EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Geometric-algebra affine projection adaptive filter

Yuetao Ren, Yongfeng Zhi* and Jun Zhang

*Correspondence: yongfeng@nwpu.edu.cn
The Research and Development Institute in Shenzhen, Northwestern Polytechnical University, Shenzhen, China

Abstract

Geometric algebra (GA) is an efficient tool to deal with hypercomplex processes due to its special data structure. In this article, we introduce the affine projection algorithm (APA) in the GA domain to provide fast convergence against hypercomplex colored signals. Following the principle of minimal disturbance and the orthogonal affine subspace theory, we formulate the criterion of designing the GA-APA as a constrained optimization problem, which can be solved by the method of Lagrange Multipliers. Then, the differentiation of the cost function is calculated using geometric calculus (the extension of GA to include differentiation) to get the update formula of the GA-APA. The stability of the algorithm is analyzed based on the mean-square deviation. To avoid ill-posed problems, the regularized GA-APA is also given in the following. The simulation results show that the proposed adaptive filters, in comparison with existing methods, achieve a better convergence performance under the condition of colored input signals.

Keywords: Adaptive filter, Geometric algebra, Affine projection, Hypercomplex process, Colored signal

1 Introduction

With the development of sensor technology, there are more and more data sources for recording the same process. For example, 3-D wind speed, dynamic pressure, aircraft rotation axis (roll, pitch, yaw), and angle of attack are used to predict the attitude of the aircraft [1]. The electromagnetic vector-sensor consists of 6 spatially arranged antennas, which measure the electric and magnetic field signals in the three directions of the incident wave [2]. These signals derive from observations of different dimensions. However, they are constructed into vectors and processed as multi-channel signals in most existing literature. In geometric algebra (GA)-based algorithms, the hypercomplex signals are transformed into multivectors, such as complex entries, quaternion entries, and higher dimensional entries [3], and handled holistically [4]. The product operation of GA, namely geometric product, allows a set of vectors to be mapped to scalars and hypersurfaces. Besides, geometric calculus (GC) [5, 6] can perform calculus with hypercomplex numbers clearly and compactly. Owing to the convenience of GA-based models, GA has been studied in many applications, such as classification, direction of arrival estimation, and image processing [7–9].

Adaptive filters (AF) have been extensively applied in many areas such as system identification, active noise control, and echo cancellation during the past decades [10]. The least mean square (LMS) and normalized LMS (NLMS) are widely used owing to their simplicity and ease of implementation. However, they show a slow convergence speed with highly colored input signals. The affine projection algorithm (APA), suggested by Ozeki and Umeda [11], is one method to overcome this problem. The APA and its variants [12–14] were found to be attractive choices with faster convergence than the NLMS and lower computational complexity than the recursive least squares (RLS).

However, standard AFs treat different dimensions as multi-channel signals, which may lose the structural information between different dimensions [15]. The GA-based adaptive filters (GAAF) have been used for simultaneous filtering of multi-dimensional signals for its faster convergence speed and suitability of multivector for multi-dimensional signal modeling. For example, quaternion adaptive filters were used to forecast Saito's Chaotic Signal and wind speed with superior performance [16]. GA-based beamformer of electromagnetic vector-sensor arrays has a better convergence performance than the standard beamformer [17, 18]. GAAF algorithms have great advantages in processing multi-dimensional signals.

Hitzer extended the quaternion AF to the GA-based nonlinear AF for hypercomplex signals of high dimensions [19]. Afterward, the authors in [1] proposed the GA-LMS to estimate the rotor in a three-dimensional point-clouds registration problem and analyzed its performance. The GA-LMS was later used to recover the 6-degrees-of-freedom alignment of two point clouds [20]. The authors in [21] developed a robust adaptive filter based on maximum GA correntropy criteria against non-Gaussian noise. The GA-based Normalized Least Mean Fourth (GA-NLMF) and GA-NLMS derived in [22] have a certain improvement in the convergence speed compared with NLMS. However, their convergence speed is reduced considerably by the colored input signals, commonly encountered in real applications.

Based on the principle of minimal disturbance and the orthogonal affine subspace theory, this article introduces the APA in the GA domain to deal with the hypercomplex system with colored signals. Firstly, the fundamentals of geometric algebra are presented in Sect. 2. We then propose the algorithm and analyse its stability in Sect. 3. Finally, several simulations are conducted to analyse the performance and the stability of the proposed algorithm.

2 Fundamentals of geometric algebra

The GA $\boldsymbol{G}(\mathbb{R}^n)$ was introduced by William K. Clifford, also called Clifford algebra. The GA enables the algebraic representation of magnitude and orientations and provides a coordinate-free framework to make the calculations efficiently.

The GA can be viewed as a geometric extension of the linear algebra $\mathbb{R}^n$. Vectors in $\mathbb{R}^n$ are also vectors in $\boldsymbol{G}(\mathbb{R}^n)$. Take a, b vectors in $\mathbb{R}^n$, the geometric product of a and b is defined as $ab = a \cdot b + a \wedge b$, on the basis of the inner $(\cdot)$ and outer $(\wedge)$ product. Since the outer product doesn't satisfy the commutative law, namely $a \wedge b = -(b \wedge a)$, the geometric product is also non-commutative in general. Unless otherwise specified, all products in this article are geometric products.

Take $\mathbb{R}^3$ for example, $\boldsymbol{G}(\mathbb{R}^3)$ has $2^3 = 8$ dimensions, with basis $\{1, e_1, e_2, e_3, e_{12}, e_{23}, e_{31}, I\}$. All bases can be divided into four parts, namely one scalar, three unit orthogonal vectors e_i (basis for $\mathbb{R}^3$), three bivectors $e_{ij} \triangleq e_i e_j = e_i \wedge e_j, i \neq j$ ($e_i \cdot e_j = 0, i \neq j$), and one trivector $I \triangleq e_{123} = e_1 e_2 e_3 = e_1 \wedge e_2 \wedge e_3$. To illustrate, take $a = e_1$ and $b = 3e_1 + 2e_2$. Then, $ab = e_1(3e_1 + 2e_2) = e_1 \cdot (3e_1 + 2e_2) + e_1 \wedge (3e_1 + 2e_2) = 3 + 2(e_1 \wedge e_2) = 3 + 2e_{12}$ (a scalar plus a bivector).

Multivector is the fundamental element of GA. The multivector A consists of its r-vectors $\langle\cdot\rangle_r$ as follows:

$$A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2 + \cdots = \sum_r \langle A\rangle_r, \tag{1}$$

in which $\langle A\rangle_0, \langle A\rangle_1$, and $\langle A\rangle_2$ are scalar, vector, and bivector, respectively. The foundation of GA theory is the ability to fuse scalar, vector, and hyper-planes into a unique element, namely multivector.

Analogous to the conjugation of complex numbers and quaternion, the reverse of the multivector A is defined as

$$\widetilde{A} \triangleq \sum_{r=0}^{n} (-1)^{r(r-1)/2} \langle A\rangle_r. \tag{2}$$

Taking the bivector $A = \langle A\rangle_0 + \langle A\rangle_1 + \langle A\rangle_2$ for an example, its reverse is $\widetilde{A} = \widetilde{\langle A\rangle_0} + \widetilde{\langle A\rangle_1} + \widetilde{\langle A\rangle_2} = \langle A\rangle_0 + \langle A\rangle_1 - \langle A\rangle_2$.

The scalar product $(*)$ is defined as $A * B = \langle AB\rangle$, in which $\langle\cdot\rangle \equiv \langle\cdot\rangle_0$. In addition, the magnitude of a multivector is defined as $|A|^2 = \widetilde{A} * A = \langle \widetilde{A} A\rangle$.

3 Methods

An array of multivector consists of a collection of multivectors. Give M multivectors $\{U_1, U_2, \ldots, U_M\}$ in $\boldsymbol{G}(\mathbb{R}^3)$, the $M \times 1$ array collects them as follows:

$$\boldsymbol{u} = \begin{bmatrix} U_1 \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} u(1,0) + u(1,1)e_1 + \cdots + u(1,7)I \\ \vdots \\ u(M,0) + u(M,1)e_1 + \cdots + u(M,7)I \end{bmatrix}. \tag{3}$$

The reverse transpose operation, denoted by $(\cdot)^*$, is the extension of the reverse operation of multivector to arrays of multivectors. For example, the reverse transpose of the array (3) is $\boldsymbol{u}^* = \begin{bmatrix} \widetilde{U}_1 & \widetilde{U}_2 & \ldots & \widetilde{U}_M \end{bmatrix}$.

Consider reference data $D(k)$, which is a multivector, observed at time k that comes from the linear model

$$\begin{aligned} D(k) &= \boldsymbol{u}^*(k)\boldsymbol{w}^o + V(k) \\ &= \sum_{i=1}^{M} \widetilde{U}(k+1-i) W_i^o + V(k), \end{aligned} \tag{4}$$

where $\boldsymbol{w}^o = [W_1^o \; W_2^o \; \ldots \; W_M^o]^{\mathrm{T}}$ is an unknown $M \times 1$ array of multivector to be estimated with $(\cdot)^{\mathrm{T}}$ denotes transpose, $V(k)$ accounts for measurement noise, $U(k)$ denotes input signal observed at time k, and $\boldsymbol{u}(k) = [U(k) \; U(k-1) \; \ldots \; U(k+1-M)]^{\mathrm{T}}$.

The model allows one to assign heterogeneous signals from different sources $s_i(k)$, $i = 0, 1, \ldots, 2^n - 1$, to each entries of the multivector, e.g., $U(k) = s_0(k) + s_1(k)e_1 + s_2(k)e_2 + s_3(k)e3 + s_4(k)e_{12} + s_5(k)e_{23} + s_6(k)e_{31} + s_7(k)I$. For example, fusion and linear prediction of aircraft parameters can be assigned as follows: $s_0(k)$ is angle of attack, $s_1(k)$ is East-West wind, $s_2(k)$ is North-South wind, $s_3(k)$ is vertical wind, $s_4(k)$ is roll, $s_5(k)$ is yaw, $s_6(k)$ is pitch, $s_7(k)$ is dynamic pressure.

The squared Euclidean norm providing a measure of distance in LA is represented by the array product $||\boldsymbol{u}||^2 \triangleq \boldsymbol{u}^*\boldsymbol{u}$, which is a scalar. However, the result of the array product in GA is a multivector rather than a scalar. Therefore, we take the scalar part of the array product $\langle \boldsymbol{u}^*\boldsymbol{u} \rangle$ as the distance measure of the array of multivectors.

3.1 GA affine projection algorithm

The GA-APA also follows the principle of minimal disturbance and the orthogonal affine subspace theory as the standard APA. In mathematical terms, the criterion for designing the affine projection filter can be formulated as an optimization problem subject to multiple constraints. We will minimize the scalar product of the change of the estimated weight array and its reverse transpose (the distance measure of the array space of $\boldsymbol{w}^o$), which is defined as

$$\langle ||\delta_{\boldsymbol{w}}||^2 \rangle = (\widehat{\boldsymbol{w}}^*(k+1) - \widehat{\boldsymbol{w}}^*(k)) * (\widehat{\boldsymbol{w}}(k+1) - \widehat{\boldsymbol{w}}(k)), \tag{5}$$

subject to the set of N constraints

$$D(k-n) = \boldsymbol{u}^*(k-n)\widehat{\boldsymbol{w}}(k+1) \text{ for } n = 0, 1, 2, \ldots, N-1, \tag{6}$$

where N is smaller than or equal to the length M of the weight array. The number of constraints N can be viewed as the order of the affine projection algorithm.

We apply the method of Lagrange multipliers to solve this optimization problem. Combining formulas (5) and (6), then we get the following cost function

$$J(k) = \langle ||\delta_{\boldsymbol{w}}||^2 \rangle + \sum_{n=0}^{N-1} \langle \widetilde{E}(n)\lambda_n \rangle, \tag{7}$$

where $E(n) = D(k-n) - \boldsymbol{u}^*(k-n)\widehat{\boldsymbol{w}}(k+1)$ and λ_n is a multivector. For convenience of presentation, we introduce the following definitions:

- An $M \times N$ matrix $\boldsymbol{U}(k)$ defined by

$$\boldsymbol{U}(k) = [\boldsymbol{u}(k)\ \boldsymbol{u}(k-1) \cdots \boldsymbol{u}(k-N+1)]. \tag{8}$$

- An $N \times 1$ array $\boldsymbol{d}(k)$ defined by

$$\boldsymbol{d}(k) = [D(k)\ D(k-1)\ \cdots\ D(k-N+1)]^{\mathrm{T}}. \tag{9}$$

- An $N \times 1$ array $\boldsymbol{\lambda}$ defined by

$$\boldsymbol{\lambda} = [\lambda_0\ \lambda_1\ \cdots\ \lambda_{N-1}]^{\mathrm{T}}. \tag{10}$$

Then, the second term of the cost function (7) can be represent as

$$\begin{aligned}\sum_{n=0}^{N-1}\langle\widetilde{E}(n)\lambda_n\rangle &= \langle(\boldsymbol{d}(k)-\boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k+1))^*\boldsymbol{\lambda}\rangle \\ &= (\boldsymbol{d}^*(k)-\widehat{\boldsymbol{w}}^*(k+1)\boldsymbol{U}(k))*\boldsymbol{\lambda}.\end{aligned} \tag{11}$$

Now, we will get the derivative of the cost function $J(k)$ with respect to the weight array $\boldsymbol{w}(k+1)$ following the rules of GC. In GA, the differential operator $\partial_w = \partial_{\widehat{\boldsymbol{w}}(k+1)}$ has the algebra properties of a multivector in $\boldsymbol{G}(\mathbb{R}^n)$. In other words, the gradient $\partial_w J(k)$ can be calculated by the geometric product of the multivector-valued quantities ∂_w and $J(k)$.

Any multivector $A \in G(\mathbb{R}^n)$ can be decomposed into blades [5, Eq. (3.20)] via

$$A = \sum_{i=0}^{2^n-1} e_i\langle e^i A\rangle = \sum_{i=0}^{2^n-1} e^i\langle e_i A\rangle = \sum_{i=0}^{2^n-1} e^i A^i, \tag{12}$$

in which A^i is a scalar valued, and $\{e_i\}$ and $\{e^i\}$, $i = 0,\ldots,2^n-1$ are two different bases of $\boldsymbol{G}(\mathbb{R}^n)$. $\{e^i\}$ is the reciprocal blade basis, which is an important analytical tool for differentiation in GA. The reciprocal blade basis can convert non-orthogonal to orthogonal vectors, vice versa. Since orthogonal elements cancel out mutually, the analytical procedure is simplified. Suffice to know that the following relation holds for reciprocal bases $e_i \cdot e^j = \delta_i^j$, where $\delta_i^j = 1$ for $i = j$ and $\delta_i^j = 0$ for $i \neq j$ (Kronecker delta). In particular, applying (12) to ∂_w results in

$$\partial_w \triangleq \sum_{l=0}^{2^n-1} e^l\langle e_l\partial_w\rangle = \sum_{l=0}^{2^n-1} e^l\partial_{w,l}. \tag{13}$$

The gradient $\partial_w J(k)$ is obtained by multiplying (13) and (7), yielding

$$\begin{aligned}\partial_w J(k) &= \sum_{l=0}^{2^n-1} e^l\partial_{w,l}\Big(\langle||\delta_{\boldsymbol{w}}||^2\rangle + \langle(\boldsymbol{d}^*(k)-\widehat{\boldsymbol{w}}^*(k+1)\boldsymbol{U}(k))\boldsymbol{\lambda}\rangle\Big) \\ &= \sum_{l=0}^{2^n-1} e^l\left(\partial_{w,l}\langle\delta_{\boldsymbol{w}}^*\delta_{\boldsymbol{w}}\rangle + \partial_{w,l}\langle-\widehat{\boldsymbol{w}}^*(k+1)\boldsymbol{U}(k)\boldsymbol{\lambda}\rangle\right) \\ &= \sum_{l=0}^{2^n-1} e^l\Big(\partial_{w,l}^1 + \partial_{w,l}^2\Big),\end{aligned} \tag{14}$$

in which $\partial_{w,l}^1 = \partial_{w,l}\langle\delta_{\boldsymbol{w}}^*\delta_{\boldsymbol{w}}\rangle$ and $\partial_{w,l}^2 = \partial_{w,l}\langle-\widehat{\boldsymbol{w}}^*(k+1)\boldsymbol{U}(k)\boldsymbol{\lambda}\rangle$. As a matter of fact, arrays of multivectors can be decomposed into blades. Thus, employing (12) once again, we can rewrite δ_w and δ_w^* in term of their 2^n blades as follows:

$$\delta_{\boldsymbol{w}} = \sum_{p=0}^{2^n-1} e_p\delta_{w,p} \text{ and } \delta_{\boldsymbol{w}}^* = \sum_{q=0}^{2^n-1} \widetilde{e}_q\delta_{w,q}^{\mathrm{T}}. \tag{15}$$

Plugging (15) into $\partial_{w,l}^1$, we have

$$\begin{aligned}\partial_{w,l}^{1} &= \partial_{w,l}\langle \sum_{p,q=0}^{2^n-1} \widetilde{e}_q \delta_{w,q}^{\mathrm{T}} e_p \delta_{w,p}\rangle \\ &= \sum_{p,q=0}^{2^n-1} \langle \widetilde{e}_q e_p\rangle \partial_{w,l}(\delta_{w,q}^{\mathrm{T}} \delta_{w,p}) \\ &= \sum_{p,q=0}^{2^n-1} \langle \widetilde{e}_q e_p\rangle (\dot{\partial}_{w,l} \dot{\delta}_{w,q}^{\mathrm{T}} \delta_{w,p} + \dot{\partial}_{w,l} \delta_{w,q}^{\mathrm{T}} \dot{\delta}_{w,p}) \\ &= \sum_{p,q=0}^{2^n-1} \langle \widetilde{e}_q e_p\rangle (\delta_l^q \delta_{w,p} + \delta_l^p \delta_{w,q}).\end{aligned} \tag{16}$$

Thus, the first term of the gradient (14) can be obtained by

$$\begin{aligned}\sum_{l=0}^{2^n-1} e^l \partial_{w,l}^{1} &= \sum_{l=0}^{2^n-1} e^l \sum_{p,q=0}^{2^n-1} \langle \widetilde{e}_q e_p\rangle (\delta_l^q \delta_{w,p} + \delta_l^p \delta_{w,q}) \\ &= \sum_{l=0}^{2^n-1} e^l \Big(\sum_{p=0}^{2^n-1} \langle \widetilde{e}_l e_p\rangle \delta_{w,p} + \sum_{q=0}^{2^n-1} \langle \widetilde{e}_q e_l\rangle \delta_{w,q}\Big) \\ &= \sum_{l=0}^{2^n-1} e^l (\langle \widetilde{e}_l \delta_w\rangle + \langle \widetilde{\delta}_w e_l\rangle) \\ &= 2\widetilde{\delta}_{\boldsymbol{w}}.\end{aligned} \tag{17}$$

Then, we calculate the second term of the formula (14) and get

$$\begin{aligned}\sum_{l=0}^{2^n-1} e^l \partial_{w,l}^{2} &= -\sum_{l=0}^{2^n-1} e^l \partial_{w,l} \langle \sum_{p=0}^{2^n-1} \widetilde{e}_p \widehat{\boldsymbol{w}}_p^{\mathrm{T}}(k+1)\boldsymbol{U}(k)\boldsymbol{\lambda}\rangle \\ &= -\sum_{l=0}^{2^n-1} e^l \sum_{p=0}^{2^n-1} \langle \widetilde{e}_p \partial_{w,l} \widehat{\boldsymbol{w}}_p^{\mathrm{T}}(k+1)\boldsymbol{U}(k)\boldsymbol{\lambda}\rangle \\ &= -\sum_{l=0}^{2^n-1} e^l \langle \widetilde{e}_l \boldsymbol{U}(k)\boldsymbol{\lambda}\rangle \\ &= -(\widetilde{\boldsymbol{U}(k)\boldsymbol{\lambda}}).\end{aligned} \tag{18}$$

Taking the results of (17) and (18), and setting the gradient (14) equal to zero, we get $2\widetilde{\delta}_w = (\widetilde{\boldsymbol{U}(k)\boldsymbol{\lambda}})$. Taking the reverse of both sides of the equation yields

$$\widehat{\boldsymbol{w}}(k+1) - \widehat{\boldsymbol{w}}(k) = \frac{1}{2}\boldsymbol{U}(k)\boldsymbol{\lambda}. \tag{19}$$

Next, we will eliminate the Lagrange vector λ from (19). Firstly, we use the definitions of (8) and (9) to rewrite (6) in the equivalent form

$$\boldsymbol{d}(k) = \boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k+1). \tag{20}$$

Premultiplying both sides of (19) by $\boldsymbol{U}^*$ and then using (20) to eliminate the updated weight array $\widehat{\boldsymbol{w}}(k+1)$ yields

$$\boldsymbol{d}(k) = \boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k) + \frac{1}{2}\boldsymbol{U}^*(k)\boldsymbol{U}(k)\boldsymbol{\lambda}. \tag{21}$$

Based on the data available, the difference $\boldsymbol{e}(k)$ between $\boldsymbol{d}(k)$ and $\boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k)$ at the adaptation cycle k is a $N \times 1$ error array denoted

$$\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k). \tag{22}$$

Assuming the array product $\boldsymbol{U}^*(k)\boldsymbol{U}(k)$ to be invertible [23] allows us to solve (21) for $\boldsymbol{\lambda}$, yielding

$$\boldsymbol{\lambda} = 2(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k). \tag{23}$$

Substituting this solution into (19), we obtain the optimum change of the weight array

$$\widehat{\boldsymbol{w}}(k+1) - \widehat{\boldsymbol{w}}(k) = \boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k). \tag{24}$$

Finally, we introduce the step-size parameter μ into (24), yielding

$$\widehat{\boldsymbol{w}}(k+1) = \widehat{\boldsymbol{w}}(k) + \mu\boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k), \tag{25}$$

which is the desired update formula of the GA-APA.

The algorithm is summarized in Algorithm 1. We can notice that GA-APA has the same format as standard APA adaptive filters. Since quaternion, complex numbers, and real numbers are subalgebras of geometric algebra, the Quaternion APA [24], the Complex APA [12], and the real-entries APA can be recovered by the GA-APA. In other words, GA-APA is a unified expression of the above algorithms.

1 *Remark*

APA is the same as NLMS in the LA domain when the order $N = 1$*. But, the update equation of the first order GA-APA is different from GA-NLMS proposed in [22]. Specially, the update term of the first order GA-APA is* $\mu u(k)(u(k)^*u(k))^{-1}e(k)$ *which is similar to the update term of the GA-NLMS* $\mu u(k)\langle u(k)^*u(k)\rangle^{-1}e(k)$*. We will compare them in the simulation section.*

Algorithm 1 GA-APA

Require: Input signal multivector $U(k)$ and reference signal multivector $D(k)$ constructed from multi-dimensional signals, $k = 1, 2, 3, \ldots$.
Ensure: The estimated coefficients multivector array $\widehat{\boldsymbol{w}}(k+1)$.
1: Initialize with $\boldsymbol{U}(k)$, $\boldsymbol{d}(k)$ as in (8) and (9).
2: Compute the estimation error: $\boldsymbol{e}(k) = \boldsymbol{d}(k) - \boldsymbol{U}^*(k)\widehat{\boldsymbol{w}}(k)$.
3: Update the estimated coefficients array: $\widehat{\boldsymbol{w}}(k+1) = \widehat{\boldsymbol{w}}(k) + \mu\boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k)$.
4: Update the time index: $k = k+1$, and repeat from Step 1.

3.2 Stability of the GA-APA

The mismatch between w^o and $\widehat{\boldsymbol{w}}(k)$ is measured by weight-error array

$$\boldsymbol{\epsilon}(k) = \boldsymbol{w}^o - \widehat{\boldsymbol{w}}(k). \tag{26}$$

Thus, subtracting (25) from w^o, we get

$$\boldsymbol{\epsilon}(k+1) = \boldsymbol{\epsilon}(k) - \mu \boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k). \tag{27}$$

We base the stability analysis of the GA-APA on the mean-square deviation $y(k) = \mathbb{E}[\langle||\boldsymbol{\epsilon}(k)||^2\rangle]$, where $\mathbb{E}[\cdot]$ accounts for expectation. Taking the distance measure of both sides of (27), rearranging terms, and taking expectations, then we get

$$\begin{aligned} y(k+1) - y(k) =& \mu^2 \mathbb{E}[\langle||\boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k)||^2\rangle] \\ &- 2\mu \mathbb{E}[\boldsymbol{\epsilon}^*(k) * \boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k)]. \end{aligned} \tag{28}$$

From the equation above, we see that the GA-APA algorithm is stable in the mean-square-error sense provided the mean-square deviation $y(k)$ decreases with the increasing number of adaptation cycles k. Therefore, the step-size parameter μ is bounded as follows:

$$0 < \mu < \frac{2\mathbb{E}[\boldsymbol{\epsilon}^*(k) * \boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k)]}{\mathbb{E}[\langle||\boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k))^{-1}\boldsymbol{e}(k)||^2\rangle]}. \tag{29}$$

3.3 Computational complexity analysis

As we can see from Algorithm 1, the main calculations are in Step 2 and Step 3. The number of real multiplications in Step 2 is $NM\alpha^2$, where $\alpha = 2^n$ represents the number of basis, N and M are the order of GA-APA and the length of $\widehat{\boldsymbol{w}}(k)$, respectively. The computational complexity of multivector matrix inversion is $N^3\alpha^2\beta$, where β represents the computational complexity of the inverse of a multivector. Therefore, the computation in Step 3 requires approximately $\alpha^2(N^3\beta + N^2(M+1) + NM)$ real multiplications. The total number of multiplications in GA-APA is $\alpha^2(N^3\beta + N^2(M+1) + 2NM)$ per adaptation cycle.

3.4 Regularized GA-APA

Since a matrix inversion $(\boldsymbol{u}^*\boldsymbol{u})^{-1}$ is required within the GA-APA, ill-posed problems usually occur, especially under the condition of noisy observation data. To avoid this problem, we regularize the matrix that needs to be inverted. Then we get the update equation of the regularized GA-APA (R-GA-APA)

$$\widehat{\boldsymbol{w}}(k+1) = \widehat{\boldsymbol{w}}(k) + \mu \boldsymbol{U}(k)(\boldsymbol{U}^*(k)\boldsymbol{U}(k) + \gamma \boldsymbol{I})^{-1}\boldsymbol{e}(k), \tag{30}$$

where γ is the regularization parameter, and $\boldsymbol{I}$ is the $N \times N$ identity matrix of real number.

4 Results and discussion

The proposed algorithm's performance is evaluated and analyzed in this section. We compare the proposed algorithm with the GA-LMS and GA-NLMS in Sect. 4.1. The impact of order N and step size μ on algorithm performance is analyzed in Sect. 4.2.

For the sake of generality, the underlying GA in all cases is $\boldsymbol{G}(\mathbb{R}^3)$. The regularization parameter of the R-GA-APA is $\gamma = 10^{-3}$. The measurement noise is zero-mean uniform distributed sequences. The blades coefficients of the colored signal $U(k)$ are obtained by filtering $2^3 = 8$ white zero-mean Gaussian random sequences through a first-order system $G(z) = 1/(1 - 0.9z^{-1})$, respectively. All simulation results are obtained by averaging 100 independent trials.

4.1 Performance comparison

The variance of measurement noise is $\sigma_V^2 = \mathbb{E}[|V(k)|^2] = 10^{-3}$. The variance of $U(k)$ is 0.1 for both white and colored signals. In this subsection, the step size value is $\mu = 0.2$ for all algorithms. The length M of the optimal weight is 10, namely $\boldsymbol{w}^o = [W_1^o \ W_2^o \cdots W_{10}^o]^{\mathrm{T}}$.

Figure 1 shows several mean-square error (MSE) $\mathbb{E}[|D(k) - \boldsymbol{u}^*(k)\widehat{\boldsymbol{w}}(k)|^2]$ learning curves for the GA-LMS [3], GA-NLMS, GA-APA with $N = 1$ and R-GA-APA with $N = 1$ with both white and colored input signals. All the multivectors entries W_i^o are the same, namely $W_i^o = W_1 = 0.25e_0 - 1.5e_1 - 0.5e2 + 0.75e_{12} - 0.4e23 + 0.3e31 - 0.25I, i = 1, \ldots, 10$. The coefficients of W_1 are selected in an aleatory manner. As we can see, all algorithms can converge at the same speed with white signals under some given parameters. These experiments show that the GA-APA and R-GA-APA, the same as the GA-LMS, are capable of identifying multivector-valued linear systems. But the GA-LMS and GA-NLMS suffer from slow convergence with colored input. The GA-APA and R-GA-APA achieve better performance with the colored signal with the same parameter. Comparing the GA-APA and the R-GA-APA, we conclude that their performance is roughly the same when the regularization parameter γ is small. We will focus our simulations on the R-GA-APA since it reaches the same performance as the GA-APA and avoids ill-posed problems. Additionally, the first order GA-APA reaches a much faster convergence than

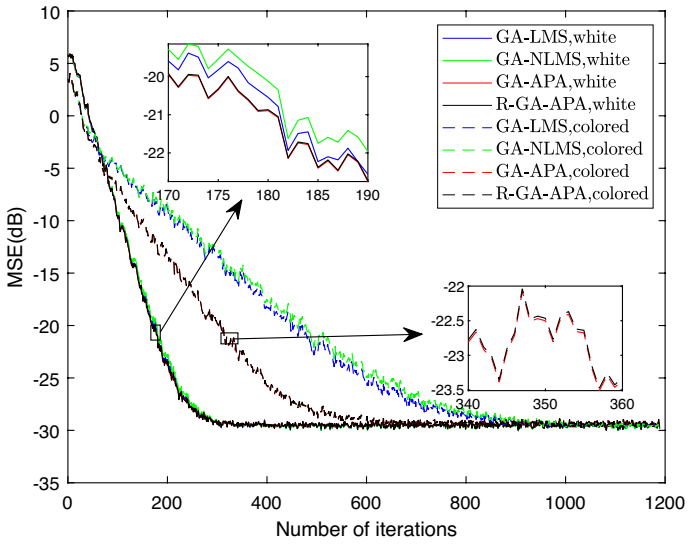

Fig. 1 MSE learning curves with white (solid) and colored (dashed) input signals with $\mu = 0.2, M = 10$ and $\sigma_V^2 = 10^{-3}$. The order of GA-APA and R-GA-APA is $N = 1$

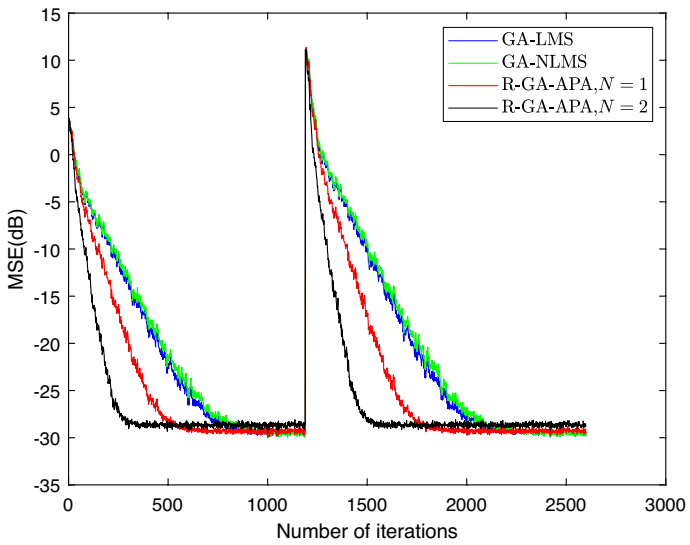

Fig. 2 MSE learning curves when system weights suddenly change at 1190

Table 1 Time required for each iteration

	GA-LMS	**GA-NLMS**	**GA-APA,** $N=1$	**GA-APA,** $N=2$
Time (ms)	2.2	2.4	5.1	9.1

the GA-NLMS. We think it's because GA-NLMS doesn't follow the principle of minimal disturbance. Although the GA-NLMS introduces the normalization comparing with the GA-LMS, it suffers from slow convergence with colored signals.

Figure 2 shows the convergence performance under colored input signals when filter weights $\boldsymbol{w}^o$ change after 1190 iterations. The filter weights change from the weights W_1 in the above experiments to $W_i^o = W_2 = 0.5e_0 + 1.8e_1 - 2e2 + 0.86e_3 + 0.31e_{12} - 0.9e23 - 0.4e31 + 0.34I, i = 1, \ldots, 10$. We can see from Fig. 2 that the proposed R-GA-APA can track the changes of weights faster than the GA-LMS and GA-NLMS after the filter weights change suddenly.

The time required for each iteration of different algorithms is given in Table 1. The results are obtained using Python on an Inter Core i7 CPU running at 3.6GHz and 16 GB of RAM. We can see that the proposed algorithm requires more calculation time. As the order of the algorithm N increases, the calculation time also increases.

4.2 Parameters analysis

The comparison results of different orders of the R-GA-APA under colored input signals are shown in Fig. 3. The filter weights are $\boldsymbol{w}^o = [W_1^o, W_2^o, \ldots, W_{30}^o]$, where $W_i^o = W_1$ (given in Sect. 4.1), $i = 1, 2, \ldots, 30$. The variance of $V(k)$ and colored $U(k)$ are $\{10^{-2}, 10^{-4}\}$ and 10^{-1}, respectively. The step size μ is set to 0.2. As we can see, the convergence rate speeds up with the order N increases. The misadjustment of APA increases when the order N increases. As expected, this is also the case with the R-GA-APA, as

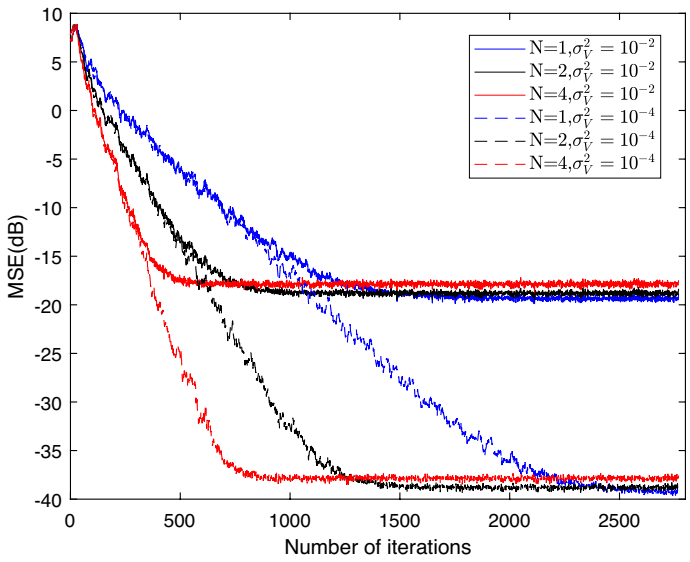

Fig. 3 MSE learning curves of the R-GA-APA of different orders N with $\mu = 0.2, M = 30$ and $\sigma_V^2 = \{10^{-2}, 10^{-4}\}$

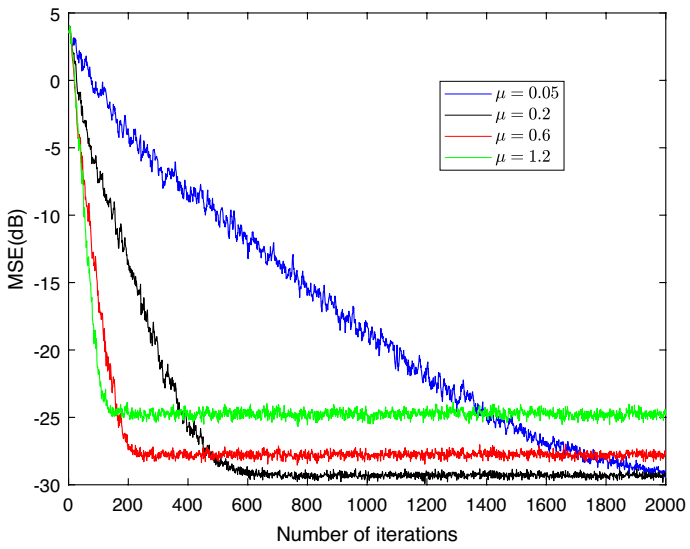

Fig. 4 MSE learning curves of the first order R-GA-APA of different step sizes μ with $M = 10$

supported in Fig. 3. Despite the speed of convergence, the steady-state error and the computational complexity of matrix inversion limit the choice of higher orders.

Finally, we evaluate the impact of the step size on convergence rate and steady-state error under colored input signals. The order of the R-GA-APA is set to 1. The σ_V^2 and σ_U^2 is set to 10^{-3} and 10^{-1}, respectively. The length of the filter weight is set to 10, and each multivector entries W_i^o is the same as W_1 in Sect. 4.1. Figure 4 shows the MSE learning curves of the R-GA-APA with step size $\mu = \{0.05, 0.2, 0.6, 1.2\}$. We can see that the algorithm will have a faster convergence rate with a larger step size. However,

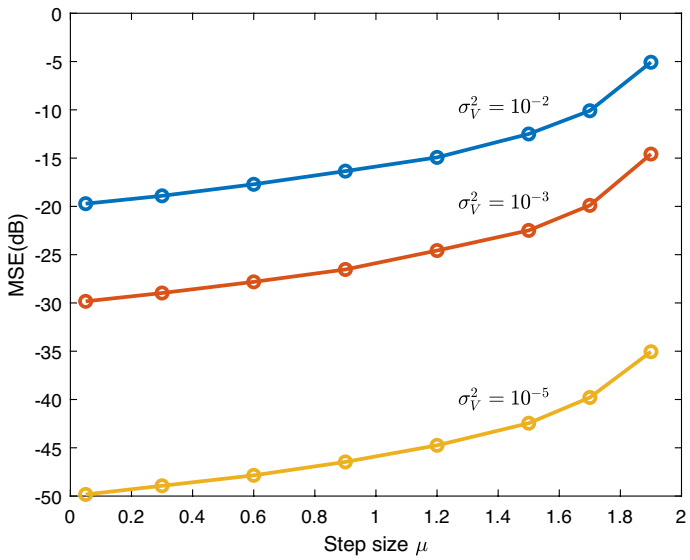

Fig. 5 MSE of the first order R-GA-APA of different step sizes μ with $M = 10$

the steady-state error goes higher with the step size increases. Therefore, it's important to choose an appropriate step size. We then examine the steady-state errors at different step sizes. Figure 5 depicts the steady-state errors at different step sizes for $\sigma_V^2 = \{10^{-2}, 10^{-3}, 10^{-5}\}$. According to the stability analysis in Sect. 3.2, the step size is roughly limited to the interval $0 < \mu < 2$ when $\sigma_U^2 = 10^{-1}$. The simulation shows a good agreement with the theoretical analysis.

5 Conclusions

The GA-APA and the R-GA-APA proposed in this article have improved estimation capabilities with highly colored input signals for hypercomplex processes. The structure of multivectors, allowing us to deal with the hypercomplex processes as a "package", seems to be naturally suited for fusing different dimensional signals. With the increase of application scenarios, new types of GA-based AFs, i.e., the RLS and other variants, are the main subjects to be studied in the future.

Abbreviations

GA: Geometric algebra; APA: Affine projection algorithm; AF: Adaptive filters; LMS: Least mean square; NLMS: Normalized Least mean square; RLS: Recursive least squares; GAAF: Geometric-algebra-based adaptive filter; LA: Linear algebra; GC: Geometric calculus; GA-LMS: Geometric-algebra least mean square; GA-NLMF: Geometric-algebra-based Normalized Least Mean Fourth; GA-NLMS: Geometric-algebra Normalized Least mean square; GA-APA: Geometric-algebra affine projection algorithm.

Acknowledgements

The authors would like to acknowledge all the participants for their contributions to this research study.

Authors' contributions

Yuetao Ren contributed to developments of the theory and the experiments; Yongfeng Zhi and Jun Zhang contributed to the problem formulation, and overall project guidance. All authors read and approved the final manuscript.

Funding

This work is supported by the Science and Technology on Electromechanical Dynamic Control Laboratory of China (No. 6142601200605), the Science, Technology and Innovation Commission of Shenzhen Municipality (No. JCYJ20170815161351983), and the National Nature Science Foundation of China (Nos. U20B2040 and 61671379).

Availability of data and materials
There is no additional data to be made available.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 20 April 2021 Accepted: 4 September 2021
Published online: 17 September 2021

References

1. W.B. Lopes, A. Al-Nuaimi, C.G. Lopes, Geometric-algebra LMS adaptive filter and its application to rotation estimation. IEEE Signal Process. Lett. **23**(6), 858–862 (2016). https://doi.org/10.1109/LSP.2016.2558461
2. J.F. Jiang, J.Q. Zhang, Geometric algebra of Euclidean 3-space for electromagnetic vector-sensor array processing, part i: modeling. IEEE Trans. Antennas Propag. **58**(12), 3961–3973 (2010). https://doi.org/10.1109/TAP.2010.2078468
3. W.B. Lopes, C.G. Lopes, Geometric-algebra adaptive filters. IEEE Trans. Signal Process. **67**(14), 3649–3662 (2019). https://doi.org/10.1109/TSP.2019.2916028
4. E. Hitzer, Introduction to Clifford's Geometric Algebra. arxiv:1306.1660
5. D. Hestenes, G. Sobczyk, *Clifford Algebra to Geometric Calculus: a Unified Language for Mathematics and Physics*, Reprint edn. *Fundamental theories of physics*, vol. 5. Reidel. OCLC: 552321534
6. E.M.S. Hitzer, Multivector differential calculus. Adv. Appl. Clifford Algebras **12**(2), 135–182 (2002). https://doi.org/10.1007/BF03161244
7. T. Meng, M. Wu, N. Yuan, DOA estimation for conformal vector-sensor array using geometric algebra. EURASIP J. Adv. Signal Process. **2017**(1), 64 (2017). https://doi.org/10.1186/s13634-017-0503-y
8. X. Gong, Z. Liu, Y. Xu, Quad-quaternion MUSIC for DOA estimation using electromagnetic vector sensors. EURASIP J. Adv. Signal Process. (2009). https://doi.org/10.1155/2008/213293
9. E. Hitzer, T. Nitta, Y. Kuroe, Applications of Clifford's geometric algebra. Adv. Appl. Clifford Algebras **23**(2), 377–404 (2013). https://doi.org/10.1007/s00006-013-0378-4. arXiv: 1305.5663
10. S. Haykin, *Adaptive Filter Theory*, 5th edn. Pearson
11. K. Ozeki, T. Umeda, An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. Electron. Commun. Jpn. Part I: Commun. **67**(5), 19–27 (1984). https://doi.org/10.1002/ecja.4400670503
12. Y. Xia, C.C. Took, D.P. Mandic, An augmented affine projection algorithm for the filtering of noncircular complex signals. Signal Process. **90**(6), 1788–1799 (2010). https://doi.org/10.1016/j.sigpro.2009.11.026
13. J. Benesty, P. Duhamel, Y. Grenier, A multichannel affine projection algorithm with applications to multichannel acoustic echo cancellation. IEEE Signal Process. Lett. **3**(2), 35–37 (1996). https://doi.org/10.1109/97.484209
14. Y.-F. Zhi, F.-F. Shang, J. Zhang, Z. Wang, Optimal step-size of pseudo affine projection algorithm. Appl. Math. Comput. **273**, 82–88 (2016). https://doi.org/10.1016/j.amc.2015.09.059
15. K.J. Sangston, Geometry of complex data. IEEE Aerospace Electron. Syst. Mag. **31**(3), 32–69 (2016). https://doi.org/10.1109/TAES.2016.150029
16. B.C. Ujang, C.C. Took, D.P. Mandic, Split quaternion nonlinear adaptive filtering. Neural Netw. **23**(3), 426–434 (2010). https://doi.org/10.1016/j.neunet.2009.10.006
17. X. Zhang, W. Liu, Y. Xu, Z. Liu, Quaternion-valued robust adaptive beamformer for electromagnetic vector-sensor arrays with worst-case constraint. Signal Process. **104**, 274–283 (2014). https://doi.org/10.1016/j.sigpro.2014.04.006
18. X. Gou, Y. Xu, Z. Liu, X. Gong, Quaternion-capon beamformer using crossed-dipole arrays, in *2011 4th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications* (2011), pp. 34–37. https://doi.org/10.1109/MAPE.2011.6156140
19. E. Hitzer, Algebraic foundations of split hypercomplex nonlinear adaptive filtering. Math. Methods Appl. Sci. **36**(9), 1042–1055 (2013). https://doi.org/10.1002/mma.2660
20. A. Al-Nuaimi, E. Steinbach, W.B. Lopes, C.G. Lopes, 6dof point cloud alignment using geometric algebra-based adaptive filtering, in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9. https://doi.org/10.1109/WACV.2016.7477642
21. W. Wang, H. Zhao, X. Zeng, Geometric algebra correntropy: definition and application to robust adaptive filtering. IEEE Trans. Circuits Syst. II Express Briefs **67**(6), 1164–1168 (2019). https://doi.org/10.1109/TCSII.2019.2931507
22. R. Wang, M. Liang, Y. He, X. Wang, W. Cao, A normalized adaptive filtering algorithm based on geometric algebra. IEEE Access **8**, 92861–92874 (2020). https://doi.org/10.1109/ACCESS.2020.2994230
23. E. Hitzer, S. Sangwine, Multivector and multivector matrix inverses in real Clifford algebras. Appl. Math. Comput. **311**, 375–389 (2017). https://doi.org/10.1016/j.amc.2017.05.027
24. C. Jahanchahi, C. Cheong Took, D.P. Mandic, A class of quaternion valued affine projection algorithms. Signal Process. **93**(7), 1712–1723 (2013). https://doi.org/10.1016/j.sigpro.2012.12.019

Publisher's Note