Zero-Attracting Recursive Least Squares Algorithms

Xia Hong, *Senior Member, IEEE*, Junbin Gao, and Sheng Chen, *Fellow, IEEE*

***Abstract*—The l_1-norm sparsity constraint is a widely used technique for constructing sparse models. In this paper, two zero-attracting recursive least squares algorithms, which are referred to as ZA-RLS-I and ZA-RLS-II, are derived by employing the l_1-norm of the parameter vector constraint to facilitate model sparsity. To achieve a closed-form solution, the l_1-norm of the parameter vector is approximated by an adaptively weighted l_2-norm in which the weighting factors are set as the inversion of the associated l_1-norm of parameter estimates that are readily available in the adaptive learning environment. ZA-RLS-II is computationally more efficient than ZA-RLS-I by exploiting the known results from linear algebra and the sparsity of the system. The proposed algorithms are proven to converge, and adaptive sparse channel estimation is used to demonstrate the effectiveness of the proposed approach.**

***Index Terms*—Adaptive channel estimation, l_1-norm, sparse model, zero-attracting recursive least squares algorithm.**

I. Introduction

ADAPTIVE filtering and system identification algorithms [1], e.g., the least mean squares (LMS), normalized least mean squares (NLMS), and recursive least squares (RLS) algorithms, are widely used in estimation problems such as channel estimation. In communications, the multipath wireless channel is characterized by multipath taps that are widely spread in time, with only a few significant components. Intuitively, this inherent sparsity of the channel impulse response (CIR) should be exploited to improve the quality of channel estimation. However, neither RLS nor LMS exploits the underlying sparsity in the data process, and their achievable system performance can be seriously impaired.

Alternatively, the sparse representation of an observed signal, in which the given signal is modeled as a linear combination of some significant atoms taken from an overcomplete dictionary, has been widely researched in the areas of computational biology, medicine, neuroscience, and compressive sensing. With regard to sparse modeling for a given dictionary, many algorithms exist, which are divided into three categories: optimization-based methods, greedy-based methods, and thresholding-based methods. Basis pursuit (BP) is a commonly used optimization method, which uses a convex optimization method to minimize the l_1-norm of the sparse coefficient vector [2], [3]. The computational complexity of the BP is very high, and therefore, it is not suitable for large-scale problems. In comparison, matching pursuit (MP), which is a greedy algorithm, has significantly lower complexity than BP, particularly when the sparsity level is low [4]. A popular extension of MP is the orthogonal MP [5], [6], which iteratively refines a sparse representation by successively identifying one atom at a time that yields the greatest improvement in modeling quality until an expected sparsity level is achieved or the approximation error is below the given threshold. The thresholding-based methods contain algorithms that do not require an estimation of sparsity. In these algorithms, the hard thresholding operator gives way to a soft thresholding operator with a positive threshold, such as in the iterative hard thresholding algorithm [7] and the hard thresholding pursuit [8]. Another important sparse modeling method is the message-passing algorithm studied in [9]. Unfortunately, all of the aforementioned algorithms are not designed for time-varying environments, such as vehicular communication applications; thus, they are not appropriate for the problem of sparse channel estimation in real time.

The LMS algorithm is one of the most popular adaptive algorithms for channel estimation since it has very low computational complexity and is easy to implement at the mobile handset receiver. Several adaptive sparse modeling algorithms have been recently proposed based on LMS [10]–[13]. A good example of them is the zero-attracting LMS (ZA-LMS) algorithm proposed in [10]. This algorithm can achieve a faster convergence rate, while reducing the steady-state excess mean square error, compared with the classic LMS algorithm. The ZA-LMS algorithm introduces an l_1-norm of the parameter vector in the cost function of the LMS algorithm, which modifies the parameter vector update equation with a zero attractor term. Similarly, the zero-attracting NLMS (ZA-NLMS) algorithm has been introduced based on NLMS, which yields better performance than the NLMS [12]. The l_0-norm, which is defined as the number of nonzero terms in the parameter vector, is a more appropriate measure of sparsity, and the work in [13] introduces an l_0-norm of the parameter vector in the cost function of the LMS algorithm. However, a nonlinear approximation to the l_0-norm is needed in practical implementation, and this requires an additional tuning parameter [12].

Manuscript received January 26, 2015; revised September 9, 2015 and January 21, 2016; accepted February 20, 2016. Date of publication February 24, 2016; date of current version January 13, 2017. This work was supported by the Australian Research Council under Grant DP140102270. The review of this paper was coordinated by Prof. S. Muhaidat.

X. Hong is with the Department of Computer Science, School of Mathematical and Physical Sciences, University of Reading, Reading RG6 6AY, U.K. (e-mail: x.hong@reading.ac.uk).

J. Gao is with the Discipline of Business Analytics, The University of Sydney Business School, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: junbin.gao@sydney.edu.au).

S. Chen is with the Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K., and also with King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: sqc@ecs.soton.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2016.2533664

Recently, sparse solutions have been proposed based on RLS algorithms [14]–[21]. The so-called SPARLS algorithm is introduced in [14] using an expectation-maximization (EM) approach. In [15], an adaptive version of the greedy least squares method using partial orthogonalization to systems is proposed. In [16], the RLS algorithm is modified by using a general convex function of the system parameters, resulting in the l_0-RLS and l_1-RLS algorithms. In comparison to the LMS-based algorithms, the RLS-based algorithms have a faster convergence speed and yield more accurate parameter estimates.

Against this background, in this paper, we propose two zero-attracting RLS (ZA-RLS) algorithms, which are referred to as ZA-RLS-I and ZA-RLS-II. Similar to [10] and [16], the l_1-norm of the parameter vector penalty is added to the RLS cost function. For tractability, we further approximate the l_1-norm of the parameter vector penalty term as an adaptively weighted l_2-norm of the parameter vector term, in which the weights are readily given by the inversion of the associated l_1-norm of the parameter estimates that are currently available in the adaptive learning environment. We initially derive ZA-RLS-I, which, however, has a higher computational cost than the RLS algorithm, due to the need for a matrix inversion at each adapting step. To overcome this limitation, ZA-RLS-II is then designed, which has a computational cost comparable to the RLS algorithm, and this is achieved by exploiting the matrix theory and structural properties of the matrices involved. Additionally, an analysis on the convergence of the proposed algorithms is given. Sparse channel identification results are included to demonstrate that our ZA-RLS approach achieves better performance, in comparison to the existing l_1-RLS algorithm in [16] and the SPARLS algorithm in [14].

Throughout this paper, $(\ )^*$ denotes complex conjugate, whereas $(\ )^T$ and $(\ )^H$ denote the vector or matrix transpose and Hermitian operators, respectively. $(\ )^{-1}$ stands for the inverse operation, and the expectation operator is denoted by $E\{\ \}$. Furthermore, $\boldsymbol{I}$ denotes the identity matrix with an appropriate dimension, and $\mathrm{diag}\{d_0, d_1, \ldots, d_L\}$ is the diagonal matrix with $d_0, d_1, \ldots, d_L$ as its diagonal elements, whereas $\mathrm{tr}\{\ \}$ denotes the matrix trace operation.

II. Adaptive Algorithms With l_1-Norm Sparsity

Consider a transmission link that is represented by a finite-duration impulse response filter of order L. It is assumed that the channel is inherently sparse in which only a few CIR coefficients are dominant with large values, but most of the CIR taps are zero or close to zero. Given the input signal $x(k) \in \mathbb{C}$, the received output signal $y(k) \in \mathbb{C}$ is described by

$$y(k) = \sum_{i=0}^{L} h_i x(k-i) + n(k) = \boldsymbol{x}^T(k)\boldsymbol{h} + n(k) \quad (1)$$

where k denotes the symbol index, and $n(k) \in \mathbb{C}$ is the channel additive white Gaussian noise with power of $N_o = E\{|n(k)|^2\} = E\{n(k)n^*(k)\} = 2\sigma_n^2$, whereas $\boldsymbol{h} = [h_0\ h_1, \ldots, h_L]^T$ denotes the CIR coefficient vector, and $\boldsymbol{x}(k) = [x(k)\ \ x(k-1), \ldots, x(k-L)]^T$ is the input vector.

In adaptive filtering and system identification, the parameters are computed recursively in time, so that the estimate $\widehat{\boldsymbol{h}}(k)$, as an estimate of $\boldsymbol{h}$ at time k, is given as a modification of $\widehat{\boldsymbol{h}}(k-1)$, based on the error signal $e(k) = y(k) - \boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1)$ upon the arrival of the new data $\{x(k), y(k)\}$. Note that model sparsity can be achieved by adding an l_1-norm penalty to its parameters in the cost function and, here, we briefly review several l_1-norm-based adaptive algorithms that will be used in our comparative studies.

ZA-LMS Algorithm: In [10] minimizing the cost function given by

$$V_{\mathrm{ZA-LMS}}(\widehat{\boldsymbol{h}}) = \frac{1}{2}|e(k)|^2 + \rho_{\mathrm{ZA-LMS}} \sum_{i=0}^{L} |\widehat{h}_i| \quad (2)$$

where $\rho_{\mathrm{ZA-LMS}} > 0$ is a small regularization parameter, is proposed. It can be seen that the cost function (2) is obtained by adding the l_1-norm of the parameter vector, i.e., $\widehat{\boldsymbol{h}} = [\widehat{h}_0\ \ \widehat{h}_1, \ldots, \widehat{h}_L]^{\mathrm{T}}$, to the LMS cost function based on the instantaneous squared error, i.e., $(1/2)|e(k)|^2$. This results in the following simple update equation for the ZA-LMS algorithm [10]:

$$\widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + \mu \cdot \boldsymbol{x}^*(k)e(k) - \mu \cdot \rho_{\mathrm{ZA-LMS}} \cdot \mathrm{sgn}\left(\widehat{\boldsymbol{h}}(k-1)\right) \quad (3)$$

where $\mu > 0$ is a preset small learning rate parameter, and $\mathrm{sgn}(u)$ is the component-wise sign function defined by

$$\mathrm{sgn}(u) = \begin{cases} \frac{u}{|u|}, & \text{if } u \neq 0 \\ 0, & \text{if } u = 0. \end{cases}$$

The algorithmic parameters in ZA-LMS can be set by using the criteria proposed in [22] and [23]. Specifically, these are based on steady-state mean squares deviation convergence analysis using white input signal [22] and application of LMS to a distributed network [22], respectively.

ZA-NLMS Algorithm: The ZA-NLMS algorithm [12] is given as

$$\widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + \mu \cdot \frac{\boldsymbol{x}^*(k)e(k)}{\boldsymbol{x}^H(k)\boldsymbol{x}(k)} - \mu \cdot \rho_{\mathrm{ZA-NLMS}} \cdot \mathrm{sgn}\left(\widehat{\boldsymbol{h}}(k-1)\right) \quad (4)$$

where $\rho_{\mathrm{ZA-NLMS}} > 0$ is a small regularization parameter. The work [22] and [23] can be extended to ZA-NLMS in selecting its algorithmic parameters.

l_1-RLS Algorithm: The l_1-RLS algorithm [16] is based on minimizing the following cost function:

$$V(\widehat{\boldsymbol{h}}) = \sum_{s=1}^{k} \lambda^{k-s}|e(k)|^2 + \rho \sum_{i=0}^{L} |\widehat{h}_i| \quad (5)$$

where λ is a forgetting factor that is slightly less than 1, in the range of 0.95–0.99, and $\rho > 0$ is a small regularization

parameter. The parameters are updated using the following equations:

$$\begin{cases} e(k) = y(k) - \boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1) \\ \boldsymbol{P}(k) = \frac{1}{\lambda}\left(\boldsymbol{P}(k-1) - \frac{\boldsymbol{P}(k-1)\boldsymbol{x}^*(k)\boldsymbol{x}^T(k)\boldsymbol{P}(k-1)}{\lambda + \boldsymbol{x}^T(k)\boldsymbol{P}(k-1)\boldsymbol{x}^*(k)}\right) \\ \widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + \boldsymbol{P}(k)\boldsymbol{x}^*(k)e(k) \\ \qquad -\rho(1-\lambda)\boldsymbol{P}(k)\mathrm{sgn}\left(\widehat{\boldsymbol{h}}(k-1)\right) \end{cases}$$

where $\widehat{\boldsymbol{h}}(0)$ is initialized as a zero or a small random vector, and $\boldsymbol{P}(k)$ is the covariance matrix that is initialized to $\boldsymbol{P}(0) = (1/\delta)\boldsymbol{I}$, with δ being a very small positive number.

SPARLS Algorithm: The SPARLS algorithm [14] also uses an l_1 penalty, which modifies a wavelet-based image restoration algorithm [24] into an adaptive filtering setting. It is a probability modeling approach based on the Gaussian noise assumption and the penalized maximum-likelihood estimation. Since this penalized maximum-likelihood estimation problem is hard to solve, the idea in [24] is to decompose the noise into a sum of two Gaussian components, which leads to the use of an iterative EM algorithm as the solver. The advantage of the SPARLS algorithm is that it has guaranteed theoretical convergence. The details of this algorithm can be found in [14].

III. Proposed Zero-Attracting Recursive Least Squares Algorithms

We initially introduce ZA-RLS-I by modifying the conventional RLS to include the l_1-norm sparsity constraint. ZA-RLS-II is then proposed to improve computational efficiency.

A. ZA-RLS-I

Similar to the l_1-RLS algorithm in [16], the two proposed ZA-RLS algorithms are also based on the cost function (5). However, since the cost function (5) does not lend to a closed-form solution, we use the strategy of relaxation by approximating (5) as

$$V_{\mathrm{ZA-RLS}}(\widehat{\boldsymbol{h}}) = \sum_{s=1}^{k} \lambda^{k-s} |e(k)|^2 + \rho \cdot \widehat{\boldsymbol{h}}^H \boldsymbol{D}(k)\widehat{\boldsymbol{h}} \quad (6)$$

where $\boldsymbol{D}(k) = \mathrm{diag}\{d_0(k), d_1(k), \ldots, d_L(k)\}$ with $d_i(k) = 1/(|\widehat{h}_i(k-1)| + \epsilon)$ for $0 \le i \le L$, whereas $\epsilon > 0$ is a very small positive number, e.g., $\epsilon = 10^{-7}$, which is introduced for numerical stability reasons. Denote $\boldsymbol{y}(k) = [y(1)\ y(2) \cdots y(k)]^T$, $\boldsymbol{\Lambda}(k) = \mathrm{diag}\{\lambda^{k-1}, \cdots, \lambda, 1\}$, and

$$\boldsymbol{X}(k) = \begin{bmatrix} \boldsymbol{X}(k-1) \\ \boldsymbol{x}^T(k) \end{bmatrix}$$

with $\boldsymbol{X}(1) = \boldsymbol{x}^T(1) = [x(1)\ 0 \cdots 0]$. The cost function (6) can be equivalently represented as

$$V_{\mathrm{ZA-RLS}}(\widehat{\boldsymbol{h}}) = \left(\boldsymbol{y}(k) - \boldsymbol{X}(k)\widehat{\boldsymbol{h}}\right)^H \boldsymbol{\Lambda}(k)\left(\boldsymbol{y}(k) - \boldsymbol{X}(k)\widehat{\boldsymbol{h}}\right) + \rho \cdot \widehat{\boldsymbol{h}}^H \boldsymbol{D}(k)\widehat{\boldsymbol{h}}. \quad (7)$$

The minimizer of (7) is given by

$$\widehat{\boldsymbol{h}}(k) = \boldsymbol{P}(k)\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{y}(k) \quad (8)$$

where $\boldsymbol{P}(k) = (\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{X}(k) + \rho\boldsymbol{D}(k))^{-1}$. At time index $(k-1)$, (8) is in the form of

$$\widehat{\boldsymbol{h}}(k-1) = \boldsymbol{P}(k-1)\boldsymbol{X}^H(k-1)\boldsymbol{\Lambda}(k-1)\boldsymbol{y}(k-1). \quad (9)$$

It is easy to verify that

$$\boldsymbol{P}^{-1}(k) = \lambda\boldsymbol{P}^{-1}(k-1) + \boldsymbol{x}^*(k)\boldsymbol{x}^T(k) + \rho\left(\boldsymbol{D}(k) - \lambda\boldsymbol{D}(k-1)\right) \quad (10)$$

$$\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{y}(k) = \lambda\boldsymbol{X}^H(k-1)\boldsymbol{\Lambda}(k-1)\boldsymbol{y}(k-1) + \boldsymbol{x}^*(k)y(k). \quad (11)$$

Substituting (9) and (10) into (11) and noting $e(k) = y(k) - \boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1)$ yield

$$\begin{aligned} &\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{y}(k) \\ &= \lambda\boldsymbol{P}^{-1}(k-1)\widehat{\boldsymbol{h}}(k-1) + \boldsymbol{x}^*(k)y(k) \\ &= \left(\boldsymbol{P}^{-1}(k) - \boldsymbol{x}^*(k)\boldsymbol{x}^T(k) - \rho\left(\boldsymbol{D}(k) - \lambda\boldsymbol{D}(k-1)\right)\right) \\ &\quad \times \widehat{\boldsymbol{h}}(k-1) + \boldsymbol{x}^*(k)y(k) \\ &= \boldsymbol{P}^{-1}(k)\widehat{\boldsymbol{h}}(k-1) - \rho\left(\boldsymbol{D}(k) - \lambda\boldsymbol{D}(k-1)\right)\widehat{\boldsymbol{h}}(k-1) \\ &\quad + \boldsymbol{x}^*(k)e(k). \end{aligned} \quad (12)$$

Substituting (12) into (8) leads to the recursive formula for updating the parameter vector, i.e.,

$$\widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) - \rho\boldsymbol{P}(k)\left(\boldsymbol{D}(k) - \lambda\boldsymbol{D}(k-1)\right)\widehat{\boldsymbol{h}}(k-1) + \boldsymbol{P}(k)\boldsymbol{x}^*(k)e(k). \quad (13)$$

We now derive the recursive formula for calculating $\boldsymbol{P}(k)$. Note that by defining $\boldsymbol{Q}^{-1}(k) = \boldsymbol{P}^{-1}(k) - \rho\boldsymbol{D}(k)$, (10) is equivalent to

$$\boldsymbol{Q}^{-1}(k) = \lambda\boldsymbol{Q}^{-1}(k-1) + \boldsymbol{x}^*(k)\boldsymbol{x}^T(k). \quad (14)$$

Using the famous matrix inversion lemma, we have

$$\boldsymbol{Q}(k) = \frac{1}{\lambda}\left(\boldsymbol{Q}(k-1) - \frac{\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)\boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)}{\lambda + \boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)}\right). \quad (15)$$

On the other hand, if we apply the matrix inversion lemma based on

$$\boldsymbol{P}^{-1}(k) = \boldsymbol{Q}^{-1}(k) + \rho\boldsymbol{D}(k) \quad (16)$$

we have

$$\boldsymbol{P}(k) = \boldsymbol{H}(k) - \boldsymbol{H}(k)\left(\boldsymbol{H}(k) + \boldsymbol{Q}(k)\right)^{-1}\boldsymbol{H}(k) \quad (17)$$

where

$$\boldsymbol{H}(k) = (\rho \boldsymbol{D}(k))^{-1} = \mathrm{diag}\left\{ \frac{\left(\left|\widehat{h}_0(k-1)\right|+\epsilon\right)}{\rho}, \ldots, \frac{\left(\left|\widehat{h}_L(k-1)\right|+\epsilon\right)}{\rho} \right\}. \tag{18}$$

We summarize the proposed ZA-RLS-I algorithm in Algorithm 1. Clearly, the ZA-RLS-I algorithm has a higher computational cost than the standard RLS, owing to the fact that the matrix inversion is still needed to calculate $\boldsymbol{P}(k)$. When $\rho = 0$, it reduces to the conventional RLS algorithm since it can be shown that $\boldsymbol{P}(k) = \boldsymbol{Q}(k)(\boldsymbol{H}(k) + \boldsymbol{Q}(k))^{-1}\boldsymbol{H}(k)$, in which the term $(\boldsymbol{H}(k) + \boldsymbol{Q}(k))^{-1}\boldsymbol{H}(k)$ tends to the identity matrix.

Algorithm 1 ZA-RLS-I algorithm

1: Initialize $\widehat{\boldsymbol{h}}(0)$ as a zero or a small random vector. Set $\boldsymbol{Q}(0) = (1/\delta)\boldsymbol{I}$, with δ being a very small positive number. Initialize both $\boldsymbol{D}(0)$ and $\boldsymbol{D}(1)$ as a zero matrix.
2: **for** time step $k = 1, 2, \ldots,$ **do**
3: Calculate

$$\begin{cases} e(k) = y(k) - \boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1) \\ \boldsymbol{H}(k) = \mathrm{diag}\left\{ \frac{\left(\left|\widehat{h}_0(k-1)\right|+\epsilon\right)}{\rho}, \ldots, \frac{\left(\left|\widehat{h}_L(k-1)\right|+\epsilon\right)}{\rho} \right\} \\ \boldsymbol{D}(k) = \mathrm{diag}\left\{ \frac{1}{\left|\widehat{h}_0(k-1)\right|+\epsilon}, \ldots, \frac{1}{\left|\widehat{h}_L(k-1)\right|+\epsilon} \right\}, \text{ for } k>1 \\ \boldsymbol{Q}(k) = \frac{1}{\lambda}\left(\boldsymbol{Q}(k-1) - \frac{\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)\boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)}{\lambda + \boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)} \right) \\ \boldsymbol{P}(k) = \boldsymbol{H}(k) - \boldsymbol{H}(k)\left(\boldsymbol{Q}(k) + \boldsymbol{H}(k)\right)^{-1}\boldsymbol{H}(k) \\ \widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) - \rho\boldsymbol{P}(k)\left(\boldsymbol{D}(k) - \lambda\boldsymbol{D}(k-1)\right)\widehat{\boldsymbol{h}}(k-1) \\ \qquad\qquad + \boldsymbol{P}(k)\boldsymbol{x}^*(k)e(k) \end{cases}$$

4: **end for**

B. ZA-RLS-II

We now propose a more efficient version of ZA-RLS, which is referred to as the ZA-RLS-II algorithm, by exploiting the structural properties of the matrices involved. Note that by denoting $\boldsymbol{Q}^{[\mathrm{inv}]}(k) = \boldsymbol{Q}^{-1}(k)$, (14) can be alternatively represented by

$$\boldsymbol{Q}^{[\mathrm{inv}]}(k) = \lambda\boldsymbol{Q}^{[\mathrm{inv}]}(k-1) + \boldsymbol{x}^*(k)\boldsymbol{x}^T(k). \tag{19}$$

The basic idea of the ZA-RLS-II algorithm is to calculate $\boldsymbol{S}(k) = (\boldsymbol{H}(k) + \boldsymbol{Q}(k))^{-1}$ in terms of $\boldsymbol{Q}^{[\mathrm{inv}]}(k)$, which can be easily updated recursively. The ZA-RLS-II algorithm also makes use of the sparsity property of the channel. Specifically, we note that as a result of the sparse channel, $\boldsymbol{H}(k)$ has a low rank, i.e., it is a diagonal matrix with only a few dominant components. We start with introducing a mathematical theorem [25].

Lemma 1 ([25]): If $\boldsymbol{Q}$ and $\boldsymbol{H}$ are nonsingular square matrices of the same dimension and $\boldsymbol{H}$ has rank one, then

$$(\boldsymbol{Q} + \boldsymbol{H})^{-1} = \boldsymbol{Q}^{-1} - \frac{1}{1+g}\boldsymbol{Q}^{-1}\boldsymbol{H}\boldsymbol{Q}^{-1} \tag{20}$$

where $g = \mathrm{tr}\{\boldsymbol{H}\boldsymbol{Q}^{-1}\}$.

Theorem 1 ([25]): Let $\boldsymbol{Q}$ and $\boldsymbol{H}$ be nonsingular square matrices of the same dimension. Suppose that $\boldsymbol{H}$ has rank $r > 0$ and is decomposed as $\boldsymbol{H} = \sum_{i=1}^{r}\boldsymbol{H}_i$, where each $\boldsymbol{H}_i$ has rank one. Denote $\boldsymbol{Q}_{i+1} = \boldsymbol{Q} + \sum_{j=1}^{i}\boldsymbol{H}_j$ with $\boldsymbol{Q}_1 = \boldsymbol{Q}$. Then, by making use of Lemma 1, we have

$$\boldsymbol{Q}_{i+1}^{-1} = \boldsymbol{Q}_i^{-1} - \frac{1}{1+g_i}\boldsymbol{Q}_i^{-1}\boldsymbol{H}_i\boldsymbol{Q}_i^{-1} \tag{21}$$

where $g_i = \mathrm{tr}\{\boldsymbol{H}_i\boldsymbol{Q}_i^{-1}\}$. In particular

$$(\boldsymbol{Q} + \boldsymbol{H})^{-1} = \boldsymbol{Q}_{r+1}^{-1} = \boldsymbol{Q}_r^{-1} - \frac{1}{1+g_r}\boldsymbol{Q}_r^{-1}\boldsymbol{H}_r\boldsymbol{Q}_r^{-1}. \tag{22}$$

Consider applying Theorem 1 to $\boldsymbol{S}(k) = (\boldsymbol{H}(k) + \boldsymbol{Q}(k))^{-1}$, by decomposing $\boldsymbol{H}(k)$ as a series of r rank-one matrices, where r is the number of nonzero taps in $\widehat{\boldsymbol{h}}(k-1)$. Specifically, at each time step k, we find the integer set ω as

$$\omega = \left\{ i | 0 \le i \le L, \left|\widehat{h}_i(k-1)\right| > \xi \right\} \tag{23}$$

where ξ is a small positive number, e.g., 10^{-3}, and $r = |\omega|$. The elements of ω point to the positions of the nonzero taps in $\widehat{\boldsymbol{h}}(k-1)$. For example, if $r = 2$, $|\widehat{h}_0(k-1)| > \xi$ and $|\widehat{h}_3(k-1)| > \xi$, then $\omega(1) = 0$ and $\omega(2) = 3$. Clearly, we can decompose $\boldsymbol{H}(k) = \sum_{i=1}^{r}\boldsymbol{H}_i(k)$, where $\boldsymbol{H}_i(k)$ has all zero elements except $|\widehat{h}_{\omega(i)}(k-1)|/\rho$ at the diagonal position $(\omega(i)+1, \omega(i)+1)$, which matches a corresponding diagonal value in $\boldsymbol{H}(k)$ that is significantly larger than zero.

Similarly, denoting $\boldsymbol{Q}_i^{[\mathrm{inv}]}(k) = \boldsymbol{Q}_i^{-1}(k)$ with $\boldsymbol{Q}_1^{[\mathrm{inv}]}(k) = \boldsymbol{Q}^{[\mathrm{inv}]}(k)$ and applying Theorem 1 yield

$$\begin{aligned} \boldsymbol{Q}_{i+1}^{[\mathrm{inv}]}(k) &= \boldsymbol{Q}_i^{[\mathrm{inv}]}(k) - \frac{1}{1+g_i(k)}\boldsymbol{Q}_i^{[\mathrm{inv}]}(k)\boldsymbol{H}_i(k)\boldsymbol{Q}_i^{[\mathrm{inv}]}(k) \\ &= \boldsymbol{Q}_i^{[\mathrm{inv}]}(k) - \frac{\left|\widehat{h}_i(k-1)\right|}{\rho + \left|\widehat{h}_i(k-1)\right|\widetilde{q}_i(k)}\widetilde{\boldsymbol{q}}_i(k)\widetilde{\boldsymbol{q}}_i^H(k) \end{aligned} \tag{24}$$

for $i = 1, 2, \ldots, r$, in which $g_i(k) = \mathrm{tr}\{\boldsymbol{H}_i(k)\boldsymbol{Q}_i^{[\mathrm{inv}]}(k)\}$, $\widetilde{q}_i(k)$ is the $(\omega(i)+1)$th diagonal element of $\boldsymbol{Q}_i^{[\mathrm{inv}]}(k)$, and $\widetilde{\boldsymbol{q}}_i(k)$ is the $(\omega(i)+1)$th column of $\boldsymbol{Q}_i^{[\mathrm{inv}]}(k)$. Note that we have $\boldsymbol{S}(k) = \boldsymbol{Q}_{r+1}^{[\mathrm{inv}]}(k)$.

We summarize the proposed ZA-RLS-II algorithm in Algorithm 2.

Algorithm 2 ZA-RLS-II algorithm

1: Initialize $\widehat{\boldsymbol{h}}(0)$ as a zero or a small random vector. Set $\boldsymbol{Q}(0)=(1/\delta)\boldsymbol{I}$, with δ being a very small positive number, and set $\boldsymbol{Q}^{[\mathrm{inv}]}(0)=\delta\boldsymbol{I}$. Initialize both $\boldsymbol{D}(0)$ and $\boldsymbol{D}(1)$ as a zero matrix.
2: **for** time step $k=1,2,\ldots,$ **do**
3: Calculate

$$\begin{cases} e(k)=y(k)-\boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1) \\ \boldsymbol{H}(k)=\mathrm{diag}\left\{\frac{(|\widehat{h}_0(k-1)|+\epsilon)}{\rho},\ldots,\frac{(|\widehat{h}_L(k-1)|+\epsilon)}{\rho}\right\} \\ \boldsymbol{D}(k)=\mathrm{diag}\left\{\frac{1}{|\widehat{h}_0(k-1)|+\epsilon},\ldots,\frac{1}{|\widehat{h}_L(k-1)|+\epsilon}\right\},\ \text{for}\ k>1 \\ \boldsymbol{Q}^{[\mathrm{inv}]}(k)=\lambda\boldsymbol{Q}^{[\mathrm{inv}]}(k-1)+\boldsymbol{x}^*(k)\boldsymbol{x}^T(k) \\ \boldsymbol{Q}(k)=\frac{1}{\lambda}\left(\boldsymbol{Q}(k-1)-\frac{\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)\boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)}{\lambda+\boldsymbol{x}^T(k)\boldsymbol{Q}(k-1)\boldsymbol{x}^*(k)}\right). \end{cases}$$

4: Given $\boldsymbol{Q}^{[\mathrm{inv}]}(k)$, $\widehat{\boldsymbol{h}}(k-1)$, ξ, find ω according to (23), set $r=|\omega|$ and set $\boldsymbol{Q}_1^{[\mathrm{inv}]}(k)=\boldsymbol{Q}^{[\mathrm{inv}]}(k)$.
5: **for** $i=1,2,\ldots,r,$ **do**
6: Calculate

$$\boldsymbol{Q}_{i+1}^{[\mathrm{inv}]}(k)=\boldsymbol{Q}_i^{[\mathrm{inv}]}(k)-\frac{\left|\widehat{h}_i(k-1)\right|}{\rho+\left|\widehat{h}_i(k-1)\right|\widetilde{q}_i(k)}\widetilde{\boldsymbol{q}}_i(k)\widetilde{\boldsymbol{q}}_i^H(k).$$

7: **end for**
8: $\boldsymbol{S}(k)=\boldsymbol{Q}_{r+1}^{[\mathrm{inv}]}(k)$.
9: Calculate

$$\begin{cases} \boldsymbol{P}(k)=\boldsymbol{H}(k)-\boldsymbol{H}(k)\boldsymbol{S}(k)\boldsymbol{H}(k) \\ \widehat{\boldsymbol{h}}(k)=\widehat{\boldsymbol{h}}(k-1)-\rho\boldsymbol{P}(k) \\ \qquad\times\left(\boldsymbol{D}(k)-\lambda\boldsymbol{D}(k-1)\right)\widehat{\boldsymbol{h}}(k-1)+\boldsymbol{P}(k)\boldsymbol{x}^*(k)e(k) \end{cases}$$

10: **end for**

C. Convergence Analysis

The exponential convergence of the standard RLS with an exponential forgetting factor was studied in [26], which focuses on the "homogeneous" case that if $y(k)$ is exactly given as $\boldsymbol{x}^T(k)\boldsymbol{h}$, then $\widehat{\boldsymbol{h}}(k)$ converges to $\boldsymbol{h}$ exponentially fast, provided that $\boldsymbol{x}(k)$ is persistently exciting. Similarly, for the proposed algorithm, if we define a parameter estimation error vector, i.e.,

$$\widetilde{\boldsymbol{h}}(k)=\boldsymbol{h}-\widehat{\boldsymbol{h}}(k) \tag{25}$$

and consider the "homogeneous" case for (13) with $y(k)=\boldsymbol{x}^T(k)\boldsymbol{h}$, then

$$\widetilde{\boldsymbol{h}}(k)=\left(\boldsymbol{I}-\rho\boldsymbol{P}(k)\left(\boldsymbol{D}(k)-\lambda\boldsymbol{D}(k-1)\right)-\boldsymbol{P}(k)\boldsymbol{x}^*(k)\boldsymbol{x}^T(k)\right)\times\widetilde{\boldsymbol{h}}(k-1)+\rho\boldsymbol{P}(k)\left(\boldsymbol{D}(k)-\lambda\boldsymbol{D}(k-1)\right)\boldsymbol{h}. \tag{26}$$

We are ready to provide the exponential convergence of the proposed ZA-RLS algorithms.

Theorem 2: If $\boldsymbol{P}(k)$ is invertible, then the ZA-RLS algorithms are exponentially stable.

Proof: We choose a Lyapunov function as

$$\mathcal{V}(k)=\left(\boldsymbol{P}^{-1}(k)\widetilde{\boldsymbol{h}}(k)-\rho\boldsymbol{D}(k)\boldsymbol{h}\right)^H\left(\boldsymbol{P}^{-1}(k)\widetilde{\boldsymbol{h}}(k)-\rho\boldsymbol{D}(k)\boldsymbol{h}\right)>0. \tag{27}$$

Substituting (10) into (26) yields

$$\widetilde{\boldsymbol{h}}(k)=\lambda\boldsymbol{P}(k)\boldsymbol{P}^{-1}(k-1)\widetilde{\boldsymbol{h}}(k-1)+\rho\boldsymbol{P}(k)\left(\boldsymbol{D}(k)-\lambda\boldsymbol{D}(k-1)\right)\boldsymbol{h}. \tag{28}$$

Substituting (28) into the Lyapunov function (27) results in

$$\mathcal{V}(k)-\mathcal{V}(k-1)=(\lambda^2-1)\mathcal{V}(k-1)<0 \tag{29}$$

$$\mathcal{V}(k)<\lambda\mathcal{V}(k-1)<\cdots<\lambda^k\mathcal{V}(0). \tag{30}$$

This proves the exponential convergence of the ZA-RLS algorithms. Finally, we conclude from $\boldsymbol{P}^{-1}(k)\widetilde{\boldsymbol{h}}(k)-\rho\boldsymbol{D}(k)\boldsymbol{h}\to\boldsymbol{0}$ that

$$\widehat{\boldsymbol{h}}(k)\to\left(\boldsymbol{I}-\rho\boldsymbol{P}(k)\boldsymbol{D}(k)\right)\boldsymbol{h}=\left(\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{X}(k)+\rho\boldsymbol{D}(k)\right)^{-1}\boldsymbol{X}^H(k)\boldsymbol{\Lambda}(k)\boldsymbol{X}(k)\boldsymbol{h} \tag{31}$$

exponentially fast. ■

Remark 1: The rank-1 updates are a standard way of building the inversion of the covariance matrix, e.g., used in deriving RLS algorithms. The process can be unstable for ill-conditioned data sets. However, in the proposed ZA-RLS algorithms, $\boldsymbol{P}(k)$ is always invertible due to regularization. Specifically, observe that the rank-1 updates (24) are guaranteed to be well conditioned owing to the regularization $\rho>0$. In fact, the regularization introduced ensures that our ZA-RLS algorithms have even better numerical stability than the standard RLS algorithm. The standard RLS algorithm is, of course, well known to be numerically stable under typical floating-point implementation that has sufficient precision.

Remark 2: From (31), clearly, $\widehat{\boldsymbol{h}}(k)$ is a biased estimator of $\boldsymbol{h}$ for nonzero ρ. However, if $\rho=0$, the parameter estimate will have a large variance due to the ill condition of the covariance matrix, particularly for sparse channels. In statistical estimation theory, this is well known as the bias and variance tradeoff in choosing the regularization parameter ρ. Note that the regularization parameter selection criteria for the LMS algorithm [22], [23] are not applicable to RLS-based algorithms that have much better performance than LMS-based algorithms. In this paper, we empirically choose appropriate values of ρ with respect to the system's signal-to-noise ratio (SNR) conditions. It is highly desired to investigate how to choose ρ optimally by minimizing the estimation error $\|\widetilde{\boldsymbol{h}}(k)\|^2$, which is defined as the squared norm of $\widetilde{\boldsymbol{h}}(k)$, using, for example, approximate Bayesian models. This will be our future study.

D. Computational Complexity Analysis

The key difference between ZA-RLS-I and ZA-RLS-II algorithms lies in how to calculate $\boldsymbol{P}(k)$. Each recursive step of the ZA-RLS-I algorithm has a computational cost on the order of $\mathsf{O}((L+1)^3)$, but for the ZA-RLS-II algorithm, this is reduced to a computational cost on the order of $\mathsf{O}(r(L+1)^2)$, where r is the estimated sparsity level at each recursion. This is because the computation in Algorithm 2 is essentially a series of r rank-1 updates, each with complexity on the order of $\mathsf{O}((L+1)^2)$. Moreover, r varies during the recursive updating procedure. However, since $r \ll L+1$ all the time, the complexity of the ZA-RLS-II algorithm is approximately on the order of $\mathsf{O}((L+1)^2)$. This is because the underlying system that we consider is very sparse, with the true sparsity level much smaller than $L+1$. Based on the standard initialization of $\widehat{\boldsymbol{h}}(0)$, i.e., setting the elements of $\widehat{\boldsymbol{h}}(0)$ to zero or, randomly, to small values, which is the most widely adopted initialization for recursive identification, it is most unlikely that r at any recursion step will ever reach a high value close to $L+1$. Rather, it is most likely that r will remain to be much smaller than $L+1$ all the time. In the simulation study, we will verify this analysis.

Thus, the computational cost of the ZA-RLS-II algorithm is only slightly higher than that of the standard RLS algorithm, which also has complexity on the order of $\mathsf{O}((L+1)^2)$. More specifically, at each recursive step, in addition to update/store the matrix variable $\boldsymbol{Q}(k)$ as also required by the conventional RLS algorithm, the ZA-RLS-II algorithm is required to update/store $\boldsymbol{Q}^{[\mathrm{inv}]}(k)$, but this only involves a negligible additional computational cost.

It is also obvious that the proposed ZA-RLS-II algorithm, the l_1-RLS algorithm [16], and the SPARLS algorithm [14] all have similar computational complexity, on the order of $\mathsf{O}((L+1)^2)$ at each recursive step.

IV. Simulation Study

Simulations were carried out with the channel input signal $x(k)$ taking values from the $M=64$ quadrature amplitude modulation symbol set. The transmit signal was normalized to have a unity average symbol energy, and thus, the average energy per bit was given by $E_b = 1/\log_2(M) = 1/6$. The received SNR was defined as $10\log_{10}(E_b/N_o)$. The sparse multipath channel had a length of 100, i.e., $L=99$, with its true sparsity level denoted by r_{true}, which was the number of the nonzero elements in $\boldsymbol{h}$. Based on the average results over 100 random trials, we tested the performance of the proposed algorithms in three aspects: 1) the performance comparison with some other well-known l_1-norm-based adaptive algorithms; 2) the effects of the sparsity levels on the proposed sparse algorithms; and 3) the effects of the regularization parameter with respect to the SNR. For each trial, the training data length was set to 1000. The positions of the significant r_{true} taps were randomly selected within the channel length. The CIR $\boldsymbol{h}$ was kept constant in the first half of the each trial, and it suddenly changed at the beginning of the second half of the each trial in terms of both the tap positions and values. The values of channel taps followed the Gaussian distribution and were normalized to $\|\boldsymbol{h}\|^2 = 1$. The estimation performance was evaluated based on the average mean absolute deviation (MAD), which is defined as

$$\mathrm{MAD}\left\{\widehat{\boldsymbol{h}}(k)\right\} = E\left\{\left|\widehat{\boldsymbol{h}}(k)-\boldsymbol{h}\right|\right\} = E\left\{\sum_{i=0}^{L}\left|\widehat{h}_i(k)-h_i\right|\right\} \tag{32}$$

and calculated based on the average value over 100 independent random trials.

Comparison With Other l_1-Norm-Based Complex-Valued Sparse Adaptive Algorithms: First, we compared the proposed algorithms with several known l_1-norm-based algorithms under the conditions of SNR = 15 dB and 30 dB, whereas the number of significant taps was fixed to $r_{\mathrm{true}} = 10$. Specifically, the proposed ZA-RLS-I and ZA-RLS-II algorithms were compared with the ZA-LMS, the ZA-NLMS, the conventional RLS, the SPARLS algorithm [14], the l_1-RLS algorithm [16], and the oracle-RLS algorithm, which is just the ordinary RLS given the nonzero tap locations. The oracle-RLS algorithm obviously will attain the best performance, but it is impractical as the nonzero tap locations are unknown. The reasons for choosing the SPARLS algorithm [14] and the l_1-RLS algorithm [16] as baseline algorithms are that they are both based on the l_1 cost function with different algorithm designs and that they have the same computational complexity as the proposed ZA-RLS-II. Moreover, all these algorithms are designed for complex-valued channel identification.

For the ZA-LMS algorithm, we set $\mu = 0.01$, and $\rho_{\mathrm{ZA-LMS}} = 0.01$, whereas for the ZA-NLMS algorithm, we set $\mu = 0.8$, and $\rho_{\mathrm{ZA-LMS}} = 0.01$, since these values were found to give the best results for the respective algorithms. For all the RLS-based algorithms, the forgetting factor $\lambda = 0.975$. We also set $\xi = 0.0001$ in ZA-RLS-II. The parameters used in the SPARLS algorithm [14] and the l_1-RLS algorithm [16] were empirically tuned to give the best performance possible. In addition, in the case of the l_1-RLS algorithm, the performance was very bad at the beginning of the data set; hence, the ordinary RLS algorithm was used for the first 120 data points. For the ZA-RLS-I and ZA-RLS-II algorithms, we set $\rho = 0.5$ for SNR = 15 dB and $\rho = 0.1$ for SNR = 30 dB.

The MAD results obtained by various adaptive algorithms are compared in Fig. 1(a) and (b), respectively, for the two SNR settings. As expected, all the RLS-based algorithms attain much better MAD performance than the ZA-LMS and ZA-NLMS algorithms. It is also seen that the results of the two proposed algorithms are identical, and they significantly outperform the l_1-RLS algorithm. Fig. 1 also shows that the ZA-RLS-II algorithm achieves a smaller steady-state error with a faster convergence rate, compared with the SPARLS algorithm. These results are significant, particularly considering that the ZA-RLS-II algorithm has similar computational complexity as the l_1-RLS algorithm and the SPARLS algorithm. In Fig. 2, we plot the estimated sparsity level r as recoded by the ZA-RLS-II algorithm at each recursive step, averaged over the 100 independent random trials. It can be observed in Fig. 2 that $r \ll L+1$ all the time.

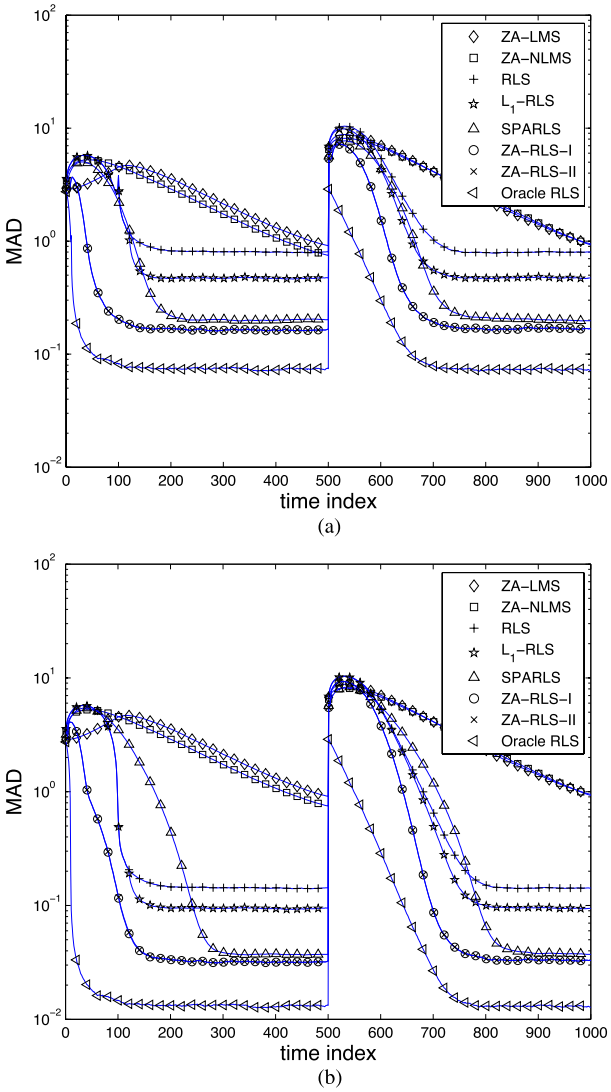

Fig. 1. Comparison of the MADs of the parameter estimates for various adaptive algorithms. (a) SNR = 15 dB and (b) SNR = 30 dB. The channel input signal is complex valued, and the channel sparsity level $r_{\text{true}} = 10$.

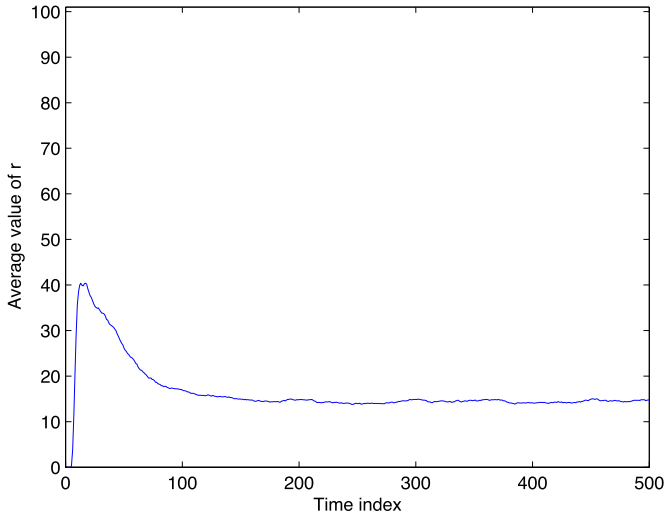

Fig. 2. Estimated sparsity level r as recorded by the ZA-RLS-II algorithm at each recursive step, averaged over 100 runs. The channel input signal is complex valued, the channel length $L + 1 = 100$, the channel sparsity level $r_{\text{true}} = 10$, and SNR = 15 dB.

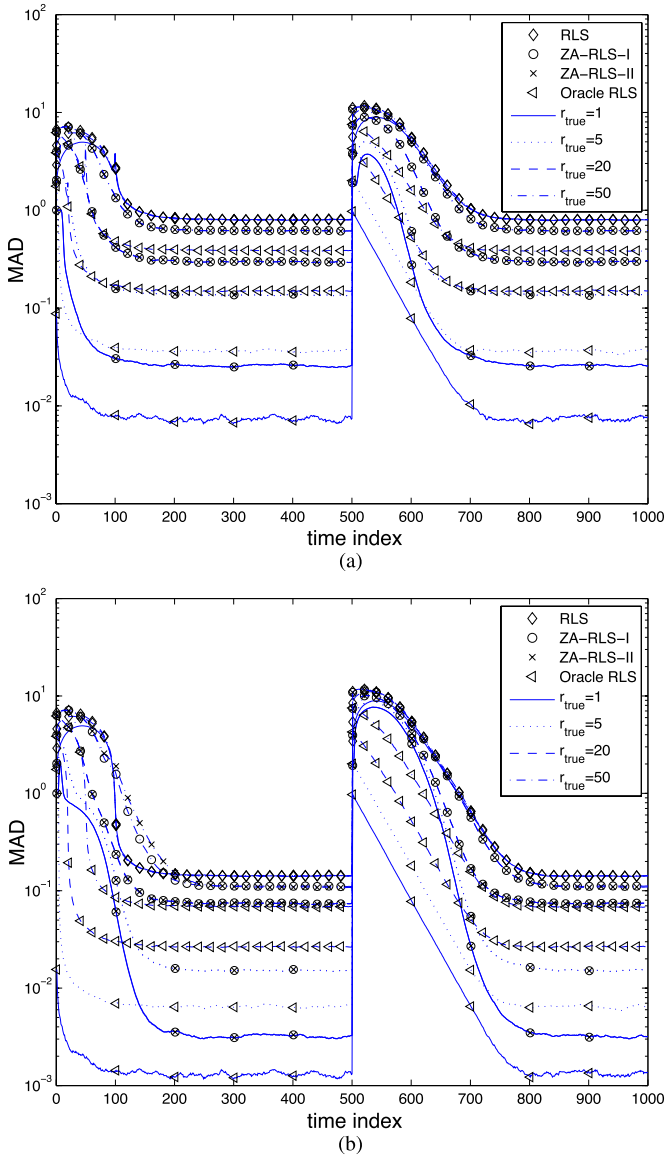

Fig. 3. Comparison of the MADs of the parameter estimates for various sparsity levels. (a) SNR = 15 dB and (b) SNR = 30 dB. The channel input signal is complex valued.

Performance Evaluation for Various Sparsity Levels: To investigate the performance of the proposed algorithms for different sparsity levels, we further experimented by changing the number of significant taps r_{true}, also under the conditions of SNR = 15 dB and 30 dB. Specifically, $r_{\text{true}} = 1$, 5, 20, and 50 was experimented. For clarity, only the oracle-RLS and ordinary RLS algorithms were used for comparison, and the results obtained were plotted in Fig. 3(a) and (b) for the two SNR settings, respectively. Since the channel taps were normalized in spite of different sparsity levels, we used different values of ρ as appropriate for ZA-RLS-I and ZA-RLS-II. Specifically, when SNR = 30 dB, we empirically set $\rho = 0.1$ for $r_{\text{true}} = 1$, 5, and 20, but $\rho = 0.01$ for $r_{\text{true}} = 50$. When SNR = 15 dB, we empirically found $\rho = 1$ for $r_{\text{true}} = 1$ and 5, $\rho = 0.3$ for $r_{\text{true}} = 20$, whereas $\rho = 0.05$ for $r_{\text{true}} = 50$. The results in Fig. 3 clearly show that it is most beneficial to use the sparse adaptive algorithms when the sparsity level is high, i.e., the

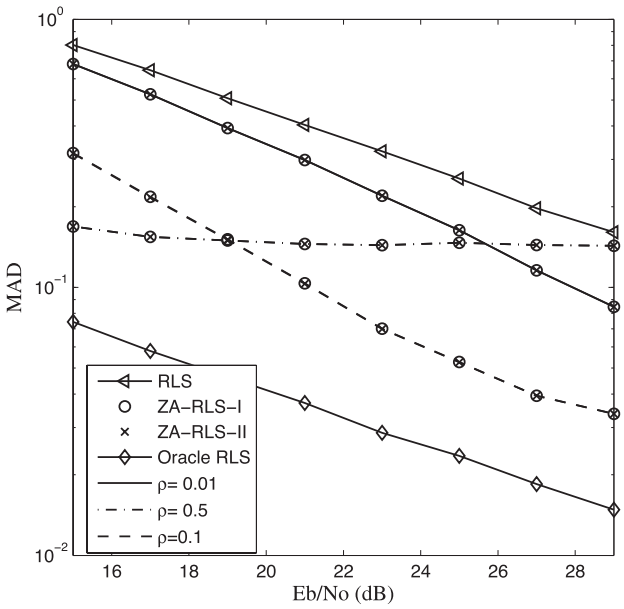

Fig. 4. Comparison of the steady-state MADs of the parameter estimates as functions of SNR. The channel input signal is complex valued, the channel sparsity level $r_{\rm true} = 10$, and three different regularization parameter values are tested for the proposed ZA-RLS algorithms.

value of $r_{\rm true}$ is small. As $r_{\rm true}$ increases to the full channel length, the oracle-RLS algorithm becomes the ordinary RLS algorithm. In this case, there exists no sparsity to be exploited, and ρ should be chosen as a very small positive number close to zero just for numerical stability considerations.

Performance Evaluation for Different Regularization Parameter and SNR Settings: To study the effects of ρ with respect to SNR levels for a fixed sparsity level of $r_{\rm true} = 10$, we recorded the results of MAD$\{\widehat{\boldsymbol{h}}(300)\}$ as the functions of SNR in Fig. 4, based on $\rho = 0.5$, $\rho = 0.1$, and $\rho = 0.01$, respectively. It is shown in Fig. 4 that for the proposed ZA-RLS algorithms, a relatively larger ρ should be used under a high-noise condition, whereas when the noise level is low, ρ should be set relatively small. However, a too small ρ value leads to poor performance.

Comparison With the l_1-Norm-Based Real-Valued Sparse Adaptive Algorithms in [19]: The l_1-norm-based sparse RLS algorithms in [19] are designed for real-valued signals, and they cannot be directly applied to our application of sparse channel identification involving complex-valued signals. By contrast, our algorithm and the other algorithms compared in the above experiment can deal with both real-valued and complex-valued signals. To compare our algorithm with the algorithms in [19], we specifically design an application involving real-valued signals. The experiment is the same as the experiment in Fig. 1, but the input signal $x(k)$ is changed to be real valued and is generated as a uniformly and randomly distributed signal in [0, 1]. The variance of the noise is set to 0.05, and the resulting SNR is approximately 21 dB. Note that the two online algorithms mentioned in [19], i.e., OCCD-TWL and OCCD-TNWL, only OCCD-TWL is explicitly derived in [19]. Since the work in [19] does not provide how OCCD-TNWL is actually realized, we implement OCCD-TNWL based on our understanding from the offline TNWL algorithm given in [19].

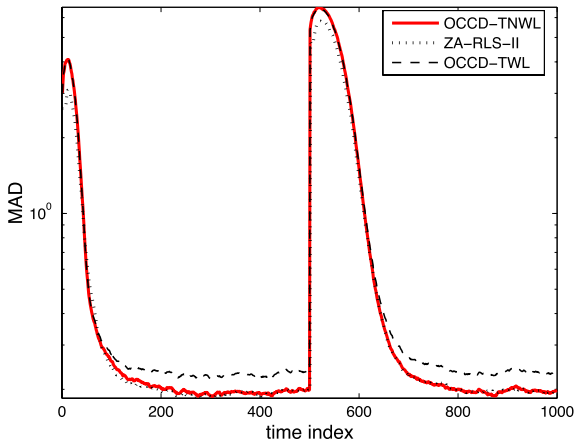

Fig. 5. Comparison of the MADs of the parameter estimates for various adaptive algorithms. The channel input signal is real valued, the channel sparsity level $r_{\rm true} = 10$, and SNR $= 21$ dB.

The OCCD-TNWL algorithm is much more complicated than the OCCD-TWL algorithm. Specifically, at each recursion, the key adaptive parameter of the algorithm is weighted, and the weighting factor depends on the full RLS channel estimate. Therefore, an additional full RLS algorithm is required to run in parallel to provide the full channel estimate at each recursion.

The MAD performances of our ZA-RLS-II, OCCD-TWL, and OCCD-TNWL are compared in Fig. 5, where it is shown that both ZA-RLS-II and OCCD-TNWL outperform OCCD-TWL. It is also shown in Fig. 5 that our ZA-RLS-II and OCCD-TNWL achieve the same steady-state performance, but our ZA-RLS-II has an additional advantage of having slightly better initial transition performance.

V. Conclusion

In this paper, we have introduced two ZA-RLS algorithms for the sparse channel identification problem by using the l_1-norm sparsity constraint adaptively. The basic idea in achieving a closed-form solution is to use an adaptively weighted l_2-norm of the parameter vector term to approximate the l_1-norm of the parameter vector in which the weighting factors are readily calculated as the inversion of the associated l_1-norm of the parameter estimates. As a variant of ZA-RLS-I, the ZA-RLS-II algorithm has focused on improving the computational efficiency by exploiting the channel sparsity and matrix theory. Consequently, the computational complexity of the ZA-RLS-II algorithm is only slightly higher than that of the standard RLS algorithm. The proposed ZA-RLS-II algorithm is compared with a number of adaptive algorithms that also use l_1-norm sparsity constraints, and the simulation results have demonstrated that the proposed ZA-RLS approach is highly effective in real-time sparse channel estimation. In particular, it has been shown that the proposed ZA-RLS-II algorithm outperforms the existing l_1-RLS and SPARLS algorithms that have similar computational complexity and also use the same l_1 cost function but with different approximations for algorithm design. Our future work will study an efficient tuning algorithm for optimizing the regularization parameter.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive comments that helped improve this paper. They would also like to thank B. Babadi for providing the SPARLS MATLAB code.

REFERENCES

[1] S. Haykins, *Adaptive Filter Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.
[2] D. L. Donoho and M. Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via l_1 minimization," *Proc. Nat. Acad. Sci. USA*, vol. 100, no. 5, pp. 2197–2202, Mar. 2003.
[3] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.
[4] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.
[5] D. Needell, J. A. Tropp, and R. Vershynin, "Greedy signal recovery review," in *Proc. 42nd Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, Oct. 26–29, 2008, pp. 1048–1050.
[6] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmonic Anal.*, vol. 53, no. 12, pp. 301–321, May 2009.
[7] T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 298–309, Apr. 2010.
[8] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM J. Numer. Anal.*, vol. 49, no. 6, pp. 2543–2563, 2011.
[9] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. IEEE Inf. Theory Workshop*, Cairo, Egypt, Jan. 6–8, 2010, pp. 1–5.
[10] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for system identification," in *Proc. ICASSP*, Taipei, China, Apr. 19–24, 2009, pp. 3125–3128.
[11] O. Taheri and S. A. Vorobyov, "Sparse channel estimation with l_p-norm and reweighed l_1-norm penalized least mean squares," in *Proc. ICASSP*, Prague, Czech Republic, May 22–27, 2011, pp. 2864–2867.
[12] G. Gui and F. Adachi, "Improved least mean square algorithm with application to adaptive sparse channel estimation," *EURASIP J. Wireless Commun. Netw.*, vol. 204, pp. 1–18, 2013.
[13] Y. T. Gu, J. Jin, and S. Mei, "l_0-norm constraint LMS algorithm algorithm for sparse system identification," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 774–777, Sep. 2009.
[14] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The sparse RLS algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 8, pp. 4013–4025, Aug. 2010.
[15] B. Dumitrescu, A. Onose, P. Helin, and I. Tabus, "Greedy sparse RLS," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2194–2207, May 2012.
[16] E. M. Eksioglu and A. Korhan, "RLS algorithm with convex regularization," *IEEE Signal Process. Lett.*, vol. 18, no. 8, pp. 470–473, Aug. 2011.
[17] E. M. Eksioglu, "Sparsity regularised recursive least squares adaptive filtering," *IET Signal Process.*, vol. 5, no. 5, pp. 480–487, Aug. 2011.
[18] Y. Zakharov and V. H. Nascimento, "Sparse RLS adaptive filter with diagonal loading," in *Proc. 46th Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, USA, Nov. 4–7, 2012, pp. 806–810.
[19] 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, Jul. 2010.
[20] Z. Liu, Y. Liu, and C. Li, "Distributed sparse recursive least-squares over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 6, pp. 1386–1395, Mar. 2014.
[21] Y. V. 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, Jun. 2013.
[22] K. Shi and P. Shi, "Convergence analysis of sparse LMS algorithms with l_1-norm penalty based on white input signal," *Signal Process.*, vol. 90, no. 12, pp. 3289–3293, Dec. 2010.
[23] Y. Liu, C. Li, and Z. Zhang, "Diffusion sparse least-mean squares over networks," *IEEE Trans. Signal Process.*, vol. 90, no. 8, pp. 4480–4485, Aug. 2012.
[24] M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 906–916, Aug. 2003.
[25] K. S. Miller, "On the inverse of the sum of matrices," *Math. Mag.*, vol. 54, no. 2, pp. 67–72, 1981.
[26] R. M. Johnstone, C. R. Johnson, R. R. Bitmead, and B. D. O. Anderson, "Exponential convergence of recursive least squares with exponential forgetting factor," in *Proc. IEEE 21st Conf. Decision Control*, Orlando, FL, USA, Dec. 8–10, 1982, pp. 994–997.

Xia Hong (SM'02) received the B.Sc. and M.Sc. degrees from National University of Defense Technology, Changsha, China, in 1984 and 1987, respectively, and the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 1998, all in automatic control.

During 1987–1993, she was a Research Assistant with Beijing Institute of Systems Engineering, Beijing, China. During 1997–2001, she was a Research Fellow with the Department of Electronics and Computer Science, University of Southampton, Southampton, U.K. She is currently a Professor with the Department of Computer Science, School of Mathematical and Physical Sciences, University of Reading, Reading, U.K. She has published over 200 research papers and coauthored a research book. Her research interests include nonlinear systems identification, data modeling, estimation and intelligent control, neural networks, pattern recognition, learning theory, and their applications.

Dr. Hong received a Donald Julius Groen Prize from the Institution of Mechanical Engineers in 1999.

Junbin Gao received the B.Sc. degree in computational mathematics from Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982 and the Ph.D. degree from Dalian University of Technology, Dalian, China, in 1991.

He is a Professor of big data analytics with The University of Sydney Business School, The University of Sydney, Sydney, NSW, Australia. Prior to 2016, he was a Professor in computer science with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW. From 2001 to 2005, he was a Lecturer and a Senior Lecturer in computer science with the University of New England, Armidale, NSW. From 1982 to 2001, he was an Associate Lecturer, a Lecturer, an Associate Professor, and a Professor with the Department of Mathematics, HUST. His main research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.

Sheng Chen (M'90–SM'97–F'08) received the B.Eng. degree in control engineering from the East China Petroleum Institute, Dongying, China, in 1982; the Ph.D. degree in control engineering from City University London, London, U.K., in 1986; and the D.Sc. degree from the University of Southampton, Southampton, U.K., in 2005.

From 1986 to 1999, he held research and academic appointments with the University of Sheffield, Sheffield, U.K.; the University of Edinburgh, Edinburgh, U.K.; and the University of Portsmouth, Portsmouth, U.K. Since 1999, he has been with the Department of Electronics and Computer Science, University of Southampton, where he is currently a Professor of intelligent systems and signal processing. He has published over 550 research papers. His research interests include adaptive signal processing, wireless communications, modeling and identification of nonlinear systems, neural network and machine learning, intelligent control system design, evolutionary computation methods, and optimization.

Dr. Chen is a Fellow of the Institution of Engineering and Technology, a Distinguished Adjunct Professor with King Abdulaziz University, Jeddah, Saudi Arabia, and an Institute for Scientific Information (ISI) Highly Cited Researcher in engineering (March 2004). He was an elected Fellow of the United Kingdom Royal Academy of Engineering in 2014.