ONLINE KERNEL RECEIVER FOR MULTIACCESS MIMO CHANNELS

Konstantinos Slavakis[1] *Pantelis Bouboulis*[2] *Sergios Theodoridis*[2]

[1] University of Peloponnese,
Dept. of Telecommunications Science and Technology,
Tripolis 22100, Greece.
Email: slavakis@uop.gr.

[2] University of Athens,
Dept. of Informatics and Telecommunications,
Athens 15784, Greece.
Email: {turambar,stheodor}@di.uoa.gr.

ABSTRACT

This paper introduces an online receiver for multiaccess multiple-input multiple-output (MIMO) channels by using kernel functions. The receiver implicitly operates, with linear complexity, in a general infinite dimensional Reproducing Kernel Hilbert Space (RKHS). Given a training sequence of symbols, the problem is viewed as a sequential multiregression oprimization problem, where any continuous convex cost function, even non-differentiable, can be used. Numerical examples show that the proposed receiver outperforms linear receivers, built upon the well-known orthogonal space-time block codes, in cases where severe multiaccess interference is present.

1. INTRODUCTION

Kernel methods have become recently the main tool for translating classical linear supervised and unsupervised techniques to nonlinear learning algorithms [1]. The key point of kernel methods is a nonlinear (implicit) mapping, which "transfers" the processing from a low dimensional Euclidean data space to a very high (possibly infinite) dimensional Reproducing Kernel Hilbert Space (RKHS) [1]. This nonlinear mapping is supplied by a kernel function which, basically, defines the RKHS.

Kernel methods have been shown to be highly successful mainly via batch settings, i.e., a finite number of data are available beforehand, like the celebrated Support Vector Machine (SVM) framework [1]. The SVM approach has been already employed with success to MIMO channels [2]. However, batch methods show prohibitively high computational complexity problems when applied to online settings, i.e., cases where data arrive sequentially, and where environments exhibit slow variation. Although sliding window versions of the SVM methodology have been already developed, genuine online kernel methods are needed [3, 4].

Orthogonal Space-Time Block Codes (OSTBC) is a powerful technique which enjoys full diversity gain and low decoding complexity in Multiple-Input Multiple-Output (MIMO) environments [5, 6]. In the case where only one transmitter (Tx) is present (point-to-point MIMO), in additive Gaussian channels, the optimal Maximum Likelihood (ML) detector, for this class of codes, is realized by a simple linear filter followed by a symbol-by-symbol decoder [7]. However, in multiaccess environments, where a multiple number of Txs is present (see Fig. 1), the complexity of the ML detector becomes high, and the performance of the OSTBC is degraded due to severe MultiAccess Interference (MAI), which can no longer be modeled as white noise [7, 8]. Suboptimal linear receivers, with excellent performance and low computational complexity have already been proposed for OSTBC in multiaccess MIMO channels, e.g., refer to [7, 8] and the references therein. However, in order for the linear receivers to successfully suppress MAI, self-interference, and noise, a sufficient number of degrees of freedom is neccessary, imposing thus limitations on the number of Txs, as well as on the rate of the OSTBC employed (see Section 2).

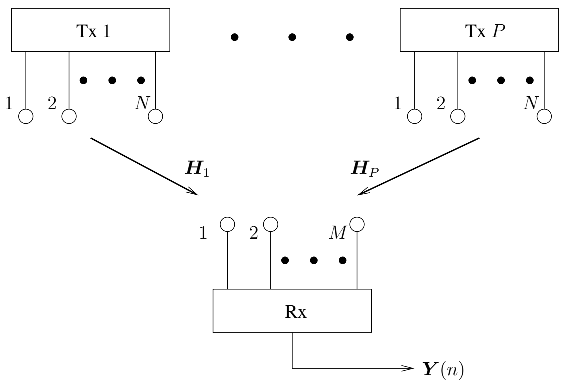

Fig. 1. Multiaccess MIMO model. A number of P transmitters (Txs), with N antennas per Tx, send symbols via the channel matrices $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_P$, to a receiver (Rx) with M antennas.

This paper introduces a kernel-based online receiver for MIMO systems. Unlike the SVM approach [2], here data are considered to be of sequential nature, as dictated by possibly slow changing channels. The receiver implicitly operates in a generally infinite dimensional RKHS, and the detection problem is modeled as an online multiregression optimization task. As opposed to [2, 3], where only a quadratic cost function was employed, the introduced algorithm is capable of accommodating any continuous convex function, even non-differentiable. The introduced receiver is of linear complexity with respect to the unknown parameters. Sparsification issues are also discussed, in order to deal with computational complexity and memory limitation issues of online systems. The numerical results show that the proposed receiver operates well in cases where linear receivers face problems due to limited degrees of freedom, as imposed, for example, by a large number of Txs, a small number of antennas at the Rx or by the rate of the employed OSTBC.

2. THE MULTIACCESS MIMO MODEL

We will denote the set of all integers, nonnegative integers, positive integers, real and complex numbers by $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{Z}_{>0}$, $\mathbb{R}$ and $\mathbb{C}$ respectively. For any integers $j_1 \leq j_2$, we denote by $\overline{j_1, j_2}$ the set $\{j_1, j_1+1, \ldots, j_2\}$.

As in [7, 8], we consider in Fig. 1 a multiaccess MIMO system with P users or transmitters (Txs), N antennas per Tx, and M antennas at the receiver (Rx). Henceforth, the index n will enumerate the number of data blocks, both for the transmitted and the received signals. Since the nature of data is considered to be sequential, we let $n \in \mathbb{Z}_{\geq 0}$. In order to unify the notation, we follow the one developed for the OSTBC frame in [7, 8]. As such, the n-th block of signals available at the Rx is given by

$$\boldsymbol{Y}(n) = \sum_{p=1}^{P} \boldsymbol{X}_p(\boldsymbol{s}_p(n))\boldsymbol{H}_p + \boldsymbol{V}(n) \quad \in \mathbb{C}^{T\times M}. \tag{1}$$

The vector $\boldsymbol{s}_p(n) := [s_{p1}(n), \ldots, s_{pK}(n)]^t \in \mathbb{C}^K$, where t stands for transposition, is the vector that gathers a number of K symbols to be encoded at every n-th block. The mapping $\boldsymbol{X}_p : \mathbb{C}^K \to \mathbb{C}^{T\times N} : \boldsymbol{s}_p(n) \mapsto \boldsymbol{X}_p(\boldsymbol{s}_p(n))$ stands for the encoder at the Tx, where the integer T is the number of symbols after the encoding procedure. To keep the discussion simple, we assume here that all Txs have identical block space-time encoders, i.e., $\boldsymbol{X}_p = \boldsymbol{X}$, $\forall p \in \overline{1,P}$. The rate of the code is denoted by $R := K/T$. The symbol $\boldsymbol{H}_p \in \mathbb{C}^{N\times M}$ is the channel matrix between the p-th Tx and the Rx. The (x, y)-th component of $\boldsymbol{H}_p$ gives the complex gain between the x-th antenna of the p-th Tx and the y-th antenna of Rx. The symbol $\boldsymbol{V}(n) \in \mathbb{C}^{T\times M}$ stands for the noise matrix, whose elements are i.i.d. zero-mean complex random variables, with variance σ_V^2.

For the present study, the channel matrices $\boldsymbol{H}_1, \ldots, \boldsymbol{H}_P$ are assumed uknown. For a linear receiver to ideally suppress MAI, and self-interference, a sufficient number of degrees of freedom is necessary. This is quantified by the inequality $PK < MT$ [7, 8]. The proposed kernel-based receiver will not follow such a restriction. On the contrary, the numerical examples will be developed for the case where $PK \geq MT$.

Regarding the encoding procedure, we consider in this study two cases.

1. The Orthogonal Space-Time Block Codes (OSTBC) frame, where the mapping $\boldsymbol{X}_p$ is defined in [6]. For more details on the encoding, the decoding, and their application, the reader is referred to [6–8], and to the references therein.
2. The encoding procedure for the proposed kernel-based receiver is given by the following simple mapping: let $K := T := 1$, i.e., each block contains only a single symbol, and

$$\boldsymbol{X}_p : \mathbb{C} \to \mathbb{C}^{1\times N} : s \mapsto [s, s, \ldots, s], \quad \forall p \in \overline{1,P}.$$

   In other words, at every block n, a single symbol is transmitted simultaneously by every antenna of the Tx. The decoder operates simply on a symbol-by-symbol basis.

Among the P Txs, we identify a number of $L \leq P$ of them as the Transmitters Of Interest (TOI), whose transmitted symbols need to be identified.

2.1. Processing prior the signal enters the receiver

To make notation easier to follow, and to work into real Euclidean spaces instead of complex ones, we introduce some preprocessing for the received signal, prior this enters the receiver (see Fig. 2).

First, let us define the mapping $\underline{\boldsymbol{M}} := \mathrm{vect}(\boldsymbol{M})$, where vect stands for the standard column stacking operation, and $\boldsymbol{M}$ is any matrix. Now, take $\underline{\boldsymbol{Y}}(n)$ from (1), and define also

$$\boldsymbol{\mathcal{Y}}_1(n) := \begin{bmatrix} \Re(\underline{\boldsymbol{Y}}(n)) \\ \Im(\underline{\boldsymbol{Y}}(n)) \end{bmatrix}, \quad \boldsymbol{\mathcal{Y}}_2(n) := \begin{bmatrix} \Im(\underline{\boldsymbol{Y}}(n)) \\ -\Re(\underline{\boldsymbol{Y}}(n)) \end{bmatrix}, \tag{2}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ denote the real and imaginary parts of a complex vector, respectively. The reason for introducing the above quantities is the following. Assume that the vector-valued function $\boldsymbol{f}$ in

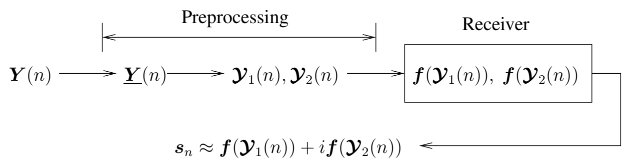
Preprocessing
Receiver
$\boldsymbol{Y}(n)$
$\underline{\boldsymbol{Y}}(n)$
$\boldsymbol{\mathcal{Y}}_1(n), \boldsymbol{\mathcal{Y}}_2(n)$
$\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_1(n)),\ \boldsymbol{f}(\boldsymbol{\mathcal{Y}}_2(n))$
$s_n \approx \boldsymbol{f}(\boldsymbol{\mathcal{Y}}_1(n)) + i\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_2(n))$

Fig. 2. Model of the receiver. The receiver is the vector-valued function $\boldsymbol{f} := [f_1, \ldots, f_L]^t$, where each component f_l is assumed to belong to some Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_l$. Our goal is to find such an $\boldsymbol{f}$ such that the difference $\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_1(n)) + i\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_2(n)) - \boldsymbol{s}_n$, when viewed via some user-defined loss function, becomes sufficiently small.

Fig. 2 is linear (and continuous). Then, each component f_l of $\boldsymbol{f}$ is also linear. By the Riesz representation theorem, there exists a unique vector $\boldsymbol{v}_l$, such that $f_l(\boldsymbol{\mathcal{Y}}) = \boldsymbol{v}_l^t\boldsymbol{\mathcal{Y}}$, $\forall \boldsymbol{\mathcal{Y}}$ (the vectors $\boldsymbol{v}_l$, $\boldsymbol{\mathcal{Y}}$ are of appropriate dimension). Partition now the vector $\boldsymbol{v}_l$ in two vectors $\boldsymbol{v}_{l1}, \boldsymbol{v}_{l2}$ of equal length, as in $\boldsymbol{v}_l = [\boldsymbol{v}_{l1}^t, \boldsymbol{v}_{l2}^t]^t$. Then, it is easy to verify by (2) that

$$f_l(\boldsymbol{\mathcal{Y}}_1(n)) = \boldsymbol{v}_l^t\boldsymbol{\mathcal{Y}}_1(n) = \Re(\boldsymbol{w}_l^*\underline{\boldsymbol{Y}}(n)),$$
$$f_l(\boldsymbol{\mathcal{Y}}_2(n)) = \boldsymbol{v}_l^t\boldsymbol{\mathcal{Y}}_2(n) = \Im(\boldsymbol{w}_l^*\underline{\boldsymbol{Y}}(n)),$$

where $\boldsymbol{w}_l := \boldsymbol{v}_{l1} + i\boldsymbol{v}_{l2}$, and $\boldsymbol{w}_l^*$ is its complex conjugate transpose vector. Hence, Fig. 2 covers the linear receiver case, whenever $\boldsymbol{f}$ is assumed linear. However, the present study considers a much wider choice for the receiver, since it allows each component of $\boldsymbol{f}$ to belong to a strictly larger superset of the Euclidean space, namely a Reproducing Kernel Hilbert Space (RKHS) (see Section 3.1).

Assume, now, the sequence of incoming data in the receiver as

$$\boldsymbol{y}_n := \boldsymbol{\mathcal{Y}}_{n \bmod 2+1}(n), \quad \forall n \in \mathbb{Z}_{\geq 0}.$$

This sequence $(\boldsymbol{y}_n)_{n\in\mathbb{Z}_{\geq 0}}$ together with the a known training sequence of symbols $(s_n)_{n\in\mathbb{Z}_{\geq 0}}$ transmitted from the TOI, will be used for the training of our receiver.

3. PRELIMINARIES

3.1. Reproducing Kernel Hilbert Space (RKHS).

Henceforth, the symbol $\mathcal{H}$ will stand for a (general infinite dimensional) Hilbert space equipped with an inner product denoted by $\langle\cdot,\cdot\rangle$. The induced norm is $\|\cdot\| := \langle\cdot,\cdot\rangle^{1/2}$.

Assume a real Hilbert space $\mathcal{H}$ whose elements are functions defined on $\mathbb{R}^m$, i.e., a point in $\mathcal{H}$ is a function $f : \mathbb{R}^m \to \mathbb{R}$, for some $m \in \mathbb{Z}_{>0}$. The function $\kappa(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{H}$ if (i) $\kappa(\boldsymbol{y},\cdot) \in \mathcal{H}$, $\forall \boldsymbol{y} \in \mathbb{R}^m$, and (ii) the following *reproducing property* holds

$$f(\boldsymbol{y}) = \langle f, \kappa(\boldsymbol{y},\cdot)\rangle, \quad \forall \boldsymbol{y} \in \mathbb{R}^m, \forall f \in \mathcal{H}. \tag{3}$$

In this case, $\mathcal{H}$ is called a *Reproducing Kernel Hilbert Space (RKHS)* [1].

Celebrated examples of reproducing kernels are i) the linear kernel $\kappa(\boldsymbol{y}_1, \boldsymbol{y}_2) := \boldsymbol{y}_1^t\boldsymbol{y}_2$, $\forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^m$ (here the associated RKHS is the space $\mathbb{R}^m$ itself [1]), and ii) the Gaussian kernel $\kappa(\boldsymbol{y}_1, \boldsymbol{y}_2) := \exp(-\frac{(\boldsymbol{y}_1-\boldsymbol{y}_2)^t(\boldsymbol{y}_1-\boldsymbol{y}_2)}{2\sigma^2})$, $\forall \boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^m$, where $\sigma > 0$ (here the RKHS is of infinite dimension [1]).

Now, assume a number of L (see Fig. 2) RKHS $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_L$, and define the Cartesian product space $\boldsymbol{\mathcal{H}} := \mathcal{H}_1 \times \cdots \times \mathcal{H}_L$. An element of $\boldsymbol{\mathcal{H}}$ can be written apparently as $\boldsymbol{f} := [f_1, \ldots, f_L]^t$, where $f_1 \in \mathcal{H}_1, \ldots, f_L \in \mathcal{H}_L$. To make $\boldsymbol{\mathcal{H}}$ a Hilbert space, we need to define a suitable inner product. In the case where the Hilbert spaces $\mathcal{H}_1, \ldots, \mathcal{H}_L$ are identical, we achieve this via a positive definite symmetric matrix $\boldsymbol{P}$: for any $\boldsymbol{f}, \boldsymbol{g} \in \boldsymbol{\mathcal{H}}$, $\langle\langle \boldsymbol{f}, \boldsymbol{g} \rangle\rangle := \sum_{l,l'=1}^{L} P_{ll'} \langle f_l, g_{l'} \rangle = \operatorname{trace}(\boldsymbol{P}[\langle f_l, g_{l'} \rangle])$, where the symbol $[\langle f_l, g_{l'} \rangle]$ stands for a matrix whose (l, l')-th element is $\langle f_l, g_{l'} \rangle$. Otherwise, in the case where the RKHS $\mathcal{H}_1, \ldots, \mathcal{H}_L$ are in generall different, we use a block diagonal positive definite matrix $\boldsymbol{P}$. Clearly, for both cases, the induced norm $|||\cdot||| := \langle\langle \cdot, \cdot \rangle\rangle^{1/2}$.

3.2. Subgradient

Now, we give a concept of fundamental importance for the developement of the proposed algorithm in Section 5.

Given a continuous convex function $\mathcal{L} : \boldsymbol{\mathcal{H}} \to \mathbb{R}$, any element of the *subdifferential* of $\mathcal{L}$, i.e., of

$$\partial\mathcal{L}(\boldsymbol{f}) := \{\boldsymbol{h} \in \boldsymbol{\mathcal{H}} : \langle\langle \boldsymbol{g} - \boldsymbol{f}, \boldsymbol{h} \rangle\rangle + \mathcal{L}(\boldsymbol{f}) \leq \mathcal{L}(\boldsymbol{g}), \forall \boldsymbol{g} \in \boldsymbol{\mathcal{H}}\},$$

is called a subgradient of $\mathcal{L}$ at $\boldsymbol{f}$, and will be denoted by $\mathcal{L}'(\boldsymbol{f})$. In other words, $\mathcal{L}'(\boldsymbol{f}) \in \partial\mathcal{L}(\boldsymbol{f})$. Note that for any continuous convex function $\mathcal{L}$ we can always define a subgradient, since for such a function, we have $\partial\mathcal{L}(\boldsymbol{f}) \neq \emptyset, \forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}$. Note also that $0 \in \partial\mathcal{L}(\boldsymbol{f})$ iff $\boldsymbol{f} \in \arg\min_{\boldsymbol{g} \in \boldsymbol{\mathcal{H}}} \mathcal{L}(\boldsymbol{g})$. The function $\mathcal{L}$ has a unique subgradient at $\boldsymbol{f}$, if $\mathcal{L}$ is differentiable at $\boldsymbol{f}$. Then, this unique subgradient is nothing but the well-known derivative $\mathcal{L}'(\boldsymbol{f})$.

4. COST FUNCTIONS

As we have seen in Fig. 2, the objective of our receiver is to find $\boldsymbol{f} \in \boldsymbol{\mathcal{H}}$ such that the difference $\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_1(n)) + i\boldsymbol{f}(\boldsymbol{\mathcal{Y}}_2(n)) - \boldsymbol{s}_n$ becomes sufficiently small, when evaluated by a user-defined cost function. In what follows, we introduce a framework such that any continuous convex function, even non-differentible, can be used as the cost function in the design. Due to lack of space, only two examples of cost functions will be presented, leaving the discussion of a larger variety of examples for a future work.

Choose any convex, continuous function $\mathcal{L} : \boldsymbol{\mathcal{H}} \to \mathbb{R}$, which quantifies the "discrepancy" between the output of the receiver $\boldsymbol{f}(\boldsymbol{y})$ and the desired responce $\boldsymbol{s}$. Choose also a nonnegative number ϵ in order to obtain a robust version, called ϵ*-insensitive* function, of $\mathcal{L}$:

$$\mathcal{L}_\epsilon(\boldsymbol{f}) := \max\{0, \mathcal{L}(\boldsymbol{f}) - \epsilon\}, \quad \forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}. \tag{4}$$

Notice that $\mathcal{L}_\epsilon$ is a nonnegative function. Such a modification of an objective function is widely used in robust statistics with numerous applications in modern pattern recognition and kernel methods [1].

Example 1 (ϵ-insensitive quadratic loss function with l_2 norm.) *Given a positive definite symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{L \times L}$, define the inner product in the Euclidean space $\mathbb{R}^L$ as $\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle_{\boldsymbol{Q}} := \boldsymbol{x}_1^t \boldsymbol{Q} \boldsymbol{x}_2$, $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^L$. The norm, thus, in $\mathbb{R}^L$ becomes $\|\cdot\|_{\boldsymbol{Q}} := \langle \cdot, \cdot \rangle_{\boldsymbol{Q}}^{1/2}$.*

Define now the function $\mathcal{L}(\boldsymbol{f}) := \|\boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{s}\|_{\boldsymbol{Q}}^2$, $\forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}$. The subdifferential of the ϵ-insensitive function of (4) *becomes as follows, depending on the location of the point $\boldsymbol{f}$.*

1. *$\mathcal{L}(\boldsymbol{f}) < \epsilon$. Then, we have $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{0\}$.*
2. *$\mathcal{L}(\boldsymbol{f}) > \epsilon$. Then, $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{\boldsymbol{P}^{-1}[2\sum_{l'=1}^{L} Q_{l'l}(f_{l'}(\boldsymbol{y}) - s_{l'})\kappa_l(\boldsymbol{y}, \cdot)]\}$, where the symbol $[2\sum_{l'=1}^{L} Q_{l'l}(f_{l'}(\boldsymbol{y}) - s_{l'})\kappa_l(\boldsymbol{y}, \cdot)]$ stands for a vector of length L, whose l-th component is the quantity $2\sum_{l'=1}^{L} Q_{l'l}(f_{l'}(\boldsymbol{y}) - s_{l'})\kappa_l(\boldsymbol{y}, \cdot)$.*
3. *$\mathcal{L}(\boldsymbol{f}) = \epsilon$. Then, $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{\theta\boldsymbol{P}^{-1}[2\sum_{l'=1}^{L} Q_{l'l}(f_{l'}(\boldsymbol{y}) - s_{l'})\kappa_l(\boldsymbol{y}, \cdot)] : \theta \in [0, 1]\}$.*

Example 2 (ϵ-insensitive linear loss function with l_1 norm.) *Let $\mathcal{L}(\boldsymbol{f}) := \|\boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{s}\|_1^2$, $\forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}$, where $\|\boldsymbol{x}\|_1 := \sum_{l=1}^{L} |x_l|$, for any $\boldsymbol{x} := [x_1, \ldots, x_L]^t \in \mathbb{R}^L$.*

The function $\mathcal{L}_\epsilon$ is not everywhere differentiable. Its subdifferential, instead, can be calculated and is given as follows. Given an $\boldsymbol{f}$, define first the index set $J = \{l_1, \ldots, l_\nu\}$, which contains all those indices l for which $f_l(\boldsymbol{y}) - s_l = 0$, i.e., $J := \{l \in \overline{1, L} : f_l(\boldsymbol{y}) - s_l = 0\}$. Then, the subdifferential is obtained depending on where the point $\boldsymbol{f}$ is and whether the index J is the empty set or not. Hence,

1. *$\mathcal{L}(\boldsymbol{f}) < \epsilon$. Then, we have $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{0\}$.*
2. *$\mathcal{L}(\boldsymbol{f}) > \epsilon$, and $J = \emptyset$. Then, $\mathcal{L}_\epsilon$ is differentiable at such an $\boldsymbol{f}$, so that its subdifferential contains only one element, its derivative: $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{\boldsymbol{P}^{-1}[\operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot)]\}$, where the symbol $[\operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot)]$ stands for the vector of length L, whose l-th component is given by $\operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot)$.*
3. *$\mathcal{L}(\boldsymbol{f}) > \epsilon$, and $J \neq \emptyset$. Then the subdifferential becomes $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \operatorname{conv}\{\boldsymbol{P}^{-1}\boldsymbol{V}_1, \ldots, \boldsymbol{P}^{-1}\boldsymbol{V}_{2^\nu}\}$, where the operator* conv *stands for the convex hull of a set, and $\forall \lambda \in \overline{1, 2^\nu}$,*

$$V_{\lambda l} := \begin{cases} \operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot), & \text{if } l \notin J, \\ \pm\kappa_l(\boldsymbol{y}, \cdot), & \text{if } l \in J. \end{cases}$$

   In other words, the vector-valued functions $\boldsymbol{V}$ contain either $+\kappa_l(\boldsymbol{y}, \cdot)$ or $-\kappa_l(\boldsymbol{y}, \cdot)$ in the slots l determined by the index set J, and $\operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot)$ in all the other positions. This is why the total number of $\boldsymbol{V}_\lambda$ becomes 2^ν.
4. *$\mathcal{L}(\boldsymbol{f}) = \epsilon$, and $J = \emptyset$. Then, we obtain $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{\theta\boldsymbol{P}^{-1}[\operatorname{sign}(f_l(\boldsymbol{y}) - s_l)\kappa_l(\boldsymbol{y}, \cdot)] : \theta \in [0, 1]\}$.*
5. *$\mathcal{L}(\boldsymbol{f}) = \epsilon$, and $J \neq \emptyset$. Then, $\partial\mathcal{L}_\epsilon(\boldsymbol{f}) = \{0, \boldsymbol{P}^{-1}\boldsymbol{V}_1, \ldots, \boldsymbol{P}^{-1}\boldsymbol{V}_{2^\nu}\}$.*

5. ALGORITHM

We present now the algorithm, based on the Adaptive Projected Subgradient Method (APSM) [9]. The algorithm can accommodate any nonnegative continuous convex function.

Each $(\boldsymbol{y}_n, \boldsymbol{s}_n)$, taken from the sequence of training data, defines a cost function similarly to the examples given in Section 4, i.e., it defines an $\mathcal{L}_\epsilon(\boldsymbol{f}; \boldsymbol{y}_n, \boldsymbol{s}_n)$, $\forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}$, where in this notation we explicitly state the dependence of the function $\mathcal{L}_\epsilon$ on the parameters $(\boldsymbol{y}_n, \boldsymbol{s}_n)$. However, to simplify notation, we let from now and on, $\mathcal{L}_n := \mathcal{L}_\epsilon(\cdot; \boldsymbol{y}_n, \boldsymbol{s}_n), \forall n \in \mathbb{Z}_{\geq 0}$.

The introduced algorithm is capable of processing multiple cost functions at each index n. Hence, at each n, apart from the natural choice $\mathcal{L}_n$, we allow the freedom of elaborating also cost functions that correspond to indexes previous than n, i.e., $\mathcal{L}_{n-1}, \ldots, \mathcal{L}_{n-q+1}$, for some positive integer q. In other words, *data reuse* or *concurrent processing* is possible. To identify such cost functions processed at each index n, we introduce here the index set $\mathcal{J}_n$ defined as:

$$\mathcal{J}_n := \begin{cases} \overline{0, n}, & \text{if } n < q - 1, \\ \overline{n - q + 1, n}, & \text{if } n \geq q - 1. \end{cases}$$

That is, at every iteration index n, the algorithm processes the cost functions $\mathcal{L}_j$ where $j \in \mathcal{J}_n$.

Now, given the point $\boldsymbol{f}_n \in \boldsymbol{\mathcal{H}}$, let us define a subset of $\mathcal{J}_n$:

$$\mathcal{I}_n := \{i \in \mathcal{J}_n : \mathcal{L}_i(\boldsymbol{f_n}) \neq 0\}.$$

Let also a number of convex weights $\{\omega_i^{(n)}\}_{i\in\mathcal{I}_n} \subset (0,1]$ such that $\sum_{i\in\mathcal{I}_n} \omega_i^{(n)} = 1$. Then, for an arbitrary starting point $\boldsymbol{f}_0 \in \boldsymbol{\mathcal{H}}$, form the sequence of estimates as

$$\boldsymbol{f}_{n+1} = \begin{cases} \boldsymbol{f}_n - \mu_n \sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i(\boldsymbol{f}_n)}{|||\mathcal{L}_i'(\boldsymbol{f}_n)|||^2} \mathcal{L}_i'(\boldsymbol{f}_n), & \text{if } \mathcal{I}_n \neq \emptyset, \\ \boldsymbol{f}_n, & \text{if } \mathcal{I}_n = \emptyset, \end{cases} \tag{5}$$

where the extrapolation parameter μ_n can be chosen by the designer from the interval $[0, 2\mathcal{M}_n]$, with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i^2(\boldsymbol{f_n})}{|||\mathcal{L}_i'(\boldsymbol{f_n})|||^2}}{|||\sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i(\boldsymbol{f_n})}{|||\mathcal{L}_i'(\boldsymbol{f_n})|||^2} \mathcal{L}_i'(\boldsymbol{f}_n)|||^2}, & \text{if } \mathcal{I}_n \neq \emptyset, \\ 1, & \text{if } \mathcal{I}_n = \emptyset, \end{cases} \tag{6}$$

and $\mathcal{M}_n \geq 1$.

Under mild condition, the above algorithm enjoys several desirable properties for online designs, like monotone approximation to the set of optimum receivers, strong convergence, asymptotic optimality, etc, which will be presented elsewhere.

6. SPARSIFICATION

It can be shown (proof omitted due to lack of space) that the algorithm in Section 5 gives a sequence of estimates $(\boldsymbol{f}_n)_{n\in\mathbb{Z}_{\geq 0}}$ which are expressed as a vector-valued series of kernels; given the arbitrary starting point $\boldsymbol{f}_0 \in \boldsymbol{\mathcal{H}}$,

$$\boldsymbol{f}_n = \sum_{j=0}^{n-1} \begin{bmatrix} \gamma_{j1}(n)\kappa_1(\boldsymbol{y}_j, \cdot) \\ \vdots \\ \gamma_{jL}(n)\kappa_L(\boldsymbol{y}_j, \cdot) \end{bmatrix}, \quad \forall n \in \mathbb{Z}_{>0}. \tag{7}$$

To keep the notation compact, we define here

$$\boldsymbol{k}_j(n) := \begin{bmatrix} \gamma_{j1}(n)\kappa_1(\boldsymbol{y}_j, \cdot) \\ \vdots \\ \gamma_{jL}(n)\kappa_L(\boldsymbol{y}_j, \cdot) \end{bmatrix}.$$

As the number of blocks n advances, more and more terms are added in the series expansion of (7). This has an immediate unpleasant effect on the computational complexity and the memory requirements of the system. To deal with such problems, sparsification of the kernel series in (7) is necessary.

Our sparsification strategy is based on the addition of an extra constraint on the optimization problem. Given a δ, define now the closed ball

$$B[0,\delta] := \{\boldsymbol{f} \in \boldsymbol{\mathcal{H}} : |||f||| \leq \delta\}.$$

Such a set is a closed convex set. Associated to a closed convex set C is the metric projection mapping defined as the mapping $P_C : \boldsymbol{\mathcal{H}} \to C : \boldsymbol{f} \mapsto P_C(\boldsymbol{f})$, where $P_C(\boldsymbol{f})$ is the unique point of C such that $|||\boldsymbol{f} - P_C(\boldsymbol{f})||| = \inf\{|||\boldsymbol{f} - \boldsymbol{g}||| : \boldsymbol{g} \in C\}$. In particular, the projection mapping for the closed ball above takes the following simple form:

$$P_{B[0,\delta]}(\boldsymbol{f}) = \begin{cases} \boldsymbol{f}, & \text{if } |||\boldsymbol{f}||| \leq \delta, \\ \frac{\delta}{|||\boldsymbol{f}|||}\boldsymbol{f}, & \text{if } |||\boldsymbol{f}||| > \delta, \end{cases} \quad \forall \boldsymbol{f} \in \boldsymbol{\mathcal{H}}. \tag{8}$$

We force now the sequence of estimates $(\boldsymbol{f}_n)$ to belong to $B[0,\delta]$. As more and more terms gather in the kernel series of (7), it is likely that at some point n, the estimate $\boldsymbol{f}_n$ will have a norm greater than δ, i.e., $|||\boldsymbol{f}_n||| > \delta$. By (8), we see that if we project $\boldsymbol{f}_n$ onto $B[0,\delta]$, we obtain

$$P_{B[0,\delta]}(\boldsymbol{f}_n) = \sum_{j=0}^{n-1} \frac{\delta}{|||\boldsymbol{f}_n|||} \begin{bmatrix} \gamma_{j1}(n)\kappa_1(\boldsymbol{y}_j, \cdot) \\ \vdots \\ \gamma_{jL}(n)\kappa_L(\boldsymbol{y}_j, \cdot) \end{bmatrix}.$$

Since $\frac{\delta}{|||\boldsymbol{f}_n|||} < 1$, we can easily see that $\frac{\delta}{|||\boldsymbol{f}_n|||}|\gamma_{jl}(n)| < |\gamma_{jl}(n)|$, $\forall l \in \overline{1,L}$. Note that the terms in the kernel series with indexes less than $n-q+1$, i.e., $\overline{0,n} \setminus \mathcal{J}_n$, are not affected by the recursive procedure (5), since only the terms in the last q slots of the kernel series change. If we additionaly assume that the above hypothesis stands for a consecutive number of times n_0, then, the coefficient $\frac{\delta^{n_0}}{|||\boldsymbol{f}_{n+n_0-1}|||\cdots|||\boldsymbol{f}_n|||} < 1$ multiplies those terms in the kernel series. Hence, as more and more blocks of data arrive at the receiver, the old terms in the kernel series expansion potentially tend to small absolute values.

Note that all of the above calculations are of linear complexity with respect to the unknown parameters, i.e., the coefficients of the kernel series in (7).

As a step further, we introduce also a buffer which keeps only a number of L_b kernel terms. In other words, we introduce here a modification of the algorithm (5), (6):

$$\tilde{\boldsymbol{f}}_{n+1} = \begin{cases} P_{B[0,\delta]}(\tilde{\boldsymbol{f}}_n - \mu_n \sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i(\tilde{\boldsymbol{f}}_n)}{|||\mathcal{L}_i'(\tilde{\boldsymbol{f}}_n)|||^2} \mathcal{L}_i'(\tilde{\boldsymbol{f}}_n)), \\ \qquad\qquad\qquad\qquad \text{if } \mathcal{I}_n \neq \emptyset, \\ P_{B[0,\delta]}(\tilde{\boldsymbol{f}}_n), \qquad\qquad \text{if } \mathcal{I}_n = \emptyset, \end{cases}$$

where the extrapolation parameter μ_n can be chosen by the designer from the interval $[0, 2\mathcal{M}_n]$, with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i^2(\tilde{\boldsymbol{f}}_n)}{|||\mathcal{L}_i'(\tilde{\boldsymbol{f}}_n)|||^2}}{|||\sum_{i\in\mathcal{I}_n} \omega_i^{(n)} \frac{\mathcal{L}_i(\tilde{\boldsymbol{f}}_n)}{|||\mathcal{L}_i'(\tilde{\boldsymbol{f}}_n)|||^2} \mathcal{L}_i'(\tilde{\boldsymbol{f}}_n)|||^2}, & \text{if } \mathcal{I}_n \neq \emptyset, \\ 1, & \text{if } \mathcal{I}_n = \emptyset, \end{cases}$$

and

$$\tilde{\boldsymbol{f}}_n = \sum_{j=n-L_b}^{n-1} \begin{bmatrix} \tilde{\gamma}_{j1}(n)\kappa_1(\boldsymbol{y}_j, \cdot) \\ \vdots \\ \tilde{\gamma}_{jL}(n)\kappa_L(\boldsymbol{y}_j, \cdot) \end{bmatrix}, \quad \forall n \in \mathbb{Z}_{>0}. \tag{9}$$

To keep the number of kernel terms fixed and equal to L_b, a strategy has to be introduced whenever the buffer becomes full and new data arrive at the receiver. Apparently, for every new term, another one has to be removed. How do we choose such a term? We remove the term with the least contribution, with respect to norm, in the kernel series (9). That is, we search in the buffer for the term indexed by $j_*(n) := \min\{\hat{j}(n) \in \arg\min\{|||\boldsymbol{k}_j(n)||| : j \in \overline{n-L_b+1,n} \setminus \mathcal{J}_n\}\}$. In other words, we look first among those terms in the buffer not concurrently processed at iteration n, i.e., $j \in \overline{n-L_b+1,n} \setminus \mathcal{J}_n$. We search for the terms with the least contribution, with respect to norm, in the kernel series (7). Then among these, we select the index $j_*(n)$ which has stayed the longest in the buffer.

7. NUMERICAL EXAMPLES

We consider the system of Fig. 1, with a number of $P = 5$ Txs and an Rx with $M = 2$ antenna elements. For linear designs, we employ

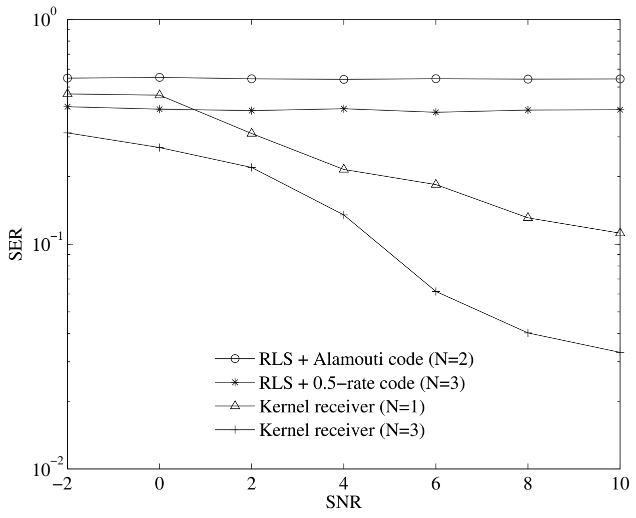

Fig. 3. Symbol Error Rates (SER) vs. SNR. Two linear RLS receivers, together with the OSTBC scheme are employed. The proposed online kernel receiver was elaborated with a number of $N = 1$ and $N = 3$ array elements per Tx.

the Orthogonal Space-Time Block Codes (OSTBC) for increased diversity gain, together with the well-known Recursive Least Squares (RLS) receiver in order to identify the unknown channel matrices $\boldsymbol{H}_p$, $p \in \overline{1, P}$, seen in (1). Each component of the channel matrix $\boldsymbol{H}_p$, $p \in \overline{1, P}$, is modeled as an i.i.d. zero-mean complex Gaussian random variable with variance equal to 1 [8]. Moreover, independency is assumed among all the components of each of $\boldsymbol{H}_p$, and among all channel matrices. Each component of the additive noise $(\boldsymbol{V}(n))$, in (1), is modeled as an i.i.d. Gaussian random process. Here also, independency is assumed among all the components of $(\boldsymbol{V}(n))$. We set $L := 2$ as the number of Transmitters Of Interest (TOI), whose symbols are to be identified. For simplicity, all the Txs are assumed to have equal power. The SNR in Fig. 3 is measured as the ratio of a single Tx's power to the noise's one. Depending on the elaborated OSTBC, a different number of array elements N, per Tx, are used. Here, we employ the Alamouti scheme [5], [6, (32)] where $N = 2$, $K = T = 2$, and a $1/2$-rate code [6, (37)] where $N = 3$ and $K = 4$, $T = 8$.

For the proposed nonlinear kernel receiver, we employed the Gaussian kernel function, with a variance $\sigma^2 := 1$, for working in an infinite dimensional Reproducing Kernel Hilbert Space (RKHS). We elaborated the non-differentiable cost function of Example 2, with the parameter $\epsilon := 0.1$. The matrix $\boldsymbol{P}$ used for the definition of the Hilbert space $\boldsymbol{\mathcal{H}}$ was set to $\boldsymbol{P} := (1/L)\boldsymbol{I}_L$. A number of 8000 complex training data ($= 16000$ real data according to the preprocessing of Fig. 2) were used for the learning process of the kernel receiver. The number of concurrent data, used at every index n, was set equal to $q = 3$. We used a buffer length of $L_b := 10000$, with a ball radius $\delta := 300$. Separate testing data are used to validate the Symbol Error Rate (SER) of each scheme, and a number of 100 experiments are performed to average such a procedure. The relatively large number of used training data, for having the method converge, is mainly due to the limited number of Rx antennas and the severe MAI of the adopted scenario. Other scenarios, as well as techniques to increase the speed of convergence, and to reduce the computational load of the method will be discussed elsewhere.

As we have seen in Section 2, any linear receiver needs a sufficient number of degrees of freedom to ideally suppress MAI and self-interference. This is quantified by the inequality $PK < MT$. However, in the present case, where the MAI is severe, we notice that for both the Alamouti scheme $PK = 5 \cdot 2 \geq 2 \cdot 2 = MT$, and for the $1/2$-rate code $PK = 5 \cdot 4 \geq 2 \cdot 8 = MT$. In other words, the RLS does not have enough degrees of freedom to combat MAI, as we readily see by Fig. 3. We notice that the $1/2$-rate code gives better results than the Alamouti one, due to the increased value of MT/K and N.

On the other hand, since the proposed nonlinear receiver operates implicitly in infinite dimensional Reproducing Kernel Hilbert Spaces (RKHS), Fig. 3 demonstrates that the obstacle of severe MAI is eventually surmounted. We also notice that the proposed receiver behaves well, even when a number of $N = 1$ array elements are used per Tx. For the case where $PK < MT$, the proposed kernel receiver produced similar results to the OSTBC scheme. Due to lack of space, more simulation results, which will thoroughly validate the various system parameters, will be presented elsewhere.

8. CONCLUSIONS

A novel online receiver for multiaccess MIMO channels was introduced by using kernel functions. The receiver is built by a number of Reproducing Kernel Hilbert Spaces (RKHS), and viewed as an online multiregression optimization task, where any continuous convex cost function, even non-differentiable can be employed. The computational complexity of the design scales linearly to the number of unknown parameters. Sparsification issues were also addressed to deal with computational and memory limitations of the online system. Simulation results showed that the proposed nonlinear receiver overcomes the physical limitations, which are inherent in linear receivers with Orthogonal Space-Time Block Codes (OSTBC), in cases where the MultiAccess Interference (MAI) is severe.

9. REFERENCES

[1] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2001.

[2] M. Sánchez-Fernández, M. De-Prado-Cumpildo, J. Arenas-García, and F. Pérez-Cruz, "SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2298–2307, Aug. 2004.

[3] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[4] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 2781–2796, 2008.

[5] S. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 45, pp. 1451–1458, Oct. 1998.

[6] V. Tarokh, H. Jafarkhani, and A. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Trans. Information Theory*, vol. 45, no. 5, pp. 1456–1467, 1999.

[7] S. Shahbazpanahi, M. Beheshti, A. Gershman, M. Gharavi-Alkhansari, and K. Wong, "Minimum variance linear receivers for multiaccess MIMO wireless systems with space-time block coding," *IEEE Trans. Signal Processing*, vol. 52, no. 12, pp. 3306–3313, 2004.

[8] R. Cavalcante and I. Yamada, "Multiaccess interference supression in orthogonal space-time block coded MIMO systems by adaptive projected subgradient method," *IEEE Trans. Signal Processing*, vol. 56, no. 3, pp. 1028–1042, 2008.

[9] K. Slavakis, I. Yamada, and N. Ogura, "The Adaptive Projected Subgradient Method over the fixed point set of strongly attracting nonexpansive mappings," *Numerical Functional Analysis and Optimization*, vol. 27, no. 7&8, pp. 905–930, 2006.