Modified Combined-Step-Size Affine Projection Sign Algorithms for Robust Adaptive Filtering in Impulsive Interference Environments

Guoliang Li, Hongbin Zhang * and Ji Zhao *

School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; lgl-flash@std.uestc.edu.cn

* Correspondence: zhanghb@uestc.edu.cn (H.Z.); zhaoji@std.uestc.edu.cn (J.Z.);
Tel.: +86-028-6183-0106 (H.Z.); +86-130-8666-1621 (J.Z.)

Received: 1 January 2020; Accepted: 11 February 2020; Published: 3 March 2020

Abstract: In this paper, to further improve the filtering performance and enhance the poor tracking capability of the conventional combined step-size affine projection sign algorithm (CSS-APSA) in system identification, we propose a simplified CSS-APSA (SCSS-APSA) by applying the first-order Taylor series expansion to the sigmoidal active function (of which the independent variable is symmetric) of CSS-APSA. SCSS-APSA has lower computational complexity, and can achieve comparable, or even better filtering performance than that of CSS-APSA. In addition, we propose a modification of the sigmoidal active function. The modified sigmoidal active function is a form of scaling transformation based on the conventional one. Applying the modified function to the CSS-APSA, we can obtain the modified CSS-APSA (MCSS-APSA). Moreover, the extra parameter of MCSS-APSA provides the power to accelerate the convergence rate of CSS-APSA. Following the simplification operations of SCSS-APSA, the computational complexity of MCSS-APSA can also be reduced. Therefore, we get the simplified MCSS-APSA (SMCSS-APSA). Simulation results demonstrate that our proposed algorithms are able to achieve a faster convergence speed in system identification.

Keywords: affine projection sign algorithm (APSA); sigmoidal active function; combined step-size; system identification; non-Gaussian noise

1. Introduction

System identifications, including channel estimation, seismic system identification, noise or echo cancellation, and image recovery have been successfully realized by adaptive filters (AFs) [1]. Among conventional AF algorithms, the least-mean-square (LMS) and normalized LMS (NLMS) algorithms play important roles due to their low computational complexity and mathematical tractability [2,3]. However, such LMS-type algorithms suffer from filtering performance degradation under the heavy-tailed impulsive interference and heavy-colored input signal [4].

To remit the negative influence of heavy-colored input data on the convergence performance [5], derived the affine projection algorithm (APA), which is based on the idea of affine subspace projection. After that, the family of APA is developed by various methods [5–16]. Rather than a single current data vector used in LMS and NLMS, APA updates weight coefficients through multiple, most recent input vectors. Increasing the projection order of APA accelerates the convergence speed at the expense of computational complexity. However, APA is derived by the minimization of $\ell_2$-norm, which results in dramatic performance degradation under the non-Gaussian noise, especially for impulsive interference.

It is known that lower-order norms are robust against impulsive disturbances. From the perspective of computational simplicity, a great number of algorithms are derived by minimizing...
the $\ell_1$-norm to achieve robust filtering [10,17,18]. Moreover, [19] combined the AP method with $\ell_1$-norm optimization to derive a robust AP-like algorithm, which is termed the affine projection sign algorithm (APSA). To satisfy the contradictory requirements between fast convergence speed and small steady-state misalignment, the variable step-size APSA algorithms (VSS-APSAs) were proposed [20–22]. However, such VSS-APSAs reduce the tracking performance when the system abruptly changes. Therefore, to enhance the tracking performance of VSS-APSAs, two convex combination methods are applied to APSA from different viewpoints. One is called the combination of APSA (CAPSA), and the other is named the combined step-size APSA (CSS-APSA). The CSS-APSA uses two different step sizes—the large one is related to faster convergence, and the small one is related to a lower steady-state misadjustment. In addition, the convex combination of affine projection-type algorithms using an impulsive noise indicator was developed [23]. However, [23] needs more calculations to realize good convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

In this paper, applying the first-order Taylor series expansion to the sigmodal active function (SAF) of CSS-APSA, we were inspired by [24] and will propose a simplified CSS-APSA (SCSS-APSA), which has lower computational complexity. However, although CSS-APSA realizes fast convergence rates and small steady-state misadjustment, there is still an open problem of enhancing convergence performance. After that, the cost function based on the sigmodal active function was applied to a family of robust adaptive filtering algorithms, and it can be further simplified by the first-order Taylor series expansion [24].

2. Affine Projection Sign Algorithm Based on Step-size Adjustment

We review the traditional APSA and its variable step-size forms, which include the VSS-APSA [21] and CAPSA.

2.1. APSA

In the conventional APSA, the desired output signal is given by

$$d(n) = u^T(n)w^0 + v(n),$$  

where $n$ is the time index; superscript $T$ represents transposition of matrix or vector; $w^0 = [w_0^0, w_1^0, ..., w_{M-1}^0]^T$ denotes an unknown system weight vector, which needs to be estimated; $u(n) = [u(n), u(n-1), ..., u(n-M+1)]^T$ is the input signal vector with length $M$; and $v(n)$ is the background noise with the addition of an interference signal.

Let $U(n)$, $d(n)$, $e(n)$, and $e_p(n)$ be the input matrix, desired signal vector, error vector, and a posteriori error vector, respectively, as follows:

$$U(n) = [u(n), u(n-1), ..., u(n-K+1)],$$

$$d(n) = [d(n), d(n-1), ..., d(n-K+1)]^T,$$

$$e(n) = d(n) - U^T(n)w(n-1),$$
\[ e_p(n) = d(n) - U^T(n)w(n), \tag{5} \]

where \( K \) is the affine projection order. The APSA algorithm can be derived by minimizing the \( \ell_1 \)-norm of a posteriori error vector \( e_p(n) \) with a constraint on the filter coefficients, as

\[
\min_{w(n)} \| d(n) - U^T(n)w(n) \|_1, \tag{6}
\]

subject to \( \| w(n) - w(n-1) \|^2_2 \leq \mu^2, \tag{7} \)

where \( \mu^2 \) is a parameter which is used to ensure the stability of the weight coefficients vector \( w(n) \) during updating. Using the method of Lagrange multipliers \([19]\), the cost function can be solved by combining (6) and (7). Then, the updated equation for the weight vector is

\[
w(n) = w(n-1) + \mu \frac{U_s(n-1)}{\sqrt{\xi_1 + \| U_s(n-1) \|^2_2}}, \tag{8}
\]

where \( \mu > 0 \) is the step size; \( 0 < \xi_1 \ll 1 \) is to avoid division by zero; \( \| \cdot \|_2 \) is the \( \ell_2 \)-norm of the vector; and \( U_s(n-1) = U(n-1)\text{sgn}(e(n-1)) \) with \( \text{sgn}(e(n-1)) \) being the sign operation on each element of the error vector.

### 2.2. VSS-APSA and CAPSA

A variable step size is based on the minimization of mean-square deviation \([21]\), and given as follows:

\[
\mu_{vs}(n) = \alpha_{vs}\mu_{vs}(n-1) + (1 - \alpha_{vs}) \times \min\left(\frac{\| e(n) \|_1}{\sqrt{\| U_s(n-1) \|^2_2}}, \mu_{vs}(n-1)\right) \tag{9}
\]

where \( \alpha_{vs} \) is the smoothing factor. Although the VSS-APSA has good performance in terms of fast convergence speed and small steady-state misalignment, its tracking performance is very poor when an abrupt change occurs.

Therefore, to enhance the tracking performance of VSS-APSA, the convex combination method was applied to APSA (CAPSA). For APSA algorithms, two different step sizes, that is, \( \mu_1 > \mu_2 > 0 \), are related to the faster convergence rate and lower steady-state error, respectively. The corresponding weight updates are defined as

\[
w_1(n) = w_1(n-1) + \mu_1 \frac{U_s(n-1)}{\sqrt{\xi_1 + \| U_s(n-1) \|^2_2}}, \tag{10}
\]

and

\[
w_2(n) = w_2(n-1) + \mu_2 \frac{U_s(n-1)}{\sqrt{\xi_1 + \| U_s(n-1) \|^2_2}}. \tag{11}
\]

Combining \( w_1(n) \) with \( w_2(n) \) by a variable mixing factor \( \lambda(n) \in [0,1] \) results in:

\[
w(n) = \lambda(n)w_2(n) + (1 - \lambda(n))w_1(n). \tag{12}
\]

Generally, \( \lambda(n) \) is updated indirectly based on reduction of an overall error \( e(n) \) through a sigmoidal active function (SAF) \([27,29]\), that is,
\[
\lambda(n) = \frac{1}{1 + \exp(-s(n))}.
\]

where \( s(n) \) is based on a gradient descent scheme by using the \( \ell_1 \)-norm minimization to ensure robustness in the presence of impulsive noise, as

\[
s(n) = s(n - 1) - \mu \frac{\partial |e(n)|}{\partial s(n - 1)}
= s(n - 1) + \mu \mu_1 12 \frac{\mathbf{u}^T(n)\mathbf{u}_s(n-1)\text{sgn}(e(n))}{\sqrt{\xi_1 + \|\mathbf{u}_s(n-1)\|^2}}.
\]

where \( \mu > 0 \) is a step size, and \( \mu_1 12 = (\mu_2 - \mu_1)[\lambda(n - 1)(1 - \lambda(n - 1)) + \xi_2] \), in which the factor \( \xi_2 \) is positive and prevents the update of \( s(n) \) from halting when \( \lambda(n - 1) \) is equal to 1 or 0. Generally, \( s(n) \) is restricted in a symmetric interval, that is, \([-4, 4]\). (12) is the weight update of CAPSA when \( s(n) < 4 \) and becomes \( \mathbf{w}(n) = \alpha_c \mathbf{w}_2(n) + (1 - \alpha_c) \mathbf{w}_1(n) + \mu_2 \frac{\mathbf{u}_s(n-1)}{\sqrt{\xi_1 + \|\mathbf{u}_s(n-1)\|^2}} \) only if \( s(n) = 4 \). \( \alpha_c (0 \leq \alpha_c \leq 1) \) has the same meaning as \( \alpha_{\text{us}} \). However, from [25], we know that there is still an open problem to enhance the convergence performance of CAPSA.

### 3. Proposed Algorithms

We start by introducing the idea of the combined step-size APSA (CSS-APSA) algorithm. Then, we analyze the behavior of the mixing factor for CSS-APSA, and explain the reason why the mixing factor can be simplified. Finally, we derive the proposed algorithms.

In order to better identify the optimization problem, the cost functions need to be known. The CSS-APSA algorithm is derived by minimizing the \( \ell_1 \)-norm of a posteriori error vector \( \mathbf{e}_p(n) \) with a constraint on the filter coefficients, as

\[
\min_{\mathbf{w}(n)} \|\mathbf{d}(n) - \mathbf{U}^T(n)\mathbf{w}(n)\|_1,
\]

subject to \( \|\mathbf{w}(n) - \mathbf{w}(n-1)\|^2_2 \leq \mu_{12}^2 \),

where

\[
\mu_{12} = \lambda(n)\mu_2 + (1 - \lambda(n))\mu_1.
\]

Similar to the APSA [19], we can obtain the following:

For the SCSS-APSA:

\[
J_s(\mathbf{w}(n)) = \|\mathbf{d}(n) - \mathbf{U}^T(n)\mathbf{w}(n)\|_1 + \varphi[\|\mathbf{w}(n) - \mathbf{w}(n-1)\|^2_2 - \mu_{s12}^2]
\]

For the MCSS-APSA:

\[
J_\beta(\mathbf{w}(n)) = \|\mathbf{d}(n) - \mathbf{U}^T(n)\mathbf{w}(n)\|_1 + \varphi[\|\mathbf{w}(n) - \mathbf{w}(n-1)\|^2_2 - \mu_{p12}^2]
\]

For the SMCSS-APSA:

\[
J_{s\beta}(\mathbf{w}(n)) = \|\mathbf{d}(n) - \mathbf{U}^T(n)\mathbf{w}(n)\|_1 + \varphi[\|\mathbf{w}(n) - \mathbf{w}(n-1)\|^2_2 - \mu_{sb12}^2]
\]
where φ is a Lagrange multiplier, μ₁₂ = λ₁(n)μ₂ + (1 − λ₁(n))μ₁, μ₂₁ = λ₂(n)μ₂ + (1 − λ₂(n))μ₁, μ₁₂ = λ₀(n)μ₂ + (1 − λ₀(n))μ₁, J₁(·), J₂(·), and J₀(·) represent the cost functions and discuss the computational complexity.

3.1. CSS-APSA

Substituting (10) and (11) into (12) yields the weight update of CSS-APSA. (According to the symmetry, (12) can also be written as \( w(n) = \lambda(n)w_1(n) + (1 - \lambda(n))w_2(n) \) [25]) as

\[
\begin{align*}
\lambda(n) &= \frac{C}{1 + \exp(-s(n))} - \left(\frac{C}{2} - \frac{1}{2}\right), \\
\lambda(n) &= \frac{w(n) - 1 + [\lambda(n)μ₂ + (1 - \lambda(n))μ₁] - \lambda(n)w_1(n) + (1 - \lambda(n))w_2(n)}{\sqrt{\gamma_1 + \|U_1(n - 1)\|^2}}. \\
\end{align*}
\]

(21)

3.2. Analysis of Variable Mixing Factors for CSS-APSA

Actually, the CSS-APSA algorithm uses the following SAF to adjust the mixing parameter

\[
\begin{align*}
\lambda(n) &= \frac{C}{1 + \exp(-s(n))} - \left(\frac{C}{2} - \frac{1}{2}\right), \\
s(n) &= \begin{cases} 
-\ln\left(\frac{C+1}{2}\right), & s(n) < -\ln\left(\frac{C+1}{2}\right) \\
\ln\left(\frac{C+1}{2}\right), & s(n) > \ln\left(\frac{C+1}{2}\right) \\
\end{cases} \\
\theta(n), & \text{elsewhere.} \\
\end{align*}
\]

(23)

According to (15), we can obtain

\[
\frac{\partial \lambda(n)}{\partial s(n)} = \frac{C \exp(-s(n))}{(1 + \exp(-s(n)))^2}. \\
\]

(24)

From (24), we know that \( \lambda(n) \) is a monotone increasing function of \( s(n) \). In other words, the increase rate of \( \lambda(n) \) from 0 to 1 can be accelerated by increasing the value of \( C \). Figure 1 shows the function curves of (22) and (23). From this figure, we know that \( \lambda(n) \) is approximately equal to zero or one when \( |s(n)| = 0.3365 \). Actually, the range of \( s(n) \) is affected by two constraint conditions, namely, \( s(n) \in [-4, 4] \) and \( \lambda(n) \in [0, 1] \). Therefore, from (23) and such two constraints, we can obtain

\[
\begin{align*}
s(n) &\in [s_-, s_+] \\
s_- &= \max\left(-4, \ln\left(\frac{C+1}{2}\right)\right) \\
s_+ &= \min\left(4, \ln\left(\frac{C+1}{2}\right)\right) \\
\end{align*}
\]

(25)

Furthermore, from (23) and (25), we know that:

- When the \( C \) value is small, such as \( C = 2 \), we can get \( s_- = -\ln\left(\frac{C+1}{2}\right) = -1.0986 \) and \( s_+ = 1.0986 \). When \( |s(n)| = 0.3365 \), \( \lambda(n) \) is approximately equal to 1 or 0. However, such two \( \lambda(n) \) values will be slowly increased to 1 or decreased to 0 as \( |s(n)| \) becomes larger. Therefore, the effective range for \( s(n) \) can be treated as \([-0.3365, 0.3365]\).

- From (23), as the \( C \) value becomes larger, \( |s(n)| \) becomes closer to 0.3365 because \( \ln\left(\frac{C+1}{2}\right) \) is a monotone decreasing function of \( C \). Also, when the \( C \) value is relatively large, such as \( C = 6 \), \( \lambda(n) \) will be very close to 1 or 0. Therefore, for the large \( C \) value, we can constrain \( s(n) \in [-0.3365, 0.3365] \).
3.3. Practical Considerations about Proposed Algorithms

In this paper, to reduce the error of $\lambda(n)$ being 1 or 0, as shown in Figure 1 when $C \leq 5$, we can appropriately extend the effective range of $s(n)$ to $[-0.4, 0.4]$. In addition, when $s(n) = 0.4$, the difference between $\exp(-s(n))$ and $1 - s(n)$ is 0.0703. Ignoring such differences and applying the first-order Taylor series expansion to $\exp(-s(n))$, we can use $1 - s(n)$ to approximate $\exp(-s(n))$. Hence, (22) can be simplified as

$$
\lambda_s(n) = \frac{C}{2 - s(n)} - \left(\frac{C}{2} - \frac{1}{2}\right). \tag{26}
$$

Based on (26), we get

$$
\begin{align*}
\{ s(n) = \frac{-2}{C+1} \rightarrow & \quad \lambda(n) = 0 \\
\{ s(n) = \frac{2}{C+1} \rightarrow & \quad \lambda(n) = 1.
\end{align*}
$$

Therefore, injecting (27) into (26) yields

$$
\lambda_s(n) = \begin{cases} 
0, & s(n) < \frac{-2}{C+1} \\
1, & s(n) > \frac{2}{C+1} \\
\frac{C}{2 - s(n)} - \left(\frac{C}{2} - \frac{1}{2}\right), & \text{ elsewhere}
\end{cases} \tag{28}
$$

with $C > 1$. Also, applying (28) to CSS-APSA, we can get the simplified CSS-APSA (denoted as: SCSS-APSA).

Comparing (22) with (13), one can find that (22) is a kind of scaling transformation of (13). Such a transformation can speed up the change rate of $\lambda(n)$. Based on this idea, (22) can be modified as follows:

$$
\lambda_{\beta}(n) = \frac{C}{1 + \exp(-\beta s(n))} - \left(\frac{C}{2} - \frac{1}{2}\right), \tag{29}
$$
where $\beta$ is a positive constant, and $s(n)$ is restricted as

$$
s(n) = \begin{cases} 
-\ln\left(\frac{C+1}{C-1}\right)/\beta, & s(n) < -\ln\left(\frac{C+1}{C-1}\right)/\beta \\
\ln\left(\frac{C+1}{C-1}\right)/\beta, & s(n) > \ln\left(\frac{C+1}{C-1}\right)/\beta \\
s(n), & \text{elsewhere}
\end{cases}
$$

(30)

Following the modification of (29), different $\beta$ values can accelerate or slow down the change rate of $\lambda_{\beta}(n)$. The change rate may result in a faster convergence rate of the proposed algorithm. Therefore, CSS-APSA armed with (29) and (30) is called a modified CSS-APSA (denoted as MCSS-APSA). Obviously, CSS-APSA is a special case of MCSS-APSA when the $\beta = 1$. In addition to (29), when $C = 1$, $s(n)$ is restricted as

$$
s(n) = \begin{cases} 
-4/\beta, & s(n) < -4/\beta \\
4/\beta, & s(n) > 4/\beta \\
s(n), & \text{elsewhere}
\end{cases}
$$

(31)

Based on (16) and (23), the range of $\beta s(n)$ for MCSS-APSA is the same as that of $s(n)$ for CSS-APSA. Therefore, according to the operations for (26), when $C > 1$ and $\beta > 0$, (29) can also be simplified as

$$
\lambda_{\beta}(n) = \frac{C}{2 - \beta s(n)} - \left(\frac{C}{2} - \frac{1}{2}\right),
$$

(32)

and $s(n)$ is restricted as

$$
s(n) = \begin{cases} 
\frac{-2}{(C-1)\beta}, & s(n) < \frac{-2}{(C-1)\beta} \\
\frac{2}{(C+1)\beta}, & s(n) > \frac{2}{(C+1)\beta} \\
s(n), & \text{elsewhere}
\end{cases}
$$

(33)

Therefore, injecting (32) and (33) into CSS-APSA, we can obtain the simplified MCSS-APSA (denoted as SMCSS-APSA). Algorithm 1 summarizes the proceeding of the proposed algorithms.

### 3.4. Computational Complexity

In Table 1, the computational complexity of the proposed SCSS-APSA, MCSS-APSA, and SMCSS-APSA are compared with that of APSA, CAPSA, and CSS-APSA in terms of the total number of additions, multiplications, comparisons, and exponents. It is clear that the simplified algorithms are more efficient in computation than CAPSA and CSS-APSA.

**Table 1.** Computational complexity of APSA, CAPSA, CSS-APSA, SCSS-APSA, MCSS-APSA, and SMCSS-APSA at each iteration.

| Algorithms       | Multiplications | Additions | Comparisons | Exponents |
|------------------|-----------------|-----------|-------------|-----------|
| APSA             | $2M(K+1) + 1$   | $M(2K+1) - 1$ | 0           | 0         |
| CAPSA            | $2M(2K+3) + 12$ | $M(4K+4) + 6$ | 2           | 0         |
| CSS-APSA         | $2M(K+1) + M + 8$ | $M(2K+2) + 4$ | 2           | 2         |
| SCSS-APSA        | $2M(K+1) + M + 8$ | $M(2K+2) + 4$ | 2           | 0         |
| MCSS-APSA        | $2M(K+1) + M + 8$ | $M(2K+2) + 4$ | 2           | 2         |
| SMCSS-APSA       | $2M(K+1) + M + 8$ | $M(2K+2) + 4$ | 2           | 0         |
Algorithm 1: The proposed Algorithms

Initialization:
0 < \mu_2 < \mu_1, 0 < \xi_1 \leq 1, s(0) = \xi_1, 1 < C, 1 \leq \beta, and w_1(0) = w_2(0) = 0.

Computation:
while \{u(n), d(n)\}_{n \geq 1} available do
1 : w_1(n) = w_1(n) + \mu_1 \frac{U_s(n-1)}{\sqrt{\xi_1 + \|U_s(n-1)\|_2^2}} w_s(n),
2 : w_2(n) = w_2(n) + \mu_2 \frac{U_s(n-1)}{\sqrt{\xi_1 + \|U_s(n-1)\|_2^2}} w_s(n),
3 : w(n) = \lambda(n)w_2(n) + (1 - \lambda(n))w_1(n),
4 : e(n) = d(n) - u^n w(n),
5 : s(n) = s(n - 1) - \mu_2 \frac{\partial e(n)}{\partial|e(n)|}, for SCSS-APSA
6 : \lambda_s(n) = \frac{C}{2-s(n)} - \left(\frac{C}{2} - \frac{1}{2}\right),
7 : if s(n) < -ln \left(\frac{C+1}{C} \right):
then s(n) = -ln \left(\frac{C+1}{C} \right),
8 : else if s(n) > ln \left(\frac{C+1}{C} \right):
then s(n) = ln \left(\frac{C+1}{C} \right),
9 : else s(n) = s(n), end.
for MCSS-APSA
6 : \lambda_\beta(n) = \frac{C}{1+exp(-\beta s(n))} - \left(\frac{C}{2} - \frac{1}{2}\right),
7 : if s(n) < -ln \left(\frac{C+1}{C} \right)/\beta:
then s(n) = -ln \left(\frac{C+1}{C} \right)/\beta,
8 : else if s(n) > ln \left(\frac{C+1}{C} \right)/\beta:
then s(n) = ln \left(\frac{C+1}{C} \right)/\beta,
9 : else s(n) = s(n), end.
for SMCS-APSA
6 : \lambda_\beta(n) = \frac{C}{2-\beta s(n)} - \left(\frac{C}{2} - \frac{1}{2}\right),
7 : if s(n) < -\frac{1}{(C-1)\beta}:
then s(n) = -\frac{1}{(C-1)\beta},
8 : else if s(n) > -\frac{1}{(C+1)\beta}:
then s(n) = -\frac{1}{(C+1)\beta},
9 : else s(n) = s(n), end.
end while

3.5. Steady-State Analysis

According to [24], the weight-error vector is defined as
$$\bar{w}(n) = w^o - w(n).$$ (34)

Using (12), we can obtain:
$$\bar{w}(n) = \bar{w}(n-1) - \mu_{\bar{d}} \frac{U_s(n-1)}{\sqrt{\|U_s(n-1)\|_2^2}}$$ (35)
where $E$ is the expectation operator. The mean-square-deviation $E[\|\hat{w}(n)\|^2]$ must be less than $E[\|\hat{w}(n-1)\|^2]$ to ensure the algorithm is stable, so the step-size $\mu_{12}$ must satisfy the following condition:

$$0 < \mu_{12} < 2 \times \frac{E[\hat{w}^T(n-1)U_s(n-1)]}{E[\|U_s(n-1)\|^2/\sqrt{\|U_s(n-1)\|^2}]}.$$  \hspace{1cm} (36)

For the SCSS-APSA, the step-size should satisfy the condition below:

$$0 < \mu_{s12} = \lambda_s(n)\mu_2 + (1 - \lambda_s(n))\mu_1 < 2 \times \frac{E[\hat{w}^T(n-1)U_s(n-1)]}{E[\|U_s(n-1)\|^2/\sqrt{\|U_s(n-1)\|^2}]}.$$  \hspace{1cm} (38)

For the MCSS-APSA, the step-size should satisfy the condition below:

$$0 < \mu_{b12} = \lambda_b(n)\mu_2 + (1 - \lambda_b(n))\mu_1 < 2 \times \frac{E[\hat{w}^T(n-1)U_s(n-1)]}{E[\|U_s(n-1)\|^2/\sqrt{\|U_s(n-1)\|^2}]}.$$  \hspace{1cm} (39)

For the SMCSS-APSA, the step-size should satisfy the condition below:

$$0 < \mu_{bs12} = \lambda_{sb}(n)\mu_2 + (1 - \lambda_{sb}(n))\mu_1 < 2 \times \frac{E[\hat{w}^T(n-1)U_s(n-1)]}{E[\|U_s(n-1)\|^2/\sqrt{\|U_s(n-1)\|^2}]}.$$  \hspace{1cm} (40)

In addition, $\lambda(n) = 0$ in the transient state and $\lambda(n) = 1(n \to \infty)$ in the steady state, such that $\mu_1$ or $\mu_2$, should also satisfy condition (36). Also, from [19], we know that $0 < \mu_2, \mu_1 < 1$, so the ranges of $\mu_{s12}, \mu_{b12},$ and $\mu_{bs12}$ are restricted as

$$0 < \mu_{s12}, \mu_{b12}, \mu_{bs12} < 1.$$  \hspace{1cm} (41)

4. Numerical Simulation Results

In this part, we test the filtering performance of the proposed algorithms in system identification. In this work, all simulation results have been averaged over 50 independent trials. The signal-to-noise ratio (SNR) was set to 30 dB, and background noise has been added to $d(n)$.

The impulsive noise $v(n)$ is modeled by $b(n)G(n)$, where $b(n)$ is a Bernoulli process with a probability of success of $P[b(n) = 1] = p_r$, and $G(n)$ denotes zero-mean Gaussian noise with the variance $\sigma_G^2 = 1000\sigma_v^2$, where $\sigma_v^2$ is the average power of the desired output $d(n) = u^T(n)w^s$. We set $\xi_1 = \xi_2 = 0.001, K = 2, p_r = 0.1, \mu_1 = 0.01,$ and $\mu_2 = 0.0001$ in all simulations. Following normalized MSD,

$$\text{NMSD} = 10 \log_{10} \left( \frac{\|w - w(n)\|_2}{\|w^s\|_2} \right)$$  \hspace{1cm} (42)

was used to measure the performance of the algorithms.

The color input datum were generated by filtering the white and zero-mean Gaussian signals with a unit power through a first-order system, as

$$H_1 = \frac{1}{1 - 0.7z^{-1}}.$$  \hspace{1cm} (43)
The unknown weight $w^o$ was randomly generated with $M = 32$ taps, and turned into $-w^o$ at the middle of the iterations. First of all, we investigated the influence of the parameter $C$ on the performance of CSS-APSA. We chose $C \in \{1.1, 2, 3, 4, 5, 6, 7\}$, $\mu_s = 1$. Figure 2 shows the corresponding NMSD curves. From this figure, we know that CSS-APSA with $C = 5$ and $C = 6$ can realize better filtering performance than that of other $C$ values. In other words, proper $C$ values can enhance the filtering performance of CSS-APSA. This result demonstrates the effectiveness of constraint on $s(n) \in [-0.3365, 0.3365]$.

**Remark 1.** For CSS-type algorithms, when $C$ is in a certain interval (such as $C \in (1, 6]$), the performance of CSS-APSA can be improved by increasing the value of $C$. In addition, we can have following observations for $C \in (1, 6]$:

- If the value of $C$ is relatively small (such as $C \leq 2$), the change rate of $\lambda(n)$ from 0 to 1 will be reduced. In this case, the combined algorithm has slower convergence speed. Figures 1 and 2 illustrate such behavior.
- If the value of $C$ is too large (such as $C = 7$), the change rate of $\lambda(n)$ from 0 to 1 is increased in the process of iteration. In this case, $\lambda(n)$ tends to 1 too quickly, resulting in $w(n)$ dominated by $w_2(n)$, which is related to the lower steady-state error. Hence, the combined algorithm will still realize a slower convergence speed.

Secondly, we investigate the influence of $\mu_s$ on CSS-APSA with $C \in \{5, 6\}$, since the $\mu_s$ as a step-size factor affects the update rate of $s(n)$ and indirectly influences the change rate of $\lambda(n)$. In this part, we set $\mu_s \in \{0.1, 1, 10\}$. Figure 3 plots the corresponding NMSD curves. From this figure, we know that $\mu_s = 1$ is a better choice than other values. Hence, except where mentioned otherwise, we set $\mu_s = 1$ in the following experiments. Furthermore, we compare the filtering performance between CSS-APSA and SCSS-APSA. Figure 4 plots the NMSD curves of the corresponding algorithms. Such results reveal that:

- CSS-APSA with $C = 6$ realizes faster convergence than $C = 5$ after an abrupt change;
- SCSS-APSA with $C = 6$ outperforms CSS-APSA in terms of convergence rate in the whole filtering process.

Moreover, Table 2 compares the execution times of CSS-APSA, SCSS-APSA, MCSS-APSA, and SMCSS-APSA at each iteration (computation platform: Intel(R) Core(TM) i7-4770K, CPU @ 3.50 GHz). One can clearly see that SCSS-APSA is more computationally efficient than CSS-APSA.

![Figure 2](image-url)  
**Figure 2.** The NMSD curves of APSA and CSS-APSA with different $C$ values.
Thirdly, we investigate the effect of parameter \( \beta \) on the performance of MCSS-APSA. We set \( C = 6 \) and \( \beta \in \{1, 5, 10, 20\} \). The NMSD curves are plotted in Figure 5, which shows that:

- With \( \beta = 5 \), MCSS-APSA realizes better filtering accuracy at a transient process than \( \beta \in \{1, 10, 20\} \);
- With \( \beta = 1 \), MCSS-APSA becomes to CSS-APSA, while MCSS-APSA with other \( \beta \) values (such as \( \beta = 5 \) or 10) outperforms CSS-APSA. This observation demonstrates the usefulness of modifications for CSS-APSA.
- Increasing the \( \beta \) value from 1 to 20 cannot always improve the filtering performance of MCSS-APSA. Similarly to the effect of the \( C \) value, a \( \beta \) value which is too large leads to \( \mathbf{w}(n) \) dominated by \( \mathbf{w}_2(n) \) in iterations, and slows down the convergence speed of MCSS-APSA.

Next, we compare the filtering performance between SMCSS-APSA and MCSS-APSA. In this part, we set \( \beta = 5 \) and \( C = 6 \), since such two parameters can improve the tracking performance of MCSS-APSA, as demonstrated in Figure 5. The compared NMSD results are plotted in Figure 6. From this figure, we can know that SMCSS-APSA achieves slightly better filtering behavior than that of MCSS-APSA, while the former is more computationally efficient than the latter, as demonstrated in Table 2.
Table 2. Execution time of CSS-APSA, SCSS-APSA, MCSS-APSA, and SMCSS-APSA at each iteration.

| Algorithms      | Parameters | Execution Time (s) |
|-----------------|------------|--------------------|
| CSS-APSA        | C = 6      | 1.7132             |
| SCSS-APSA       | C = 6      | 1.6952             |
| MCSS-APSA       | C = 6, β = 5 | 1.7467           |
| SMCSS-APSA      | C = 6, β = 5 | 1.7108           |

Figure 5. The NMSD curves of MCSS-APSA with different β, C = 6.

Figure 6. The NMSD curves of MCSS-APSA and SMCSS-APSA with β = 5, C = 6.

Finally, the filtering performance of SMCSS-APSA is compared with APSA, CAPSA, and SCSS-APSA (From [25], we know the VSS-APSA [21] need to reset algorithms when an abrupt change occurs, so we ignore the VSS-APSA's comparison in system identification). Figure 7 shows the NMSD curves of these algorithms. The results demonstrate that SMCSS-APSA outperforms APSA, CAPSA, and SCSS-APSA in terms of the convergence rate.
Symmetry 2020, 12, 385

Figure 7. The NMSD learning curves of APSA, CAPSA, SCSS-APSA, and SMCSS-APSA with colored input data $H_1(z)$, (a): $\mu_1 = 0.01$; (b): $\mu_2 = 0.0001$; (c): $\mu_1 = 0.01, \mu_2 = 0.0001$; (d): $C = 6$; (e): $C = 6$, $\beta = 5$.

5. Conclusions

In this paper, SCSS-APSA, MCSS-APSA, and SMCSS-APSA were derived to improve the convergence speed of CSS-APSA. To enhance the tracking performance of CAPSA and CSS-APSA, MCSS-APSA was proposed by using a modified sigmoidal active function. Furthermore, the simplification method was used on MCSS-APSA, resulting in SMCSS-APSA, which was found to achieve better filtering performance than that of MCSS-APSA. In addition, SCSS-APSA not only reduced the computational complexity of CSS-APSA, but also sped up the convergence under certain conditions. Simulations have shown that the proposed SCSS-APSA, MCSS-APSA, and SMCSS-APSA are superior to CSS-APSA.

Author Contributions: G.L. presented the conception of this article and carried out the experimental work; project administration, H.Z.; J.Z. reviewed this article and put forward some constructive suggestions for the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grants No. 61971100).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Guo, Y.; Li, J.; Li, Y. Diffusion correntropy subband adaptive filtering (SAF) algorithm over distributed smart dust networks. Symmetry 2019, 11, 1335. [CrossRef]
2. Duttweiler. D.L. Proportionate normalized least-mean-squares adaptation in echo cancelers. IEEE Trans. Speech Audio Process. 2000, 8, 508–517. [CrossRef]
3. Wang, Y.; Li, Y. Norm penalized joint-optimization NLMS algorithms for broadband sparse adaptive channel estimation. Symmetry 2017, 9, 133. [CrossRef]
4. Min, S.; Nikias, C.L. Signal processing with fractional lower order moments: Stable processes and their applications. Proc. IEEE 1993, 81, 986–1010.
5. Ozeki, K.; Umeda, T.; Members, R. An adaptive filtering algorithm using an orthogonal projection on to an affine subspace and its properties. Electron. Commun. Jpn. 1984, 67, 126–132. [CrossRef]
6. Shin, H.C.; Sayed, A.H.; Song, W.J. Variable step-size nlms and affine projection algorithms. IEEE Signal Process. Lett. 2004, 11, 132–135. [CrossRef]
7. Lee, C.H.; Park, P. Optimal step-size affine projection algorithm. *IEEE Signal Process. Lett.* 2012, 19, 431–434. [CrossRef]

8. Paul, T.K.; Ogunfunmi, T. On the convergence behavior of the affine projection algorithm for adaptive filters. *IEEE Trans. Circuits Syst. Regul. Pap.* 2011, 58, 1813–1826. [CrossRef]

9. Yin, W.; Mehr, A.S. A variable regularization method for affine projection algorithm. *IEEE Trans. Circuits Syst. I: Express Briefs* 2010, 57, 476–480.

10. Zhao, J.; Zhang, H.; Wang, G.; Liao, X. Modified memory-improved proportionate affine projection sign algorithm based on correntropy induced metric for sparse system identification. *Electron. Lett.* 2018, 54, 630–632. [CrossRef]

11. Li, Y.; Jiang, Z.; Yin, J. Mixed norm constrained sparse apa algorithm for satellite and network echo channel estimation. *IEEE Access* 2018, 6, 65901–65908. [CrossRef]

12. Sivabalan, S.R.A. Za-apa with zero attractor controller selection criterion for sparse system identification. *Signal Image Video Process.* 2018, 12, 371–377.

13. Meng, R.; Lamare, R.C.D.; Nascimento, H. Sparsity-aware affine projection adaptive algorithms for system identification. In Proceedings of the Sensor Signal Processing for Defence (SSPD 2011), London, UK, 27–29 September 2011; pp. 27–29.

14. Li, Y.; Zhang, C.; Wang, S. Low-complexity non-uniform penalized affine projection algorithm for sparse system identification. *Circuits Syst. Signal Process.* 2016, 35, 1611–1624. [CrossRef]

15. Li, Y.; Li, W.; Yu, W.; Wan, J.; Li, Z. Sparse adaptive channel estimation based on lp-norm-penalized affine projection algorithm. *Int. Antennas Propag.* 2014, 17, 23–24.

16. Li, Y.; Hamamura, M. Smooth approximation l0-norm constrained affine projection algorithm and its applications in sparse channel estimation. *Sci. World J.* 2014, 937252.

17. Arikan, O.; Cetin, A.E.; Erzin, E. Adaptive filtering for non-gaussian stable processes. *IEEE Signal Process. Lett.* 1994, 1, 163–165. [CrossRef]

18. Kwong, C. Dual sign algorithm for adaptive filtering. *IEEE Trans. Commun.* 1986, 34, 1272–1275. [CrossRef]

19. Shao, T.; Zheng, Y.R.; Member, S.; Benesty, J.; Member, S. An affine projection sign algorithm robust against impulse interferences. *IEEE Signal Process. Lett.* 2010, 17, 327–330. [CrossRef]

20. Yoo, J.; Shin, J.; Park, P. Variable step-size affine projection sign algorithm. *IEEE Trans. Circuits Syst. II Express Briefs* 2014, 61, 274–278.

21. Shin, J.; Yoo, J.; Park, P. Variable step-size affine projection sign algorithm. *Electron. Lett.* 2012, 48, 9–10. [CrossRef]

22. Zhang, S.; Zhang, J. Modified variable step-size affine projection sign algorithm. *Electron. Lett.* 2013, 49, 1264–1265. [CrossRef]

23. Kim, S.H.; Jeong, J.J. Robust convex combination of affine projection-type algorithms using an impulse noise indicator. *Signal Process.* 2016, 129, 33–37. [CrossRef]

24. Huang, F.; Zhang, J.; Zhang, S. A family of robust adaptive filtering algorithms based on sigmoid cost. *Signal Process.* 2018, 149, 179–192. [CrossRef]

25. Huang, F.; Zhang, J.; Zhang, S. Combined-step-size affine projection sign algorithm for robust adaptive filtering in impulse interference environments. *IEEE Trans. Circuits Syst. Express Briefs* 2016, 63, 493–497. [CrossRef]

26. Choi, J.H.; Cho, H.; Jeong, J.J.; Kim, W. Combination of step sizes for affine projection algorithm with variable mixing parameter. *Electron. Lett.* 2013, 49, 29–30. [CrossRef]

27. Arenas-García, J.; Gómez-Verdejo, V. New algorithms for improved adaptive convex combination of lms transversal filters. *IEEE Trans. Instrum. Meas.* 2005, 54, 2239–2249. [CrossRef]

28. Kozat, S.S.; Erdogan, A.T.; Singer, A.C.; Sayed, A.H. Steady-state mse performance analysis of mixture approaches to adaptive filtering. *IEEE Trans. Signal Process.* 2010, 58, 4050–4063. [CrossRef]

29. Ferrer, M.; Gonzalez, A.; Member, S.; Diego, M.D. Convex combination filtered-x algorithms for active noise control systems. *IEEE Trans. Audio Speech Lang. Process.* 2013, 21, 156–167. [CrossRef]