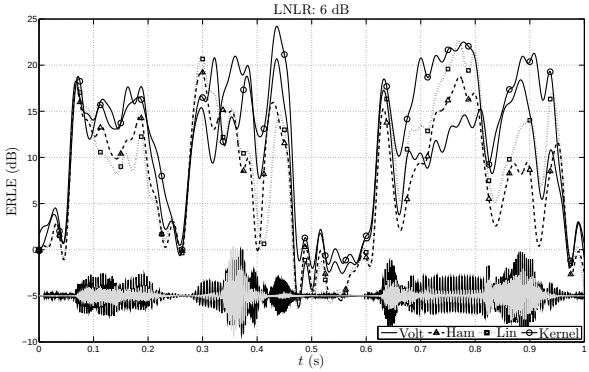
We now calculate the resulting filter vector lengths to obtain the overall computational complexity when using an APA update. The filter coefficient vector for linear APA and the input dimension for SWL-KAPA is $L = 400$. The Hammerstein filter vector has $L_{\mathrm{H}} = 3L = 1200$ coefficients and the Volterra filter vector has $L_{\mathrm{V}} = L + L_2(L_2+1)/2 + L_3(L_3+1)(L_3+2)/6 = 7269025$ coefficients, where the memory of the Volterra kernels is $L_2 - 1 = L_3 - 1 = 349$. The projection order is $P = 24$ except for the Volterra filter, which has $P_{\mathrm{V}} = 3$. Table I shows the overall complexity of the four algorithms.

SWL-KAPA is only 4.6 times more computationally expensive than the linear APA and 1.6 times that of the 3rd-order Hammerstein filter. On the other hand, the 3rd-order Volterra with $P_{\mathrm{V}} = 3$ is 88 times more expensive than the SWL-KAPA and 413 times more than the linear APA.

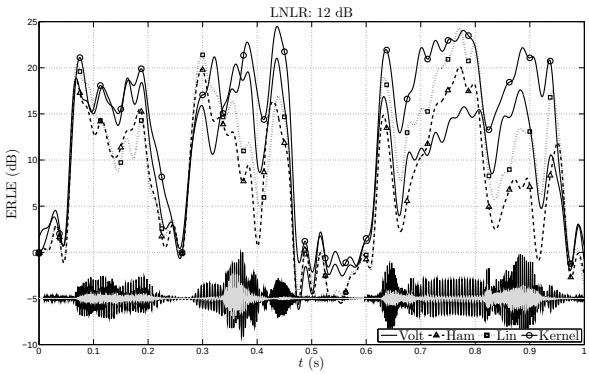
One can argue that the employed Volterra filter is merely the direct symmetric implementation [46], which indeed is not as cheap as the simplified Volterra filters of [8]. One can also argue that better convergence rates can be obtained using orthogonalized versions [10] or frequency-domain versions [14]. However, the complexity of the SWL-KAPA can also be reduced by implementing a sliding Gram matrix or error reuse [17]. So the aim here is not to make an exhaustive comparison of the different methods but to demonstrate the potential of an alternative approach to NLAEC based on kernel adaptive methods. Much more work has to be done in this context, particularly in view of complexity reduction, optimization of the sliding window with respect to the leakage term, the use of other kernels, the "kernelization" of other adaptive filters used in AEC, the dynamic selection of the kernel weights, etc.

V. Conclusions

The sliding-window leaky KAPA (SWL-KAPA) has been introduced and its validity and potential in tackling the NLAEC problem has been demonstrated. SWL-KAPA uses a kernel function that consists of a weighted sum of linear and Gaussian kernels. The motivation for using this kernel is to separate the problem into linear and nonlinear subproblems, where weighting the kernels also imposes different forgetting mechanisms in the sliding window, which in turn translates to more flexible regularization. The result of this is that large performance improvements can be achieved by choosing different kernel weight values at different times, depending on the instantaneous LNLR. The combination of SWL-KAPA with the proposed kernel function results in excellent ERLE performance compared to that of a linear APA, 3rd-order Hammerstein, and Volterra filters. In our experiment set-up, the proposed SWL-KAPA consistently outperforms the linear APA, resulting in 12-dB maximum ERLE improvement at a computational cost that is only 4.6 times higher. Moreover, it is shown that the SWL-KAPA outperforms, by 4 to 6 dB, the Volterra-based NLAEC, which itself is 413 times more expensive than the linear APA.

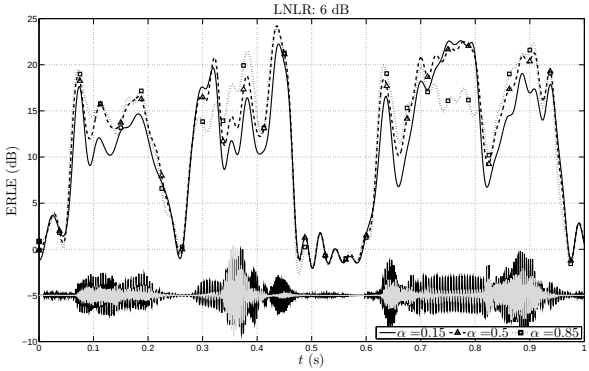


(a)

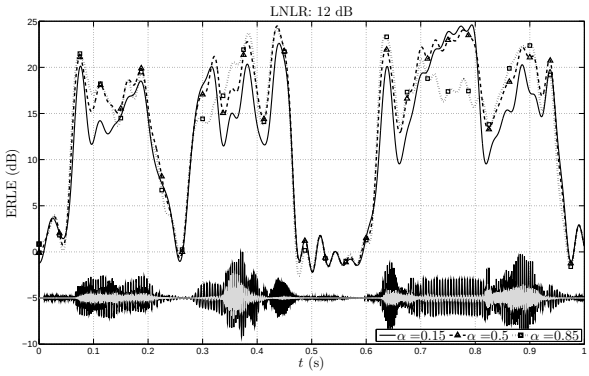


(b)

Fig. 2. ERLE comparison of four algorithms: Volterra (solid), Hammerstein (triangle-dashed), linear APA (square-dotted) and SWL-KAPA (circle-solid). (a) 6-dB LNLR. (b) 12-dB LNLR.

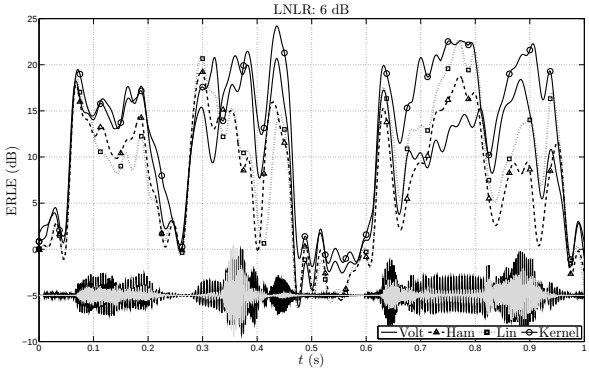


(a)

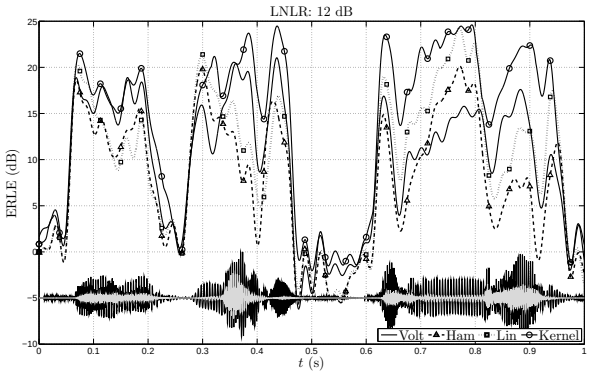


(b)

Fig. 3. ERLE comparison of SWL-KAPA using different kernel weights: $\alpha = 0.15$ (solid), $\alpha = 0.5$ (triangle-dashed) and $\alpha = 0.85$ (square-dotted). (a) 6-dB LNLR. (b) 12-dB LNLR.



(a)



(b)

Fig. 4. ERLE comparison of four algorithms: Volterra (solid), Hammerstein (triangle-dashed), linear APA (square-dotted) and SWL-KAPA (circle-solid). (a) Best weights $(0.15 - 0.5 - 0.85)$ with 6-dB LNLR. (b) Best weights $(0.15 - 0.5 - 0.85)$ with 12-dB LNLR

REFERENCES

[1] J. Benesty, T. Gänsler, D. R. Morgan, M. M. Sondhi, and S. L. Gay, *Advances in Network and Acoustic Echo Cancellation*. Berlin: Springer-Verlag, 2001.

[2] P. Gil-Cacho, T. van Waterschoot, M. Moonen, and S. H. Jensen, "Multi-microphone acoustic echo cancellation using multi-channel warped linear prediction of common acoustical poles," in *Proc. 18th European Signal Process. Conf.*, (EUSIPCO'10), Aalborg, Denmark, Aug. 2010, pp. 2121–2125.

[3] ——, "Study and characterization of odd and even nonlinearities in electrodynamic loudspeakers by means of periodic random-phase multisine," in *Preprints AES 127th convention*, Oct. 2009, Preprint 7841.

[4] A. N. Birkett and R. A. Goubran, "Nonlinear echo cancellation for handsfree telephony using neural networks," in *Proc. 1994 IEEE Workshop Neural Netw. Signal Process.*, (NNSP'94), Ermioni , Greece, Sept. 1994, vol. 1, pp. 249 – 258.

[5] A. B. Rabaa and R. Tourki, "Nonlinear echo cancellation for hands-free telephony," in *Proc. 10th European Signal Process. Conf.*, (EUSIPCO'00), Tampere, Finland, Sept. 2000, pp. 1875–1878.

[6] B. S. Nollett and D. L. Jones, "Nonlinear echo cancellation for hands-free speakerphones," in *Proc. 1997 IEEE Workshop Nonlinear Signal and Image Process.*, (NSIP'97), London, UK, May 1997, vol. 1, Paper number 521.

[7] A. Stenger and W. Kellermann, "Adaptation of a memoryless preprocessor for nonlinear acoustic echo cancelling," *IEEE Trans. Signal Process.*, vol. 80, no. 9, pp. 1747–1760, Sept. 2000.

[8] A. Fermo, A. Carini, and G. L. Sicuranza, "Simplified Volterra filters for acoustic echo cancellation in GSM receivers," in *Proc. 10th European Signal Process. Conf.*, (EUSIPCO'00), Tampere, Finland, Sept. 2000, pp. 2413–2416.

[9] G. L. Sicuranza, A. Carini, and A. Fermo, "Nonlinear adaptive filters for acoustic echo cancellation in mobile terminals," in *Nonlinear Signal and Image Processing: Theory, Methods, and Applications*, ser. The Electrical Engineering and Applied Signal Processing Series, K. E. Barner and G. R. Arce, Eds. Florida: CRC Press LLC, 2004, ch. 7, pp. 223–255.

[10] F. Kuech and W. Kellerman, "Orthogonalized power filters for nonlinear acoustic echo cancellation," *Signal Process.*, vol. 86, pp. 1168–1181, Nov. 2006.

[11] A. Stenger and R. Rabenstein, "Adaptive Volterra filters for acoustic echo cancellation," in *Proc. 1999 IEEE-EURASIP Workshop Nonlinear Signal and Image Process.*, (NSIP'99), Antalya, Turkey, June 1999.

[12] J. P. Costa, A. Lagrange, and A. Arliaud, "Acoustic echo cancellation using nonlinear cascade filters," in *Proc. 2003 IEEE Int. Conf. Acoust. Speech Signal Process.*, (ICASSP'03), Hong Kong, China, Apr. 2003, pp. 389–392.

[13] F. Kuech and W. Kellerman, "A novel multidelay adaptive algorithm for Volterra filters in diagonal coordinates," in *Proc. 2004 IEEE Int. Conf. Acoust. Speech Signal Process.*, (ICASSP'04), Montreal, Quebec, Canada, May 2004, pp. 869–872.

[14] F. Küch, "Adaptive polynomial filters and their application to nonlinear acoustic echo cancellation," Ph.D. dissertation, Friedrich–Alexander–Universität Erlangen–Nürnberg, 2005.

[15] A. Guérin, G. Faucon, and R. L. Bouquin-Jeannés, "Nonlinear acoustic echo cancellation based on Volterra filters," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 672–683, Nov. 2003.

[16] L. A. Azpicueta-Ruiz, M. Zeller, J. Arenas-García, and W. Kellermann, "Novel schemes for nonlinear acoustic echo cancellation based on filter combinations," in *Proc. 2010 IEEE Int. Conf. Acoust. Speech Signal Process.*, (ICASSP'10), Taipei, Taiwan, Apr. 2010, pp. 193–196.

[17] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Hoboken, NY: Wiley, 2010.

[18] S. V. Vaerenbergh, "Kernel methods for nonlinear identification, equalization and separation of signals," Ph.D. dissertation, Universidad de Cantabria, 2010.

[19] P. Bouboulis and S. Theodoridis, "The complex Gaussian kernel LMS algortihm," in *Proc. 20th Int. Conf. Artificial Neural Netw.*, (ICANN'10), Thessaloniki, Greece, Aug. 2010, pp. 11–20.

[20] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[21] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 165–176, Aug. 2004.

[22] B. Schölkopf and A. J. Smola, *Learning with kernels*. Cambridge, MA: MIT Press, 2002.

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

[24] J. M. Gil-Cacho, T. van Waterschoot, M. Moonen, and S. H. Jensen, "Nonlinear acoustic echo cancellation based on a parallel-cascade kernel affine projection algorithm," in *Proc. 2012 IEEE Int. Conf. Acoust. Speech Signal Process.*, (ICASSP'12), Kyoto, Japan, Mar. 2012, pp. 33–36.

[25] T. van Waterschoot, G. Rombouts, P. Verhoeve, and M. Moonen, "Double-talk-robust prediction error identification algorithms for acoustic echo cancellation," *IEEE Trans. Signal Process.*, vol. 55, no. 3, pp. 846–858, Mar. 2007.

[26] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NY: Prentice Hall, 2002.

[27] P. S. R. Diniz, *Adaptive filtering: Algorithms and Practical Implementations*. Boston, MA: Springer, 2008.

[28] T. van Waterschoot, G. Rombouts, and M. Moonen, "Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement," *Signal Process.*, vol. 88, no. 3, pp. 594–611, Mar. 2008.

[29] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*. New York: Wiley, 1977.

[30] A. Neumaier, "Solving ill-conditioned and singular linear systems: A tutorial on regularization," *SIAM Rev.*, vol. 40, no. 3, pp. 636–666, 1998.

[31] 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. 2924–2936, Oct. 2004.

[32] R. D. Gitlin, H. C. Meadors, and S. B. Weinstein, "The tap-leakage algorithm: An algorithm for the stable operation of a digitally-implemented fractionally-spaced equalizer," *Bell Sys. Tech. J.*, vol. 61, no. 10, pp. 1817–1840, Oct. 1982.

[33] K. Mayyas and T. Aboulnasr, "Leaky LMS algorithm: MSE analysis for Gaussian data," *IEEE Trans. Signal Process.*, vol. 45, no. 4, pp. 927–934, Apr. 1997.

[34] B. Widrow and M. Hoff, "Adaptive switching circuits," in *Proc. WESCON Conv. Rec.*, vol. 4, 1960, pp. 96–140.

[35] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electronics and Communication in Japan*, vol. 67, no. 5, pp. 19–27, Aug. 1984.

[36] S. S. Oh, M. Ali, and D. Linebarger, "Adaptive filtering method and apparatus employing modified fast affine projection algorithm," US Patent No: 6 137 881, Oct. 24, 2000.

[37] G. Q. Jin, "Modified affine projection algorithm for non-stationary signal," US Patent No: 7 280 654, Oct. 09, 2007.

[38] B. Baykal and A. Constantinides, "Underdetermined-order recursive least-squares adaptive filtering: the concept and algorithms," *IEEE Trans. Signal Process.*, vol. 45, no. 2, pp. 346–362, Feb. 2008.

[39] D. R. Morgan and S. G. Kratzer, "On a class of computationally efficient, rapidly converging, generalized nlms algorithms," *IEEE Signal Process. Lett.*, vol. 3, pp. 245 – 247, Aug. 1996.

[40] P. Gil-Cacho, T. van Waterschoot, M. Moonen, and S. H. Jensen, "Regularized adaptive notch filters for acoustic howling

suppression," in *Proc. 17th European Signal Process. Conf.*, (EUSIPCO'09), Glasgow, Scotland, UK, Aug. 2009, pp. 2574–2578.

[41] B. D. Rigling, "Subspace leaky LMS," *IEEE Signal Process. Lett.*, vol. 11, no. 2, pp. 136 – 139, Feb. 2004.

[42] G. B. Chandra and A. Mitra, "A fast adaptive echo canceler with leaky proportionate NLMS algorithm," in *Proc. IET-UK Int. Conf. Inf. Commun. Technol. Elect. Sc.*, (ICTES'07), Tamil Nadu, India, Dec. 2007, pp. 611–615.

[43] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. T. Roy. Soc. A*, vol. 209, pp. 415–446, 1909.

[44] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning," *Automat. Rem. Cont.*, vol. 25, pp. 821–837, 1964.

[45] M. O. Franz and B. Schölkopf, "A unifying view of Wiener and Volterra theory and polynomial kernel regression," *Neural Computation*, vol. 18, no. 12, pp. 3097–3118, 2006.

[46] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley, 2000.

[47] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[48] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[49] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proc. 14th Ann. Conf. Comput. Learning Theory*, (COLT'01), Amsterdam, The Netherlands, 2001, pp. 416–426.

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

[51] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 275–285, Aug. 2004.

[52] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 950–961, Dec. 2004.

[53] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares support vector machines," in *Proc. 2000 IEEE Int. Symp. Circuits Syst.*, (ISCAS'00), Geneva, Switzerland, May 2000, vol. 2, pp. 757–760.

[54] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058 – 1067, Mar. 2009.

[55] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672 – 4682, Sept. 2012.

[56] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. D. Moor, "A comparison of pruning algorithms for sparse least squares support vector machines," *Lect. Notes Comp. Sc.*, vol. 1, pp. 247–253, 2004.

[57] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2003.

[58] S. V. Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *Proc. 2010 IEEE Int. Conf. Acoust. Speech Signal Process.*, (ICASSP'10), Dallas, TX, USA, Mar. 2010, pp. 882–885.

[59] S. V. Vaerenbergh, J. Vía, and I. Santamaría, "Nonlinear system identification using a new sliding-window kernel RLS algorithm," *J. Commun.*, vol. 2 no. 3, pp. 1–8, May 2007.

[60] G. Wahba, *Spline models for observational data*. Philadelphia, PA: SIAM, 1990.

[61] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Sept. 2004.

[62] C. Berg, J. Peter, R. Christensen, and P. Ressel, *Harmonic Analysis on Semigroups*. New York: Springer-Verlag, 1984.

[63] A. J. M. Kaizer, “Modeling of the nonlinear response of an electrodynamic loudspeaker by a Volterra series expansion,” *J. Audio Eng. Soc.*, vol. 35, no. 6, pp. 421–432, June 1987.

[64] M. J. Reed and M. O. J. Hawksford, “Identification of discrete Volterra series using maximum length sequences,” *IEE Proc.-Circuits Devices Syst.*, vol. 143, no. 5, pp. 241 – 248, Oct. 1996.