RLS Adaptive Filtering with Sparsity Regularization

Asst. Prof. Ender M. EKŞİOĞLU

Istanbul Technical University
Electronics and Communications Engineering Department

Main Headings

Main Headings

- Introduction

Main Headings

- Introduction
- ℓ_1-RLS Algorithm

Main Headings

- Introduction
- ℓ_1-RLS Algorithm
- Simulation Results

Main Headings

- Introduction
- ℓ_1-RLS Algorithm
- Simulation Results
- Conclusions

Introduction

- Sparse adaptive filtering, where the impulse response for the system to be identified is assumed to be of a sparse form has acquired attention recently.

Introduction

- Sparse adaptive filtering, where the impulse response for the system to be identified is assumed to be of a sparse form has acquired attention recently.
- The sparsity prior has applications in acoustic and network echo cancellation and communication channel identification.

Introduction

- Sparse adaptive filtering, where the impulse response for the system to be identified is assumed to be of a sparse form has acquired attention recently.
- The sparsity prior has applications in acoustic and network echo cancellation and communication channel identification.
- Proportionate adaptive algorithm is a well-known approach to the problem.

Introduction

- Recently, novel LMS type algorithms which incorporate the sparsity condition directly into the cost function have been developed.

Introduction

- Recently, novel LMS type algorithms which incorporate the sparsity condition directly into the cost function have been developed.
- The common idea is to add a penalty term in the form of an ℓ_p norm of the weight vector into the overall cost function to be minimized.

Introduction

- Recently, novel LMS type algorithms which incorporate the sparsity condition directly into the cost function have been developed.
- The common idea is to add a penalty term in the form of an ℓ_p norm of the weight vector into the overall cost function to be minimized.
- Sparsity based adaptive algorithms have been mostly confined to the LMS domain.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.
- In this paper, we propose an RLS adaptive algorithm for sparse system identification.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.
- In this paper, we propose an RLS adaptive algorithm for sparse system identification.
- The algorithm will utilize the modified RLS cost function with an additional sparsity inducing ℓ_1 penalty term.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.
- In this paper, we propose an RLS adaptive algorithm for sparse system identification.
- The algorithm will utilize the modified RLS cost function with an additional sparsity inducing ℓ_1 penalty term.
- We find the recursive minimization procedure in a manner similar to the conventional RLS approach.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.
- In this paper, we propose an RLS adaptive algorithm for sparse system identification.
- The algorithm will utilize the modified RLS cost function with an additional sparsity inducing ℓ_1 penalty term.
- We find the recursive minimization procedure in a manner similar to the conventional RLS approach.
- The difference occurs in the weight vector update equation, where a novel zero-attracting, sparsity inducing additional term is included.

Introduction

- Recursive least squares (RLS) adaptive filtering is another important modality in the adaptive system identification setting.
- In this paper, we propose an RLS adaptive algorithm for sparse system identification.
- The algorithm will utilize the modified RLS cost function with an additional sparsity inducing ℓ_1 penalty term.
- We find the recursive minimization procedure in a manner similar to the conventional RLS approach.
- The difference occurs in the weight vector update equation, where a novel zero-attracting, sparsity inducing additional term is included.
- We will call this new algorithm as the ℓ_1-RLS.

Introduction

- Firstly give a brief outline of the adaptive system identification setting.

Introduction

- Firstly give a brief outline of the adaptive system identification setting.
- Then, we develop the novel ℓ_1-RLS algorithm by outlining the similarities to the development of regular RLS.

Introduction

- Firstly give a brief outline of the adaptive system identification setting.
- Then, we develop the novel ℓ_1-RLS algorithm by outlining the similarities to the development of regular RLS.
- We give the final form of ℓ_1-RLS algorithm.

Introduction

- Firstly give a brief outline of the adaptive system identification setting.
- Then, we develop the novel ℓ_1-RLS algorithm by outlining the similarities to the development of regular RLS.
- We give the final form of ℓ_1-RLS algorithm.
- We will present simulation results comparing the novel ℓ_1-RLS algorithm to regular RLS, regular LMS and other adaptive algorithms.

ℓ_1-RLS Algorithm

- Consider the system identification setting given by the following input-output equation.

$$y(n) = \mathbf{h}^T \mathbf{x}(n) + \eta(n) \tag{1}$$

ℓ_1-RLS Algorithm

- Consider the system identification setting given by the following input-output equation.

$$y(n) = \mathbf{h}^T \mathbf{x}(n) + \eta(n) \tag{1}$$

- The aim of the adaptive system identification algorithm is to estimate the system parameters $\mathbf{h}$ from the input and output signals in a sequential manner.

ℓ_1-RLS Algorithm

- Consider the system identification setting given by the following input-output equation.

$$y(n) = \mathbf{h}^T \mathbf{x}(n) + \eta(n) \qquad (1)$$

- The aim of the adaptive system identification algorithm is to estimate the system parameters $\mathbf{h}$ from the input and output signals in a sequential manner.
- In conventional RLS, the cost function to be minimized by the weight estimate is given by

$$\mathcal{E}(n) = \sum_{m=0}^{n} \lambda^{n-m} |e(m)|^2. \qquad (2)$$

ℓ_1-RLS Algorithm

- We assume that the underlying filter coefficient vector $\mathbf{h}$ has a sparse form.

ℓ_1-RLS Algorithm

- We assume that the underlying filter coefficient vector $\mathbf{h}$ has a sparse form.
- Hence, we want to modify the cost function in a manner that underlines this a priori information.

ℓ_1-RLS Algorithm

- We assume that the underlying filter coefficient vector $\mathbf{h}$ has a sparse form.
- Hence, we want to modify the cost function in a manner that underlines this a priori information.
- A tractable way to force sparsity is by using the ℓ_1-norm of the weight vector.

ℓ_1-RLS Algorithm

- We assume that the underlying filter coefficient vector $\mathbf{h}$ has a sparse form.
- Hence, we want to modify the cost function in a manner that underlines this a priori information.
- A tractable way to force sparsity is by using the ℓ_1-norm of the weight vector.
- Hence, we regularize the RLS cost function by including the weighted ℓ_1 norm of the current tab estimate as a sparsifying term.

ℓ_1-RLS Algorithm

$$J(n) = \frac{1}{2}\mathcal{E}(n) + \gamma\|\mathbf{h}(n)\|_1 \tag{3}$$

ℓ_1-RLS Algorithm

$$J(n) = \frac{1}{2}\mathcal{E}(n) + \gamma\|\mathbf{h}(n)\|_1 \tag{3}$$

- Here, $\gamma > 0$ is a parameter that governs the tradeoff between sparsity and estimation error.

ℓ_1-RLS Algorithm

$$J(n) = \frac{1}{2}\mathcal{E}(n) + \gamma \|\mathbf{h}(n)\|_1 \tag{3}$$

- Here, $\gamma > 0$ is a parameter that governs the tradeoff between sparsity and estimation error.
- $\|\mathbf{h}(n)\|_1$ is the ℓ_1 norm of the weight vector and is given by

$$\|\mathbf{h}(n)\|_1 = \sum_{k=0}^{N-1} |h_k(n)| \tag{4}$$

ℓ_1-RLS Algorithm

- We want to minimize this regularized cost function $J(n)$ with respect to the filter tab weights.

ℓ_1-RLS Algorithm

- We want to minimize this regularized cost function $J(n)$ with respect to the filter tab weights.
- In the standard RLS case when the cost function is simply $\mathcal{E}(n)$, the minimization condition is written in terms of the gradient of $\mathcal{E}(n)$ with respect to $\mathbf{h}(n)$.

ℓ_1-RLS Algorithm

- We want to minimize this regularized cost function $J(n)$ with respect to the filter tab weights.
- In the standard RLS case when the cost function is simply $\mathcal{E}(n)$, the minimization condition is written in terms of the gradient of $\mathcal{E}(n)$ with respect to $\mathbf{h}(n)$.
- However, the ℓ_1 norm term $\|\mathbf{h}(n)\|_1$ in $J(n)$ in (3) is nondifferentiable at any point where $h_k(n) = 0$.

ℓ_1-RLS Algorithm

- We want to minimize this regularized cost function $J(n)$ with respect to the filter tab weights.
- In the standard RLS case when the cost function is simply $\mathcal{E}(n)$, the minimization condition is written in terms of the gradient of $\mathcal{E}(n)$ with respect to $\mathbf{h}(n)$.
- However, the ℓ_1 norm term $\|\mathbf{h}(n)\|_1$ in $J(n)$ in (3) is nondifferentiable at any point where $h_k(n) = 0$.
- A substitute for the gradient in the case of nondifferentiable convex functions such as $\|\mathbf{h}(n)\|_1$ here is offered by the definition of the subgradient.

ℓ_1-RLS Algorithm

- One subgradient vector of the penalized cost function $J(n)$ with respect to the weight vector $\mathbf{h}(n)$ can be written as

$$\nabla^S J(n) = \frac{1}{2}\nabla\mathcal{E} + \gamma\,\mathrm{sgn}\big(\mathbf{h}(n)\big) \tag{5}$$

ℓ_1-RLS Algorithm

- One subgradient vector of the penalized cost function $J(n)$ with respect to the weight vector $\mathbf{h}(n)$ can be written as

$$\nabla^S J(n) = \frac{1}{2}\nabla\mathcal{E} + \gamma \operatorname{sgn}\big(\mathbf{h}(n)\big) \tag{5}$$

- The i^{th} element of this vector is calculated as below.

$$\left\{\nabla^S J(n)\right\}_i = -\sum_{m=0}^{n} \lambda^{n-m} e(m) x^*(m-i+1) + \gamma \operatorname{sgn}\big(h_i(n)\big) \tag{6}$$

ℓ_1-RLS Algorithm

- We set the subgradient equal to zero to find the optimal least squares solution, namely $\hat{\mathbf{h}}(n)$.

ℓ_1-RLS Algorithm

- We set the subgradient equal to zero to find the optimal least squares solution, namely $\hat{\mathbf{h}}(n)$.

$$-\sum_{m=0}^{n}\lambda^{n-m}\left\{y(m)-\sum_{k=0}^{N-1}\hat{h}_k(n)x(m-k)\right\}x^*(m-i+1)=-\gamma\,\mathrm{sgn}\big(\hat{h}_i(n) \tag{7}$$

ℓ_1-RLS Algorithm

- We set the subgradient equal to zero to find the optimal least squares solution, namely $\hat{\mathbf{h}}(n)$.

$$-\sum_{m=0}^{n} \lambda^{n-m} \Big\{ y(m) - \sum_{k=0}^{N-1} \hat{h}_k(n) x(m-k) \Big\} x^*(m-i+1) = -\gamma \, \mathrm{sgn}\big(\hat{h}_i(n) \quad (7)$$

- Written for all $i = 1, \ldots, N$ together in a matrix form, results in the modified deterministic normal equations.

ℓ_1-RLS Algorithm

$$\boldsymbol{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma \operatorname{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

ℓ_1-RLS Algorithm

$$\boldsymbol{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma\,\mathrm{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

- Here, $\boldsymbol{\Phi}(n)$ is the exponentially weighted deterministic autocorrelation matrix estimate.

ℓ_1-RLS Algorithm

$$\mathbf{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma\,\mathrm{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

- Here, $\mathbf{\Phi}(n)$ is the exponentially weighted deterministic autocorrelation matrix estimate.
- $\mathbf{r}(n)$ is the deterministic cross-correlation estimate between $y(n)$ and $\mathbf{x}(n)$.

ℓ_1-RLS Algorithm

$$\mathbf{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma \, \mathrm{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

- Here, $\mathbf{\Phi}(n)$ is the exponentially weighted deterministic autocorrelation matrix estimate.
- $\mathbf{r}(n)$ is the deterministic cross-correlation estimate between $y(n)$ and $\mathbf{x}(n)$.
- These two quantities can be updated by rank-one recursive equations.

ℓ_1-RLS Algorithm

$$\mathbf{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma \operatorname{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

- Here, $\mathbf{\Phi}(n)$ is the exponentially weighted deterministic autocorrelation matrix estimate.
- $\mathbf{r}(n)$ is the deterministic cross-correlation estimate between $y(n)$ and $\mathbf{x}(n)$.
- These two quantities can be updated by rank-one recursive equations.

$$\mathbf{\Phi}(n) = \lambda\mathbf{\Phi}(n-1) + \mathbf{x}^*(n)\mathbf{x}^T(n)$$

ℓ_1-RLS Algorithm

$$\mathbf{\Phi}(n)\hat{\mathbf{h}}(n) = \mathbf{r}(n) - \gamma\,\mathrm{sgn}\big(\hat{\mathbf{h}}(n)\big) \tag{8}$$

- Here, $\mathbf{\Phi}(n)$ is the exponentially weighted deterministic autocorrelation matrix estimate.
- $\mathbf{r}(n)$ is the deterministic cross-correlation estimate between $y(n)$ and $\mathbf{x}(n)$.
- These two quantities can be updated by rank-one recursive equations.

$$\mathbf{\Phi}(n) = \lambda\mathbf{\Phi}(n-1) + \mathbf{x}^*(n)\mathbf{x}^T(n)$$

$$\mathbf{r}(n) = \lambda\mathbf{r}(n-1) + y(n)\mathbf{x}^*(n)$$

ℓ_1-RLS Algorithm

- Instead of solving the normal equations for the optimal least squares solution $\hat{\mathbf{h}}(n)$ directly, search for an iterative solution.

ℓ_1-RLS Algorithm

- Instead of solving the normal equations for the optimal least squares solution $\hat{\mathbf{h}}(n)$ directly, search for an iterative solution.
- We assume that the sign of the weight values do not change significantly in a single time step.

ℓ_1-RLS Algorithm

- Instead of solving the normal equations for the optimal least squares solution $\hat{\mathbf{h}}(n)$ directly, search for an iterative solution.
- We assume that the sign of the weight values do not change significantly in a single time step.
- The normal equation can be rewritten as

$$\hat{\mathbf{h}}(n) = \mathbf{P}(n)\boldsymbol{\theta}(n) \tag{9}$$

where $\mathbf{P}(n)$ is the inverse of the autocorrelation matrix.

$$\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n)$$

ℓ_1-RLS Algorithm

- We come up with the following result.

$$\hat{\mathbf{h}}(n) = \mathbf{P}(n-1)\boldsymbol{\theta}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\boldsymbol{\theta}(n-1)$$
$$+ y(n)\mathbf{k}(n) + \gamma\left(\frac{\lambda-1}{\lambda}\right)\times$$
$$\left\{\mathbf{P}(n-1)\,\mathrm{sgn}\left(\hat{\mathbf{h}}(n-1)\right) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\,\mathrm{sgn}\left(\hat{\mathbf{h}}(n-1)\right)\right\}$$

ℓ_1-RLS Algorithm

- We come up with the following result.

$$\hat{\mathbf{h}}(n) = \mathbf{P}(n-1)\boldsymbol{\theta}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\boldsymbol{\theta}(n-1)$$
$$+ y(n)\mathbf{k}(n) + \gamma\left(\frac{\lambda - 1}{\lambda}\right) \times$$
$$\left\{\mathbf{P}(n-1)\,\mathrm{sgn}\left(\hat{\mathbf{h}}(n-1)\right) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\,\mathrm{sgn}\left(\hat{\mathbf{h}}(n-1)\right)\right\}$$

- Here, $\mathbf{k}(n)$ is the gain vector.

$$\mathbf{k}(n) = \frac{\mathbf{P}(n-1)\mathbf{x}^*(n)}{\lambda + \mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)} \tag{10}$$

ℓ_1-RLS Algorithm

- Using the matrix inversion lemma, it can be shown that the time update for the inverse correlation matrix can be performed by the well known Riccati equation.

$$\mathbf{P}(n) = \lambda^{-1}\Big\{\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big\} \qquad (11)$$

ℓ_1-RLS Algorithm

- Using the matrix inversion lemma, it can be shown that the time update for the inverse correlation matrix can be performed by the well known Riccati equation.

$$\mathbf{P}(n) = \lambda^{-1}\Big\{\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big\} \qquad (11)$$

- The recursive update for the tab weight vector assumes its final form.

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mathbf{k}(n)\Big\{y(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n)\Big\} + \gamma\Big(\frac{\lambda-1}{\lambda}\Big)\Big\{\mathbf{I}_N - \mathbf{k}(n)\mathbf{x}^T(n)\Big\}\mathbf{P}(n-1)\mathrm{sgn}\big(\hat{\mathbf{h}}(n-1)\big) \qquad (12)$$

ℓ_1-RLS Algorithm

- Using the matrix inversion lemma, it can be shown that the time update for the inverse correlation matrix can be performed by the well known Riccati equation.

$$\mathbf{P}(n) = \lambda^{-1}\Big\{\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)\Big\} \qquad (11)$$

- The recursive update for the tab weight vector assumes its final form.

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mathbf{k}(n)\Big\{y(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n)\Big\} + \gamma\Big(\frac{\lambda-1}{\lambda}\Big)\Big\{\mathbf{I}_N - \mathbf{k}(n)\mathbf{x}^T(n)\Big\}\mathbf{P}(n-1)\mathrm{sgn}\big(\hat{\mathbf{h}}(n-1)\big) \qquad (12)$$

- This update equation finalizes the ℓ_1-RLS algorithm.

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda, \ \gamma, \ x(n), \ y(n)$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda,\ \gamma,\ x(n),\ y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}, \quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda,\ \gamma,\ x(n),\ y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}, \quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: λ, γ, $x(n)$, $y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}$, $\quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda, \ \gamma, \ x(n), \ y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}, \quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
- $\mathbf{k}(n) = \dfrac{\mathbf{k}_\lambda(n)}{\lambda + \mathbf{x}^T(n)\mathbf{k}_\lambda(n)}$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: λ, γ, $x(n)$, $y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}$, $\quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
- $\mathbf{k}(n) = \dfrac{\mathbf{k}_\lambda(n)}{\lambda + \mathbf{x}^T(n)\mathbf{k}_\lambda(n)}$
- $\xi(n) = y(n) - \mathbf{h}^T(n-1)\mathbf{x}(n)$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: λ, γ, $x(n)$, $y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}$, $\quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
- $\mathbf{k}(n) = \dfrac{\mathbf{k}_\lambda(n)}{\lambda + \mathbf{x}^T(n)\mathbf{k}_\lambda(n)}$
- $\xi(n) = y(n) - \mathbf{h}^T(n-1)\mathbf{x}(n)$
- $\mathbf{P}(n) = \dfrac{1}{\lambda}\left[\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{k}_\lambda^H(n)\right]$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda,\ \gamma,\ x(n),\ y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}$, $\quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
- $\mathbf{k}(n) = \dfrac{\mathbf{k}_\lambda(n)}{\lambda + \mathbf{x}^T(n)\mathbf{k}_\lambda(n)}$
- $\xi(n) = y(n) - \mathbf{h}^T(n-1)\mathbf{x}(n)$
- $\mathbf{P}(n) = \dfrac{1}{\lambda}\left[\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{k}_\lambda^H(n)\right]$
-

$$\mathbf{h}(n) = \mathbf{h}(n-1) + \mathbf{k}(n)\xi(n) + \gamma\left(\frac{\lambda-1}{\lambda}\right)\left\{\mathbf{I}_N - \mathbf{k}(n)\mathbf{x}^T(n)\right\}\mathbf{P}(n-1)\mathrm{sgn}\big(\mathbf{h}(n-1)\big)$$

ℓ_1-RLS Algorithm

ℓ_1 regularized RLS (ℓ_1-RLS) algorithm.

- inputs: $\lambda,\ \gamma,\ x(n),\ y(n)$
- initial values: $\mathbf{h}(-1) = \mathbf{0}, \quad \mathbf{P}(-1) = \delta^{-1}\mathbf{I}$
- for $n := 0, 1, 2, \ldots$
- $\mathbf{k}_\lambda(n) = \mathbf{P}(n-1)\mathbf{x}^*(n)$
- $\mathbf{k}(n) = \dfrac{\mathbf{k}_\lambda(n)}{\lambda + \mathbf{x}^T(n)\mathbf{k}_\lambda(n)}$
- $\xi(n) = y(n) - \mathbf{h}^T(n-1)\mathbf{x}(n)$
- $\mathbf{P}(n) = \dfrac{1}{\lambda}\left[\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{k}_\lambda^H(n)\right]$
-
$$\begin{aligned}\mathbf{h}(n) &= \mathbf{h}(n-1) + \mathbf{k}(n)\xi(n) \\ &+ \gamma\left(\frac{\lambda-1}{\lambda}\right)\left\{\mathbf{I}_N - \mathbf{k}(n)\mathbf{x}^T(n)\right\}\mathbf{P}(n-1)\mathrm{sgn}\big(\mathbf{h}(n-1)\big)\end{aligned}$$
- endfor

ℓ_1-RLS Algorithm

- When we compare the ℓ_1-RLS weight update with the regular RLS update equation, we see that the last term starting with $\gamma\left(\frac{\lambda-1}{\lambda}\right)$ constitutes the difference from regular RLS.

Simulation results

- We compare the performance of the novel ℓ_1-RLS algorithm to the regular RLS, regular LMS and other sparsity oriented adaptive algorithm.

Simulation results

- We compare the performance of the novel ℓ_1-RLS algorithm to the regular RLS, regular LMS and other sparsity oriented adaptive algorithm.
- The first experiment considers the tracking capabilities of ℓ_1-RLS, RLS, ZA-LMS (Chen2009) and LMS algorithms under white excitation.

Simulation results

- The first experiment considers the tracking capabilities of ℓ_1-RLS, RLS, ZA-LMS (Chen2009) and LMS algorithms under white excitation.

Simulation results

- The first experiment considers the tracking capabilities of ℓ_1-RLS, RLS, ZA-LMS (Chen2009) and LMS algorithms under white excitation.

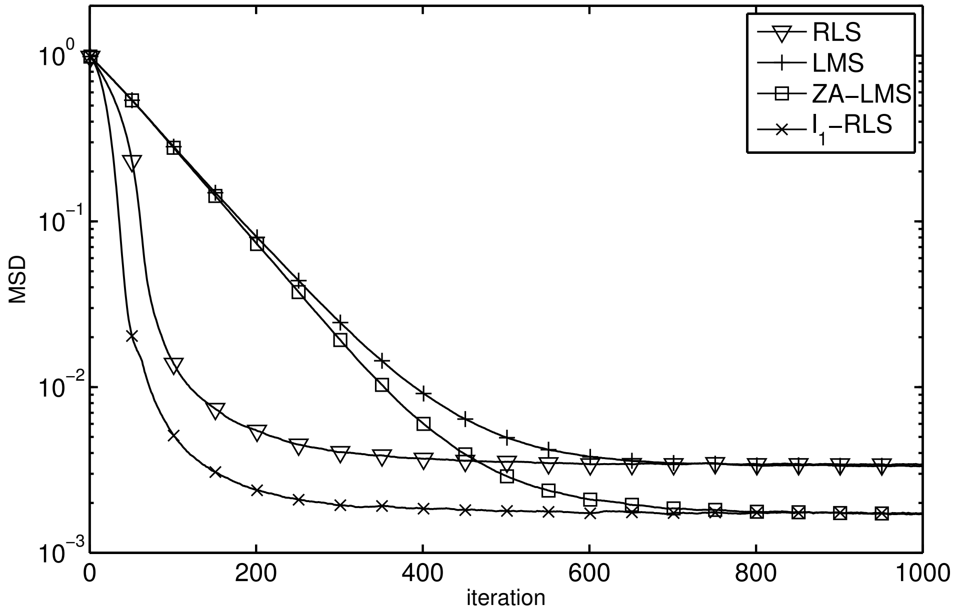

Figure 1: Learning curves for ℓ_1-RLS, RLS, ZA-LMS and LMS.

Simulation results

- The first experiment considers the tracking capabilities of ℓ_1-RLS, RLS, ZA-LMS (Chen2009) and LMS algorithms under white excitation.

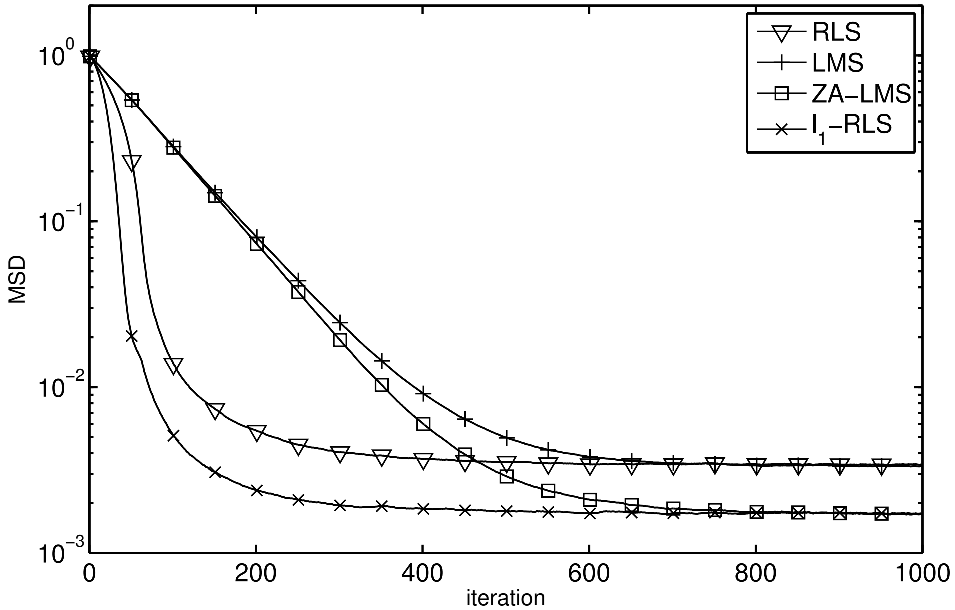

Figure 1: Learning curves for ℓ_1-RLS, RLS, ZA-LMS and LMS.

- ℓ_1-RLS presents convergence and steady-state error improvements over the regular RLS algorithm.

Simulation results

- In the second experiment we compare the performance of the novel ℓ_1-RLS algorithm to the regular RLS under different SNR values.

Simulation results

- In the second experiment we compare the performance of the novel ℓ_1-RLS algorithm to the regular RLS under different SNR values.

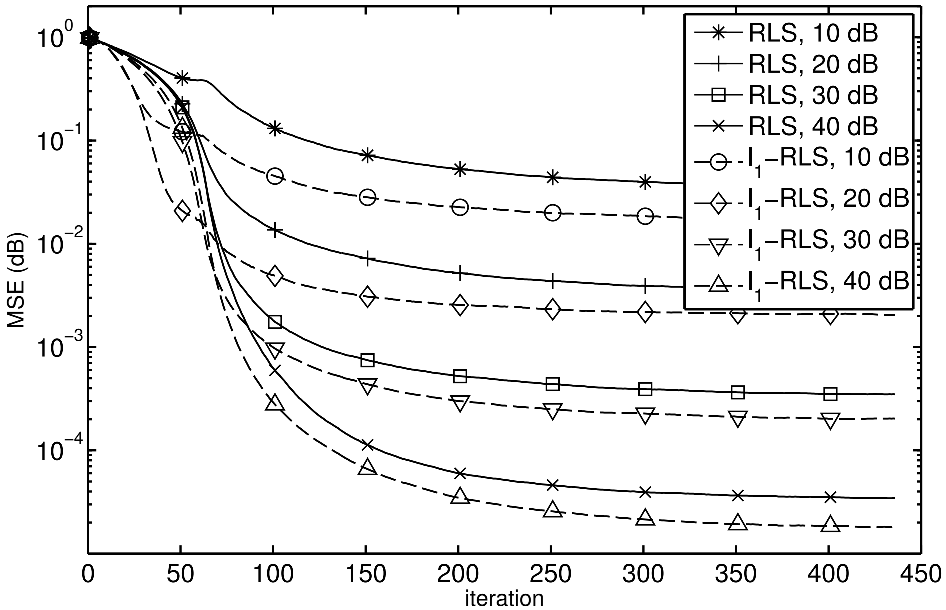

Figure 2: Learning curves for ℓ_1-RLS and RLS for SNR=40, 30, 20 and 10 dB.

Simulation results

- In the second experiment we compare the performance of the novel ℓ_1-RLS algorithm to the regular RLS under different SNR values.

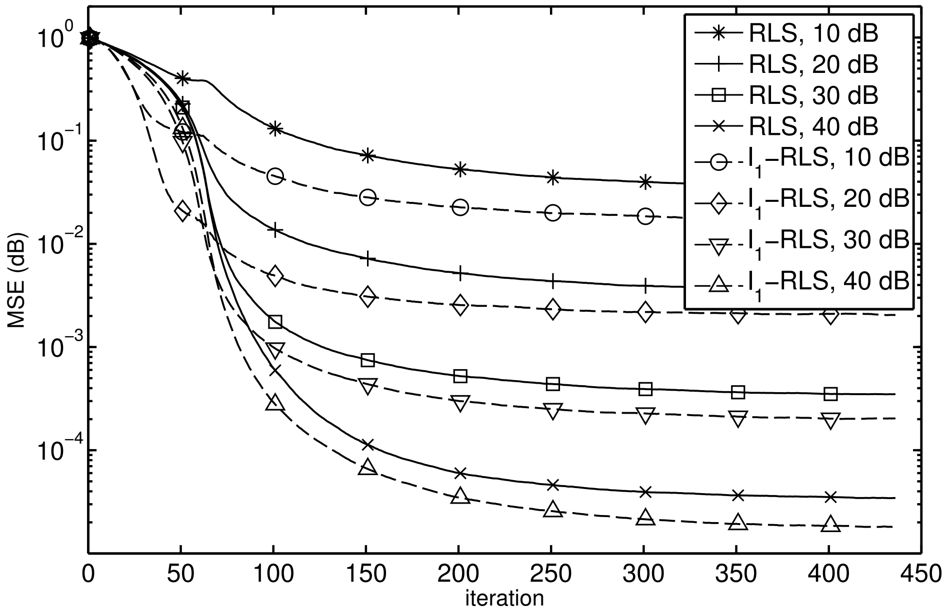

Figure 2: Learning curves for ℓ_1-RLS and RLS for SNR=40, 30, 20 and 10 dB.

- The ℓ_1-RLS has better convergence and steady-state properties than the regular RLS.

Conclusions

- This paper introduced a new RLS algorithm, namely ℓ_1-RLS, applicable for the adaptive identification of systems with sparse impulse response.

Conclusions

- This paper introduced a new RLS algorithm, namely ℓ_1-RLS, applicable for the adaptive identification of systems with sparse impulse response.
- The novel update equations for this algorithm are developed by regularizing the cost function with an ℓ_1 norm term.

Conclusions

- This paper introduced a new RLS algorithm, namely ℓ_1-RLS, applicable for the adaptive identification of systems with sparse impulse response.
- The novel update equations for this algorithm are developed by regularizing the cost function with an ℓ_1 norm term.
- Numerical simulations demonstrate that the algorithm indeed brings about better convergence and steady state performance than regular RLS.

Conclusions

- This paper introduced a new RLS algorithm, namely ℓ_1-RLS, applicable for the adaptive identification of systems with sparse impulse response.
- The novel update equations for this algorithm are developed by regularizing the cost function with an ℓ_1 norm term.
- Numerical simulations demonstrate that the algorithm indeed brings about better convergence and steady state performance than regular RLS.
- Future work might include theoretical analysis for the steady state error and simulations studying performance of the proposed algorithm in the case of sparse, slowly time-varying systems.

Thanks

Thanks

Thanks for listening.