SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Supplementary file for 'A Recursive Least Squares Algorithm with ℓ_1 Regularization for Sparse Representation'

Di LIU[1,2], Simone BALDI[1,3], Quan LIU[4] & Wenwu YU[1,3]

[1]*School of Cyber Science and Engineering, Southeast University, Nanjing* 210096*, China;*
[2] *School of Computation, Information and Technology, Technical University of Munich, Garching* 85748*, Germany;*
[3]*School of Mathematics, Center for Mobile Communication and Security, Southeast University, Nanjing* 210096*, China;*
[4]*School of Artificial Intelligence, Southeast University, Suzhou* 215100*, China*

Appendix A: Derivation of the proposed ℓ_1^2-RLS method

Consider an input-output parameter estimation setting given by the standard relation:

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

where $\boldsymbol{h} = [h_0\, h_1 \cdots h_{N-1}]^T$ denotes the unknown system vector to be estimated, $\boldsymbol{x}(k) = [x_0\, x_1 \cdots x_{N-1}]^T$ is the input vector signal, $y(k)$ is the output signal, $n(k)$ is the measurement noise and k denotes the time index. The system is sparse when only a few elements of $\boldsymbol{h}$ are non-zero. The estimation goal is to provide an estimate of $\boldsymbol{h}$ at time k (call it $\widehat{\boldsymbol{h}}(k)$) by using input and output signals collected up to time k. The estimation is recursive when the estimate $\widehat{\boldsymbol{h}}(k)$ is updated online when new data $\boldsymbol{x}(k)$, $y(k)$ are collected at time k.

To formulate the recursive estimation problem, define the following cost function associated with a least squares problem with ℓ_1-regularization

$$J_N(k) = \frac{1}{2}\sum_{i=1}^{k}\big(y(i) - \widehat{\boldsymbol{h}}^T(k)\boldsymbol{x}(i)\big)^2 + \frac{1}{2}(\widehat{\boldsymbol{h}}^T(k) - \widehat{\boldsymbol{h}}^T(0))P^{-1}(0)(\widehat{\boldsymbol{h}}(k) - \widehat{\boldsymbol{h}}(0)) + \frac{\rho}{2}||\widehat{\boldsymbol{h}}(k)||_1^2 \tag{2}$$

where $\widehat{\boldsymbol{h}}(0)$ is an initial estimate of $\boldsymbol{h}$, $P(0)$ is an initial covariance matrix used for initialization, and $\rho > 0$ is the regularizing parameter. Note that (2) contains the square of the ℓ_1-norm, which is consistent with the presence of the square of the ℓ_2-norm in Tikhonov regularization, whose cost can be written as

$$J_N(k) = \frac{1}{2}\sum_{i=1}^{k}\big(y(i) - \widehat{\boldsymbol{h}}^T(k)\boldsymbol{x}(i)\big)^2 + \frac{1}{2}(\widehat{\boldsymbol{h}}^T(k) - \widehat{\boldsymbol{h}}^T(0))P^{-1}(0)(\widehat{\boldsymbol{h}}(k) - \widehat{\boldsymbol{h}}(0)) + \frac{\rho}{2}||\widehat{\boldsymbol{h}}(k)||_2^2$$

with the ℓ_2-norm instead of the ℓ_1-norm. In this sense, the parameters $P^{-1}(0)$ and ρ in (2) play the same role as $P^{-1}(0)$ and ρ in Tikhonov regularization[1)]. A large value of $P^{-1}(0)$ gives more weight to minimize the l_2 norm of $\widehat{\boldsymbol{h}}$, and a large value of ρ gives more weight to minimize the l_1 norm of $\widehat{\boldsymbol{h}}$. However, if these parameters are chosen very large, less weight will be given to the estimation error in the first term of eq. (2), which will inevitably become worse. Therefore, in practice, it is up to the designer to

* Corresponding author (email:)

1) It is worth mentioning that the second term of the cost function $(\widehat{\boldsymbol{h}}^T(k) - \widehat{\boldsymbol{h}}^T(0))P^{-1}(0)(\widehat{\boldsymbol{h}}(k) - \widehat{\boldsymbol{h}}(0))$ is often omitted in standard Tikhonov regularization, in favour of the term $\rho||\widehat{\boldsymbol{h}}(k)||_2^2$. Mathematically speaking, such term should not be omitted, because it is needed to correctly initialize the recursive equations with $\widehat{\boldsymbol{h}}(0)$ and $P^{-1}(0)$ [1]. If the second term is omitted in favour of $\rho||\widehat{\boldsymbol{h}}(k)||_2^2$, then one should initialize $\widehat{\boldsymbol{h}}(0) = 0$ and $P(0) = \rho^{-1}I$.

tune the parameters to find a good trade-off between having small norms of $\widehat{\boldsymbol{h}}$ (i.e. sparsity) and having a good estimation.

We will now aim to find the vector of coefficients which minimizes the cost function (2) in a recursive manner. Let us denote by $\boldsymbol{X}(k) = [\boldsymbol{x}(k)\ \boldsymbol{x}(k-1)\cdots\boldsymbol{x}(1)]^T$ the collection of past inputs, and by $\boldsymbol{Y}(k) = [y(k)\ y(k-1)\cdots y(1)]^T$ the collection of past outputs. Let $\widehat{\boldsymbol{h}}(k)$ denote the estimate of vector $\boldsymbol{h}$ at time k. This estimate is optimal when it is the vector that makes the gradient of the cost function (2) equal to zero:

$$\nabla J_N(k) = -\boldsymbol{X}^T(k)\big(\boldsymbol{Y}(k) - \boldsymbol{X}(k)\widehat{\boldsymbol{h}}(k)\big) + P^{-1}(0)(\widehat{\boldsymbol{h}}(k) - \widehat{\boldsymbol{h}}(0)) + \rho\nabla||\widehat{\boldsymbol{h}}(k)||_1^2 = 0 \tag{3}$$

where $\nabla||\widehat{\boldsymbol{h}}(k)||_1^2$ indicates one of possible subgradients of the nonsmooth function $||\widehat{\boldsymbol{h}}(k)||_1^2$, which in this work is taken as

$$\nabla||\widehat{\boldsymbol{h}}(k)||_1^2 = \begin{bmatrix} sgn(\widehat{h}_0(k)) \\ sgn(\widehat{h}_1(k)) \\ \vdots \\ sgn(\widehat{h}_{N-1}(k)) \end{bmatrix} \Big[sgn(\widehat{h}_0(k))\ sgn(\widehat{h}_1(k))\ \cdots\ sgn(\widehat{h}_{N-1}(k))\Big] \begin{bmatrix} \widehat{h}_0(k) \\ \widehat{h}_1(k) \\ \vdots \\ \widehat{h}_{N-1}(k) \end{bmatrix} \tag{4}$$

where we have used the relation

$$||\widehat{\boldsymbol{h}}(k)||_1 = \widehat{h}_0(k)sgn(\widehat{h}_0(k)) + \widehat{h}_1(k)sgn(\widehat{h}_1(k)) + \cdots + \widehat{h}_{N-1}(k)sgn(\widehat{h}_{N-1}(k)).$$

Here, *sgn* is the sign function applied to each component of the vector:

$$sgn(\widehat{h}_i(k)) = \begin{cases} \dfrac{\widehat{h}_i(k)}{|\widehat{h}_i(k)|} & \text{if } \widehat{h}_i(k) \neq 0 \\ 0 & \text{if } \widehat{h}_i(k) = 0. \end{cases} \tag{5}$$

A more compact form for (4) can be obtained by introducing the notation

$$\overline{\boldsymbol{sgn}}(k) = \begin{bmatrix} sgn(\widehat{h}_0(k)) \\ sgn(\widehat{h}_1(k)) \\ \vdots \\ sgn(\widehat{h}_{N-1}(k)) \end{bmatrix},$$

so that we can write $\nabla||\widehat{\boldsymbol{h}}(k)||_1^2 = \overline{\boldsymbol{sgn}}(k)\,\overline{\boldsymbol{sgn}}^T(k)\,\widehat{\boldsymbol{h}}(k)$. It is easy to get the following equation from (3)

$$(\boldsymbol{X}^T(k)\boldsymbol{X}(k) + P^{-1}(0) + \rho\overline{\boldsymbol{sgn}}(k)\,\overline{\boldsymbol{sgn}}^T(k))\widehat{\boldsymbol{h}}(k) = P^{-1}(0)\widehat{\boldsymbol{h}}(0) + \boldsymbol{X}^T(k)\boldsymbol{Y}(k). \tag{6}$$

We can see from (6) that $\overline{\boldsymbol{sgn}}(k)$ should be calculated based on the vector $\widehat{\boldsymbol{h}}(k)$ which is not yet available. A similar issue arises in other ℓ_1-regularization methods, such as [3–5], and it is solved assuming that the signs of the coefficients vector estimate values do not change significantly in a single iteration. Therefore, if $\widehat{\boldsymbol{h}}(k-1)$ is the optimal estimate obtained using the data from 0 to $k-1$, we can use $\overline{\boldsymbol{sgn}}(k-1)$ to replace $\overline{\boldsymbol{sgn}}(k)$.

Therefore, the estimate of $\widehat{\boldsymbol{h}}(k)$ which minimizes the cost function can be written as

$$\widehat{\boldsymbol{h}}(k) = (\boldsymbol{X}^T(k)\boldsymbol{X}(k) + P^{-1}(0) + \rho\overline{\boldsymbol{sgn}}(k-1)\,\overline{\boldsymbol{sgn}}^T(k-1))^{-1}\big(P^{-1}(0)\widehat{\boldsymbol{h}}(0) + \boldsymbol{X}^T(k)\boldsymbol{Y}(k)\big). \tag{7}$$

It is clear that (7) is a non-recursive[2)] relation, since it uses all the data $\boldsymbol{X}(k)$, $\boldsymbol{Y}(k)$ collected up to time k. In the following we want to derive recursive relations to update the estimate. In order to achieve this, let us now derive the recursive formula for calculating $P(k)$. Let us define

$$P^{-1}(k) = \boldsymbol{X}^T(k)\boldsymbol{X}(k) + P^{-1}(0) + \rho\overline{\boldsymbol{sgn}}(k-1)\,\overline{\boldsymbol{sgn}}^T(k-1). \tag{8}$$

2) To further highlight the difference with Tikhonov regularization, recall that that the non-recursive relation for $\widehat{\boldsymbol{h}}(k)$ in Tikhonov regularization is

$$\widehat{\boldsymbol{h}}(k) = (\boldsymbol{X}^T(k)\boldsymbol{X}(k) + P^{-1}(0) + \rho I)^{-1}\big(P^{-1}(0)\widehat{\boldsymbol{h}}(0) + \boldsymbol{X}^T(k)\boldsymbol{Y}(k)\big)$$

which shows the role of $P^{-1}(0)$ and ρ in the regularization.

Then, we can get the recursive relationship between $P^{-1}(k)$ and $P^{-1}(k-1)$ as follows:

$$\begin{aligned} P^{-1}(k) - P^{-1}(k-1) &= \boldsymbol{X}^T(k)\boldsymbol{X}(k) - \boldsymbol{X}^T(k-1)\boldsymbol{X}(k-1) \\ &\quad + \rho\overline{\boldsymbol{sgn}}(k-1)\overline{\boldsymbol{sgn}}^T(k-1) - \rho\overline{\boldsymbol{sgn}}(k-2)\overline{\boldsymbol{sgn}}^T(k-2) \\ &= \boldsymbol{x}^T(k)\boldsymbol{x}(k) \\ &\quad + \rho\overline{\boldsymbol{sgn}}(k-1)\overline{\boldsymbol{sgn}}^T(k-1) - \rho\overline{\boldsymbol{sgn}}(k-2)\overline{\boldsymbol{sgn}}^T(k-2). \end{aligned} \tag{9}$$

However, in order to avoid the calculation of the inverse of $P(k)$, a more convenient recursive relation is the one between $P(k)$ and $P(k-1)$ (rather than between $P^{-1}(k)$ and $P^{-1}(k-1)$). To this purpose, we can use the well-known matrix inversion lemma, that is $(A+BC)^{-1} = A^{-1} - A^{-1}B(I+CA^{-1}B)^{-1}CA^{-1}$. When applying the matrix inversion lemma to (9), we can define

$$A = P^{-1}(k-1), \quad B = [\boldsymbol{x}(k) \ \sqrt{\rho}\overline{\boldsymbol{sgn}}(k-1) \ -\sqrt{\rho}\overline{\boldsymbol{sgn}}(k-2)], \quad C = \begin{bmatrix} \boldsymbol{x}^T(k) \\ \sqrt{\rho}\overline{\boldsymbol{sgn}}^T(k-1) \\ \sqrt{\rho}\overline{\boldsymbol{sgn}}^T(k-2) \end{bmatrix},$$

so we can get the recursive form of $P(k)$ as follows:

$$P(k) = P(k-1) - P(k-1)Q(k)\Big(I + S(k)P(k-1)Q(k)\Big)^{-1}S(k)P(k-1) \tag{10}$$

where we have defined $Q(k) = [\boldsymbol{x}(k) \ \sqrt{\rho}\overline{\boldsymbol{sgn}}(k-1) \ -\sqrt{\rho}\overline{\boldsymbol{sgn}}(k-2)]$ and $S(k) = [\boldsymbol{x}(k) \ \sqrt{\rho}\overline{\boldsymbol{sgn}}(k-1) \ \sqrt{\rho}\overline{\boldsymbol{sgn}}(k-2)]^T$.

From (10) it is easy to obtain the recursive relation between $\widehat{\boldsymbol{h}}(k)$ and $\widehat{\boldsymbol{h}}(k-1)$. In fact, by combining (6) and (7), we can get

$$\begin{aligned} \widehat{\boldsymbol{h}}(k) &= P(k)(P^{-1}(0)\widehat{\boldsymbol{h}}(0) + \boldsymbol{X}^T(k-1)\boldsymbol{Y}(k-1) + \boldsymbol{x}(k)y(k)) \\ &= P(k)\big(P^{-1}(k-1)\widehat{\boldsymbol{h}}(k-1) + \boldsymbol{x}(k)y(k)\big) \\ &= P(k)\Big(P^{-1}(k)\widehat{\boldsymbol{h}}(k-1) - \boldsymbol{x}(k)\boldsymbol{x}^T(k)\widehat{\boldsymbol{h}}(k-1) - \rho[\overline{\boldsymbol{sgn}}(k-1) \ -\overline{\boldsymbol{sgn}}(k-2)] \\ &\qquad \cdot \begin{bmatrix} \overline{\boldsymbol{sgn}}^T(k-1) \\ \overline{\boldsymbol{sgn}}^T(k-2) \end{bmatrix} \widehat{\boldsymbol{h}}(k-1) + \boldsymbol{x}(k)y(k)\Big). \end{aligned} \tag{11}$$

As a result, we get the recursive update equation for $\widehat{\boldsymbol{h}}(k)$ from (11) as follows:

$$\widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + P(k)\boldsymbol{x}(k)e(k) - \rho P(k)[\overline{\boldsymbol{sgn}}(k-1) \ -\overline{\boldsymbol{sgn}}(k-2)] \begin{bmatrix} \overline{\boldsymbol{sgn}}^T(k-1) \\ \overline{\boldsymbol{sgn}}^T(k-2) \end{bmatrix} \widehat{\boldsymbol{h}}(k-1) \tag{12}$$

where $e(k)$ is the instantaneous error term given by $e(k) = y(k) - \widehat{\boldsymbol{h}}^T(k-1)\boldsymbol{x}(k)$. The resulting ℓ_1^2-RLS method is summarized in Algorithm 1.

Algorithm 1: Proposed Sparsity ℓ_1^2 Regularized Recursive Least Squares (ℓ_1^2-RLS)

Data and parameters: $\boldsymbol{x}(n), y(n)$, ρ, $\widehat{\boldsymbol{h}}(0)$, $P(0)$.

for time step $k = 1, 2, \cdots, n$, **do**

let $e(k) = y(k) - \widehat{\boldsymbol{h}}^T(k-1)\boldsymbol{x}(k)$ and

$$\overline{\boldsymbol{sgn}}(k-1) = \left[sgn(\widehat{h}_0(k-1)) \;\; sgn(\widehat{h}_1(k-1)) \cdots sgn(\widehat{h}_{N-1}(k-1))\right]^T$$

let $Q(k) = [\boldsymbol{x}(k) \;\; \sqrt{\rho}\overline{\boldsymbol{sgn}}(k-1) \;\; -\sqrt{\rho}\overline{\boldsymbol{sgn}}(k-2)]$

let $S(k) = [\boldsymbol{x}(k) \;\; \sqrt{\rho}\,\overline{\boldsymbol{sgn}}(k-1) \;\; \sqrt{\rho}\,\overline{\boldsymbol{sgn}}(k-2)]^T$

update $P(k) = P(k-1) - P(k-1)Q(k)$

$$\times\left(I + S(k)P(k-1)Q(k)\right)^{-1} S(k)P(k-1)$$

update $\widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + P(k)\boldsymbol{x}(k)e(k)$

$$-\rho P(k)[\overline{\boldsymbol{sgn}}(k-1) \;\; -\overline{\boldsymbol{sgn}}(k-2)] \begin{bmatrix} \overline{\boldsymbol{sgn}}^T(k-1) \\ \overline{\boldsymbol{sgn}}^T(k-2) \end{bmatrix} \widehat{\boldsymbol{h}}(k-1)$$

end for

Appendix B: Details on comparative experiments

This appendix gives more details about the comparisons of the proposed method with other methods. The performance of the proposed ℓ_1^2-RLS method is evaluated as compared to the standard RLS, to ℓ_1-RRLS [5] and to ZA-RLS [6][3]. For completeness, let us recall the last two algorithms.

The cost function of ℓ_1-RRLS is as follows:

$$J_{RRLS}(k) = \frac{1}{2}\sum_{s=1}^{k}\lambda^{k-s}|e(k)|^2 + \frac{1}{2}\,\rho||\boldsymbol{W}\boldsymbol{h}(k)||_1 \tag{13}$$

where $||\boldsymbol{W}\boldsymbol{h}(k)||_1$ stands for the weighted ℓ_1 norm of the vector estimate, that is

$$||\boldsymbol{W}\boldsymbol{h}(k)||_1 = \sum_{i=0}^{N-1} w_i|h_i(k)| \tag{14}$$

and $w_i, i = 0, 1, 2, \cdots, N-1$ are valued weighting parameters. Accordingly, the matrix $\boldsymbol{W}$ denotes the $N \times N$ diagonal matrix with the elements w_i on the main diagonal. In order to make (16) consistent with the proposed cost function (2), we will choose $\boldsymbol{W}$ as the identity matrix.

The ℓ_1-RRLS algorithm can be written as follows:

$$\begin{cases} \boldsymbol{k}_\lambda(k) = P(k-1)\boldsymbol{x}(k) \\ \boldsymbol{k}(k) = \dfrac{\boldsymbol{k}_\lambda(k)}{\lambda + \boldsymbol{x}^T(k)\boldsymbol{k}_\lambda(k)} \\ e(k) = y(k) - \widehat{\boldsymbol{h}}^T(k-1)\boldsymbol{x}(k) \\ P(k) = \dfrac{1}{\lambda}[P(k-1) - \boldsymbol{k}(k)\boldsymbol{k}_\lambda^T(k)] \\ \widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) + \boldsymbol{k}(k)e(k) + \rho(\dfrac{\lambda-1}{\lambda})(I - \boldsymbol{k}(k)\boldsymbol{x}^T(k))P(k-1)\dfrac{sgn((\widehat{\boldsymbol{h}}(k-1)))}{|\widehat{\boldsymbol{h}}(k-1)| + \epsilon} \end{cases} \tag{15}$$

where $\widehat{\boldsymbol{h}}(0) = 0$ is the initial estimate, $P(0) = \dfrac{1}{\delta}I_N$ is the initial covariance matrix with δ being a small positive number and $\epsilon > 0$ is a very small positive number, e.g. $\epsilon = 10^{-7}$, which is introduced for numerical stability reasons.

The cost function of ZA-RLS is as follows:

$$J_{ZA-RLS}(k) = \frac{1}{2}\sum_{s=1}^{k}\lambda^{k-s}|e(k)|^2 + \frac{1}{2}\,\rho\boldsymbol{h}^H\boldsymbol{D}(k)\boldsymbol{h} \tag{16}$$

where $(\cdot)^H$ denotes the Hermitian operator (which coincides with the transpose operator for real vectors), $\boldsymbol{D}(k)$=diag$\{d_0(k), d_1(k), \cdots, d_{N-1}(k)\}$ with $d_i(k) = \dfrac{1}{|h_i(k-1)| + \epsilon}$ for $0 \leqslant i \leqslant N-1$, while $\epsilon > 0$ is a very small positive number, e.g. $\epsilon = 10^{-7}$, which is introduced for numerical stability reasons.

The ZA-RLS algorithm can be written as follows:

$$\begin{cases} e(k) = y(k) - \widehat{\boldsymbol{h}}^T(k-1)\boldsymbol{x}(k) \\ H(k) = diag\Big\{(|\widehat{h}_0(k-1)| + \epsilon)/\rho, \cdots, (|\widehat{h}_L(k-1)| + \epsilon)/\rho\Big\} \\ D(k) = diag\Big\{\dfrac{1}{(|\widehat{h}_0(k-1)| + \epsilon)}, \cdots, \dfrac{1}{(|\widehat{h}_L(k-1)| + \epsilon)}\Big\},\ for\, k > 1 \\ P(k) = \dfrac{1}{\lambda}\left(P(k-1) - \dfrac{P(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)P(k-1)}{\lambda + \boldsymbol{x}^T(k)P(k-1)\boldsymbol{x}(k)}\right) \\ \widehat{P}(k) = H(k) - H(k)(P(k) + H(k))^{-1}H(k) \\ \widehat{\boldsymbol{h}}(k) = \widehat{\boldsymbol{h}}(k-1) - \rho\widehat{P}(k)(D(k) - \lambda D(k-1))\widehat{\boldsymbol{h}}(k-1) + \widehat{P}(k)\boldsymbol{x}(k)e(k) \end{cases} \tag{17}$$

3) Two ZA-RLS methods have been proposed in [6], namely ZA-RLS-I and ZA-RLS-II. Both methods give the same estimation performance, but the algorithm in ZA-RLS-II is computationally more efficient. This study considers ZA-RLS-I since its algorithm is closer to RLS and thus easier to understand. ZA-RLS-II would give exactly the same results.

where $\widehat{\boldsymbol{h}}(0)$ is the initial estimate, $P(0) = \dfrac{1}{\delta} I_N$ is the initial covariance matrix δ being a small positive number, while $D(0)$ and $D(1)$ can be initialized as zero matrices.

A *benchmark study* is set up as follows. The input $\boldsymbol{x}(k)$ is assumed to be white, and additive white Gaussian noise (AWGN) is added to the system output with a certain signal-to-noise ratio (SNR). For each trial, the length of the training data was set to 3000. The sparse system in each experiment has a total of 10 tabs where only K of them are nonzero. The positions of the nonzero tab value are chosen randomly, but for every trial the norm of $\boldsymbol{h}$ is normalized in such a way that $||\boldsymbol{h}||_1 = 1$, which of course also implies $||\boldsymbol{h}||_1^2 = 1$. This means that we can choose the same ρ for all algorithms, because the penalty would have a similar effect for all algorithms, despite using the ℓ_1 penalty or the ℓ_1^2 penalty. This is done in such a way to make the comparisons fair. For example, comparing the cost (13) of ℓ_1-RRLS with the cost (16) of ZA-RLS one can see that their cost is the same (for a small ϵ in ZA-RLS), therefore they can adopt the same ρ. Then, when looking at the proposed cost (2), again the cost is the same when $||\boldsymbol{h}||_1 = 1$ (consider that the term with $P^{-1}(0)$ is also present in ℓ_1-RRLS and ZA-RLS due to the initialization step, although this term is not explicitly reported in the corresponding literature). In other words, this benchmark study has been designed on purpose to make the parametric conditions fair for all algorithms used in the testing.

In order to make the comparisons as consistent as possible, we will adopt the following settings:

- The initial values $\widehat{\boldsymbol{h}}(0)$ and $P(0)$, and the regularization parameter ρ are chosen the same for all algorithms;
- We choose $\lambda = 1$ for ZA-RLS, in such a way that (16) is consistent with the proposed cost (2). However, we cannot choose $\lambda = 1$ for ℓ_1-RRLS, otherwise this algorithm will degenerate to the standard RLS (note that the term multiplying (1-λ) in (15) would disappear, resulting in a standard RLS). Therefore, we will choose $\lambda = 0.9999$ for ℓ_1-RRLS[4)].

In addition, the results are averaged over 1000 random trials, in order to calculate an average performance. The estimation performance is evaluated based on the ℓ_1-norm and ℓ_2-norm error with the true parameters, defined as

$$||\widehat{\boldsymbol{h}}_{FIN} - \boldsymbol{h}||_1 \tag{18}$$

$$||\widehat{\boldsymbol{h}}_{FIN} - \boldsymbol{h}||_2 \tag{19}$$

where $\widehat{\boldsymbol{h}}_{FIN}$ is the estimated $\widehat{\boldsymbol{h}}$ after the final iteration. The norms (18)-(19) are also calculated based on the average of 1000 independent trials.

The performance of all methods is tested in four aspects:

1) the effect of regularization parameter on the performance;
2) the effect of sparsity on the performance;
3) the effect of signal-to-noise ratio on the performance;
4) the convergence rate.

Effect of regularization parameter on the performance

We analyze the effect of the regularizing parameter ρ on different methods. The system to be identified presents three circumstances: i) it has a total of 10 coefficients where 3 are nonzero; ii) it has a total of 10 coefficients where 5 are nonzero; iii) it has a total of 10 coefficients where 7 are nonzero. This is done because increasing ρ increases the zero-attracting effect (driving the estimate towards zero). Therefore, it is very relevant to test the zero-attracting effect for different levels of sparsity. The SNR is 3 dB in all cases.

The results are shown in Table 1, Table 2 and Table 3. The parameters for the different algorithms are chosen as below:

- RLS, ℓ_1-RRLS, ZA-RLS, ℓ_1^2-RLS (proposed): $P(0) = 10^3 \times I$, $\widehat{\boldsymbol{h}}(0) = 0$
- ℓ_1^2-RLS (proposed), ZA-RLS: $\lambda = 1$,
- ℓ_1-RRLS: $\lambda = 0.9999$.

The regularizing parameter changes from 0.1 up to 5. Except for the standard RLS, the performance of ℓ_1^2-RLS (proposed), ZA-RLS, and ℓ_1-RRLS is sensitive to the regularizing parameter. This is because a large parameter ρ increases the effect of attracting the estimate towards zero. It is worth mentioning

4) We have also tried other values for ℓ_1-RRLS, such as $\lambda = 0.99$ or $\lambda = 0.999$, but we have experienced unstable behavior in some scenarios. Therefore, we have eventually chosen $\lambda = 0.9999$ which gives a stable behaviour in all scenarios we tested.

Table 1 Effect of regularizing parameter on performance (when K=3).

$\boldsymbol{\ell_1}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.1032	0.1032	0.1032	0.1032	0.1032	0.1032
ℓ_1-RRLS	0.1036	0.1034	0.1031	0.1028	0.1025	0.1007
ZA-RLS	0.1030	0.1020	0.1008	0.0996	0.0984	**0.0916**
ℓ_1^2-RLS (proposed)	**0.1029**	**0.1019**	**0.1006**	**0.0994**	**0.0982**	0.0922
$\boldsymbol{\ell_2}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.0400	0.0400	0.0400	0.0400	0.0400	0.0400
ℓ_1-RRLS	0.0401	0.0400	0.0399	0.0398	0.0397	0.0391
ZA-RLS	0.0399	0.0397	0.0393	0.0390	0.0387	0.0370
ℓ_1^2-RLS (proposed)	**0.0398**	**0.0396**	**0.0392**	**0.0389**	**0.0386**	**0.0368**

Table 2 Effect of regularizing parameter on performance (when K=5).

$\boldsymbol{\ell_1}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.1032	0.1032	0.1032	0.1032	0.1032	0.1032
ℓ_1-RRLS	0.1037	0.1035	0.1033	0.1032	0.1030	0.1020
ZA-RLS	0.1030	0.1022	0.1013	0.1005	0.0996	0.0948
ℓ_1^2-RLS (proposed)	**0.1029**	**0.1017**	**0.1002**	**0.0988**	**0.0976**	**0.0926**
$\boldsymbol{\ell_2}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.0400	0.0400	0.0400	0.0400	0.0400	0.0400
ℓ_1-RRLS	0.0401	0.0401	0.0400	0.0399	0.0399	0.0395
ZA-RLS	0.0399	0.0397	0.0394	0.0392	0.0390	0.0378
ℓ_1^2-RLS (proposed)	**0.0398**	**0.0395**	**0.0391**	**0.0388**	**0.0384**	**0.0368**

Table 3 Effect of regularizing parameter on performance (when K=7).

$\boldsymbol{\ell_1}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.1032	0.1032	0.1032	0.1032	0.1032	0.1032
ℓ_1-RRLS	0.1037	0.1036	0.1035	0.1034	0.1034	0.1029
ZA-RLS	0.1031	0.1027	0.1022	0.1017	0.1012	0.0988
ℓ_1^2-RLS (proposed)	**0.1030**	**0.1020**	**0.1009**	**0.1000**	**0.0992**	**0.0979**
$\boldsymbol{\ell_2}$**-norm error**	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$	$\rho = 5$
RLS	0.0400	0.0400	0.0400	0.0400	0.0400	0.0400
ℓ_1-RRLS	0.0401	0.0401	0.0401	0.0400	0.0400	0.0398
ZA-RLS	0.0399	0.0398	0.0397	0.0395	0.0394	0.0388
ℓ_1^2-RLS (proposed)	**0.0398**	**0.0396**	**0.0393**	**0.0391**	**0.0389**	**0.0385**

that the zero-attracting effect is beneficial when the level of sparsity is large (e.g. $K = 3$): this is because a lot tabs to be estimated are indeed zero. However, as the level of sparsity decreases (e.g. with $K = 5$ or $K = 7$), attracting the estimate towards zero is not necessarily beneficial, since many elements of $\boldsymbol{h}$ are actually different than zero. This explains why ZA-RLS is good for $\rho = 5$ and $K = 3$, but it is outperformed by the proposed ℓ_1^2-RLS method when $\rho = 5$ and $K = 5$, $K = 7$. In other words, the proposed ℓ_1^2-RLS method seems to provide a good trade-off between attracting the estimate towards zero and providing a good estimate.

Effect of sparsity on the performance

The experiment dwells on the effects of the sparsity on the different methods. The sparse system to be identified has a total of 10 tabs and we change the number of nonzero tabs K to change the level of sparsity. The SNR is 3 dB in all cases. The number of nonzero coefficients varies from 1 to 9. The parameters for the different algorithms are chosen as below:

- RLS, ℓ_1-RRLS, ZA-RLS, ℓ_1^2-RLS (proposed): $P(0) = 10^3 \times I$, $\widehat{\boldsymbol{h}}(0) = 0$, $\rho = 1$
- ℓ_1^2-RLS (proposed), ZA-RLS: $\lambda = 1$,
- ℓ_1-RRLS: $\lambda = 0.9999$.

The results presented in Table 4 demonstrate that the performance of the standard RLS is independent of the system sparsity. On the other hand, the performance of ℓ_1-RRLS and ZA-RLS degrades with a decline in sparsity. The error of the proposed ℓ_1^2-RLS method first decreases and then increases as the number of nonzero terms increases, i.e. the performance seems to be best in the range 20-50% sparsity. The proposed ℓ_1^2-RLS method gives the best performance in all cases except when having only 1 non-zero tab.

Table 4 Effect of number of non-zero tabs on performance

$\boldsymbol{\ell_1}$**-norm error**	K=1	K=3	K=5	K=7	K=9
RLS	0.1032	0.1032	0.1032	0.1032	0.1032
ℓ_1-RRLS	0.1024	0.1031	0.1033	0.1035	0.1036
ZA-RLS	**0.1001**	0.1008	0.1013	0.1022	0.1029
ℓ_1^2-RLS (proposed)	0.1019	**0.1006**	**0.1002**	**0.1009**	**0.1022**
$\boldsymbol{\ell_2}$**-norm error**	K=1	K=3	K=5	K=7	K=9
RLS	0.0400	0.0400	0.0400	0.0400	0.0400
ℓ_1-RRLS	0.0397	0.0399	0.0400	0.0401	0.0401
ZA-RLS	**0.0392**	0.0393	0.0394	0.0397	0.0399
ℓ_1^2-RLS (proposed)	0.0396	**0.0392**	**0.0391**	**0.0393**	**0.0397**

Effect of signal-to-noise ratio on the performance

This experiment compares the performance of the proposed ℓ_1^2-RLS method, standard RLS, ℓ_1-RRLS and ZA-RLS under different SNR values. The underlying system has again a total of 10 coefficients where 3 are nonzero. The performance for SNR values of 1, 3, 5, 7 and 10 dB is shown in Table 5. The parameters for the different algorithms are chosen as below:

- RLS, ℓ_1-RRLS, ZA-RLS, ℓ_1^2-RLS (proposed): $P(0) = 10^3 \times I$, $\widehat{\boldsymbol{h}}(0) = 0$, $\rho = 1$
- ℓ_1^2-RLS (proposed), ZA-RLS: $\lambda = 1$,
- ℓ_1-RRLS: $\lambda = 0.9999$.

Table 5 shows that the proposed ℓ_1^2-RLS method behaves better in noisy situations, and it is only when the signal-to-noise ratio is above 10 that ZA-RLS behaves as good as the proposed method.

Convergence rate

We finally investigate the learning rate of all the algorithms for different signal-to-noise ratios (similar to Table 5). The figures show that all methods have a comparable trend (showing that all the methods used for comparison are consistent with each other). The proposed method and ZA-RLS have the fastest convergence, where the proposed method behaves a bit better in most scenarios.

Two main points can be identified regarding why the proposed algorithm can overcome some of the tested state-of-the-art algorithms:

a) The first point is that all algorithms aim to minimize a cost which has no analytic solution in general. Therefore, some approximation is necessary in order to find a minimum of the cost. The cost and the approximation method we proposed (cf. (2) in Appendix A) is the one closest to Tikhonov

Table 5 Effect of signal-to-noise ratio on performance.

$\boldsymbol{\ell_1}$**-norm error**	SNR=1	SNR=3	SNR=5	SNR=7	SNR=10
RLS	0.1299	0.1032	0.0820	0.0653	0.0461
ℓ_1-RRLS	0.1299	0.1031	0.0818	0.0649	0.0457
ZA-RLS	0.1275	0.1008	0.0795	0.0630	**0.0437**
ℓ_1^2-RLS (proposed)	**0.1273**	**0.1006**	**0.0794**	**0.0629**	**0.0437**
$\boldsymbol{\ell_2}$**-norm error**	SNR=1	SNR=3	SNR=5	SNR=7	SNR=10
RLS	0.0503	0.0400	0.0317	0.0252	0.0179
ℓ_1-RRLS	0.0503	0.0399	0.0316	0.0252	0.0177
ZA-RLS	0.0497	0.0393	0.0311	0.0249	**0.0172**
ℓ_1^2-RLS (proposed)	**0.0496**	**0.0392**	**0.0310**	**0.0248**	**0.0172**

regularization, the most standard formulation to address least-squares. This means that the proposed ℓ_1^2-RLS methodology is deeply rooted in a standard least-squares formulation.

b) The second point is that different algorithms differ regarding the way the cost is approximated, and some assumptions are needed to perform such approximation. We have discussed that algorithms as ℓ_1-RLS, ℓ_1-RRLS require $0 < \lambda < 1$, which means that their approximations cannot work when $\lambda = 1$, since they degenerate in the standard RLS. The proposed ℓ_1^2-RLS approach is potentially applicable for any value of $0 < \lambda \leqslant 1$ (upon minor modifications not shown in this wok), i.e. it can be adopted in more general settings.

Because the proposed method behaves well in sparse and non-sparse scenarios, an interesting future work is to dynamically change the size of the vector to be estimated, i.e., reduce or increase it online depending on an estimated degree of sparsity. Also, adaptation of ρ is sometimes studied in the literature: because our goal was to compare state-of-the-art algorithms under similar conditions (note that ℓ_1-RRLS, ZA-RLS do not use adaptation of ρ) we left the adaptation of ρ outside the scope of this work, which is however a relevant topic amenable for future work.

References

1 Haykin S S. Adaptive filter theory. Pearson Education India, 2008.

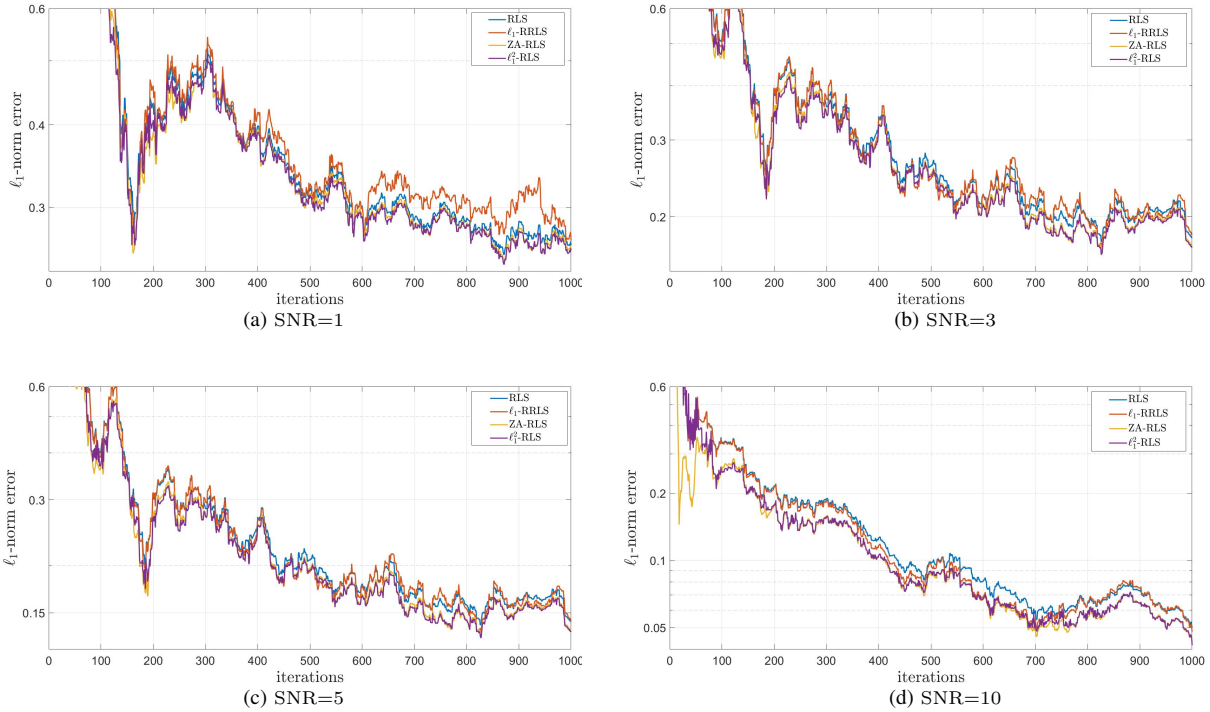
2 Li Y, Hamamura M. Zero-attracting variable-step-size least mean square algorithms for adaptive sparse channel estimation. International Journal of Adaptive Control and Signal Processing, 2015, 29: 1189-1206

3 Eksioglu E M. RLS adaptive filtering with sparsity regularization. In: 10th International Conference on Information Science, Signal Processing and their Applications (ISSPA), 2010. 550-553

4 Lim J, Lee K, Lee S. A Modified Recursive Regularization Factor Calculation for Sparse RLS Algorithm with l1-Norm. Mathematics, 2021, 9: 1580

5 Eksioglu E M. Sparsity regularised recursive least squares adaptive filtering. IET Signal Processing, 2011, 5: 480-487

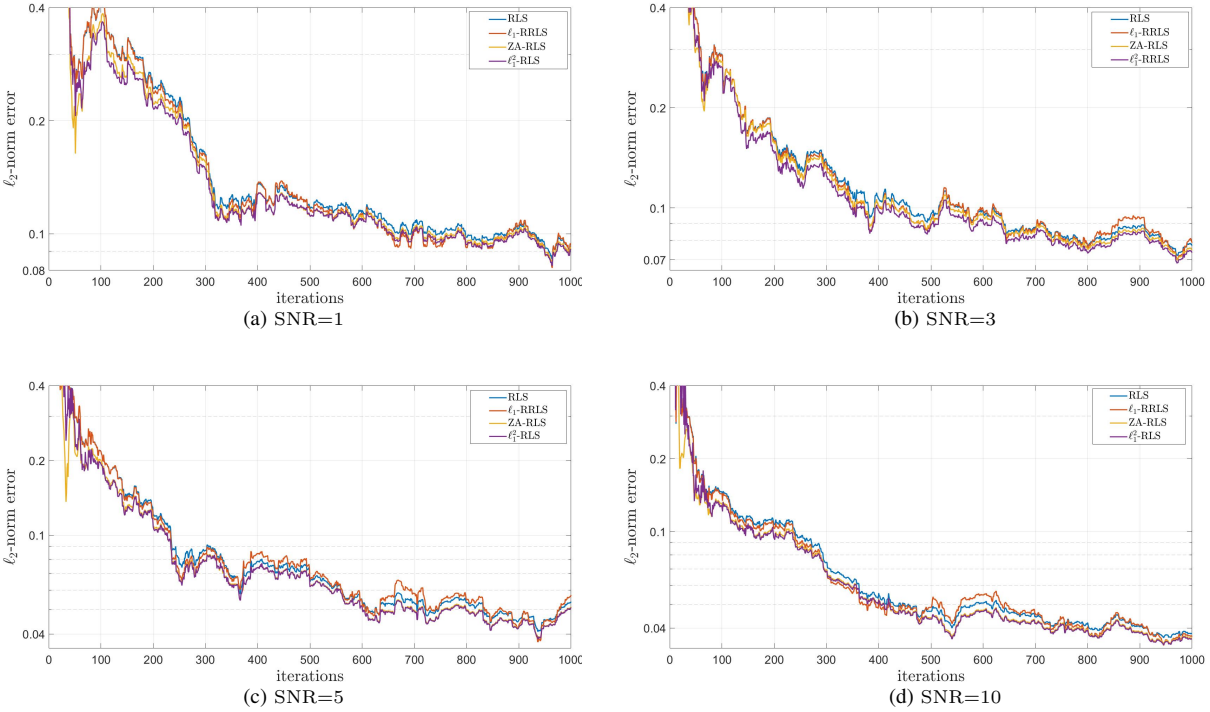
6 Hong X, Gao J, Chen S. Zero-attracting recursive least squares algorithms. IEEE Transactions on Vehicular Technology, 2016, 66: 213-221



(a) SNR=1 (b) SNR=3

(c) SNR=5 (d) SNR=10

Figure 1 Learning curves (in terms of ℓ_1-norm error) for RLS, ℓ_1-RRLS, ZA-RLS, and proposed ℓ_1^2-RLS.



(a) SNR=1 (b) SNR=3

(c) SNR=5 (d) SNR=10

Figure 2 Learning curves (in terms of ℓ_2-norm error) for RLS, ℓ_1-RRLS, ZA-RLS, and proposed ℓ_1^2-RLS.