CYCLIC ADAPTIVE MATCHING PURSUIT

Alexandru Onose, Bogdan Dumitrescu

Department of Signal Processing
Tampere University of Technology
PO BOX 553, 33101, Tampere, Finland
e-mail: alexandru.onose@tut.fi, bogdan.dumitrescu@tut.fi

ABSTRACT

We present an improved Adaptive Matching Pursuit algorithm for computing approximate sparse solutions for overdetermined systems of equations. The algorithms use a greedy approach, based on a neighbor permutation, to select the ordered support positions followed by a cyclical optimization of the selected coefficients. The sparsity level of the solution is estimated on-line using Information Theoretic Criteria. The performance of the algorithm approaches that of the sparsity informed RLS, while the complexity remains lower than that of competing methods.

Index Terms— matching pursuit, adaptive algorithm, sparse filters, channel identification

1. INTRODUCTION

In recent years, sparse approximation problems have been of a major interest due to their practical applicability in array processing, compression, denoising and many other tasks. The aim of this paper is to present a series of improvements to an adaptive version of Matching Pursuit (MP) [1] making it a viable alternative to more complex methods like [2, 3].

Let us consider a typical example of an FIR channel identification task. At time t the channel input $u(t)$ and output $d(t)$ are measured and our aim is to find the coefficients h_i, $i = 0 : N$, such that the estimation error

$$e(t) = d(t) - \sum_{i=0}^{N-1} h_i u(t-i) \tag{1}$$

is minimized. In a slow time varying environment, with λ as the forgetting factor, this can be translated into minimizing

$$J(t) = \sum_{i=1}^{t} \lambda^{t-i} |e(i)|^2. \tag{2}$$

This is equivalent to minimizing, at each time instance t, the norm of the residual $||\mathbf{b} - \mathbf{A}\mathbf{x}||_2$, where the matrix $\mathbf{A} \in \mathbb{R}^{t \times N}$ and the vector $\mathbf{b} \in \mathbb{R}^t$ are built with the input and output data, respectively. Furthermore, we consider that the coefficient vector has at most $M \ll N$ non-zero coefficients ($\mathbf{x}$ is sparse), fact usually valid in many practical applications. The number of significant coefficients and their positions are not known a priori.

For finding sparse approximate solution to the minimization problem presented in (2) we use an Adaptive Matching Pursuit (AMP) [4] algorithm combined with a Cyclic Matching Pursuit approach [5]. The algorithm provides models with different sparsity levels and we apply Information Theoretic Criteria (ITC) in a similar way as in [6] to choose the model that best fits the data. We present two algorithms, each assuming either a fixed or variable upper sparsity level bound M, and we show how the ITC can be computed; we prove, by empirical simulations, that the performance of these low-complexity algorithms is good.

This paper is organized as follows: in section 2 we present an improved adaptive matching pursuit algorithm; section 3 presents details regarding the ITC that are used to selecting the best support size while section 4 details their use in conjunction with our algorithms; section 5 contains the results of our simulations.

2. CYCLIC ADAPTIVE MATCHING PURSUIT

We begin by presenting an extension to the classic MP algorithm adjusted for an adaptive context. At the base of our method resides an improved AMP that uses a cyclic coefficient re-computation [5] to minimize the prediction error.

Like in the MP case, the algorithm selects one by one columns from the matrix $\mathbf{A}$ (named active columns) such that they are best aligned with the residual. Upon selecting the first column $\mathbf{a}_{k_1}$ best aligned with $\mathbf{b}_0 = \mathbf{b}$, the projection of the current residual, $\mathbf{b}_0$, on the direction of $\mathbf{a}_{k_1}$ is removed from itself, thus resulting a new residual $\mathbf{b}_1$. At step i, the search for the best aligned column $\mathbf{a}_{k_i}$ continues in the set $\mathcal{I}$ of columns not yet chosen; a new column is selected such that is best aligned with the current residual $\mathbf{b}_{i-1}$

$$k_i = \arg\max_{l \in \mathcal{I}} \frac{|\mathbf{a}_l^T \mathbf{b}_{i-1}|^2}{||\mathbf{a}_l||^2}; \tag{3}$$

the new coefficient represents the alignment of the column with the residual

$$x_{k_i} = \frac{\mathbf{a}_{k_i}^T \mathbf{b}_{i-1}}{||\mathbf{a}_{k_i}||^2}; \tag{4}$$

the residual $\mathbf{b}_i$ is then computed by removing the influence of the column $\mathbf{a}_{k_i}$ form $\mathbf{b}_{i-1}$

$$\mathbf{b}_i = \mathbf{b}_{i-1} - x_{k_i} \mathbf{a}_{k_i}. \tag{5}$$

Work supported by Tekes FiDiPro – Finland Distinguished Professor Programme grant. B. Dumitrescu is also with Department of Automatic Control and Computers, "Politehnica" University of Bucharest, Romania.

The selection of active columns stops once a given number of columns M is reached.

The selection of the new column based on (3) guarantees that the residual is decreased by the largest amount at each step, without changing the values of the previously computed coefficients. Once a set of M_i columns is selected, the value of the coefficients can be updated by cyclically optimizing one coefficient at a time while holding the other $M_i - 1$ coefficients fixed. An update of the coefficient x_{k_i} is performed by removing the associated column $\mathbf{a}_{k_1}$ from the active set, restoring the influence it had in decreasing the residual

$$\mathbf{b}^{'}_{M_i} = \mathbf{b}_{M_i} + x_{k_i}\mathbf{a}_{k_i} \tag{6}$$

and reintroducing it in the active set again. By plugging equation (6) into (4), the new coefficient value is

$$x^{'}_{k_i} = \frac{\mathbf{a}^T_{k_i}\mathbf{b}^{'}_{M_i}}{||\mathbf{a}_{k_i}||^2} = x_{k_i} + \gamma, \quad \text{with } \gamma = \frac{\mathbf{a}^T_{k_i}\mathbf{b}_{M_i}}{||\mathbf{a}_{k_i}||^2}. \tag{7}$$

Using the coefficient expression and the residual updates (5) and (6), the cyclic update, made in place, is summarized by

$$\mathbf{b}_{M_i} \leftarrow \mathbf{b}^{'}_{M_i} - (x_{k_i} + \gamma)\mathbf{a}_{k_i} = \mathbf{b}_{M_i} - \gamma\mathbf{a}_{k_i} \tag{8}$$

$$x_{k_i} \leftarrow x^{'}_{k_i}. \tag{9}$$

Cyclically performing the update for each coefficient a number of times further minimizes the residual.

We propose two methods for estimating the values of M coefficients; the first, named Cyclic Adaptive Matching Pursuit (CAMP) and presented in Alg. 3, cyclically updates the coefficients after all the active columns are chosen; the second, named Iterated Cyclic Adaptive Matching Pursuit (ICAMP) and presented in Alg. 4, cyclically improves the coefficients after the introduction of each new column in the active set.

The algorithms are efficiently implemented using only information about the scalar products $\Phi_{i,j} = \mathbf{a}_i^T\mathbf{a}_j$ between the columns of the matrix $\mathbf{A}$ and the scalar products $\Psi_i = \mathbf{a}_i^T\mathbf{b}$ between the the columns of $\mathbf{A}$ and output vector $\mathbf{b}$. Equations (5), (6) and (8) can be expressed with the use of the scalar products $\mathbf{\Psi}$ and $\mathbf{\Phi}$ by multiplying on the left with $\mathbf{A}^T$, while for the others, the introduction of the scalar products is immediate.

At time t, upon receiving a new input data vector $\boldsymbol{\alpha}$ and output data β, the scalar products $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are updated in place and a copy, $\tilde{\mathbf{\Psi}}$ of $\mathbf{\Psi}$, is used to store the scalar products with the current residual (Alg. 3 and 4, equation $(*)$). Due to the slow varying nature of the considered problem, when we choose the active column set (Alg. 3 and 4, step 2.1) we reuse the previous selection computed at time $t-1$ and allow changes in the column order only between neighbor positions. The only exception is the last position, for which all the remaining columns compete.

For the column selection and the coefficient estimation (Alg. 1) we use the scalar products $\tilde{\mathbf{\Psi}}$ between $\mathbf{A}$ and the current residual and update them in place (Alg. 1, steps 1.1, 2.1, 5). This is performed after each new column is selected, according to a vectorial version of (5). We note that, although the algorithm can be implemented without permuting the matrix $\mathbf{A}$, to make the presentation simpler we consider that the columns in $\mathbf{A}$ (and the other corresponding matrices) are ordered according to their influence on the residual (Alg. 1, step 3). The computation of the coefficient value (Alg. 1, step 4), if we consider the permutation influence on $\mathbf{x}$, follows directly from (4).

For both CAMP and ICAMP algorithms, after the column selection and the initial coefficient estimation, N_{it} cyclical updates are performed such that the residual is further

Alg. 1 (**Estimate the coefficient** i).

1 if $i < M$
- 1.1 update scalar product for next column $\tilde{\Psi}_{i+1} \leftarrow \tilde{\Psi}_{i+1} - \mathbf{\Phi}_{i+1,1:i-1}\mathbf{x}_{1:i-1}$
- 1.2 $k_i = \arg\max_{l\in[i,i+1]} \frac{\tilde{\Psi}_l^2}{\Phi_{l,l}}$ (find best candidate column searching between neighbors)

2 if $i == M$
- 2.1 update remaining scalar products $\tilde{\mathbf{\Psi}}_{M+1:N} \leftarrow \tilde{\mathbf{\Psi}}_{M+1:N} - \mathbf{\Phi}_{M+1:N,1:M-1}\mathbf{x}_{1:M-1}$
- 2.2 $k_i = \arg\max_{l\in[M:N]} \frac{\tilde{\Psi}_l^2}{\Phi_{l,l}}$ (find the best k_i candidate column searching all remaining columns)

3 swap columns (and lines) i and k_i in $\mathbf{\Phi}$
swap elements i and k_i in $\tilde{\mathbf{\Psi}}$, $\mathbf{x}$ and $\mathbf{\Psi}$

4 $x_i = \frac{\tilde{\Psi}_i}{\Phi_{i,i}}$ (evaluate the coefficient value)

5 $\tilde{\mathbf{\Psi}}_{1:i+1} \leftarrow \tilde{\mathbf{\Psi}}_{1:i+1} - x_i\mathbf{\Phi}_{1:i+1,i}$ (update the scalar product considering the new residual)

Alg. 2 (**Cyclic update of** M_i **coefficients**).

1 for $i = 1 : M_i$
- 1.1 $\gamma = \frac{\tilde{\Psi}_i}{\Phi_{i,i}}$
- 1.2 $x_i \leftarrow x_i + \gamma$ (update the coefficient)
- 1.3 $\tilde{\mathbf{\Psi}}_{1:M_i+1} \leftarrow \tilde{\mathbf{\Psi}}_{1:M_i+1} - \gamma\mathbf{\Phi}_{1:M_i+1,i}$

Alg. 3 (**CAMP: Cyclic adaptive matching pursuit**).

1 update scalar products with current data; save a copy of scalar products $\mathbf{\Psi}_{1:N}$

$$\left.\begin{array}{l} \mathbf{\Phi}_{1:N,1:N} \leftarrow \lambda\mathbf{\Phi}_{1:N,1:N} + \boldsymbol{\alpha}_{1:N}\boldsymbol{\alpha}^T_{1:N} \\ \mathbf{\Psi}_{1:N} \leftarrow \lambda\mathbf{\Psi}_{1:N} + \beta\boldsymbol{\alpha}_{1:N} \\ \tilde{\mathbf{\Psi}}_{1:N} = \mathbf{\Psi}_{1:N} \end{array}\right\} \quad (*)$$

2 for $i = 1 : M$ (select coefficients one by one)
- 2.1 estimate the coefficient i as in Alg. 1

3 for $l = 1 : N_{\text{it}}$
- 3.1 update $M_i = M$ coefficients as in Alg. 2

Alg. 4 (**ICAMP: Iterated cyclic adaptive matching pursuit**).

1 update scalar products like in $(*)$

2 for $i = 1 : M$ (select coefficients one by one)
- 2.1 estimate the coefficient i as in Alg. 1
- 2.2 for $l = 1 : N_{\text{it}}$
  - 2.2.1 update $M_i = i$ coefficients as in Alg. 2
- 2.3 $\check{\mathbf{x}}_{i,1:i} = \mathbf{x}_{1:i}$ (store the current coefficients)

decreased. The procedure is presented in Alg. 2 and follows closely (7) and (8), the main difference consisting in the use of the scalar products $\tilde{\mathbf{\Psi}}$ defined with the residual $\mathbf{b}_{M_i}$ from (8).

The number of operations necessary for implementing the selection of active columns and computation of the coefficients is the same for both CAMP and ICAMP $\nu \approx \frac{3}{2}N^2 + \frac{1}{2}M^2 + NM$. The added complexity due to the cyclical update is different for the two algorithms; in case of CAMP it is $\rho_0 \approx N_{\text{it}}M^2$; for ICAMP the computational burden is greater, $\tau_0 \approx \frac{1}{3}N_{\text{it}}M^3 + 4N_{\text{it}}M^2$.

3. INFORMATION THEORETIC CRITERIA

The algorithms presented so far order the elements of the solution based on their contribution in decreasing the residual. If we consider that the real cardinality L_t of the solution support is unknown and we apply the algorithms for a number of non-zero elements $M \geq L_t$ then, it is plausible to assume that, when enough data are available, the first L_t positions chosen by the CAMP and ICAMP algorithms correspond with high probability to the non-zero locations of the true solution.

To find an estimate L for the true cardinality of the support L_t we employ information theoretic criteria (ITC). We use two model selection methods, Bayesian information criterion (BIC) [7] and Predictive least squares criterion (PLS) [8] as suggested by [6] for a similar order selection task.

3.1. Bayesian information criterion (BIC)

Consider, at time t, the squared norm of the residual $\Gamma_k = \mathbf{b}_k^T\mathbf{b}_k$ computed for a solution $\mathbf{x}$ with a support cardinality k; if we use the fact that $\mathbf{b}_k = \mathbf{b}_0 - \sum_{i=1}^{k} x_i\mathbf{a}_i$, it results that

$$\begin{aligned}\mathbf{b}_k^T\mathbf{b}_k &= \left(\mathbf{b}_0 - \textstyle\sum_{i=1}^{k} x_i\mathbf{a}_i\right)^T\left(\mathbf{b}_0 - \textstyle\sum_{i=1}^{k} x_i\mathbf{a}_i\right) \\ &= \Gamma_0 - 2\textstyle\sum_{i=1}^{k} x_i\Phi_i + \sum_{i=1}^{k}\sum_{j=1}^{k} x_i x_j \Psi_{i,j}.\end{aligned} \tag{10}$$

Using the norm of the residual, we can define the BIC criterion [7] at time t as

$$\mathbf{BIC}_{k,t} = n_{\text{ef}} \ln \Gamma_k + (k+1)\ln n_{\text{ef}}, \tag{11}$$

where $n_{\text{ef}} = \sum_{i=0}^{t}\lambda^{t-i}$ is the effective number of samples used to determine the solution $\mathbf{x}$.

For the ICAMP algorithm, the criterion (11) can be computed directly with (10) and the stored values for the coefficients $\check{\mathbf{x}}_{k,1:k}$, while for CAMP the following recursion is used

$$\Gamma_k = \Gamma_{k-1} - 2x_k\Phi_k + 2x_k\mathbf{x}_{1:k}^T\Psi_{k,1:k} - x_k^2\Psi_{k,k}^2. \tag{12}$$

3.2. Predictive least squares criterion (PLS)

The PLS criterion [8] at time t is defined as

$$\mathbf{PLS}_{k,t} = \sum_{i=0}^{t}\lambda^{t-i}e_{k,i}^2, \tag{13}$$

where $e_{k,i} = \beta - \boldsymbol{\alpha}_{1:k}^T\mathbf{x}_{1:k}$ is the a priori estimation error at time i produced by the k-sparse solution $\mathbf{x}$ computed at time $i-1$, $\boldsymbol{\alpha}$ and β are the input and output at time i. The PLS criterion is given by

$$\mathbf{PLS}_{k,t} = \lambda\mathbf{PLS}_{k,t-1} + e_{k,t}^2. \tag{14}$$

Alg. 5 (**ITC CAMP**).

1 estimate the coefficients as in Alg. 3
2 estimate the support size L using BIC or PLS
3 $\tilde{\Psi}_{1:L} \leftarrow \tilde{\Psi}_{1:L} + \Phi_{1:L,L+1:M}\mathbf{x}_{L+1:M}$ (remove the contribution of the other $M-L$ columns)
4 for $l = 1 : N_{\text{it}}$
  4.1 update $M_i = L$ coefficients as in Alg. 2
5 if the variable M algorithm is used, increase or decrease M by 1 such that it approaches $L + \Delta$

Alg. 6 (**ITC ICAMP**).

1 estimate the coefficients as in Alg. 4
2 estimate the support size L using BIC or PLS
3 use the stored values $\check{\mathbf{x}}_{L,1:L}$ for the coefficients
4 if the variable M algorithm is used, increase or decrease M by 1 such that it approaches $L + \Delta$

4. ITC AND THE CAMP ALGORITHMS

We propose two variants of the algorithms. For the first we define a maximum, fixed, sparsity level M chosen large enough to accommodate all possible values for the true support size L_t and apply the ITC to choose an estimate L for the support size. The second uses a variable upper sparsity level M which is increased or decreased by one at each sample time such that it approaches $L + \Delta$, where Δ is a parameter guaranteeing that there are enough candidates for finding the best number of non-zero elements and L is the estimate for the support size. The use of a variable M ensures a lower number of operations and an improved robustness to unknown sparsity levels.

The estimation L of the true number of non-zero elements L_t results from minimizing the criterion

$$L = \arg\min_{k=1:M} \mathbf{BIC}_{k,t} \quad \text{or} \quad \mathbf{PLS}_{k,t}. \tag{15}$$

By using the BIC and PLS criteria in conjunction with CAMP and ICAMP, two algorithms result (Alg. 5 and Alg. 6).

The use of the ITC increases the complexity of the base algorithms. For CAMP, the additional complexity is $\rho_{1,\text{bic}} \approx \rho_{1,\text{pls}} \approx N_{\text{it}}L^2 + M^2$ when using BIC and PLS; for ICAMP, the added complexity is $\tau_{1,\text{bic}} \approx \frac{2}{3}M^3 + 3M^2$ for BIC or $\tau_{1,\text{pls}} \approx M^2$ for PLS.

5. SIMULATIONS

The algorithms were tested for a FIR channel identification problem (1) with $L_t = 5$ nonzero coefficients and a filter order $N = 200$. The coefficient positions are randomly chosen while their variation is described by

$$\tilde{x}_i(t) = a_i\cos(2\pi f T_s t + \phi_i), \tag{16}$$

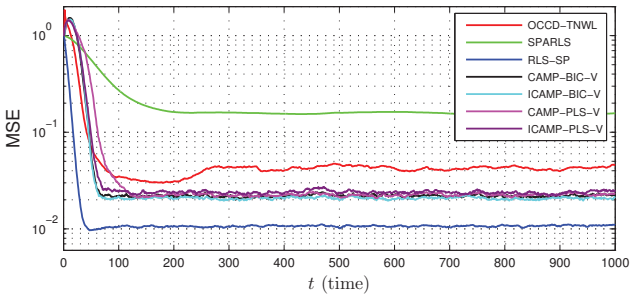

Fig. 1. Squared coefficient error for M = 5.

where the amplitude a_i and phase ϕ_i are distributed uniformly in $[0.05, 1]$ and $[0, 2\pi]$, respectively. Afterwards, the filter is normed so that its average norm is $E\{||\tilde{\mathbf{x}}||_2^2\} = 1$. The inputs are normally distributed according to $\mathcal{N}(0, 1)$ and the outputs are affected by an additive Gaussian noise with $\sigma^2 = 0.01$. The channel variation speed is controlled by the product fT_s. The measure for the performance is the coefficient mean squared error (MSE)

$$\mathbf{MSE}(t) = E\{||\tilde{\mathbf{x}} - \mathbf{x}||_2^2\}, \tag{17}$$

where $\tilde{\mathbf{x}}$ contains the real value of the coefficients and $\mathbf{x}$ their estimate. The estimate of the MSE is computed over 1000 runs for each of the variation speeds fT_s.

In Table 1 we present the MSE of several algorithms, averaged over the last 100 samples, for different variation speeds fT_s and forgetting factors λ: RLS-SP, the sparsity informed RLS algorithm with prior knowledge of the nonzero filter coefficients positions; RLS, the standard algorithm that considers a full filter of order N; GRLS, the greedy algorithm with prior knowledge of the support size from in [6]; CAMP and ICAMP, the variants of our algorithms with prior knowledge of the support size; (I)CAMP-PLS/BIC-F/V, the fixed and variable M variants of the algorithms presented in Alg. 5 and 6; SPARLS, the algorithm presented in [3]; OCCD-TNWL, the best of the algorithms from [2].

The configuration of the GRLS, SPARLS and OCCD-TNWL algorithms is done according to the recommendations from the corresponding articles. The parameter γ, used in SPARLS and presented in Table 1, was optimized using a grid search. For GRLS, CAMP and ICAMP algorithms the value for the sparsity was chosen $M = L_t$; the algorithms that employ ITC use $M = 20$ for the fixed threshold version and $\Delta = 5$ for the variable threshold algorithm; for all our algorithms the number of cyclic optimization rounds was $N_{\text{it}} = 5$.

In Fig. 1 we present the time evolution of our algorithms with a variable upper sparsity level limit together with evolution of OCCD-TNWL, SPARLS and RLS-SP for $fT_s = 0.001$.

6. CONCLUSIONS

We proposed two adaptive MP algorithm families (CAMP and ICAMP) employing a cyclical coefficient re-computation and using ITC (BIC and PLS) to estimate on-line the support size. The complexity of the algorithms is lower than that of competing methods; for the ICAMP algorithms the MSE is in general comparable with that of GRLS, while for the less complex CAMP algorithms the MSE slightly degrades; the additional complexity due to the ITC is low if M is small while giving increased robustness.

Table 1. MSE for the studied algorithms.

fT_s	0.002	0.001	0.0005	0.0002	0.0001
λ	0.90	0.92	0.94	0.96	0.98
RLS	5.3012	1.4565	0.36427	0.09774	0.04734
RLS-SP	0.0267	0.0110	0.00518	0.00246	0.00306
GRLS	0.0501	0.0178	0.00785	0.00343	0.00343
CAMP	0.0700	0.0266	0.01161	0.00453	0.00352
ICAMP	0.0534	0.0181	0.00790	0.00346	0.00343
CAMP-BIC-F	0.0805	0.0244	0.00928	0.00380	0.00367
CAMP-BIC-V	0.0649	0.0225	0.00984	0.00445	0.00379
CAMP-PLS-F	0.0899	0.0288	0.01117	0.00430	0.00371
CAMP-PLS-V	0.0745	0.0246	0.01030	0.00430	0.00366
ICAMP-BIC-F	0.0769	0.0220	0.00856	0.00377	0.00377
ICAMP-BIC-V	0.0639	0.0210	0.00992	0.00487	0.00397
ICAMP-PLS-F	0.0881	0.0249	0.00898	0.00343	0.00348
ICAMP-PLS-V	0.0683	0.0194	0.00762	0.00326	0.00340
SPARLS	0.4417	0.1578	0.04225	0.01120	0.00767
γ	170	110	75	50	75
OCCD-TNWL	0.4802	0.0436	0.01231	0.00447	0.00372

7. REFERENCES

[1] S.G. Mallat and Z. Zhang, "Matching Pursuit with Time Frequency Dictionaries," *IEEE Trans. Sgn. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[2] D. Angelosante, J.A. Bazerque, and G.B. Giannakis, "Online Adaptive Estimation of Sparse Signals: Where RLS Meets the l_1-Norm," *IEEE Trans. Sign. Proc.*, vol. 58, no. 7, pp. 3436–3447, July 2010.

[3] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The Sparse RLS Algorithm," *IEEE Trans. Sign. Proc.*, vol. 58, no. 8, 2010.

[4] S.F. Cotter and B.D. Rao, "The Adaptive Matching Pursuit Algorithm for Estimation and Equalization of Sparse Time-Varying Channels," in *34th Asilomar Conf. Sign. Syst. Comp.*, 2000, vol. 2, pp. 1772–1776.

[5] M.G. Christensen and S.H. Jensen, "The Cyclic Matching Pursuit and its Application to Audio Modeling and Coding," in *41th Asilomar Conf. Sign. Syst. Comp.*, nov. 2007, pp. 550–554.

[6] B. Dumitrescu, A. Onose, P. Helin, and I. Tăbuş, "Greedy Sparse RLS," *Submitted to IEEE Trans. Sign. Proc.*, July 2011.

[7] G. Schwarz, "Estimating the Dimension of a Model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

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