RLS adaptive filter with inequality constraints

Vítor H. Nascimento[†], *Senior Member*, *IEEE*, and Yuriy Zakharov [††], *Senior Member*, *IEEE*

Abstract

In practical implementations of estimation algorithms, designers usually have information about the range in which the unknown variables must lie, either due to physical constraints (such as power always being nonnegative) or due to hardware constraints (such as in implementations using fixed-point arithmetic). In this paper we propose a fast (that is, whose complexity grows linearly with the filter length) version of the dichotomous coordinate descent recursive least-squares adaptive filter which can incorporate constraints on the variables. The constraints can be in the form of lower and upper bounds on each entry of the filter, or norm bounds. We compare the proposed algorithm with the recently proposed normalized non-negative least mean squares (LMS) and projected-gradient normalized LMS filters, which also include inequality constraints in the variables.

Index Terms

adaptive filter, box constraint, inequality constraint, non-negativity, RLS-DCD.

I. Introduction

In estimation problems of practical importance, one often has *a priori* information about the range in which the solution must lie. This knowledge may take the form of equality constraints, such as the requirement that the solution lies in a given subspace (as in the case of the generalized sidelobe canceller [1]), or may take the form of inequality constraints. One important example of the latter is *box constraints*, that is, imagine that one must compute estimates $\boldsymbol{h}(i)$ for an unknown, possibly time-varying parameter vector $\boldsymbol{h}_{\mathrm{o}}(i) \in \mathbb{R}^N$ taking into account constraints of the type

$$a_n \leq h_n(i) \leq b_n, \tag{1}$$

with known bounds $-\infty < a_n < b_n < \infty$, where $h_n(i)$ represents the n-th entry of vector $\boldsymbol{h}(i)$. Note that either the lower or upper bound might not be present (e.g., we might require only a non-negativity constraint $h_n(i) > 0$ [2]).

The constraints may arise from physical limitations on the variables [3]—such as the maximum range of an actuator [4]—or the non-negativity of image pixels and sound intensities [5], or may be due to design choices and

[†]Dept. of Electronic Systems Engineering, Univ. of São Paulo, Brazil, vitor@lps.usp.br. [††]Department of Electronics, Univ. of York, UK, yz1@ohm.york.ac.uk. The work of Y. V. Zakharov was partly supported by the York-FAPESP grant 2014/50765-6. The work of V. H. Nascimento was partly supported by CNPq and FAPESP grants 306268/2014-0, 2014/50765-6 and 2014/04256-2.

limitations in the hardware used to implement the estimator itself (such as in fixed-point implementations). The use of constraints has a number of advantages — it may [6]:

- allow the use of simpler models [3];
- avoid the appearance of "unfeasible" or "un-physical" solutions that could arise due to noisy measurements;
- reduce the variance of estimates;
- increase the convergence speed of algorithms.

Although modified Kalman filters with inequality constraints have received considerable attention [7], in the adaptive filtering literature the focus has been on equality constraints [8]. Adaptive filtering algorithms with non-negativity constraints have recently been proposed in [2], [6], [9], and can also be obtained through appropriate model choices using projection onto convex sets as described in [10] (see also [2]). The non-negative least mean squares (NNLMS) algorithm and its variants proposed in [2], [6], [9] perform well, but may be sensitive to outliers due to a term proportional to the power $1+\gamma$ of the weight entries (i.e., $h_n^{1+\gamma}(i)$) with $0 < \gamma \leq 1$ in their update laws. This may lead to instability, similarly to what occurs for the constant-modulus algorithm (CMA) [11]. This supra-linear term makes the stability of the algorithm dependent on the initial condition $\boldsymbol{h}(0)$ [6], as also occurs in the CMA algorithm [11]. The projected-gradient NLMS algorithm also described in [2] allows for more general (convex) inequality constraints and does not suffer from this problem, but can converge slowly when the input signal is highly correlated.

In this paper we describe a novel approach for adaptive filtering with either box constraints, as in (1), or norm constraints, as in (2) below. Our algorithm is a modification of the RLS-DCD algorithm proposed in [12], which is a numerically-stable, low-cost alternative to the recursive least-squares algorithm (RLS) based on the dichotomous coordinate-descent (DCD) method for solving least-squares and other convex optimization problems. Being based on a Hessian (RLS) approach, instead of a gradient (LMS) approach as in [2], [6], [9], the algorithms proposed here converge faster than gradient-based algorithms. The DCD and RLS-DCD algorithms are optimized for implementation using finite-precision arithmetic (particularly in custom or semi-custom implementations, such as in FPGAs) [13], [14]. The use of DCD makes the algorithms numerically stable with low cost (linear on the filter length N), and also allows for easy implementation in hardware.

We initially describe our method in terms of box constraints (1), and later extend it to bounded norm constraints such as

$$\|\boldsymbol{D}\boldsymbol{h}(i)\| \leq \tau, \tag{2}$$

where $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_N)$ for any constants $d_n > 0$ and $\tau > 0$. $\|\cdot\|$ is a vector norm, such as ℓ_1, ℓ_2 or ℓ_∞. For example, sparsity-inducing algorithms [15]–[18] can be obtained using the ℓ_1 norm in (2); using the ℓ_2 (Euclidean) norm and $d_n = 1$ is equivalent to requiring that the solution $\boldsymbol{h}(i)$ stay inside a sphere of radius τ; and using the ℓ_∞ (maximum) norm, (2) reduces to (1) with $-a_n = b_n = \tau/d_n$.

A box-constrained version of the DCD algorithm is described in [19], applied to multiuser detection. In Section II we first briefly describe the RLS-DCD algorithm and then propose modifications to the box-constrained DCD

algorithm from [19] that allow for general bound constraints (such as in (2)) and make it more suitable for adaptive filtering. We then use this result to propose an RLS-DCD algorithm incorporating box (1) or bound (2) constraints. In Section III, we compare our algorithms with those proposed in [2]. Finally, in Section IV we conclude the paper.

II. Constrained RLS-DCD algorithm

Given two sequences $\left\{z(i) \in \mathbb{R}\right\}_{i=1}^{\infty}$ and $\left\{\boldsymbol{x}(i) \in \mathbb{R}^N\right\}_{i=1}^{\infty}$, we seek a vector $\boldsymbol{h}(i)$ that solves the constrained least-squares (LS) problem

$$\boldsymbol{h}(i+1) \triangleq \arg\min_{\boldsymbol{h}\in\mathcal{A}_i} \sum_{k=1}^{i} \lambda^{i-k} \left[z(k) - \boldsymbol{x}^T(k)\boldsymbol{h}\right]^2 + \lambda^i \delta \|\boldsymbol{h}\|_2^2, \tag{3}$$

where $0 < \lambda < 1$ is a forgetting factor, $\delta > 0$ is a regularization term, and $\mathcal{A}_i$ is a convex constraint, such as (1) or (2). The index i in $\mathcal{A}_i$ indicates that, in general, the constraint may change over time. However, we only consider a time-invariant $\mathcal{A}$ in this paper. For each time instant i, the cost function in (3) is a convex, differentiable cost function with convex constraints, a kind of problem for which coordinate descent optimization converges and performs well [20], [21].

An iterative solution to (3) can be found by modifying the derivation of the RLS-DCD algorithm [12] as follows. Define the autocorrelation matrix $\boldsymbol{R}(i)$ and cross-correlation vector $\boldsymbol{p}(i)$ by the recursions for $i \geq 1$

$$\boldsymbol{R}(i) = \lambda \boldsymbol{R}(i-1) + \boldsymbol{x}(i)\boldsymbol{x}^T(i), \quad \boldsymbol{R}(0) = \delta \boldsymbol{I}, \tag{4}$$

$$\boldsymbol{p}(i) = \lambda \boldsymbol{p}(i-1) + z(i)\boldsymbol{x}(i), \quad \boldsymbol{p}(0) = \boldsymbol{0}. \tag{5}$$

Since $z^2(k)$ does not depend on $\boldsymbol{h}$, (3) can be rewritten as

$$\boldsymbol{h}(i+1) = \arg\min_{\boldsymbol{h}\in\mathcal{A}} \left\{ \frac{1}{2}\boldsymbol{h}^T \boldsymbol{R}(i)\boldsymbol{h} - \boldsymbol{h}^T \boldsymbol{p}(i) \right\}. \tag{6}$$

Assuming that an approximation $\hat{\boldsymbol{h}}(i)$ to $\boldsymbol{h}(i)$ is available, we now search for an updated approximation $\hat{\boldsymbol{h}}(i+1)$. Let $\boldsymbol{h} = \hat{\boldsymbol{h}}(i) + \boldsymbol{\Delta h}$. Disregarding the terms that do not depend on $\boldsymbol{\Delta h}$, the argument of (6) can be written as

$$\begin{aligned} &\frac{1}{2}\boldsymbol{\Delta h}^T \boldsymbol{R}(i)\boldsymbol{\Delta h} + \boldsymbol{\Delta h}^T \boldsymbol{R}(i)\hat{\boldsymbol{h}}(i) - \boldsymbol{\Delta h}^T \boldsymbol{p}(i) \\ &= \frac{1}{2}\boldsymbol{\Delta h}^T \boldsymbol{R}(i)\boldsymbol{\Delta h} - \lambda \boldsymbol{\Delta h}^T \boldsymbol{r}(i) + e(i)\boldsymbol{\Delta h}^T \boldsymbol{x}(i), \end{aligned}$$

were the residue $\boldsymbol{r}(i)$ and error $e(i)$ at time i are given by

$$\boldsymbol{r}(i) \triangleq \boldsymbol{p}(i-1) - \boldsymbol{R}(i-1)\hat{\boldsymbol{h}}(i), \qquad e(i) \triangleq z(i) - \boldsymbol{x}^T(i)\hat{\boldsymbol{h}}(i).$$

Define $\boldsymbol{q}(i) = \lambda \boldsymbol{r}(i) + e(i)\boldsymbol{x}(i)$. We conclude that (6) is equivalent to letting $\boldsymbol{h}(i+1) = \hat{\boldsymbol{h}}(i) + \boldsymbol{\Delta h}(i)$, where

$$\boldsymbol{\Delta h}(i) = \arg\min_{\substack{\boldsymbol{\Delta h},\ \text{s.t.} \\ \hat{\boldsymbol{h}}(i)+\boldsymbol{\Delta h}\in\mathcal{A}}} \left\{ \frac{1}{2}\boldsymbol{\Delta h}^T \boldsymbol{R}(i)\boldsymbol{\Delta h} - \boldsymbol{\Delta h}^T \boldsymbol{q}(i) \right\}. \tag{7}$$

An approximate solution to (7) can be obtained efficiently using the DCD algorithm [22]. DCD is a coordinate

descent optimization method, with parameters adjusted so that multiplications and divisions are avoided (replaced by additions and bit shifts), making the algorithm easy to implement in semi-custom hardware [4], [13], [23]. In general $\|\Delta \boldsymbol{h}(i)\|_\infty$ will be small compared to $\|\hat{\boldsymbol{h}}(i)\|_\infty$, which implies that only a few DCD iterations applied to the problem (7) are necessary to obtain a $\widehat{\boldsymbol{\Delta h}}(i)$ such that $\hat{\boldsymbol{h}}(i+1) \triangleq \hat{\boldsymbol{h}}(i) + \widehat{\boldsymbol{\Delta h}}(i)$ is a good approximation to $\boldsymbol{h}(i+1)$. Note that for implementations in general-purpose computers or in DSPs, other versions of coordinate-descent optimization algorithms [21] would be equally effective.

The DCD algorithm in [12] does not take constraints into account. Fortunately, the inclusion of convex constraints in coordinate descent algorithms is simple [20]: at each DCD iteration, we need to check if the new candidate solution lies within the constraints. If it does not, the update is not performed. A box-constrained version of DCD was proposed in [19], but based on a cyclic version of DCD, which is not the best for adaptive filtering [12].

In Table I we introduce a *leading* box-constrained DCD algorithm (BDCD). At each iteration a better approximation to (7) is computed, updating a single entry of $\boldsymbol{h}$. The coordinate chosen for update is the one corresponding to the largest absolute entry in the current residue $\boldsymbol{r}$ (steps 1 and 12 in Table I). Note that the algorithm in Table I is designed to operate directly on $\boldsymbol{h}$, not on $\boldsymbol{\Delta h}$, thus simplifying the check of the constraints in step 7.

The algorithm's inputs are a matrix $\boldsymbol{R}$, vector $\boldsymbol{q}$, initial guess $\boldsymbol{h}$, and parameters H, M_b and N_u. The initial step size $\alpha = H > 0$ should be a power of two to reduce complexity (so all multiplications and divisions become bit shifts). The algorithm will work with any value of H, but will need fewer iterations when H corresponds to the most significant bit required to store $\boldsymbol{h}$. Choosing H as a power of two, M_b will be the number of bits used to represent the solution. N_u is the maximum number of vector operations (operations that involve N additions, see steps 1, 9 and 12 in Table I); N_u is used to limit the computational cost of the algorithm. Steps 14-16 in Table I are used to avoid the algorithm getting stalled when a constraint becomes active, that is, if the test in step 7 is false. In this case the algorithm reverts to a cyclic DCD scheme for one iteration.

The main loop of the adaptive filtering algorithm, which we call the RLS-BDCD algorithm, is described in Table II. Step 5 is a call to the BDCD algorithm of Table I.

We need now to consider the update of $\boldsymbol{R}(i-1)$ in step 3 of Table II. The update of $\boldsymbol{R}(i-1)$ following directly (4) is an $\mathcal{O}(N^2)$ task. However, if $\boldsymbol{x}(i)$ is a tap-delay line, that is, if $\boldsymbol{x}(i) = \begin{bmatrix} x(i) & x(i-1) & \dots & x(i-N+1) \end{bmatrix}^T$, then $\boldsymbol{R}(i)$ can be computed in $\mathcal{O}(N)$ operations, as follows [12]

$$\boldsymbol{R}(i) = \left[\begin{array}{c|c} [\boldsymbol{\rho}(i)]_{1,1} & [\boldsymbol{\rho}^T(i)]_{2:N} \\ \hline [\boldsymbol{\rho}(i)]_{2:N} & [\boldsymbol{R}(i-1)]_{1:N-1,1:N-1} \end{array}\right], \tag{8}$$

where $[\boldsymbol{a}]_{m:n}$ represents the entries m to n from vector $\boldsymbol{a}$, and similarly for matrices, and vector $\boldsymbol{\rho}(i)$ is the first column of $\boldsymbol{R}(i)$, with update $\boldsymbol{\rho}(i) = \lambda \boldsymbol{\rho}(i-1) + x(i)\boldsymbol{x}(i)$.

The complexity of the RLS-BDCD algorithm is upper bounded by $5N$ multiplications plus $(N_u+3)N$ additions. If $\lambda = 1 - 2^{-b}$ with an integer $b > 0$, then the complexity is upper bounded by $3N$ multiplications and $(N_u+5)N$ additions (see [12] for more details).

TABLE I
LEADING BOX-CONSTRAINED DCD ALGORITHM (BDCD).

Step	Inputs: $\boldsymbol{h}$, $\boldsymbol{q}$, $\boldsymbol{R}$, N_u, M_b, constraint set. Initialization: $\boldsymbol{r} \leftarrow \boldsymbol{q}$; $m \leftarrow 0$; $\mu \leftarrow 1$; $\alpha \leftarrow H$, $j_c \leftarrow 1$
1	$j = \arg\max_{n=1,\ldots,N} \lvert r_n \rvert$
2	if $\lvert r_j \rvert < (\alpha/2) R_{j,j}$, then
3	$\alpha \leftarrow \alpha/2$ and $m \leftarrow m+1$
4	if $m > M_b$, then the algorithm stops
5	else,
6	$w = h_j + \text{sign}(r_j)\alpha$
7	if $w \in [a_j, b_j]$
8	$h_j = w$
9	$\boldsymbol{r} \leftarrow \boldsymbol{r} - \text{sign}(r_j)\alpha \boldsymbol{R}^{(q)}$
10	$\mu \leftarrow \mu + 2$
11	if $\mu > N_u$ the algorithm stops
12	$j = \arg\max_{n=1,\ldots,N} \lvert r_n \rvert$
13	else,
14	$j = j_c$
15	if $j_c < N$, then $j_c \leftarrow j_c + 1$
16	else, $j_c \leftarrow 1$
17	Go to step 2

TABLE II
RLS-BDCD ALGORITHM.

Step	Given $\hat{\boldsymbol{h}}(1)$, $\boldsymbol{R}(0) = \delta \boldsymbol{I}$, $\delta > 0$ Let $\boldsymbol{r}(1) = -\boldsymbol{R}(0)\hat{\boldsymbol{h}}(1)$
	Repeat for $i \geq 1$:
1	$y(i) = \hat{\boldsymbol{h}}^T(i)\boldsymbol{x}(i)$
2	$e(i) = z(i) - y(i)$
3	Update $\boldsymbol{R}(i-1)$ to $\boldsymbol{R}(i)$
4	$\boldsymbol{q}(i) = \lambda \boldsymbol{r}(i) + e(i)\boldsymbol{x}(i)$
5	Apply the BDCD algorithm to the problem in (7) with $\boldsymbol{h} \leftarrow \hat{\boldsymbol{h}}(i)$, $\boldsymbol{R} \leftarrow \boldsymbol{R}(i)$, $\boldsymbol{q} \leftarrow \boldsymbol{q}(i)$, and appropriate constraint set to obtain $\hat{\boldsymbol{h}}(i+1) \leftarrow \boldsymbol{h}$ and $\boldsymbol{r}(i+1) \leftarrow \boldsymbol{r}$

Norm constraints

With norm constraints as in (2), it is convenient to introduce a new variable c to store the current value of the constraint measure. The algorithm in Table I is modified as follows to obtain the NDCD algorithm of Table III:

- Initialization: Let $c \leftarrow \|\boldsymbol{D}\hat{\boldsymbol{h}}(1)\|_1$ (if using ℓ_1 norm) or $c \leftarrow \|\boldsymbol{D}\hat{\boldsymbol{h}}(1)\|_2^2$ (if using ℓ_2 norm).
- Step 5: Apply the norm-constrained DCD algorithm (NDCD) from Table III with $\hat{\boldsymbol{h}}(i)$, $\boldsymbol{R}(i)$ and $\boldsymbol{q}(i)$ to obtain $\hat{\boldsymbol{h}}(i+1)$, $\boldsymbol{r}(i+1)$, and the updated constraint measure c.

The NDCD algorithm, summarized in Table III, is similar to the BDCD algorithm, but with an extra step to update c (step 7). This can be implemented cheaply in the case of ℓ_1 or ℓ_2 norms, since only one entry of $\boldsymbol{h}$ is modified.

ℓ_1 norm: $c \leftarrow c + d_j(|h_j + \text{sign}(r_j)\alpha| - |h_j|)$.

ℓ_2 norm: $c \leftarrow c + d_j^2(2\text{sign}(r_j)\alpha h_j + \alpha^2)$.

TABLE III
LEADING NORM-CONSTRAINED DCD ALGORITHM (NDCD).

Step	Inputs: $\boldsymbol{h}$, $\boldsymbol{q}$, $\boldsymbol{R}$, c, N_u, M_b, constraint set. Initialization: $\boldsymbol{r} \leftarrow \boldsymbol{q}$; $m \leftarrow 0$; $\mu \leftarrow 1$; $\alpha \leftarrow H$, $j_c \leftarrow 1$
1	$j = \arg\max_{n=1,\ldots,N} \lvert r_n \rvert$
2	if $\lvert r_j \rvert < (\alpha/2) R_{j,j}$, then
3	$\quad \alpha \leftarrow \alpha/2$ and $m \leftarrow m+1$
4	$\quad$ if $m > M_b$, then the algorithm stops
5	else,
6	$\quad w = h_j + \text{sign}(r_j)\alpha$
7	$\quad s \leftarrow$ updated constraint
8	$\quad$ if $s \leq \tau$
9	$\quad\quad h_j = w$, $c = s$
10	$\quad\quad \boldsymbol{r} \leftarrow \boldsymbol{r} - \text{sign}(r_j)\alpha \boldsymbol{R}^{(q)}$
11	$\quad\quad \mu \leftarrow \mu + 2$
12	$\quad\quad$ if $\mu > N_u$ the algorithm stops
13	$\quad\quad j = \arg\max_{n=1,\ldots,N} \lvert r_n(i) \rvert$
14	$\quad$ else,
15	$\quad\quad j = j_c$
16	$\quad\quad$ if $j_c < N$, then $j_c \leftarrow j_c + 1$
17	$\quad\quad$ else, $j_c \leftarrow 1$
18	Go to step 2

Replacing the call to the BDCD algorithm in step 5 in Table II with a call to the NDCD algorithm, we arrive at the RLS-NDCD algorithm. Multiplications are avoided if the d_n are chosen as powers of two. In this case, the computational complexity of the RLS-NDCD algorithm is similar to that of the RLS-BDCD algorithm.

III. NUMERICAL RESULTS

We now compare the proposed algorithms with unconstrained RLS [8], [24], unconstrained RLS-DCD [12], and with the NNLMS and projected gradient NLMS algorithms of [2] in identification scenarios, i.e., when $z(i) = \boldsymbol{h}_{\text{o}}^T \boldsymbol{x}(i) + v(i)$, where $\boldsymbol{h}_{\text{o}}$ is an unknown weight vector, and $v(i)$ is additive white noise. We plot ensemble-average estimates of the mean-square deviation (MSD), i.e., the expected value $\text{E}\{\|\hat{\boldsymbol{h}}(i) - \boldsymbol{h}_{\text{o}}\|_2^2\}$, against the time index i. The weight vector $\boldsymbol{h}_{\text{o}}$ is modified at the middle of the simulation run in order to compare the tracking ability of the algorithms.

Fig. 1 compares the RLS-BDCD (with constraints $0 \leq h_n \leq 1$), unconstrained RLS-DCD, and classical RLS algorithms. Vector $\boldsymbol{h}_{\text{o}}$ contains $N = 100$ taps uniformly distributed in the interval $[0, 1]$. The length of $\boldsymbol{h}(i)$ is also $N = 100$. The input signal $x(i)$ is white Gaussian with unit variance. All filters use $\lambda = 0.99$; N_u is set to 8; and the noise variance is $\sigma^2 = 4$. When the noise variance is small, the RLS-DCD and RLS-BDCD algorithms behave similarly (not shown here); however with a large noise variance the use of the box constraints helps reduce the variance of the estimate, without compromising the convergence speed, as seen in Fig. 1.

Fig. 2 compares the RLS-BDCD algorithm ($N_u = 2$, $\lambda = 0.992$) with the normalized NNLMS (N-NNLMS) ($\eta = 0.9444$) and projected-gradient NLMS (PG-NLMS) ($\mu = 0.2160$) algorithms from [2]. In this example the filter length is $N = 15$, the input $x(i)$ is an autoregressive process with unit variance, generated as $x(i) = 0.95x(i-1) + w(i)$, where $w(i)$ is a zero-mean iid Gaussian process. $\boldsymbol{h}_{\text{o}}$ was generated from a uniform distribution

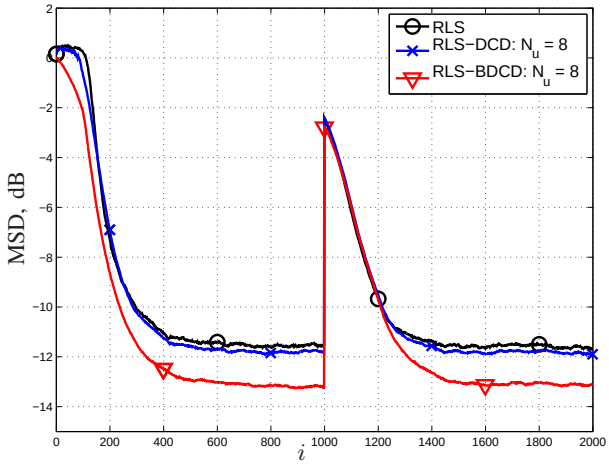

Fig. 1. MSD performance of RLS, RLS-DCD and RLS-BDCD, $\sigma^2 = 4$.

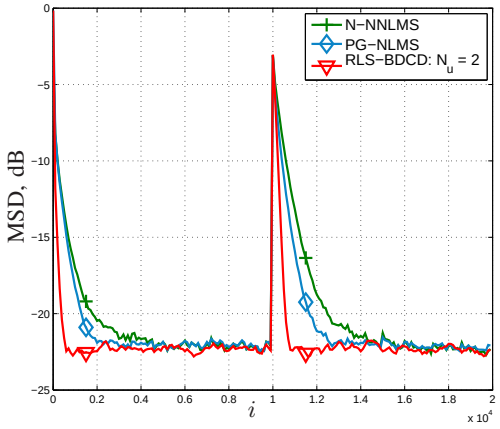

Fig. 2. Comparison of MSD performance for RLS-BDCD, normalized NNLMS and projected gradient NLMS.

in the interval $[0, 1]$. The constraints for the RLS-BDCD and PG-NLMS algorithms were set to $0 \leq h_n$ (no upper bound) to match the behavior of N-NNLMS, which enforces positive entries to the estimated vector. The adaptive filters start from an initial condition at the origin, except for the normalized NNLMS, which is initialized with all coefficients set to 0.1 (NNLMS should not be initialized at the origin). The noise variance is $\sigma^2 = 0.01$. The filter parameters were chosen to guarantee the same steady-state MSD. It can be seen that the RLS-BDCD algorithm outperforms both the normalized NNLMS and the PG-NLMS algorithms.

In Fig. 3, $\boldsymbol{h}_{\mathrm{o}}$ contains one random negative element. In this case, the optimum constrained solution $\boldsymbol{h}^*$ to (3) is not $\boldsymbol{h}_{\mathrm{o}}$. We computed $\boldsymbol{h}^*$ theoretically to plot the ensemble-average learning curves $\mathrm{E}\{\|\hat{\boldsymbol{h}}(i) - \boldsymbol{h}^*\|_2^2\}$. The conditions are the same as those of Fig. 2, except that $\eta = 0.4$ for N-NNLMS and $\mu = 0.15$ for PG-NLMS. The RLS-BDCD algorithm again converges faster.

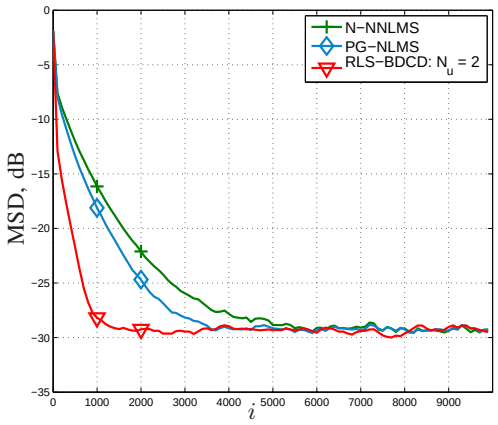

Fig. 3. Comparison of MSD performance when the signal is generated by a model in which $\boldsymbol{h}_{\rm o}$ contains one negative entry. The MSD is computed with respect to the solution to the constrained optimization problem.

An example of the performance of the RLS-NDCD algorithm is presented in Fig. 4. In this case the signal is generated using a vector $\boldsymbol{h}_{\rm o}$ of length $N = 100$ in which only ten random entries are nonzero. Each nonzero entry is obtained randomly, using a Gaussian distribution. The vector $\boldsymbol{h}_{\rm o}$ is then normalized to unit ℓ_2 norm (the resulting ℓ_1 norm is 2.49). The same vector $\boldsymbol{h}_{\rm o}$ is used for all simulation trials. We consider the unconstrained RLS-DCD and ℓ_1-constrained RLS-NDCD algorithms, with $\lambda = 0.992$, and constraint bounds $\tau = 2.6$ and 3.5. Fig. 4 shows that the norm constraint results in improved performance, either in terms of convergence rate or steady-state MSD.

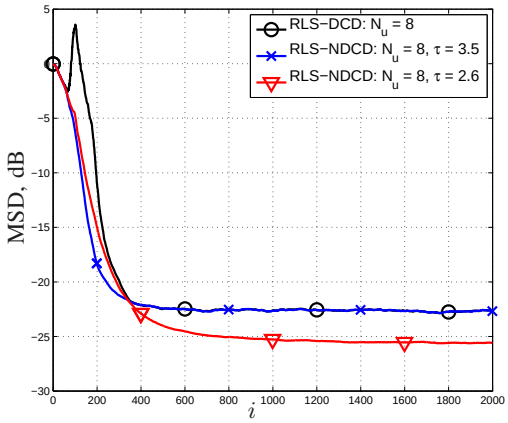

Fig. 4. Comparison of MSD performance of RLS-DCD and RLS-NDCD using ℓ_1-norm with $\tau = 2.6$ and $\tau = 3.5$, for the estimation of a sparse vector.

IV. Conclusion

We described a new family of adaptive filters that is able to easily incorporate different kinds of inequality constraints, such as box constraints or norm bounds. Several different norm bounds can be used, such as ℓ_1 and ℓ_2 norms. The algorithms are extensions of the RLS-DCD algorithm, and thus have fast convergence and low cost (their computational complexity grows only linearly with filter length), while remaining numerically stable.

The new algorithms were compared to other methods described in the literature through simulations, showing advantages in terms of convergence rate and steady-state performance.

References

[1] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*. New York, NY: John Wiley & Sons, 2002.

[2] J. Chen, C. Richard, J.-C. Bermudez, and P. Honeine, "Variants of non-negative least-mean-square algorithm and convergence analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3990–4005, 2014.

[3] C. V. Rao, J. B. Rawlings, and D. Q. Mayne, "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations," *IEEE Trans. Autom. Control*, vol. 48, no. 2, pp. 246–258, 2003.

[4] R. Schlanbusch, R. Kristiansen, and P. J. Nicklasson, "Spacecraft magnetic control using dichotomous coordinate descent algorithm with box constraints," *Modeling, Identification and Control*, vol. 31, no. 4, pp. 123–131, 2010.

[5] F. Benvenuto, R. Zanella, L. Zanni, and M. Bertero, "Nonnegative least-squares image deblurring: improved gradient projection approaches," *Inverse Problems*, vol. 26, no. 2, pp. 1–18, 2010.

[6] J. Chen, C. Richard, J. Bermudez, and P. Honeine, "Nonnegative least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5225–5235, 2011.

[7] D. Simon, "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms," *IET Control Theory & Applications*, vol. 4, no. 8, pp. 1303–1318, 2010.

[8] A. H. Sayed, *Adaptive Filters*. NJ: Wiley, 2008.

[9] J. Chen, J. Bermudez, and C. Richard, "Steady-state performance of non-negative least-mean-square algorithm and its variants," *IEEE Signal Process. Lett.*, vol. 21, no. 8, pp. 928–932, 2014.

[10] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 97–123, 2011.

[11] O. Dabeer and E. Masry, "Convergence analysis of the constant modulus algorithm," *IEEE Trans. Inf. Theory*, vol. 49, no. 6, pp. 1447–1464, Jun. 2003.

[12] Y. Zakharov, G. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3150–3161, 2008.

[13] J. Liu, Y. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 11, pp. 2425–2438, 2009.

[14] M. Algreer, M. Armstrong, and D. Giaouris, "Active online system identification of switch mode DC–DC power converter based on efficient recursive DCD-IIR adaptive filter," *IEEE Trans. Power Electron.*, vol. 27, no. 11, pp. 4425–4435, 2012.

[15] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The sparse RLS algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 8, pp. 4013–4025, 2010.

[16] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: Where RLS meets the ℓ_1 -norm," *IEEE Trans. Signal Process.*, vol. 58, no. 7, pp. 3436–3447, 2010.

[17] Y. Zakharov and V. H. Nascimento, "DCD-RLS adaptive filters with penalties for sparse identification," *IEEE Trans. Signal Process.*, vol. 61, no. 12, pp. 3198–3213, 2013.

[18] S. Oymak, C. Thrampoulidis, and B. Hassibi, "The Squared-Error of Generalized LASSO: A Precise Analysis," *arXiv:1311.0830*, Nov. 2013, arXiv: 1311.0830.

[19] Y. Zakharov and T. Tozer, "Box-constrained multiuser detection based on multiplication-free coordinate descent optimisation," in *IEEE 5th Workshop on Signal Processing Advances in Wireless Communications*, Lisbon, Portugal, 2004, pp. 483–486.

[20] Z. Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, Jan. 1992.

[21] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[22] Y. Zakharov and T. Tozer, "Multiplication-free iterative algorithm for LS problem," *Electronics Letters*, vol. 40, no. 9, pp. 567–569, 2004.

[23] Z. Quan, J. Liu, and Y. Zakharov, "FPGA implementation of DCD based CDMA multiuser detector," in *15th IEEE International Conference on Digital Signal Processing*, Cardiff, Wales, UK, July 2007, pp. 319–322.

[24] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 3rd ed. Springer, 2008.