Frequency-Domain Adaptive Filtering in Hypercomplex Systems

Francesca Ortolani, Danilo Comminiello, Michele Scarpiniti and Aurelio Uncini

Abstract In recent years, linear and nonlinear signal processing applications required the development of new multidimensional algorithms. Higher-dimensional algorithms include quaternion-valued filters. One of the drawbacks filter designers have to cope with is the increasing computational cost due to multidimensional processing. A strategy to reduce the computational complexity of long adaptive filters is to implement block algorithms and update the filter coefficients periodically. More efficient techniques embed frequency-domain processing in block algorithms with the use of the Fast Fourier Transform (FFT). Transform-domain adaptive filters in the quaternion field require quaternion-valued transforms. In this paper we also suggest a simple method to obtain a quaternionic DFT/FFT from a complex DFT/FFT. As an example, we propose the Overlap-Save Quaternion Frequency Domain algorithm.

Keywords Adaptive filters · Quaternion · Hypercomplex · Frequency domain · Overlap-Save

1 Introduction

Since the origin of science, men searched for ways to describe the surrounding world. Although maths is tough, it allows an unambiguous representation of nature and its laws. When scientists are called to face the complexity of the problems, the exist-

F. Ortolani · D. Comminiello (✉) · M. Scarpiniti · A. Uncini
Department of Information Engineering Electronics and Telecommunications (DIET),
"La Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy
e-mail: danilo.comminiello@uniroma1.it

F. Ortolani
e-mail: francesca.ortolani@uniroma1.it

M. Scarpiniti
e-mail: michele.scarpiniti@uniroma1.it

A. Uncini
e-mail: aurel@ieee.org

S. Bassis et al. (eds.), *Advances in Neural Networks*, Smart Innovation,
Systems and Technologies 54, DOI 10.1007/978-3-319-33747-0_5

ing models need to be overcome and the introduction of new numerical systems is peremptory. The most primitive way to quantify objects uses natural numbers, i.e. integer and unsigned numbers. Today we can take advantage of *Hypercomplex* algebras, whose elements consist of multidimensional numbers.

The reasons for increasing the number of dimensions of a numerical system arise from several needs. We may require a number system capable of describing a variety of geometrical objects (points, lines, planes, hyperplanes, spheres, hyperspheres). It would be impossible to draw a hyperplane by means of real numbers, whereas hypercomplex numbers allow describing this kind of object. We may also require to store different information into one single entity. For example, complex numbers carry both the information of amplitude and phase in one number:

$$z = x + \mathbf{i}y \leftrightarrow z = re^{\mathbf{i}\phi} \tag{1}$$

where $r = |z| = \sqrt{x^2 + y^2}$ is the *modulus* of z, representing its amplitude, and $\phi = \arg(z)$ is its *phase*. These needs for working with a higher-dimensional number system are interconnected from a philosophical point of view. We could say, they are geometrical issues. Therefore, a proper numerical infrastructure is necessary for decomposing a physical problem into its own dimensions and spotlighting all its faces, which might remain unseen otherwise. De facto, hypercomplex calculus is also known as *geometric* calculus. Moreover, as written in [12] "Once geometry, in particular differential geometry, is discussed, the presence of physics is unavoidable".

Currently, our activity within the hypercomplex number systems is restricted to *quaternions*, i.e. a four-dimensional hypercomplex subgroup. Specifically, quaternions are a geometric algebra in the sense of Clifford Algebras [4]. At the beginning, quaternions found application in electromagnetism, quantum physics and relativity. Thanks to their compact notation, it is possible to express equations in a more accessible form.

Recent applications, however, include aero-navigation, kinematics, 3D graphics, image processing in general (e.g. color template matching), digital forensics (e.g. DNA sequence matching), weather forecasting and 3D audio processing.

Our effort was to extend frequency-domain adaptive filtering to quaternions. Prior to our work, several quaternion-valued algorithms in the time domain [6, 7, 9, 14] and one algorithm in the frequency domain [8] were proposed. Pioneer in this field was the development of the Quaternionic Least Mean Square algorithm (QLMS) by Mandic et al. [14]. The class of algorithms presented in this paper operates weight adaptation in the frequency domain. There are two main reasons for this choice: fast computation and signal orthogonalization. A low computational cost can be achieved efficiently with the use of the Fast Fourier Transform (FFT), which allows a fast performance of convolutions and crosscorrelations. In addition, the orthogonality properties of the Discrete Fourier Transform (DFT) produce a diagonal input autocorrelation matrix as the transformation length tends to infinity. This makes it possible to get the most out of *power normalization*, a compensation technique intended to have

all the algorithm modes converging at the same rate. So, when power normalization is used, a more uniform convergence rate can be obtained easily thanks to the DFT.

Besides the linear filtering algorithms cited above, further algorithm implementations can be found in typical nonlinear environments. Studies concerning the use of quaternion neural networks, amongst many examples, include the definition and application to engineering problems of a quaternionic Multilayer Perceptron (QMLP) model [1, 10, 11] and the development of quaternionic Hopfield-type networks [5, 16]. For what concerns our contribution to the area of neural networks, our idea is to embed our quaternionic algorithms in the conventional learning schemes for neural networks.

2 Introduction to Quaternion Algebra

2.1 Quaternion Math

Quaternions are a hypercomplex normed division algebra consisting of 4 components: one scalar (real part) and 3 imaginary components (vector part):

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} = \mathrm{Re}(q) + \mathrm{Im}(q) \ . \tag{2}$$

A quaternion having zero scalar part is also known as *pure quaternion*. The imaginary units, $\mathbf{i} = (1, 0, 0)$, $\mathbf{j} = (0, 1, 0)$, $\mathbf{k} = (0, 0, 1)$, represent an orthonormal basis in $\mathbb{R}^3$.

The cross product of the versors of the basis gives as results:

$$\mathbf{ij} = \mathbf{i} \times \mathbf{j} = \mathbf{k} \quad \mathbf{jk} = \mathbf{j} \times \mathbf{k} = \mathbf{i} \quad \mathbf{ki} = \mathbf{k} \times \mathbf{i} = \mathbf{j} \tag{3}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1 \tag{4}$$

which are the fundamental properties of quaternions.

Quaternion product is non-commutative, i.e. $\mathbf{ij} \neq \mathbf{ji}$, in fact

$$\mathbf{ij} = -\mathbf{ji} \quad \mathbf{jk} = -\mathbf{kj} \quad \mathbf{ki} = -\mathbf{ik} \ . \tag{5}$$

The sum of quaternions is obtained by component-wise addition:

$$\begin{aligned} q \pm p &= \left(q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}\right) \pm \left(p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}\right) \\ &= \left(q_0 \pm p_0\right) + \left(q_1 \pm p_1\right)\mathbf{i} + \left(q_2 \pm p_2\right)\mathbf{j} + \left(q_3 \pm p_3\right)\mathbf{k} \ . \end{aligned} \tag{6}$$

The product between quaternions q_1 and q_2 is computed as

$$\begin{aligned} q_1 q_2 &= \left(a_0 + a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}\right)\left(b_0 + b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}\right) \\ &= \left(a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3\right) + \left(a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2\right)\mathbf{i} \\ &\quad + \left(a_0 b_2 - a_1 b_3 + a_2 b_0 + a_3 b_1\right)\mathbf{j} + \left(a_0 b_3 + a_1 b_2 - a_2 b_1 + a_3 b_0\right)\mathbf{k} \ . \end{aligned} \tag{7}$$

The conjugate of a quaternion is defined as

$$q^* = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k} \ . \tag{8}$$

2.2 Quaternion Discrete Fourier Transform

Frequency domain algorithms require mathematical transformations in order to (pre)process input and output signals. There exists an immediate method that exploits complex DFT/FFT to build up a quaternionic transform (QDFT/QFFT). This method was formerly developed in image processing and presented in [2, 13] with the idea of collecting colour (RGB) or luminance/chrominance signals in one quaternion, rather than treating them as independent vectors. This method is based on the Cayley-Dickson decomposition, according to which quaternions can be thought of as complex numbers whose real and imaginary parts are complex numbers in turn (Cayley-Dickson form):

$$q = s + p\mathbf{v_2} = \underbrace{\left(w + x\mathbf{v_1}\right)}_{simplex} + \underbrace{\left(y + z\mathbf{v_1}\right)}_{perplex}\mathbf{v_2} = w + x\mathbf{v_1} + y\mathbf{v_2} + z\mathbf{v_3} \ . \tag{9}$$

Both the *simplex* and *perplex* parts are isomorphic to field $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, since they are both defined in the same space of the complex operator $\mathbf{v_1}$. Hence a function $f(n)$ can be formulated in terms of an orthonormal basis $\left(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}\right)$:

$$f(n) = \left[w(n) + x(n)\mathbf{v_1}\right] + \left[y(n) + z(n)\mathbf{v_1}\right]\mathbf{v_2} \ . \tag{10}$$

The basis must be chosen in a way that $\mathbf{v_1} \perp \mathbf{v_2} \perp \mathbf{v_3}$, $\mathbf{v_1 v_2} = \mathbf{v_3}$ and $\mathbf{v_1 v_2 v_3} = -1$.

Thanks to the linearity property of the Fourier transform, the quaternion DFT (11a) and the quaternion *inverse* DFT (11b) can be decomposed into the sum of two complex-valued transforms:

$$F(u) = \sum_{n=0}^{N-1} \exp\left(-2\pi\mathbf{v}\frac{nu}{N}\right) f(n) = F_s(u) + F_p(u)\,\mathbf{v}_2 \tag{11a}$$

$$f(n) = \frac{1}{N}\sum_{u=0}^{N-1} \exp\left(2\pi\mathbf{v}\frac{nu}{N}\right) F(u) = f_s(n) + f_p(n)\,\mathbf{v}_2 \tag{11b}$$

Table 1 Kernel definitions for monodimensional QDFT

	Left	Right
Axis $\mathbf{v}$	$e^{-\mathbf{v}\omega n} f(\cdot)$	$f(\cdot)\, e^{-\mathbf{v}\omega n}$

where the subscripts s and p denote the simplex and perplex parts, respectively. Both functions $F(u)$ and $f(n)$ are quaternionic functions of N samples of the kind $f(n) = w(n) + x(n)\mathbf{i} + y(n)\mathbf{j} + z(n)\mathbf{k}$. Versor $\mathbf{v}$ is an arbitrarily chosen pure unitary quaternion versor and has to meet the constraints $\mathrm{Re}[\mathbf{v}] \perp \mathrm{Im}[\mathbf{v}]$, $\|\mathrm{Re}[\mathbf{v}]\| - \|\mathrm{Im}[\mathbf{v}]\| = 1$. After executing the two complex DFTs and reassembling the final transformed quaternion, it is sufficient to change back to the original basis by means of inverse change of basis.

Unfortunately, the non-commutativity property of the quaternionic product gives rise to a two-sided monodimensional quaternion transform, i.e. quaternion transforms can be found either in a left- or a right-handed form (transpose with one another) as summed-up in Table 1. For instance, (11a) and (11b) represent the quaternionic Fourier monodimensional left-handed transform (the exponential function is on the left) and its inverse, respectively. Tests conducted during the development of these algorithms revealed that there is no major implication from the existence of a two-sided transform on filtering applications, if the direction of rotation is kept unchanged from the beginning to the end of the algorithm.

3 Overlap-Save Quaternion Frequency Domain Adaptive Filter

We present the OS-QFDAF algorithm, also called *Fast QLMS*. It is a block algorithm, i.e. the filter coefficients $\mathbf{W}_k$ are updated at each block iteration k. A general classification discerns block algorithms by the input block length, the number of the overlapping samples, the type of transform used (for those algorithms working in a transform domain). In OS-QFDAF the usual choice for the filter length is $M_F = M$, where M is the number of the overlapping samples. In order to simplify the implementation, the transform length is chosen as $N = M + L$, where L is the block length.

The algorithm comes with power normalization, a technique intended to improve the convergence to optimum by whitening the input signal, so that the convergence modes are equalized.

Fast convolution in OS-QFDAF is performed using the *Overlap-Save* method.

3.1 Algorithm Overview

Algorithm initialization:

$$\begin{aligned} &\mathbf{W}_0 = \mathbf{0} \ \text{(2M-by-1 null vector)} \\ &\mu = \mu_0,\ P_0(m) = \delta,\ m = 0, 1, ..., N-1 \end{aligned} \tag{12}$$

$P_k(m)$: power of the m-th frequency bin at block k.
μ_0, δ: initialization constants to be chosen empirically.

Do the following steps for each new input block k:

Compute the QFFT of the filter input samples:

$$\mathbf{X}_k = \text{diag}\left\{ \text{QFFT}\left[\mathbf{x}_{old}^{M}\ \mathbf{x}_k^{L}\right]^T \right\} \tag{13}$$

Input block definition:

$\mathbf{x}_{old}^{M} = [x(kL - M + 1), \cdots, x(kL - 1)]$
$\mathbf{x}_k^{L} = [x(kL), \cdots, x(kL + L - 1)]$

Note: diagonalization in (13) allows a formalism similar to Block LMS when the filter output is computed in (14) later on [15].

Compute the filter output in the time domain:

$$\mathbf{Y}_k = \mathbf{X}_k \mathbf{W}_k \tag{14}$$

$$\hat{\mathbf{y}}_k = \left[\text{IQFFT}\left(\mathbf{X}_k \mathbf{W}_k\right)\right]^{\lfloor L \rfloor} \tag{15}$$

$\mathbf{Y}_k$: filter output in the frequency domain.
$\hat{\mathbf{y}}_k$: windowed filter output in the time domain.

Note: since filtering operations are implemented with linear convolution, proper window constraints on data are needed in order to obtain linear convolution from circular convolution.[1]

Error calculation:

$$\hat{\mathbf{d}}_k = \left[d(kL)\ d(kL + 1)\ \cdots\ d(kL + L - 1)\right]^T \tag{16}$$

$$\hat{\mathbf{e}}_k = \hat{\mathbf{d}}_k - \hat{\mathbf{y}}_k \tag{17}$$

$$\mathbf{E}_k = \text{QFFT}\left(\left[0_M\ \hat{\mathbf{e}}_k\right]^T\right) \tag{18}$$

$\hat{\mathbf{d}}_k$: desired output vector in the time domain at block k.
$\hat{\mathbf{e}}_k$: error vector in the time domain at block k.
$\mathbf{E}_k$: error vector in the frequency domain at block k.[2]

[1] Circular convolution (in the time domain) is provided by the inverse transform of the product of two DFT sequences.

[2] Zero-padding is needed before executing the QFFT.

Update the learning rates (Power Normalization):

$$\boldsymbol{\mu}_k = \mu \cdot \text{diag}\left[P_k^{-1}(0), \ldots, P_k^{-1}(N-1)\right] \tag{19}$$

$$P_k(m) = \lambda P_{k-1}(m) + (1-\lambda)\left|X_k(m)\right|^2 \tag{20}$$

$\lambda \in [0,1]$: forgetting factor.

Note: Power Normalization is a technique aimed at having all the algorithm modes converging at the same rate. Each filter weight is assigned a step size of its own, so that the disparity between the eigenvalues of the input autocorrelation matrix is reduced.

Update the filter weights:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \boldsymbol{\mu}_k \mathbf{X}_k^H \mathbf{E}_k \tag{21}$$

$\mathbf{X}_k^H \mathbf{E}_k$: gradient estimation in the frequency domain at step k.

Gradient constraint:

$$\mathbf{W}_{k+1} = \text{QFFT}\begin{pmatrix} \left[\text{IQFFT}(\mathbf{W}_{k+1})\right]^{\lceil M \rceil} \\ \mathbf{0}_L \end{pmatrix} . \tag{22}$$

If the gradient constraint $\lceil M \rceil$ is neutralized, the product $\mathbf{X}_k^H \mathbf{E}_k$ corresponds to a circular correlation in the time domain and the algorithm is said *Unconstrained* QFDAF. Usually, unconstrained algorithms exhibit a polarized convergence. In order to get convergence to optimum, the filter length M should be increased.

3.2 Computational Cost

Each QFFT requires the execution of 2 complex FFTs. The computation of one complex FFT involves $N\log_2 N$ multiplications. The OS-QFDAF algorithm includes 5 QFFTs, $16N$ multiplications to compute the filter output and $16N$ multiplications to update the filter weights. Given a block of $L = M$ samples, the computational cost for OS-QFDAF is approximately

$$C_{OS-QFDAF} \simeq 2 \cdot 5N\log_2 N + 2 \cdot 16N = 20M\log_2 2M + 64M \tag{23}$$

where $N = L + M = 2M$.

In QLMS, the computation of both the filter output and the cross-correlation in the update equation requires $4 \cdot 4M = 16M$ multiplications for each sample. Overall, for

M samples, the computational cost for QLMS is approximately $C_{QLMS} \simeq 32M \cdot M = 32M^2$. The complexity ratio between OS-QFDAF and QLMS is

$$\frac{C_{OS-QFDAF}}{C_{QLMS}} = \frac{5\log_2 2M + 16}{8M} \quad . \tag{24}$$

For example, for $M = 64$, OS-QFDAF is about 10 times faster than QLMS.

4 Simulations

The OS-QFDAF algorithm has been tested in a system identification problem. The filter quaternion input was a unit variance colored noise of the form

$$x[n] = bx[n-1] + \frac{\sqrt{1-b^2}}{\sqrt{4}}\eta[n]\,. \tag{25}$$

where $\eta[n]$ is a quaternionic zero-mean unit-variance white Gaussian noise.

OS-QFDAF with varying M: Figure 1 shows how choosing too a small block length causes an Excess Mean Square Error (EMSE) in the learning curve, i.e. there is a gap between the steady-state MSE curve and the theoretical MSE bound. As a rule of thumb, too a large M and/or too a large δ decreases the convergence rate of the algorithm.

OS-QFDAF versus QLMS with narrow-band signals: The input signal is now narrow-band (b = 0.99). In such a difficult situation it is possible to outperform the QLMS with the use of the transform domain algorithms as shown in Fig. 2.

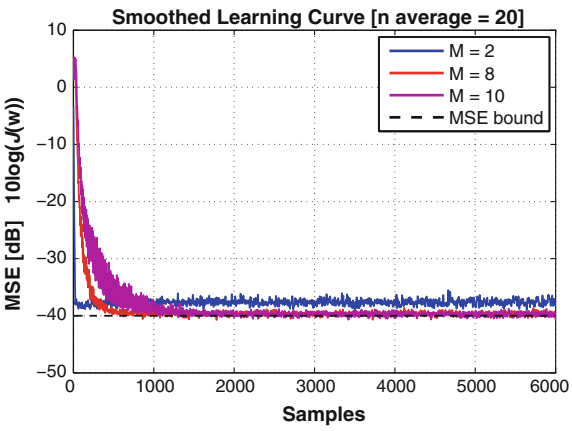

Fig. 1 OS-QFDAF with varying M (with power norm.) ($\mu_0 = 0.08$, $\lambda = 0.98, \delta = 4$)

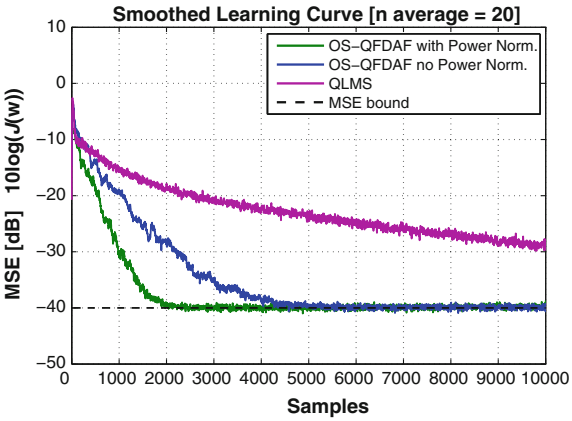

Fig. 2 OS-QFDAF versus QLMS with narrow band signal $b = 0.99$. (Power-norm. OS-QFDAF: $M = 6, \mu_0 = 0.008, \lambda = 0.999, \delta = 50$. Non-power-norm. QLMS and OS-QFDAF $M = 6, \mu = 0.008$)

The step size in a power-normalized algorithm is an overall parameter governing the diagonal inverse power matrix. Changing its value does not have a significant effect on the final result. If power normalization is not applied, the step size modifies the algorithm behavior in the same way as in QLMS [14].

5 Conclusions

Quaternions surely have an impact on physics and engineering. Within the hypercomplex number subfields, multidimensional filtering gives the possibility to process data accordingly to their typical space dimensions, thus obtaining robustness and coherence from the results. This is feasible since the correlation between different dimensions is considered and multidimensional data are processed as a whole. The processing of four-dimensional signals increases the computational complexity significantly. For this reason we implemented a quaternionic class of frequency-domain adaptive filters. For instance, in audio signal processing, long impulse response filters require a long memory. In such a case, time-domain processing is not recommended. Block filters in the frequency domain provide a fast computation and improved statistical properties of the signals, as well. Recently, the definition of a quaternionic neuron and the application of quaternion filtering to neural networks have been explored. On this trend, our next efforts are continuing on this route.

References

1. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.: Multilayer perceptrons to approximate quaternion valued functions. Neural Netw. **10**, 335–342 (1997)
2. Ell, T.A., Sangwine, S.J.: Decomposition of 2d hypercomplex fourier transforms into pairs of complex fourier transforms. In: Proceedings of 10th European Signal Processing Conference

3. Haykin, S.: Adaptive Filter Theory. Prentice Hall (1996)
4. Hitzer, E.: Introduction to Clifford's geometric algebra. SICE J. Control, Measur. Sys. Integr. **4**(1), 1–10 (2011)
5. Isokawa, T., Nishimura, H., Matsui, N.: Quaternionic multilayer perceptron with local analyticity. Information **3**, 756–770 (2012)
6. Jahanchahi, C., Took, C., Mandic, D.: A class of quaternion valued affine projection algorithms. Signal Process. **93**, 1712–1723 (2013)
7. Jahanehahi, C., Took, C.C., Mandic, D.P.: The widely linear quaternion recursive least squares filter. In: Proceedings of 2nd International Workshop Cognitive Information Processing (CIP) (2010)
8. Jiang, M., Liu, W., Li, Y.L., Zhang, X.: Frequency-domain quaternion-valued adaptive filtering and its application to wind profile prediction. In: IEEE Region 10 Conference TENCON, pp. 1–5, Oct 2013
9. Kraft, E.: A quaternion-based unscented Kalman filter for orientation tracking. In: Proceedings of 6th International Conference on Information Fusion (ISIF), pp. 47–54 (2003)
10. Kusamichi, H., Isokawa, T., Matsui, N., Y. Ogawa, Y., Maeda, K.: A new scheme for color night vision by quaternion neural network. In: Proceedings of 2nd International Conference Autonomous Robots and Agents, pp. 101–106, Dec 2004
11. Nitta, T.: An extension of the back-propagation algorithm to quaternion. In: 3rd Int. Conference on Neural Information Processing ICONIP, pp. 247–550, Sep 1996
12. Yefremov, A.P.: Number, geometry and nature. Hypercomplex Numbers Geom. Phys. **1**(1), 3–4 (2004)
13. Pei, S.C., Ding, J.J., Chang, J.H.: Efficient implementation of quaternion fourier transform, convolution, and correlation by 2-d complex fft. IEEE Trans. Signal Process. **11**(49), 2783–2797 (2001)
14. Took, C.C., Mandic, D.P.: The quaternion LMS algorithm for adaptive filtering of hypercomplex processes. IEEE Trans. Signal Process. **57**(4), 1316–1327 (2009)
15. Uncini, A.: Fundamentals of Adaptive Signal Processing. Springer (2015)
16. Yoshida, M., Kuroe, Y., Mori, T.: Models of Hopfield-type quaternion neural networks and their energy function. Int. J. Neural Syst. **15**, 129–135 (2005)