Adaptive Randomized Coordinate Descent for Sparse Systems: Lasso and Greedy Algorithms

Alexandru Onose, Bogdan Dumitrescu

***Abstract*—Coordinate descent (CD) is a simple optimization technique suited to low complexity requirements and also for solving large problems. In randomized version, CD was recently shown as very effective for solving least-squares (LS) and other optimization problems. We propose here an adaptive version of randomized coordinate descent (RCD) for finding sparse LS solutions, from which we derive two algorithms, one based on the lasso criterion, the other using a greedy technique. Both algorithms employ a novel way of adapting the probabilities for choosing the coordinates, based on a matching pursuit criterion. Another new feature is that, in the lasso algorithm, the penalty term values are built without knowing the noise level or using other prior information. The proposed algorithms use efficient computations and have a tunable trade-off between complexity and performance through the number of CD steps per time instant. Besides a general theoretical convergence analysis, we present simulations that show good practical behavior, comparable to or better than that of state of the art methods.**

***Index Terms*—adaptive algorithms, sparse filters, coordinate descent, randomization**

I. Introduction

Randomized algorithms have a long history in computing, with a sound theory for analyzing their performances. In optimization, the first randomized algorithms appeared in the 1960s. Their main attractions are simplicity and versatility. They work well with "black-box" objective functions. The random character intrinsically allows avoiding local minima. However, their low requirements are often counterbalanced by a better performance of deterministic algorithms for objective functions for which additional information (gradient, Hessian) is available.

The recent resurgence of randomized algorithms is motivated in part by the need to solve large problems, for which even the computation of the gradient is computationally demanding. Even though the problem may be simple in principle, like the linear least-squares (LS) solution of the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for a given matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$, the sheer size of the data may pose practical challenges to most optimization methods. For LS, the randomization of some simple optimization techniques showed, somewhat unexpectedly, better convergence behavior than that of their deterministic counterpart. The randomized Kaczmarz algorithm [1], giving the LS solution for consistent systems, forces the current solution approximation to satisfy a randomly chosen equation of the system; it needs only a row of the matrix $\boldsymbol{A}$ in each iteration. Randomized coordinate descent (RCD) [2] updates a single element of the solution approximation, accessing only a column of the matrix. With proper values of the probabilities for choosing a row or a column, these methods are very effective for large systems, especially when the matrix is sparse. RCD was generalized in [3] to more general functions, with proven convergence speed where no results were available for deterministic CD.

In this paper we propose *adaptive* RCD algorithms for finding a *sparse* solution to the LS problem. To our knowledge, this is the first time when RCD is used in an adaptive manner to compute online solutions. A first algorithm belongs to the lasso class, optimizing an ℓ_1-regularized LS criterion; RCD can be seen here as the randomized version of the algorithm from [4], that used cyclic sweeps over the coordinates. The second algorithm is greedy and falls into the direct line of [5], where the coordinates selection was made via a simplified version of matching pursuit.

In both cases, a distinctive new feature is the adaptation of the probabilities for choosing the coordinates. The only other work where probabilities were changed during the optimization process is [6], but in a different context (batch optimization) and using different rules: the probabilities were increased when the descent gave a significant decrease of the objective function. Here, we propose a rule suited to finding sparse solutions and with the direct purpose of obtaining higher probabilities for the nonzero coordinates and hence accelerating convergence.

Finding online sparse solutions to the LS problem is common in signal processing applications, e.g. FIR channel identification or linear prediction. Previous work is entirely deterministic and appealed to a broad range of methods. The first algorithms belonged to the LMS class, representative examples being [7], [8], [9]. Closer to our work are the algorithms from the RLS family. An approximate expectation maximization was employed in [10], later extended to nonlinear systems [11]; coordinate descent was used in [4], together with weight adaptation for an ℓ_1-regularized LS criterion; projections on weighted ℓ_1 balls were the main tool in [12]; low complexity dichotomous coordinate descent powered the algorithms from [13]; variational Bayes inference was the principle behind the work from [14]. We had some contributions in greedy algorithms [15], [5], from which we borrow some techniques, most notably those related to the use of Information Theoretic

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

The authors are with Department of Signal Processing, Tampere University of Technology, Finland. E-mail for correspondence: bogdan.dumitrescu@tut.fi. B. Dumitrescu is also with Department of Automatic Control and Computers, University Politehnica of Bucharest, Romania.

This work was supported in part by the Romanian National Authority for Scientic Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-0400.

Criteria for estimating the sparsity level of the solution. A first version of this work is [16], where we presented the greedy algorithm; the lasso algorithm, the convergence analysis and all simulations are new.

The content of the paper is as follows. In section II we briefly recall the problem. Section III contains our contributions to adaptive randomized coordinate descent, most notably the proposed probability adaptation in III-C, the lasso algorithm in III-E, the greedy algorithm in III-F and the convergence analysis in III-H (with proofs in the Appendix). Section IV is dedicated to simulations.

II. Problem formulation

We assume a linear underlying process

$$\boldsymbol{\alpha}^{(t)^T}\boldsymbol{x}^* = \beta^{(t)} + v^{(t)}, \tag{1}$$

where, at discrete time instant t, the input is the vector $\boldsymbol{\alpha}^{(t)}$ and the output is the scalar $\beta^{(t)}$. The output is corrupted with Gaussian noise $v^{(t)}$ with zero mean and variance σ^2. The unknown vector that weighs the data is $\boldsymbol{x}^*$, of size N. We assume that this vector has $L^{(t)} \ll N$ nonzero elements, i.e. it is sparse.

Using an exponential window with forgetting factor λ, we build

$$\boldsymbol{A}^{(t)} = \begin{bmatrix} \sqrt{\lambda}\boldsymbol{A}^{(t-1)} \\ \boldsymbol{\alpha}^{(t)^T} \end{bmatrix}, \quad \boldsymbol{b}^{(t)} = \begin{bmatrix} \sqrt{\lambda}\boldsymbol{b}^{(t-1)} \\ \beta^{(t)} \end{bmatrix}. \tag{2}$$

Assuming that $v^{(t)}$ is white noise, the minimization of the least-squares criterion

$$J^{(t)}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)}\boldsymbol{x}\|^2 \tag{3}$$

produces a consistent estimate $\boldsymbol{x}^{(*,t)}$ of the true parameter vector $\boldsymbol{x}^*$.

For inducing sparsity, the lasso criterion

$$G_\gamma^{(t)}(\boldsymbol{x}) = J^{(t)}(\boldsymbol{x}) + \gamma^{(t)}\|\boldsymbol{x}\|_1 \tag{4}$$

is one of the most convenient tools. If the penalty $\gamma^{(t)} > 0$ is constant, the lasso estimator produces a biased LS solution. To alleviate bias, putting different weights on the coefficients is helpful, thus giving the weighted lasso criterion

$$K_\gamma^{(t)}(\boldsymbol{x}) = J^{(t)}(\boldsymbol{x}) + \sum_{i=0}^{N-1}\gamma_i^{(t)}|x_i|. \tag{5}$$

Ideally, the weights $\gamma_i^{(t)}$ should be 1 for the zero coefficients and 0 for the nonzero coefficients. Since the support of the solution is not known, this choice is impossible. However, the weights can be changed while minimizing the criterion (5); the resulting estimation process is named reweighted lasso [17].

We will give online algorithms for finding sparse minimizers to the problems (3) and (5). In the latter case, we will also give a method for computing the penalty values $\gamma_i^{(t)}$.

Algorithm 1: ARCD: Adaptive Randomized Coordinate Descent—Principle Form

1 Parameters: R, number of CD steps per time moment
2 Initialize $x_i^{(R,0)} = 0$, $\pi_i^{(0)} = 1/N$, $i = 0 : N-1$.
3 **for** $t = 1, 2, \ldots$ **do**
4 Build $\boldsymbol{A}^{(t)}$ and $\boldsymbol{b}^{(t)}$ as in (2)
5 Set $\boldsymbol{x}^{(0,t)} = \boldsymbol{x}^{(R,t-1)}$
6 **for** $k = 1 : R$ **do**
7 Randomly choose coordinate ℓ, using probabilities $\pi_i^{(t-1)}$
8 Compute $\boldsymbol{x}^{(k,t)}$ by optimal descent on coordinate ℓ, starting from $\boldsymbol{x}^{(k-1,t)}$
9 Compute new probabilities $\pi_i^{(t)}$

III. ARCD algorithm

A. Principle form

A principle form of adaptive randomized coordinate descent is presented in Algorithm 1. This is a natural extension of RCD to the adaptive context. At each time moment we update the objective function (step 4), start the descent from the last solution approximation at previous time (step 5) and then perform R optimal CD steps on randomly chosen coordinates (steps 7, 8). The number R of RCD steps is a parameter of the method; a larger R means certainly higher complexity but probably quicker descent towards the optimum.

Unlike typical RCD, we propose to adapt the coordinates probabilities as the algorithm progresses. This feature can be especially useful for finding sparse solutions, where performing more CD steps on the coordinates of nonzero coefficients than on those of the zero coefficients is likely to increase convergence speed.

We will detail the algorithm for two criteria: least-squares (3) and weighted lasso (5). While the specific form of the CD steps is simple and their efficient implementation in an adaptive way is available from [5] (for LS, but easy to modify for lasso), there are some difficulties to be solved. A meaningful adaptation of probabilities is necessary for speeding up convergence for both criteria. For lasso, a significant problem is the choice of the weights in (5). While lasso naturally gives the support of the solution, for LS we adopt a greedy strategy, used also in [5], that determines the support using Information Theoretic Criteria.

B. Optimal CD step

We present here the outcome of an optimal CD step on coordinate i, for the LS and lasso criteria. At time t and iteration k, the residual associated with the LS problem is

$$\boldsymbol{r}^{(k,t)} = \boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)}\boldsymbol{x}^{(k,t)}. \tag{6}$$

We will also use the notation

$$\tilde{\boldsymbol{r}}^{(k,t)} = \boldsymbol{r}^{(k,t)} + x_i^{(k,t)}\boldsymbol{a}_i^{(t)}, \tag{7}$$

which is the residual for a solution approximation whose i-th element is set to zero.

For the LS criterion (3), the optimal step changes only the i-th element of the current solution

$$x_i^{(k+1,t)} = x_i^{(k,t)} + \frac{\boldsymbol{a}_i^{(t)^T} \boldsymbol{r}^{(k,t)}}{\|\boldsymbol{a}_i^{(t)}\|^2} = \frac{\boldsymbol{a}_i^{(t)^T} \tilde{\boldsymbol{r}}^{(k,t)}}{\|\boldsymbol{a}_i^{(t)}\|^2}. \quad (8)$$

The other elements of $\boldsymbol{x}^{(k+1,t)}$ are equal to those of $\boldsymbol{x}^{(k,t)}$. Due to this step, the decrease of the LS criterion is

$$J^{(t)}(\boldsymbol{x}^{(k,t)}) - J^{(t)}(\boldsymbol{x}^{(k+1,t)}) = \frac{1}{2} \frac{|\boldsymbol{a}_i^{(t)^T} \boldsymbol{r}^{(k,t)}|^2}{\|\boldsymbol{a}_i^{(t)}\|^2}. \quad (9)$$

For the lasso criterion (5), the optimal step gives

$$x_i^{(k+1,t)} = \mathrm{sgn}(\boldsymbol{a}_i^{(t)^T} \tilde{\boldsymbol{r}}^{(k,t)}) \frac{\left\{ |\boldsymbol{a}_i^{(t)^T} \tilde{\boldsymbol{r}}^{(k,t)}| - \gamma_i^{(t)} \right\}_+}{\|\boldsymbol{a}_i^{(t)}\|^2}, \quad (10)$$

where the soft thresholding operator $\{\cdot\}_+$ leaves nonnegative values unchanged and sets negative values to 0. The decrease of the LS criterion (3) is smaller than (9), since the step is not LS-optimal, unless $\gamma_i^{(t)} = 0$. (We are not especially interested to quantify the decrease of the lasso criterion (5).)

C. Adapting the probabilities

Although we will prove that the ARCD algorithm converges for almost any choice of the probabilities $\pi_i^{(t)}$, it is clear that a meaningful choice should improve convergence speed. This is especially relevant for sparse systems, where the coordinates of nonzero coefficients should receive higher probabilities.

In [2], [3], the probabilities are constant and depend only on the matrix of the LS system. The same rule could be easily adapted to a time-varying system, but it would not take into account the sparsity of the solution. These considerations clearly suggest that probabilities should be adapted as the algorithm evolves. To see how to adapt them, we appeal to basic tools of sparse approximations.

The matching pursuit (MP) selection criterion is fundamental to many greedy algorithms because it gives a measure of how much the residual can decrease by the choice of a coordinate. Let us denote $\mathcal{K}$ the set of coordinate indices randomly chosen in the R iterates at the current time t; the R elements of this set may include repetitions. In step 9 of ARCD, we propose to change only the probabilities of the coordinates in $\mathcal{K}$, making them proportional to the quantities

$$p_i = \frac{|\boldsymbol{a}_i^{(t)^T} \tilde{\boldsymbol{r}}^{(k,t)}|^2}{\|\boldsymbol{a}_i^{(t)}\|^2}. \quad (11)$$

Note that these quantities are computed anyway in (8) or (10), so they are available almost for free, but only for the coordinates in $\mathcal{K}$. We see from (9) that, barring the $1/2$ factor, the value (11) is the decrease of the LS criterion if the i-th coefficient is changed from zero to its optimal value (while the others stay unchanged). Hence, the value (11) is a measure of the importance of the i-th coefficient in the solution. It is usually large for nonzero coefficients and small for zero coefficients.

Since some values (11) may become very small compared to the others, the respective coordinates would be chosen very seldom. In the case the support changes, it is possible that a new nonzero coefficient may not be updated for a long time, due to the low probability inherited from its last update, when it was zero. To prevent this situation, we impose a minimum value $\pi_{\min}$ for the probabilities $\pi_i^{(t)}$. Moreover, to preserve their sum (equal to 1 in the initialization from step 2 of ARCD), we propose to update the probabilities with a linear transformation of the values (11):

$$\pi_i^{(t)} = \pi_{\min} + \frac{p_i}{\sum_{\ell \in \mathcal{K}} p_\ell} \left(\sum_{\ell \in \mathcal{K}} \pi_\ell^{(t-1)} - R\pi_{\min} \right), \quad i \in \mathcal{K}. \quad (12)$$

We stress that only the probabilities with indices $i \in \mathcal{K}$ are updated, in order to avoid extra computations. For $i \notin \mathcal{K}$, we keep $\pi_i^{(t)} = \pi_i^{(t-1)}$. We assume the non-restrictive condition $R > 1$, otherwise the probabilities (12) cannot change from their initial values.

For smoother variation, we can introduce a forgetting factor $0 \le \theta < 1$ and modify the values computed in (12) by

$$\pi_i^{(t)} \leftarrow (1-\theta)\pi_i^{(t)} + \theta \pi_i^{(t-1)}, \quad i \in \mathcal{K}. \quad (13)$$

Note that $\theta = 0$ means that (12) is used as such, without considering past values, and that (13) preserves the sum.

The mechanism described here for updating probabilities is used for both the LS and lasso criteria.

D. Efficient online computation

Since the data $\boldsymbol{A}^{(t)}$ and $\boldsymbol{b}^{(t)}$ grow indefinitely, relations like (8) cannot be computed as such. Using the same trick as in [15], we store instead the products

$$\boldsymbol{\Phi}^{(t)} = \boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)}, \quad \boldsymbol{\psi}^{(t)} = \boldsymbol{A}^{(t)^T} \boldsymbol{b}^{(t)}. \quad (14)$$

Then, the update (2) takes the form

$$\begin{aligned} \boldsymbol{\Phi}^{(t)} &\leftarrow \lambda \boldsymbol{\Phi}^{(t-1)} + \boldsymbol{\alpha}^{(t)} \boldsymbol{\alpha}^{(t)^T} \\ \boldsymbol{\psi}^{(t)} &\leftarrow \lambda \boldsymbol{\psi}^{(t-1)} + \beta^{(t)} \boldsymbol{\alpha}^{(t)} \end{aligned} \quad (15)$$

We assume that, at time t, we have an estimate of the support of the solution, denoted $\mathcal{C}^{(t)}$. That is, we assume $x_i^{(k,t)} = 0$, if $i \notin \mathcal{C}^{(t)}$. Using (14), the scalar product that appears in (8) and other formulas can be written as

$$\rho_i^{(k,t)} \triangleq \boldsymbol{a}_i^{(t)^T} \tilde{\boldsymbol{r}}^{(k,t)} = \psi_i^{(k,t)} - \sum_{\ell \in \mathcal{C}^{(t)} \setminus \{i\}} x_\ell^{(k,t)} \phi_{ij}^{(k,t)}. \quad (16)$$

Once this quantity is computed, the LS CD step (8) becomes

$$x_i^{(k+1,t)} = \frac{\rho_i^{(k,t)}}{\phi_{ii}^{(k,t)}}, \quad (17)$$

the lasso CD step (10) is

$$x_i^{(k+1,t)} = \mathrm{sgn}(\rho_i^{(k,t)}) \frac{\left\{ |\rho_i^{(k,t)}| - \gamma_i^{(t)} \right\}_+}{\phi_{ii}^{(k,t)}}, \quad (18)$$

and the probabilities (11) become

$$p_i = \frac{|\rho_i^{(k,t)}|^2}{\phi_{ii}^{(k,t)}}. \quad (19)$$

So, the only costly operation in a CD update is the computation of (16), whose complexity is proportional to the size of the support estimate $\mathcal{C}^{(t)}$.

E. ARCD for the lasso criterion

To optimize the lasso criterion (5) we use directly the principle form of Algorithm 1 and the implementation issues described in the previous subsection. The result is named ARCD-L (L for lasso) and is presented as Algorithm 2. To keep the algorithm simple, we take $\mathcal{C}^{(t)} = 0 : N-1$ in (16), i.e. we do not explicitly manage a set of nonzero coefficients.

This algorithm can be seen as a randomized version of the algorithm from [4] based on cyclic sweeps of the coordinates. There is another important difference, namely the adaptation of the weights $\gamma_i^{(t)}$ in (5). To eliminate the bias introduced by the lasso criterion and find the (constant) LS solution $\boldsymbol{x}^*$, the weights need to become insignificant as the data gather. If $\lambda = 1$, this means $\gamma_i^{(t)}/t \to 0$ (see Theorem 2 in Section III-H).

In [4], a fixed law was adopted for computing $\gamma_i^{(t)}$, which included the exact value of the noise variance and the estimates of the coefficient values provided by the standard RLS. We propose here an adaptive law, based only on computed estimates of the noise variance. Thus, we relate the estimated noise variance to a threshold given by a given probability that the noise is due to a coefficient outside the support. The procedure follows two steps similarly to [4]: i) compute the value $\gamma^{(t)}$ such that, when used directly in (18), it penalizes all coefficients outside the support (coefficients on the support may be also penalized); ii) compute the weight $w_i^{(t)}$ to reweight the penalty such that $\gamma_i^{(t)} = w_i^{(t)}\gamma^{(t)}$ only penalizes the coefficients outside the support in (18).

To begin, let us note that the residual (6) corresponding to the correct support of size $L^{(t)}$ and the correct coefficients, denoted $\boldsymbol{r}$ in what follows, only depends on the noise. In this case, its elements r_i are independent, zero mean, random variables with variance $\lambda^{t-i}\sigma^2$, where σ^2 is the output noise variance. For any column $\boldsymbol{a}_i^{(t)}$, the elements are known constants and we have

$$\begin{aligned} \mathrm{E}\{{\boldsymbol{a}_i^{(t)}}^T \boldsymbol{r}\} &= 0 \\ {\sigma_i^{(t)}}^2 \triangleq \mathrm{var}\{{\boldsymbol{a}_i^{(t)}}^T \boldsymbol{r}\} &= \textstyle\sum_{j=1}^{t} \lambda^{t-j} {a_{j,i}^{(t)}}^2 \sigma^2. \end{aligned} \tag{20}$$

Thus, ${\boldsymbol{a}_i^{(t)}}^T \boldsymbol{r}$ is normally distributed with zero mean and variance ${\sigma_i^{(t)}}^2$. To estimate this variance, which is important in what follows, we estimate online the noise variance σ^2 as

$$\hat{\sigma}^{(t)^2} = \lambda \hat{\sigma}^{(t-1)^2} + (1-\lambda)(\beta^{(t)} - {\boldsymbol{\alpha}^{(t)}}^T \boldsymbol{x}^{(R,t-1)})^2. \tag{21}$$

Since the estimate may be unreliable before enough data are gathered, we put a cap $\tilde{\sigma}^2$ on it, that can be comfortably larger than the true noise variance. The variances from (20) are then estimated online via

$$\hat{\sigma}_i^{(t)^2} = \min\left(\lambda \hat{\sigma}_i^{(t-1)^2} + {\alpha_i^{(t)}}^2 \hat{\sigma}^{(t)^2},\ \tilde{\sigma}^2\right) \quad i = 1 : N. \tag{22}$$

i) Choice of $\gamma^{(t)}$. The first goal is to find the value $\gamma^{(t)}$ such that all coefficients outside the solution support are penalized with a certain probability. Let us suppose that there exists a nonzero coefficient $x_j^{(k,t)} \neq 0$ outside the support. It results from (18) that, at the next iteration, $x_j^{(k+1,t)}$ becomes zero if

$$|\rho_j^{(k,t)}| \leq \gamma^{(t)}. \tag{23}$$

Algorithm 2: ARCD-L: ARCD for the lasso criterion

```
Parameters: R, number of CD steps per time moment;
  probabilities ϖ_γ, ϖ_τ, ϖ_v; exponent c
Initialize x_i^(R,0) = 0, π_i^(0) = 1/N, γ_i^(0) = 0,
  i = 0 : N − 1.
Generate set K of R indices using probabilities π_i^(0)
for t = 1, 2, ... do
    Update the data products as in (15)
    Set x^(0,t) = x^(R,t−1), γ_i^(t) = γ_i^(t−1), i ∈ K
    for i ∈ K (k = 0 : R − 1) do
        Update x_i^(k+1,t) as in (18)
        Compute probability related quantities (19)
    Compute new probabilities π_i^(t) with (12) and (13)
    Generate set K of R indices using probabilities π_i^(t)
    Estimate variances (21) and (22)
    Compute penalty value γ^(t) with (28), using (27)
    for i ∈ K do
        Compute new penalty terms γ_i^(t) = γ^(t) w_i^(t)(π_i^(t))
        using (29) and (34)–(36)
```

Assuming an exact value for the residual, we have

$$|\rho_j^{(k,t)}| \approx |{\boldsymbol{a}_j^{(t)}}^T \boldsymbol{r} + {\boldsymbol{a}_j^{(t)}}^T \boldsymbol{a}_j^{(t)} x_j^{(k,t)}| \leq |{\boldsymbol{a}_j^{(t)}}^T \boldsymbol{r}| + \|\boldsymbol{a}_j^{(t)}\|^2 |x_j^{(k,t)}|. \tag{24}$$

In order to achieve the goal of maximally penalizing the spurious coefficients, we choose $\gamma^{(t)}$ such that

$$|{\boldsymbol{a}_j^{(t)}}^T \boldsymbol{r}| + \|\boldsymbol{a}_j^{(t)}\|^2 |x_j^{(k,t)}| \leq \gamma^{(t)} \text{ with probability } \varpi_\gamma, \tag{25}$$

for given ϖ_γ.

Since $|{\boldsymbol{a}_j^{(t)}}^T \boldsymbol{r}|$ only depends on the noise with values distributed according to the half-normal distribution, we can choose $\mathrm{v}_i^{(t)}$ such that $|{\boldsymbol{a}_i^{(t)}}^T \boldsymbol{r}| < \mathrm{v}_i^{(t)}$ with probability ϖ_γ. The quantile function of the half normal distribution, with variance given by the estimation (22) instead of the true value (20), is

$$Q_{h\mathcal{N}}(\varpi) = \sqrt{2\hat{\sigma}_i^{(t)^2}}\,\mathrm{erf}^{-1}(\varpi), \tag{26}$$

with $\mathrm{erf}(\varpi) = \frac{2}{\pi}\int_0^{\varpi} e^{-t^2}\mathrm{d}t$. Note that the inverse of the error function $\mathrm{erf}^{-1}(\varpi)$ only depends on the preset probability ϖ and can be computed offline. We thus compute

$$\mathrm{v}_i = Q_{h\mathcal{N}}(\varpi_\gamma). \tag{27}$$

We insert this value into (25), which we also slightly modify in order to ensure the convergence conditions later proved in Theorem 2, i.e. $\frac{\gamma^{(t)}}{t} \to 0$, obtaining

$$\gamma^{(t)} = \max_i \|\boldsymbol{a}_i^{(t)}\|^{2c} |x_i^{(R,t)}| + \mathrm{v}_i^{(t)}, \tag{28}$$

where $0 \leq c < 1$. Note that we can afford to relax to inequality in (24) and to take the maximum in (28), thus overestimating $\gamma^{(t)}$ and hence penalizing as well coefficients on the support, due to the fine tuning provided by the weights as shown next.

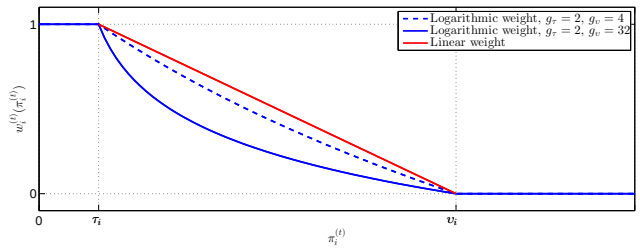

Fig. 1. The weight shaping function (29) for different parameter values.

ii) Choosing the weights. Various weights have been proposed [18]; we use the logarithmic function

$$w_i^{(t)}(\pi_i^{(t)}) = \begin{cases} 1 & \text{if } \pi_i^{(t)} \le \tau_i \\ \frac{\log_2(g_\upsilon)-\log_2(g_\tau+\vartheta)}{\log_2(g_\upsilon)-\log_2(g_\tau)} & \text{if } \tau_i < \pi_i^{(t)} < \upsilon_i \\ 0 & \text{if } \upsilon_i \le \pi_i^{(t)} \end{cases} \tag{29}$$

where $\vartheta = (g_\upsilon - g_\tau)\frac{\pi_i^{(t)}-\tau_i}{\upsilon_i-\tau_i}$ and g_υ, g_τ define the range over which the logarithm is considered; the function is drawn in Fig. 1; depending on the parameters, it may vary between a linear and a step function in the interval $[\tau_i, \upsilon_i]$. Note that the weights depend on the probability values from (13). The goal is to find the two margins τ_i and υ_i such that we penalize (mainly) the spurious coefficients.

Assuming again that the exact residual is available, for a position j outside the solution support (for which $x_j = 0$), the quantity (11) has the expression

$$p_j = \frac{(\boldsymbol{a}_j^{(t)^T}\boldsymbol{r})^2}{\|\boldsymbol{a}_j^{(t)}\|^2}. \tag{30}$$

Due to (20), the values $p_j \frac{\|\boldsymbol{a}_j^{(t)}\|^2}{\bar{\sigma}_j^{(t)^2}}$ are distributed according to the chi-squared distribution with 1 degree of freedom, which has the quantile function

$$Q_{\chi^2}(\varpi) = \int_0^t \frac{t^{\frac{\varpi}{2}-1}e^{-\frac{t}{2}}}{2^{\frac{\varpi}{2}}\Gamma(\frac{\varpi}{2})}\mathrm{d}t. \tag{31}$$

Thus, given ϖ, there exists a value $\bar{p}_j(\varpi)$ such that

$$p_j < \bar{p}_j(\varpi) \quad \text{with probability} \quad \varpi, \tag{32}$$

whose immediate estimation is

$$\bar{p}_j(\varpi) = \frac{\hat{\sigma}_j^{(t)^2}}{\|\boldsymbol{a}_j^{(t)}\|^2} Q_{\chi^2}(\varpi). \tag{33}$$

We note that the raw quantities $\bar{p}_j(\varpi)$ depend only on the noise variance estimation (22) and that the values $Q_{\chi^2}(\varpi)$ can be computed offline. We use $\bar{p}_j(\varpi)$ to determine the threshold for which the residual only depends on the noise and consequently, to determine when the position j, with $p_j < \bar{p}_j(\varpi)$, is not desirable on the support.

Analogously to (12), we transform proportionally $\bar{p}_j(\varpi)$ into values that can be used in the weight computation (29) through

$$\bar{\pi}_j(\varpi) = \pi_{\min} + (1 - N\pi_{\min})\frac{\bar{p}_j(\varpi)}{\mathrm{E}\{\sum_{i=0}^{N-1} p_i\}}, \tag{34}$$

Algorithm 3: ARCD-G: greedy ARCD for the LS criterion

1 Parameters: R, number of CD steps per time moment; Δ, extra length for PLS criterion
2 Initialize $x_i^{(R,0)} = 0$, $\pi_i^{(0)} = 1/N$, $i = 0 : N-1$, $\mathcal{A}^{(1)} = \mathcal{C}^{(1)} = \emptyset$, $\hat{L}^{(0)} = 0$.
3 **for** $t = 1, 2, \ldots$ **do**
4 Update the data products as in (15)
5 Set $\boldsymbol{x}^{(0,t)} = \boldsymbol{x}^{(R,t-1)}$
6 Generate set $\mathcal{K}$ of R indices using probabilities $\pi_i^{(0)}$
7 **for** $i \in \mathcal{K}$ *(*$k = 0 : R-1$*)* **do**
8 Update $x_i^{(k+1,t)}$ as in (17); if $i \in \mathcal{A}^{(t)}$, use $\mathcal{A}^{(t)}$ instead of $\mathcal{C}^{(t)}$ in (16)
9 Compute probability related quantities (19)
10 Compute new probabilities $\pi_i^{(t)}$ with (12) and (13)
11 Order indices in decreasing order of $\pi_i^{(k,t)}$
12 Compute the PLS criterion for the first $\hat{L}^{(t)} + \Delta$ indices
13 Update $\hat{L}^{(t)}$ like in (37), with $\tilde{L}$ given by the minimum PLS
14 Put $\mathcal{A}^{(t+1)}$ as the set of first $\hat{L}^{(t)}$ indices and $\mathcal{C}^{(t+1)}$ as the set of first $\hat{L}^{(t)} + \Delta$ indices
15 Set $x_i^{(k+1,t)} = 0$, for all $i \in \mathcal{K} \setminus \mathcal{C}^{(t)}$

normalizing with the average value of the quantities from (11), which can be estimated via

$$E\{p_i\} = E\left\{\frac{(\boldsymbol{a}_i^{(t)^T}\boldsymbol{r} + \|\boldsymbol{a}_i^{(t)}\|^2 x_i)^2}{\|\boldsymbol{a}_i^{(t)}\|^2}\right\} \approx \|\boldsymbol{a}_i^{(t)}\|^2 x_i^2 + \frac{\hat{\sigma}_i^{(t)^2}}{\|\boldsymbol{a}_i^{(t)}\|^2}. \tag{35}$$

Finally, we choose two probabilities, $\varpi_\tau < \varpi_\upsilon$, the first for penalizing spurious coefficients with maximum weight in (29) and the second for penalizing with a nonzero weight. The margins used in (29) are then

$$\tau_i = \bar{\pi}_i(\varpi_\tau), \quad \upsilon_i = \bar{\pi}_i(\varpi_\upsilon). \tag{36}$$

F. A greedy ARCD algorithm

For optimizing directly the LS criterion (3), we proposed in [16] a greedy adaptive algorithm. For the sake of completeness, we present it here as Algorithm 3, with the necessary modifications to put it in the context of this paper. We name it ARCD-G (G for Greedy).

The coordinate selection and the management of the probabilities are the same as for ARCD-L, while the CD steps are performed in accordance with the LS criterion; however, the CD step (17) is based on a relation (16) that is altered as explained below. The specific greedy character is given by the explicit construction of the support of the solution, described by steps 11-15 of Algorithm 3.

Since the quantities (11) defining the probabilities come from the MP selection, it is natural to assume that they characterize the importance of the coefficients of the solution. Like in greedy algorithms from the MP family, we sort the coefficients in the decreasing order of probabilities and

consider that the support is made of the first $\hat{L}^{(t)}$ coefficients, where $\hat{L}^{(t)}$ is an estimate of the true support length $L^{(t)}$. This estimate is computed using the Predictive Least Squares (PLS) criterion [19]. More precisely, if $\tilde{L}$ is the sparsity level for which the PLS criterion is minimum at time t, then we use the estimate

$$\hat{L}^{(t+1)} = \begin{cases} \hat{L}^{(t)} + 1, & \text{if } \tilde{L} > \hat{L}^{(t)} \\ \hat{L}^{(t)}, & \text{if } \tilde{L} = \hat{L}^{(t)} \\ \hat{L}^{(t)} - 1, & \text{if } \tilde{L} < \hat{L}^{(t)} \end{cases} \tag{37}$$

The incremental change is preferred to directly using the PLS estimate, since it leads to better performance.

We consider for the PLS criterion only the solutions having at most $M^{(t)} = \hat{L}^{(t)} + \Delta$ coefficients, where Δ is a small constant, e.g. $\Delta = 5$; this ensures that we have enough candidates for the PLS criterion, covering the case where the sparsity levels increases. The coefficients that are not among the first $M^{(t)}$ are explicitly set to zero.

For increasing the estimation quality, we employ the double residual trick proposed in [5]. Denote $\mathcal{A}^{(t)}$ the set of active coordinates, namely the first $\hat{L}^{(t)}$ that define the solution. The scalar product (16) defining the residual is computed in two different ways. For active coefficients ($i \in \mathcal{A}^{(t)}$), the residual is computed using only active coefficients, which means replacing $\mathcal{C}^{(t)}$ with $\mathcal{A}^{(t)}$ in (16). For the other coefficients the relation (16) is left as it is. Thus, the active coefficients are better estimated, since only the most trusted coefficients are used in the CD update; hence, a better solution estimation is obtained. The other coefficients are used only for building the PLS criterion, where the importance of a precise residual is not crucial.

G. Complexity

We analyze here the complexity of the proposed algorithms ARCD-L and ARCD-G, expressed in number of operations per time instant. The common products update (15) has an $O(N^2)$ complexity in the general case; however, in channel estimation, which is the main application we have in view, where the row $\boldsymbol{\alpha}^{(t)}$ is obtained by shifting $\boldsymbol{\alpha}^{(t-1)}$ and inserting a single new value, the complexity is only $O(N)$. Another common operation is the generation of the R random indices, that can be done in $O(R \log N)$ operations, by building a balanced tree of cumulated probabilities. Keeping $\boldsymbol{\pi}$ sorted as required by step 11 of ARCD-G needs also $O(R \log N)$ operations.

For ARCD-L, the update of the solution in step 8 is the most costly, requiring $O(RN)$ operations. All the other computations from steps 9 and 12-15 require $O(N)$ or even $O(R)$ operations.

For ARCD-G, denote M the average value of $M^{(t)}$, the size of the set $\mathcal{C}^{(t)}$ used in (16). Typically, this is slightly larger than the sparsity level, but can be even larger at small t; however, practically we noticed that it does not grow to a significant fraction of N. The complexity of the solution updates in step 8 is $O(RM)$. The other computations are much cheaper, e.g. $O(M)$ for evaluating the PLS criterion.

We conclude that the complexity is roughly $O(RN)$ for ARCD-L and $O(N) + O(RM)$ for ARCD-G. This compares favorably with [4], [10], [12], [11], [15], [5], where it was at least $O(MN)$, if not $O(N^2)$. A more detailed comparison with [13] is given in section IV.

More importantly, the number R of random CD steps per iterations can effectively serve for tuning the trade-off complexity/performance. We showed here that complexity is roughly proportional with R; the simulations will confirm the expected behavior: a larger R means better convergence properties.

H. Convergence

For the theoretical analysis, we assume that there is no forgetting factor, i.e $\lambda = 1$, and the data are stationary:

$$\lim_{t\to\infty} \frac{1}{t} \boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)} = \boldsymbol{\Phi}^{\infty}, \tag{38}$$

and

$$\lim_{t\to\infty} \frac{1}{t} \boldsymbol{A}^{(t)^T} \boldsymbol{b}^{(t)} = \boldsymbol{\psi}^{\infty}. \tag{39}$$

We also assume that the asymptotic covariance matrix $\boldsymbol{\Phi}^{\infty}$ is nonsingular, and so

$$\boldsymbol{x}^* = (\boldsymbol{\Phi}^{\infty})^{-1} \boldsymbol{\psi}^{\infty}. \tag{40}$$

We can prove that the ARCD algorithm is convergent in the two studied cases, for the LS and lasso criteria, regardless of the sparsity level. The convergence takes place *in average over the random choices of coordinates*. The proofs of the convergence results are given in Appendix A. We adopt also another common assumption about the probabilities for choosing the coordinates, namely that they are bounded from below by a positive constant: $\pi_i^{(t)} \geq \check{\pi} > 0$, for all coordinates i and times t. Note that this rule is actually enforced by (12) in our algorithms.

Theorem 1: Under the above assumptions, the ARCD algorithm for the LS criterion (3), using the optimal step (8), converges in average to $\boldsymbol{x}^*$. ■

We remark that the number R of CD steps per time instant appears to directly determine the convergence speed, as shown by relations (52-53). A larger R means a larger decrease of the LS criterion at time t.

We note that this Theorem does not guarantee the convergence of Algorithm ARCD-G. For this to happen, the PLS criterion must give a correct or overvalued sparsity level, i.e. $\hat{L}^{(t)} \geq L^{(t)}$ (almost always), and the MP criterion used in computing the probabilities (11) must give the highest values for the true nonzero coefficients. If the columns of $\boldsymbol{A}^{(t)}$ are very independent (e.g. almost orthogonal), then the above events are more likely; usually, this happens only when $t \gg N$. Despite the lack of a complete convergence proof, ARCD-G showed very good behavior in the numerous tests that we performed.

For the ARCD-L algorithm the convergence characterization is neater.

Theorem 2: Under the same assumptions as Theorem 1, if $\gamma_i^{(t)}/t \to 0$ then the ARCD-L algorithm converges in average to the solution of the LS problem (3). ■

We recall that the choice (28) of the penalty term in Algorithm 2 aims to respect the conditions of Theorem 2.

Theorem 2 is the randomized correspondent of the adaptive CD algorithm for the lasso criterion from [4], which used cyclic sweeps over all coordinates. Besides tunable complexity via the parameter R, ARCD-L enjoys a much simpler convergence proof than its deterministic counterpart.

I. Choice of parameters

Our algorithms have several parameters and their optimal values may depend on the data of the problem at hand. Without pretending optimality, we give here some values that appear to be robust or discuss how a parameter can be chosen. This is also an opportunity to list all the parameters.

We start with the parameters that may be considered constant; the values recommended below have been used in all tests.

- The minimum probability $\pi_{\min}$ from (12) is not critical for performance as long as it stays in $[0.3/N,\ 0.9/N]$, as shown by simulations in [16] for ARCD-G. These values are clearly better than equal probabilities $1/N$ for all coordinates. ARCD-L has similar behavior. We recommend taking $\pi_{\min} = 0.7/N$.
- The threshold $\tilde{\sigma}^2$ from (22) limits the variability of the noise estimate; given a large forgetting factor λ, large estimates of the noise (due to sharp changes of the true model) may take a long time to decay and hence produce slow convergence if $\tilde{\sigma}^2$ is large. We recommend setting the cap $\tilde{\sigma}^2 = 5\sigma^2$ for $\lambda \geq 0.98$ and removing the limit (i.e. $\tilde{\sigma}^2 = \infty$) for $\lambda < 0.98$.
- The probability ϖ_γ from (25) ensures that coefficients are penalized adequately and should be close to 1; we took $\varpi_\gamma = 0.95$.
- The asymptotic convergence factor c from (28) should respect $0 \leq c < 1$; we recommend $c = 0.9$.
- The shaping values g_τ, g_ν from (29) should respect $0 < g_\tau < g_\nu$; we took $g_\tau = 2$, $g_\nu = 4$.
- The PLS support size parameter Δ from section III-F appears to have in $\Delta = 5$ a very robust value.

The remaining parameters are tunable and we give below some empirical rules that should facilitate their choice. The simulations will illustrate our proposed rules.

- The forgetting factor λ is a common presence in adaptive algorithms and should fit the expected variability of the model.
- The number of descent steps R acts as a tradeoff between numerical complexity and solution accuracy. In principle, it is enough to take $R \geq 2$ to ensure that probabilities are allowed to change; generally for $R \geq 10$ the performance is reasonable.
- The probabilities forgetting factor θ from (13) limits spurious results and improves the stationary error; it represents a tradeoff between convergence speed and estimation error. We always take $\theta \leq \lambda$. The tests in the next section suggest that $\theta = 0.9$ produces good results.
- The probability ϖ_τ from (36) shows how likely a spurious coefficient is fully penalized in the lasso criterion. It relates to the degree of uncertainty regarding the solution. For larger uncertainty (large noise, variable coefficients) it should be lower, since penalizing the true coefficients incurs higher performance loss. The simulations use values from 0.8 to 0.9.
- The probability ϖ_υ from (36) shows how likely a spurious coefficients is at least partially penalized. It is chosen similarly to ϖ_τ, but has larger values. In simulations we took values from 0.9 to 0.999.

We remark that all tunable parameters have a significance that comes directly either from the user's needs or from a priori knowledge on the general characteristics of the problem. Note also that ARCD-L uses all the above parameters, with the exception of Δ, while ARCD-G needs $\pi_{\min}$, Δ, λ, R and θ.

IV. Simulations

Simulation conditions. A sparse FIR channel identification task is one of the main scenarios where the algorithms developed herein may be used. Based on the underlying general process (1) we produce simulation data for two specific scenarios: the solution coefficients are constant, which is the base assumption for our convergence analysis; or, the values of the coefficients change in time. In both cases we use an $L^{(t)}$-sparse filter of length $N = 200$, with $L^{(t)} = 10$ if not otherwise specified.

The coefficient variation is sinusoidal, the nonzero coefficient in position i of the FIR filter being given by

$$h_i^{(t)} = s_i \cos(2\pi f t + \phi_i). \tag{41}$$

The support is randomly chosen and for each nonzero position i the amplitude/constant s_i and the initial phase ϕ_i are uniformly distributed in $[0.05, 1]$ and $[0, 2\pi]$, respectively. The filter is normed such that the average norm over all time is 1.

The parameter f responsible for determining the variation speed in (41) is set to either 0.0002 to simulate a slow variation speed or to 0.001 for a moderately fast variation speed. The constant coefficients are generated by setting $f = 0$ and $\phi_i = 0$. For the variable channel, the forgetting factor is chosen $\lambda = 0.96$ and $\lambda = 0.92$ for the two values of f, respectively. For the constant channel we took $\lambda = 0.99$.

Some simulations include an abrupt change at a certain time instant, when 3 coefficients of the solution change position and value.

The input data from $\boldsymbol{\alpha}^{(t)}$ is normally distributed according to $\mathcal{N}(0, 1)$ and the output is corrupted by an additive Gaussian noise with variance σ^2; in most simulations we took $\sigma^2 = 0.01$. The performance of the algorithms is measured in terms of the coefficient mean square error

$$\mathrm{MSE}^{(t)} = E\{\|\boldsymbol{h}^{(t)} - \boldsymbol{x}^{(t)}\|_2^2\}, \tag{42}$$

and it is estimated by averaging the data over multiple test runs.

Our algorithms are as described in the previous section. The parameters are chosen as discussed in section III-I. Some of them have constant values in all simulation, as stated there. The other parameters are specified for each simulation setup.

In all simulations we include the results of RLS-SP, the oracle or sparsity informed RLS algorithm having prior knowledge of the position and number of coefficients and showing

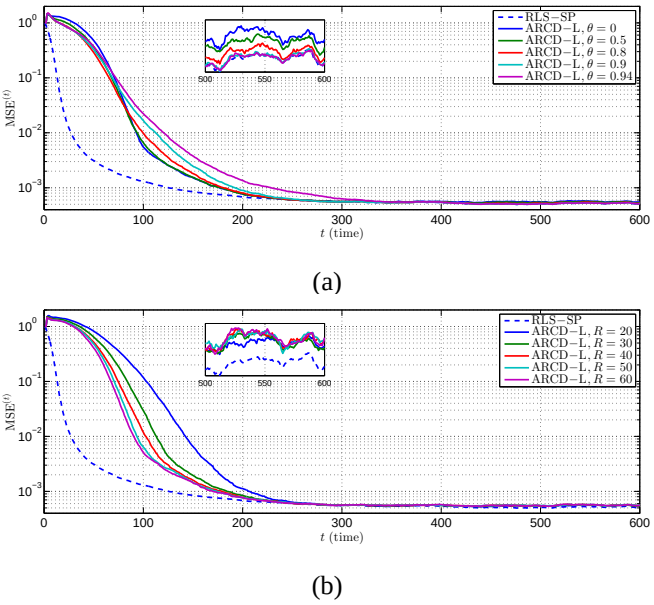

Fig. 2. $\mathrm{MSE}^{(t)}$ for ARCD-L for (a) different values of θ ($R = 50$) and for (b) different values of R ($\theta = 0$), respectively; for both figures $L^{(t)} = 10$, $f = 0$, $\lambda = 0.99$, $\varpi_\tau = 0.9$, $\varpi_\upsilon = 0.999$, $\sigma^2 = 0.01$.

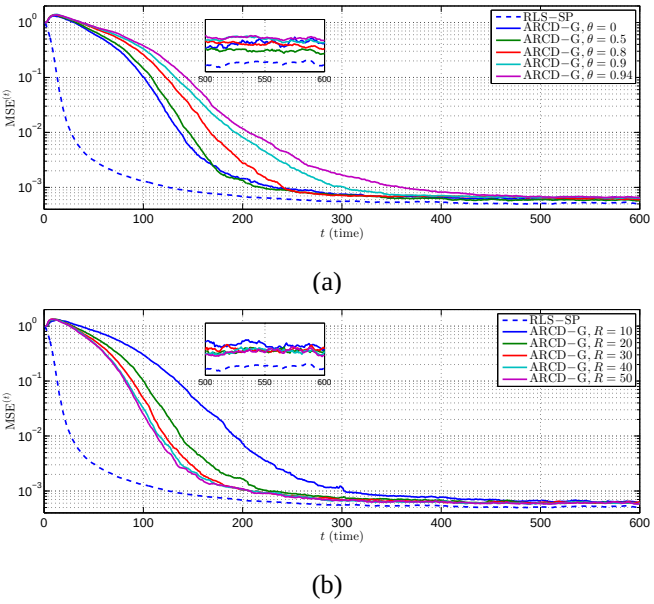

Fig. 3. $\mathrm{MSE}^{(t)}$ for ARCD-G for (a) different values of θ ($R = 20$) and for (b) different values of R ($\theta = 0$), respectively; for both figures $L^{(t)} = 10$, $f = 0$, $\lambda = 0.99$, $\sigma^2 = 0.01$.

the best attainable performance for the minimization of the LS criterion. Comparisons with other algorithms will be later discussed.

Effects of main parameters. In Fig. 2 we present the $\mathrm{MSE}^{(t)}$ for a constant channel to show the performance of ARCD-L for different values of R and θ. We present similar results in Fig. 3 for ARCD-G. It can be seen that for both algorithms the convergence speed is correlated with the number R of descent steps performed per iteration. By using a larger forgetting factor θ for the probabilities we decrease the convergence

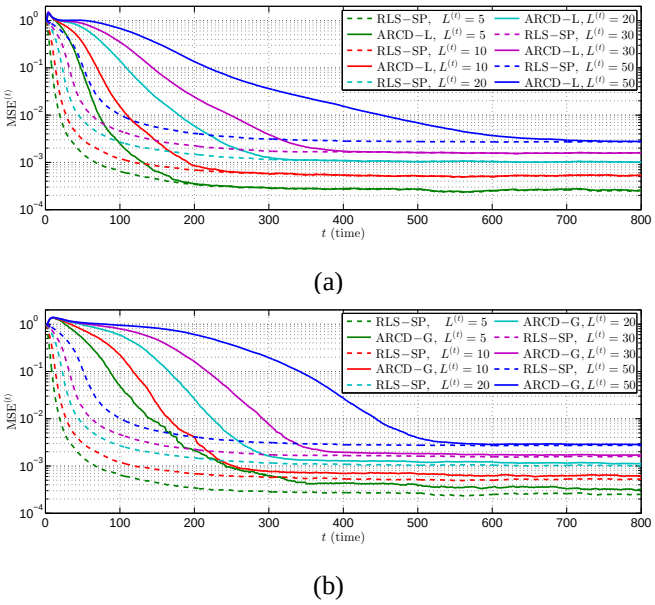

Fig. 4. $\mathrm{MSE}^{(t)}$ for different sparsity levels; (a) ARCD-L has $R = 50$; (b) ARCD-G has $R = 30$; for both figures $f = 0$, $\lambda = 0.99$, $\theta = 0.9$, $\varpi_\tau = 0.9$, $\varpi_\upsilon = 0.999$, $\sigma^2 = 0.01$.

speed but this improves the steady state performance. This is due to the mitigation of spurious changes in the values of the probabilities. The ARCD-G algorithm is able to achieve good performance with a very low number of coordinate descent steps R, however, it is outperformed by ARCD-L in terms of steady state error. We note that for larger θ in Fig. 2a we achieve almost the same $\mathrm{MSE}^{(t)}$ as RLS-SP.

In Fig. 4 we present $\mathrm{MSE}^{(t)}$ for different sparsity levels. We note that the performance scales well for larger values of the sparsity level $L^{(t)}$, with ARCD-L reaching almost identical MSE levels as RLS-SP and ARCD-G showing improved results for larger values of $L^{(t)}$, when compared to RLS-SP. As $L^{(t)}$ grows, the convergence is slower due to the larger numbers of coefficients to be identified while R is constant.

Comparisons with other algorithms. In the remainder of this section we use our algorithms with $R = 50$ and $\theta = 0.9$. We consider for comparison three recent algorithms with good performance. TNWL [4] minimizes the same criterion as ARCD-L and also uses coordinate descent, but in cyclic manner; it has $O(N^2)$ complexity. ASVB-mpL appears to be the best algorithm from [14][1]; it has no important parameters to be tuned and has an $O(N^2)$ complexity. DCD-RLS [13] uses a special form of coordinate descent on the lasso criterion; it has tunable complexity, like our algorithms, with a parameter N_u dictating the number of CD steps, like our R. Each CD step optimizes a single bit of a coefficient, choosing the coefficient that gives the largest decrease of the criterion (this strategy is sometimes called greedy CD). The steps may be successful (the decrease indeed occurs) or not, case in which one tries the bit in the next less significant position. The parameter N_u is the number of successful steps. On a general purpose

[1]We used the sources provided by the authors at http://members.noa.gr/themelis/lib/exe/fetch.php?media=code:asvb_demo_code.zip.

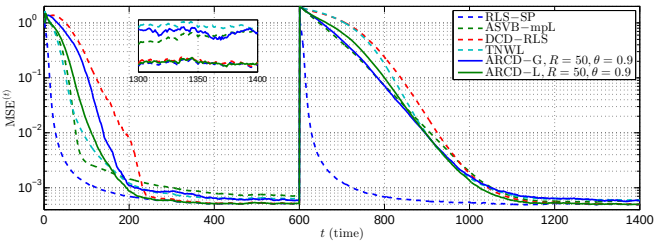

Fig. 5. $\mathrm{MSE}^{(t)}$ for a constant channel with abrupt change; the parameters are $f = 0$, $\lambda = 0.99$, $\varpi_\tau = 0.9$, $\varpi_\upsilon = 0.999$, $\sigma^2 = 0.05$.

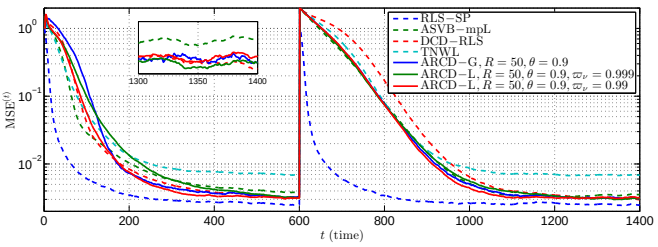

Fig. 6. $\mathrm{MSE}^{(t)}$ for a very noisy constant channel; the parameters are $f = 0$, $\lambda = 0.99$, $\varpi_\tau = 0.9$, $\sigma^2 = 0.05$.

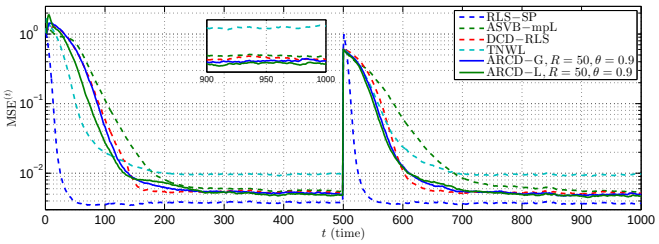

Fig. 7. $\mathrm{MSE}^{(t)}$ for a slow coefficient variation speed; the parameters are $f = 0.0002$, $\lambda = 0.96$, $\varpi_\tau = 0.8$, $\varpi_\upsilon = 0.9$, $\sigma^2 = 0.01$.

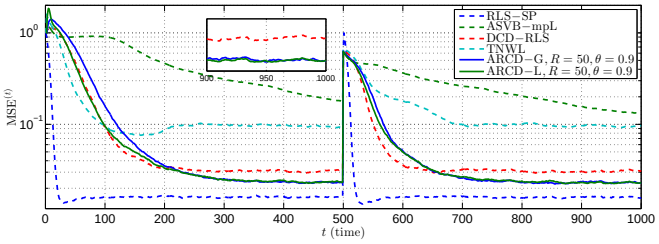

Fig. 8. $\mathrm{MSE}^{(t)}$ for a fast coefficient variation speed; the parameters are $f = 0.001$, $\lambda = 0.92$, $\varpi_\tau = 0.8$, $\varpi_\upsilon = 0.9$, $\sigma^2 = 0.01$.

processor, the complexity of a step is about $10N$ operations (some of them are shifts that can be implemented faster on specialized architecture), with $2N$ more operations if the step is successful. For $N_u < 10$, the number of actual steps is $2N_u$ or more; so, since our algorithms need about $2N$ operations per step for ARCD-L and much less for ARCD-G (see again section III-G), it appears fair to compare DCD-RLS with $N_u = 5$ with our algorithms for $R = 50$. We tried to tune the other parameters of DCD-RLS to their best values for each scenario.

The results for constant channel with abrupt change are given in Figs. 5 and 6, in the latter one the noise variance being higher. Only ARCD-L and DCD-RLS are able to attain the oracle error level, with ARCD-L having a better convergence speed. ASVB-mpL and TNWL are better in reducing the error in the initial stages, but fall behind in the long run, especially TNWL in the noisier case. ARCD-G behaves very well if its much lower complexity is taken into account. Fig. 6 also illustrates two choices for the probability ϖ_υ, namely 0.99 and 0.999. A larger value leads to slower convergence, but slightly better stationary error.

Results for variable channels. Figs. 7 and 8 show the results for the slow and fast variable channels, respectively. When the channel is variable, ARCD-L clearly benefits from lower values of the probability thresholds, namely $\varpi_\tau = 0.8$ and $\varpi_\upsilon = 0.9$. If we allow more spurious coefficients in the support, we also ensure that we recover more of the correct positions. Missing the solution support usually has a bigger hit on performance than adding a few small spurious coefficients, thus lowered probability thresholds yield better results. ARCD-G is only slightly slower than ARCD-L, but it behaves similarly. DCD-RLS has very good performance for the slow channel, but reaches a larger stationary error than our algorithms for the fast channel. ASVB-mpL is much slower (and extremely slow for the fast channel), although it reaches eventually good MSE values. TNWL has the worse stationary MSE.

We conclude that our algorithms have better behavior than the algorithms from previous work, with complexity that is lower (with respect to TNWL and ASVB-mpL) or similar (with respect to DCD-RLS).

V. Conclusions

We have presented an adaptive randomized coordinate descent (RCD) approach for finding sparse LS solutions to systems of linear equation. The probabilities that govern the selection of the CD steps are adapted online based on a matching pursuit criterion. Such adaptation allows good tracking of time varying processes.

Two RCD algorithms have been proposed. The first, ARCD-L, minimizes the ℓ_1-regularized LS criterion. In addition to the randomized character, we also proposed the adaptation of the ℓ_1 regularization term based on the online noise estimates. The second algorithm, ARCD-G, finds the sparse solution to the LS problem using a greedy approach. It orders the coefficients based on their probability (and implicitly based on their influence in decreasing the LS criterion) and relies on the PLS criterion to provide an estimate of the sparsity level. For both algorithms, the probabilistic approach drastically reduces the numerical complexity when compared to traditional deterministic methods.

Through extensive simulations we have shown that the performance of the algorithms approaches the optimal LS solution. The ARCD-L methods is more computationally demanding than ARCD-G, but it provides near oracle performance for time invariant setups. The greedy method, although very light computationally, is still able to provide good results especially for time varying scenarios, where it matches ARCD-L. Both

methods show increased performance and lower computational complexity when compared to the state of the art. Little prior knowledge is required when choosing the configuration parameters.

APPENDIX A
PROOFS OF CONVERGENCE

To show convergence, we will use the following lemma.

Lemma 1: Let $0 < \rho < 1$ and $\eta^{(t)} \geq 0$ such that $\eta^{(t)} \to 0$ as $t \to \infty$. If $\zeta^{(0)} \geq 0$ and

$$0 \leq \zeta^{(t+1)} \leq \rho \zeta^{(t)} + \eta^{(t)}, \tag{43}$$

then $\zeta^{(t)} \to 0$.

Proof. Given some $\epsilon > 0$, take $\epsilon', \epsilon'' > 0$ such that $\epsilon = \epsilon'/(1-\rho) + \epsilon''$. There exists t' such that $\eta^{(t)} < \epsilon'$ for all $t \geq t'$. The recursion (43) implies that

$$\begin{aligned} \zeta_{t+1} &\leq \sum_{\tau=0}^{t} \rho^{t-\tau} \eta^{(\tau)} \\ &= \sum_{\tau=0}^{t'-1} \rho^{t-\tau} \eta^{(\tau)} + \sum_{\tau=t'}^{t} \rho^{t-\tau} \eta^{(\tau)} \\ &\leq \rho^{t-t'} \Gamma + \epsilon'/(1-\rho), \end{aligned}$$

where $\Gamma < t' \max_{\tau=0:t'-1} \eta^{(\tau)}$ is a constant. Since $\rho < 1$, there is a t'' such that $\rho^{t-t'}\Gamma < \epsilon''$ for all $t \geq t''$. It results that $\zeta^{(t+1)} < \epsilon$ for all $t \geq \max(t', t'')$, hence $\zeta^{(t)} \to 0$. ■

For the theoretical analysis we replace (3) with

$$\bar{J}^{(t)}(\boldsymbol{x}) = \frac{1}{t} J^{(t)}(\boldsymbol{x}) \tag{44}$$

and with the notation (38), (39) we get

$$\lim_{t\to\infty} \bar{J}^{(t)}(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Phi}^{\infty} \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{\psi}^{\infty} \triangleq \bar{J}^{\infty}(\boldsymbol{x}). \tag{45}$$

The optimal solution $\boldsymbol{x}^{(*,t)}$ remains the same for both criteria, $\bar{J}^{(t)}(\boldsymbol{x})$ and $J^{(t)}(\boldsymbol{x})$.

Define the criterion errors

$$\Delta J^{(t)}(\boldsymbol{x}) = J^{(t)}(\boldsymbol{x}) - J^{(t)}(\boldsymbol{x}^{(*,t)}) \tag{46}$$

$$\Delta^* J^{(t)}(\boldsymbol{x}) = J^{(t)}(\boldsymbol{x}) - J^{\infty}(\boldsymbol{x}^*) \tag{47}$$

Similar notations are used for $\bar{J}^{(t)}(\boldsymbol{x})$ defined by (44).

Proposition 1: The average decrease of the error in a CD step of the ARCD algorithm for the LS criterion (3) obeys the relation

$$E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(k+1,t)})\right\} \leq \Delta J^{(t)}(\boldsymbol{x}^{(k,t)}) \left[1 - \check{\pi} \frac{\sigma^2_{\min}(\boldsymbol{A}^{(t)})}{\|\boldsymbol{A}^{(t)}\|_F^2}\right]. \tag{48}$$

Proof. The decrease of the LS criterion for an optimal step (8) on coordinate i is given by (9). Using the notation (46) and averaging over coordinates we obtain

$$\begin{aligned} &\Delta J^{(t)}(\boldsymbol{x}^{(k,t)}) - E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(k+1,t)})\right\} \\ &= \frac{1}{2} \sum_i \pi_i^{(t-1)} \frac{|\boldsymbol{a}_i^{(t)^T} \boldsymbol{r}^{(k,t)}|^2}{\|\boldsymbol{a}_i^{(t)}\|^2} \\ &\geq \frac{1}{2} \check{\pi} \frac{1}{\|\boldsymbol{A}^{(t)}\|_F^2} \|\boldsymbol{A}^{(t)^T} \boldsymbol{r}^{(k,t)}\|^2 \end{aligned} \tag{49}$$

Denoting $\boldsymbol{x}^{(k,t)} = \boldsymbol{x}^{(*,t)} + \boldsymbol{\delta}^{(k,t)}$, it follows that

$$\boldsymbol{A}^{(t)^T} \boldsymbol{r}^{(k,t)} = \boldsymbol{A}^{(t)^T} (\boldsymbol{r}^{(*,t)} - \boldsymbol{A}^{(t)} \boldsymbol{\delta}^{(k,t)}) = -\boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{\delta}^{(k,t)},$$

the optimal residual being orthogonal on the range of $\boldsymbol{A}^{(t)}$. It results that

$$\|\boldsymbol{A}^{(t)^T} \boldsymbol{r}^{(k,t)}\|^2 \geq \sigma^2_{\min}(\boldsymbol{A}^{(t)}) \|\boldsymbol{A}^{(t)} \boldsymbol{\delta}^{(k,t)}\|^2. \tag{50}$$

The optimal solution satisfies the normal equation

$$\boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(*,t)} = \boldsymbol{A}^{(t)^T} \boldsymbol{b}^{(t)}.$$

It follows that

$$\|\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)} \boldsymbol{x}^{(*,t)}\|^2 = \|\boldsymbol{b}^{(t)}\|^2 - \boldsymbol{b}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(*,t)}.$$

Using these equalities, we obtain

$$\begin{aligned} &\|\boldsymbol{A}^{(t)} \boldsymbol{\delta}^{(k,t)}\|^2 = \boldsymbol{x}^{(k,t)^T} \boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(k,t)} \\ &+ \boldsymbol{b}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(*,t)} - 2 \boldsymbol{b}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(k,t)} \\ &= \|\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)} \boldsymbol{x}^{(k,t)}\|^2 - \|\boldsymbol{b}^{(t)}\|^2 + \boldsymbol{b}^{(t)^T} \boldsymbol{A}^{(t)} \boldsymbol{x}^{(*,t)} \\ &= 2 \Delta J^{(t)}(\boldsymbol{x}^{(k,t)}). \end{aligned} \tag{51}$$

Combining (49), (50), and (51) gives (48). ■

Using the above results, we can prove the convergence in average of the ARCD algorithm for the LS criterion.

Proof of Theorem 1. Averaging relation (48) over the R CD steps at time t we obtain

$$E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(R,t)})\right\} \leq C^{(t)} \cdot E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(0,t)})\right\}. \tag{52}$$

where

$$C^{(t)} = \left[1 - \check{\pi} \frac{\sigma^2_{\min}(\boldsymbol{A}^{(t)})}{\|\boldsymbol{A}^{(t)}\|_F^2}\right]^R. \tag{53}$$

Due to stationarity and nonsingularity of the asymptotic covariance matrix (38), there exist a time t_0 and a constant ϵ such that with probability 1

$$\frac{\sigma^2_{\min}(\boldsymbol{A}^{(t)})}{\|\boldsymbol{A}^{(t)}\|_F^2} = \frac{\sigma_{\min}(\boldsymbol{A}^{(t)^T} \boldsymbol{A}^{(t)}/t)}{\|\boldsymbol{A}^{(t)}\|_F^2/t} \geq \epsilon$$

for all $t \geq t_0$. Then, there exist a constant $C < 1$ such that $C^{(t)} \leq C$, $\forall t \geq t_0$. Combining this with (52) and the fact that $\boldsymbol{x}^{(0,t+1)} = \boldsymbol{x}^{(R,t)}$, we obtain

$$\begin{aligned} &E\left\{\Delta^* \bar{J}^{(t+1)}(\boldsymbol{x}^{(0,t+1)})\right\} \\ &= E\left\{\bar{J}^{(t+1)}(\boldsymbol{x}^{(0,t+1)})\right\} - E\left\{\bar{J}^{(t)}(\boldsymbol{x}^{(R,t)})\right\} \\ &+ E\left\{\bar{J}^{(t)}(\boldsymbol{x}^{(R,t)})\right\} - \bar{J}^{(t)}(\boldsymbol{x}^{(*,t)}) \\ &+ \bar{J}^{(t)}(\boldsymbol{x}^{(*,t)}) - \bar{J}^{\infty}(\boldsymbol{x}^*) \\ &\leq C \cdot E\left\{\Delta^* \bar{J}^{(t)}(\boldsymbol{x}^{(0,t)})\right\} \\ &+ E\left\{|\bar{J}^{(t+1)}(\boldsymbol{x}^{(R,t)}) - \bar{J}^{(t)}(\boldsymbol{x}^{(R,t)})|\right\} \\ &+ (1+C)|\bar{J}^{(t)}(\boldsymbol{x}^{(*,t)}) - \bar{J}^{\infty}(\boldsymbol{x}^*)| \end{aligned}$$

We have thus obtained a relation like (43), with $\zeta^{(t)} = E\left\{\Delta^* \bar{J}^{(t)}(\boldsymbol{x}^{(0,t)})\right\}$. Hence, $E\left\{\Delta^* \bar{J}^{(t)}(\boldsymbol{x}^{(0,t)})\right\} \to 0$, i.e. the ARCD algorithm converges in average to the true optimum, thus obtaining the unique solution $\boldsymbol{x}^*$. ■

The convergence of ARCD-L follows via a simple remark on the structure of the soft thresholding operation.

Proof of Theorem 2. We can associate a CD step on the criterion (5) with two operations. First, an optimal LS step on coordinate i is taken, producing $\tilde{\boldsymbol{x}}^{(k,t)}$ as in (8). Note that, due to optimality, the relation

$$\boldsymbol{a}_i^{(t)^T} (\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)} \tilde{\boldsymbol{x}}^{(k,t)}) = 0 \tag{54}$$

holds. Then, the soft thresholding operation giving $\boldsymbol{x}^{(k+1,t)}$ as in (10) is performed. To this purpose a step of size at most $\gamma_i^{(t)}/\|\boldsymbol{a}_i^{(t)}\|^2$ is made. Due to (54), the increase of the criterion is at most

$$\begin{aligned} &J^{(t)}\big(\boldsymbol{x}^{(k+1,t)}\big) - J^{(t)}\big(\tilde{\boldsymbol{x}}^{(k,t)}\big) \\ &= \tfrac{1}{2}\|\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)}\tilde{\boldsymbol{x}}^{(k,t)} - \frac{\gamma_i^{(t)}}{\|\boldsymbol{a}_i^{(t)}\|^2}\boldsymbol{a}_i^{(t)}\|^2 - \tfrac{1}{2}\|\boldsymbol{b}^{(t)} - \boldsymbol{A}^{(t)}\tilde{\boldsymbol{x}}^{(k,t)}\|^2 \\ &= \frac{{\gamma_i^{(t)}}^2}{2\|\boldsymbol{a}_i^{(t)}\|^2} \end{aligned} \tag{55}$$

Averaging over coordinates and combining with the result of Proposition 1, we get instead of (52) the relation

$$E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(R,t)})\right\} \leq C^{(t)} \cdot E\left\{\Delta J^{(t)}(\boldsymbol{x}^{(0,t)})\right\} + \tilde{C}\max_i \frac{{\gamma_i^{(t)}}^2}{\|\boldsymbol{a}_i^{(t)}\|^2}, \tag{56}$$

with a constant $\tilde{C}$ whose estimate is not relevant. We divide (56) by t in order to get a similar relation for $\Delta\bar{J}^{(t)}(\cdot)$. Since $\max_i \gamma_i^{(t)}/t \to 0$ and $\|\boldsymbol{a}_i^{(t)}\|^2/t \to \phi_{ii}^{\infty}$, see (38), it results that $({\gamma_i^{(t)}}^2/\|\boldsymbol{a}_i^{(t)}\|^2)/t \to 0$. Thus, the proof of Theorem 1 can be easily adapted to (56), fitting again into the frame of Lemma 1. ■

REFERENCES

[1] T. Strohmer and R. Vershynin, "A Randomized Kaczmarz Algorithm with Exponential Convergence," *J. Fourier Anal. Appl.*, vol. 15, pp. 262–278, 2009.

[2] D. Leventhal and A. S. Lewis, "Randomized Methods for Linear Constraints: Convergence Rates and Conditioning," *Math. Oper. Res.*, vol. 35, no. 3, pp. 641–654, Aug. 2010.

[3] Yu. Nesterov, "Efficiency of coordinate descent methods on huge scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.

[4] 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 Proc.*, vol. 58, no. 7, pp. 3436–3447, July 2010.

[5] A. Onose and B. Dumitrescu, "Adaptive matching pursuit using coordinate descent and double residual minimization," *Signal Proc.*, vol. 93, no. 11, pp. 3143–3150, 2013.

[6] T. Glasmachers and U. Dogan, "Accelerated coordinate descent with adaptive coordinate frequencies," in *Proc. 5th Asian Conf. Machine Learning*, 2013, vol. 29, pp. 72–86.

[7] Y. Chen, Y. Gu, and A.O. Hero III, "Sparse LMS for System Identification," in *Int. Conf. Acoustics, Speech, Signal Proc.*, 2009, pp. 3125–3128.

[8] Y. Gu, J. Jin, and S. Mei, "ℓ_0 Norm Constraint LMS Algorithm for Sparse System Identification," *IEEE Signal Proc. Letters*, vol. 16, no. 9, pp. 774–777, Sept. 2009.

[9] G. Mileounis, B. Babadi, N. Kalouptsidis, and V. Tarokh, "An Adaptive Greedy Algorithm With Application to Nonlinear Communications," *IEEE Trans, Signal Proc.*, vol. 58, no. 6, pp. 2998–3007, June 2010.

[10] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The Sparse RLS Algorithm," *IEEE Trans. Signal Proc.*, vol. 58, no. 8, pp. 4013–4025, Aug. 2010.

[11] N. Kalouptsidis, G. Mileounis, B. Babadi, and V. Tarokh, "Adaptive Algorithms for Sparse System Identification," *Signal Proc.*, vol. 91, pp. 1910–1919, 2011.

[12] Y. Kopsinis, K. Slavakis, and S. Theodoridis, "Online Sparse System Identification and Signal Reconstruction Using Projections Onto Weighted ℓ_1 Balls," *IEEE Trans. Signal Proc.*, vol. 59, no. 3, pp. 936–952, Mar. 2011.

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

[14] K.E. Themelis, A.A. Rontogiannis, and K.D. Koutroumbas, "A Variational Bayes Framework for Sparse Adaptive Estimation," *IEEE Trans, Signal Proc.*, vol. 62, no. 18, pp. 4723–4736, Sept. 2014.

[15] B. Dumitrescu, A. Onose, P. Helin, and I. Tăbuș, "Greedy Sparse RLS," *IEEE Trans. Signal Proc.*, vol. 60, no. 5, pp. 2194–2207, May 2012.

[16] A. Onose and B. Dumitrescu, "Adaptive Randomized Coordinate Descent for Solving Sparse Systems," in *EUSIPCO*, Lisbon, Portugal, 2014.

[17] E.J. Candes, M.B. Wakin, and S.P. Boyd, "Enhancing Sparsity by Reweighted ℓ_1 Minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 877–905, 2008.

[18] H. Zou and R. Li, "One-Step Sparse Estimates in Nonconcave Penalized Likelihood Models," *Annals Stat.*, vol. 36, no. 4, pp. 1509–1533, 2008.

[19] J. Rissanen, "Order Estimation by Accumulated Prediction Errors," *J. Appl. Prob.*, vol. 23, pp. 55–61, 1986.

PLACE PHOTO HERE

Alexandru Onose was born in Bârlad, Romania, in 1985. He received the Diploma Engineer degree from University Politehnica of Bucharest, Romania, in 2009 and the Ph.D. degree from Tampere University of Technology, Tampere, Finland, in 2014.

He is currently a postdoctoral research associate with the School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, U.K.

His research interests include optimization, compressed sensing and their applications in signal processing.

PLACE PHOTO HERE

Bogdan Dumitrescu (M'01) was born in Bucharest, Romania, in 1962. He received the M.S. and Ph.D. degrees in 1987 and 1993, respectively, from University Politehnica of Bucharest, Romania.

He is now a Professor with the Department of Automatic Control and Computers, University Politehnica of Bucharest. He held several visiting research positions at Tampere International Center for Signal Processing, Tampere University of Technology, Finland, in particular that of FiDiPro fellow (2010-2013). He was Associate Editor (2008-2012) and Area Editor (2010-2014) at IEEE Transactions on Signal Processing. His scientific interests are in optimization, numerical methods, and their applications to signal processing.