SAPIENZA
UNIVERSITÀ DI ROMA

Hypercomplex Adaptive Filtering

PhD Course in Information and Communications Technologies
Curriculum in Information and Communication Engineering
XXX Cycle

Candidate

Francesca Ortolani
ID number 796175

Supervisor

Prof. Aurelio Uncini

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in ICT

October 2017
Dpt. of Information Engineering, Electronics and Telecommunications

Thesis not yet defended

Hypercomplex Adaptive Filtering

Ph.D. thesis. Sapienza – University of Rome

.

Version: October 31, 2017

Author's email: francesca.ortolani@uniroma1.it

Usque ad finem

Abstract

The degree of diffusion of hypercomplex algebras in adaptive and non-adaptive filtering research topics is growing faster and faster. The performance of hypercomplex adaptive filters has been widely experimented during the last decade. Quaternion filters, in particular, have been utilized in systems where the signals to be processed have some form of correlation. Besides correlation, the debate today concerns the usefulness and the benefits of representing multidimensional systems by means of these complicated mathematical structures and the criterions of choice between one algebra or another. One of the goals of this work is to discuss whether the choice of a certain algebra in the description of a problem/environment can play a significant role and determine an adaptive filter performance.

That said, adaptive filtering can be expanded to new numerical systems and unseen sides of physical problems can be highlighted thanks to the mathematical properties of such hypercomplex algebras. Each algebra has its own rules and calculation outcomes may not be compatible from one algebra to another. However, such peculiarities diversify algebras in a way that each of them fits specific geometrical/physical problems.

The bulk of study and experiments presented in this work was carried out in a 3-Dimensional (3D) audio context. 3D audio is the new frontier in audio technology and it is quickly taking place in many applications, from cinema to virtual reality, audio surveillance and video games. The large amount of data requires fast and compact solutions for signal processing. With this aim in view, research is moving towards the exploration of hypercomplex algebras in order to find a non-redundant and compact form for the representation of 3D sound fields without loss of information. Quaternion sound fields are currently under investigation and this thesis presents some recent results concerning the integration of hypercomplex (quaternion) adaptive signal processing into a 3D audio environment.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Aurelio Uncini for his support and guidance, invincible advises, infinite patience and for letting me try, fail, re-try and succeed.

My sincere thanks also goes to Prof. Michele Scarpiniti and Prof. Danilo Comminiello for their contribution and for constantly reviewing my work.

Thanks to Dr. Simone Scardapane, for making these 1096 days funnier.

Thanks to my PhD Advisory Board: Prof. Elio Di Claudio, Prof. Francesca Cuomo and Prof. Sergio Barbarossa. Thank you all for suggesting me some great ideas.

Heartfelt thanks to my external assessors Prof. Felix Yanovsky and Prof. Alexander Nosich. Thank you for believing in me.

Большое спасибо Мастер Анатолий Пивоваренко за физическую и психическую тренировку.

Of course, infinite thanks to mum and dad. After three years listening to my theories and reasonings, they have probably become the greatest experts on quaternion adaptive filtering on the planet.

List of Figures

List of Tables

List of Acronyms

LMS Least Mean Square

BLMS Block Least Mean Square

CLMS Complex Least Mean Square

QLMS Quaternion Least Mean Square

BQLMS Block Quaternion Least Mean Square

LMSE Linear Mean Square Estimation

TLMS Tessarine Least Mean Square

OSQFDAF Overlap-Save Quaternion Frequency Domain Adaptive Filter

OSTFDAF Overlap-Save Tessarine Frequency Domain Adaptive Filter

WL Widely Linear

WLBQLMS Widely Linear Block Quaternion Least Mean Square

WLMSE Widely Linear Mean Square Estimation

WLQLMS Widely Linear Quaternion Least Mean Square

MSE Mean Square Error

EMSE Excess Mean Square Error

Contents

Introduction to Hypercomplex Adaptive Filtering

Introduction

The enthralling side of hypercomplex algebras attracted scientists and encouraged the world of research to uncover the environments and the circumstances in which the application of hypercomplex numbers finds fertile ground. After the initial definition of these new numerical systems [6, 13, 34, 55], a wave of study touched physics and the description of classical problems [1]. Hypercomplex numbers extend the definition of real and complex algebras, and quaternions represent the simplest mathematical object right after complex numbers [34]. The compact formalism provided by quaternion maths achieved resounding success and many 3D problems in physics and their laws were reformulated in a quaternion fashion, e.g. Maxwell's equations of electromagnetism [15], Newton's laws of motion [39], gravity [93], quantum mechanics and the electron wave theory [107], etc. The new formalism surely simplified the expression and the divulgation of many scientific theories among insiders. Nevertheless, considered that these hypercomplex algebras might not be easily understandable for those people having an unspecialized knowledge of mathematics, soon a question arose: do we really need quaternions and octonions? Probably, the answer arrived from the world of engineering. The challenge has been the provision of convincing reasons why hypercomplex signal processing is favorable (or not). It is known that the orientation information is intrinsic to quaternion-valued objects [9]. Therefore, quaternions found a new revival in 3D rotation applications. These include avionics, 3D graphics and virtual reality, modeling in chemistry, etc. The experimentation of quaternion algebra and the implementation of quaternion systems determined a first motivation why quaternions are irreplaceable in these fields: before quaternions, all rotations were traditionally expressed in Euler angles. Unfortunately, such a representation is affected by system deadlocks occurring at critical angles. Practically, the rotation system may degenerate and lose a degree of freedom [35, 80].

One of the trends in the last two decades in digital signal processing has been the exploration of hypercomplex algebras for multidimensional signal processing with particular regard to adaptive filtering and intelligent systems [23, 40, 63, 74, 98]. Since complex numbers were widely experimented and studied in both linear and nonlinear environments [56], the immediate step forward in hypercomplex signal processing has considered quaternions [98] and octonions [48]. Adaptive filters are

self-adjusting systems (Fig. 0.1), so human intervention is restricted to the filter development and parameter setup phases.

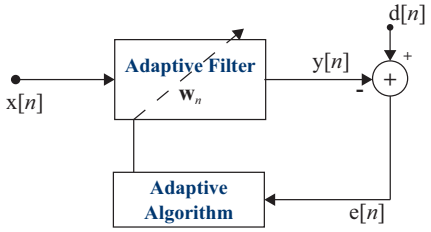


Figure 0.1. Adaptive filter.

Scientists studied how such *automatic* systems behave with changing the mathematical format of input and output signals and the filter architecture, accordingly [96,101]. It was experimented that signal component correlation plays a significant role in the adaptation process, thus speeding up the convergence rate and improving the filter performance [99, 100]. On this trend, researchers developed several algorithms in quaternion algebra [31, 46, 59, 98, 115].

Our research group at the ISPAMM laboratory (Sapienza) also committed to this topic. Our main contributions include the development of a class of quaternion-valued adaptive filters in the frequency domain [67], the investigation of the usage of quaternion-valued filters in 3-Dimensional (3D) audio applications [66, 68, 71], and a comparison of isodimensional hypercomplex algebras [69]. Besides this, the group proposed and encouraged the formalization of *hypercircuits* and a *hypercircuit theory* [67, 72], examining in depth the properties and the differences of quaternion mathematical transforms in comparison with real and complex systems with the aim of supplying essential mathematical tools to signal processing.

Scientific and technologic motivations

Adaptive filtering finds use in many different fields. It is customary to see adaptive filters embedded in RADAR and SONAR applications, weather forecast, seismology, control applications (navigation, tracking and guidance), audio processing applications (e.g. audio restoration, immersive audio, etc.) and so on. Specifically, adaptive filters can be employed in noise, interference and echo cancelling, channel equalization, system identification, inverse system modeling [104]. As introduced in the previous paragraph, it is known that hypercomplex algebras allow the description of a wide variety of geometrical objects (from points to hyperspaces), thus simplifying the study of many physical problems. Besides that, these multidimensional structures can be used to assemble different information into one single entity. An example of that can be found in complex numbers, also. In fact, we can transport the information of amplitude and phase by means of one single numerical object. Complex numbers gave us the possibility to represent the spectrum of a signal and today they are absolutely necessary to circuit and signal theories.

That said, within the hypercomplex algebras, multidimensional filtering gives the possibility to process data accordingly to their typical space dimensions, thus obtaining robustness and coeherence from the results.

Hypercomplex filters for 3D Sound Space

Anytime we purchase a modern electronic audio device, we are likely to find some intelligent, at least *adaptive*, sub-system running in the background to accomplish some special task. Adaptive filters play a conspicuous role in noise and echo control, feedback suppression, room equalization and many other applications. Devices embedding these functionalities are smartphones, personal computers, sound correction and enhancement systems, loudspeaker systems and others. Since the methods adopted in adaptive signal processing are quite mature in the case of single and stereo channels [24, 26, 89, 108], research is focusing on finding better solutions for those systems having a larger number of channels. These systems include surround sound and 3D audio. Besides reliability, we are interested in developing a compact and efficient form for multidimensional data processing and extrapolating and exploiting from data information as much as possible, in order to minimize the error in the adaptation processes. 3D Audio has been extensively explored by our research work and further investigation is planned.

It is possible to define a set of fundamental problems concerning 3D audio processing and adaptive filtering. Firstly, a proper mathematical format, capable of revealing all the aspects and the sides of the problem, should be chosen. In this regard, we considered a quaternion representation for the Ambisonic 1st order B-Format signals [29, 30]. The B-Format is composed of 4 audio signals and each signal has been assigned to a quaternion component. This assignment is consistent with a sound space transformation: the technique decomposes the sound field into spherical harmonics and we transformed the sound space into a proper quaternion format [66, 68]. We obtained some interesting results by processing B-Format signals with different 4-dimensional algebras: quaternions vs. tessarines [69, 70]. It can be seen that the Ambisonic 3D space and the B-Format microphone array have a well-determined geometry. Results from simulations with quaternion and tessarine adaptive filters revealed that, the Ambisonic B-Format is rather inclined to be represented by quaternions than tessarines. In fact, the quaternion filters exhibit a faster convergence rate and improved stability in comparison with their tessarine counterparts. A second point in question with 3D audio processing is the computational burden. Quaternions provide a convenient representation from this point of view. Alternative representations with matrices, for instance, introduce redundant information which are not suited to real-time 3D audio. Moreover, in acoustic applications, because of the long-lasting tail of reflections to handle, the adaptive filter is designed to have a long impulse response, that is, a long memory is needed, thus resulting in a significant increase in the computational complexity of the algorithm [104]. In such a case, especially in multichannel/3D applications, time domain processing is not recommended. A wiser approach in designing the filter combines a block implementation of an FIR filter and transform domain processing. That is the reason why we started investigating the development of quaternion-valued filters operating in the frequency-domain [67]. Last but not least, we searched for ways to exploit the statistical properties of signals in order to improve the filter performance. For this purpose, we experimented the use of *widely linear* filters [66, 100] in the quaterion domain, since these algorithms include full signal second order statistics into adaptive processing. We observed from simulations that

correlated quaternion signal components require widely linear algorithms in order to obtain faster convergence to optimum.

The promising results suggest to keep on investigating in hypercomplex algebras and multidimensional problems. Changing the algebra can have an influence over convergence rate and filter stability.

Rotating systems and augmented reality

Augmented reality (AR) is a branch of signal processing aimed at enhancing the surrounding environment with extra information by elaboration of the available data. The very first AR devices supported video and geolocalization, and found place especially in civil and military avionics, where the information for pilots can be displayed on screens and on-body viewers [58, 79]. The always growing diffusion of miniaturized and wearable sensor devices determined the success and development of new AR equipment in the most various application fields, including everyday life [90, 92]. Moreover, because of the increasing demand for intelligent systems, new smart AR devices are widespread amongst digital processors. At present, the integration of full 3D audio into AR systems is one of the recent challenges in digital signal processing. The Ambisonic technique is a suitable candidate to be embedded into smart AR systems. In fact, for instance, the 1st order B-Format only employs 4 microphones and the processing effort is not extremely burdensome.

In 3D digital rotations, the so-called *Gimbal Lock* phenomenon is generated in a way similar to the case of inertial guidance in avionics, even though gyroscopes or torques are not present [9]. One of the strong points in quaternion algebra (and Clifford algebras, in general) is that object representation is coordinate-free. In other words, an object embeds a coordinate structure and motion can be described with respect to it. In view of the above, a quaternionic audio representation could espouse a quaternionic image rendering in a virtual reality context.

Chapter 1

Hypercomplex Algebras

Contents

1.1 Introduction to Hypercomplex Algebras

As implied by the name, *hypercomplex* numbers are a set of elements generalizing real and complex numbers. Some of the sub-groups belonging to this field include Quaternions (Hamilton 1843), Octonions (Graves 1843), Tessarines (Bicomplex numbers), Coquaternions, Biquaternions, Pauli algebra (Euclidean 3D algebra), Dirac algebra, Twistors (Penrose 1967) and others. In 1872, Benjamin Peirce started cataloguing these new numerical fields in [75]. Peirce's method for hypercomplex algebra identification classifies numerical sets into *idempotent* numerical systems [94], each containing one or more idempotent numbers, and *non-idempotent* systems, each containing no idempotent numbers. In other words, any given hypercomplex number system must contain an idempotent number I (i.e. $I \neq 0, \quad I^2 = I$) or, alternatively, every element of the system must be nilpotent (i.e. $I^n = 0$ for some positive integer n). Moreover, the units of the idempotent number systems are regularized with respect to one element from the set (e.g. I), namely the *basis*. That is, the units $\{e_1, e_2, e_3, ...\}$ are chosen with reference to I and they must belong to one of four groups defined in [94].

A second classification method uses the Cayley-Dickson construction. This method is based upon *involutions* to generate complex numbers, quaternions and octonions, given a real-valued basis. Although intuitive and practical, such a strategy is efficient only for a few hypercomplex sub-groups. Hurwitz theorem (normed division algebras) and Frobenius theorem (real division algebras) define the applicability limits for the Cayley-Dickson decomposition (see Par. 1.2). Nevertheless, this method has been largely exploited in this work to develop most of the algebraic quaternion-valued functions in use. Quaternions, for instance, belong to Clifford algebra (also known as Hypercomplex Algebra). Clifford algebra was introduced by William Kingdom Clifford as a sub-field of the geometric algebra previously invented by Hermann Günther Grassmann. Grassmann's geometric algebra was expanded into Clifford algebra in 1878 and named in honour of Clifford's efforts. It consists of a generalisation of the complex numbers, which have formerly existed since the 15th century [37]. In wider terms, a generic n-dimensional hypercomplex algebra has at least one *non-real* axis (such as the imaginary axis $\mathbf{j}$ in complex numbers) and is closed under addition and multiplication:

$$a = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3 + \cdots + a_n\mathbf{e}_n = \sum_{\nu=1}^{n} a_\nu \mathbf{e}_\nu \in \mathbb{A} \tag{1.1}$$

with $a_1, ..., a_n \in \mathbb{K}$ and the orthonormal basis $\mathbf{e}_1, \cdots, \mathbf{e}_n$. Clifford algebra has more than one non-real axis. Clifford unified and generalized the works of Hamilton and Grassman by finalizing the fundamental concept of *directed numbers*.

Universal property of Clifford algebra "Any isometry from the vector space V into an inner-product algebra $\mathcal{A}$ over the field $\mathbb{K}$ can be uniquely extended to an isometry from the Clifford algebra $\mathcal{C}\ell(V)$ into $\mathcal{A}$. Clifford algebra is the unique associative and multilinear algebra with this property" [37].

In Par. 1.2 we will show how the eight- or higher-dimensional Cayley-Dickson construction is not associative with respect to multiplication (the same occurs with

split-complex constructs). On the contrary, Clifford algebras are associative at any dimensionality [37].

In Clifford algebras, it is possible to define a set of bases $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_k\}$, orthogonal vectors, which are generated out of the orthogonalization of a quadratic form, with the use of a symmetric scalar product $\mathbf{u} \cdot \mathbf{v} = \frac{1}{2}(uv + vu)$ defined over the field of real numbers. The set of bases is such that

$$\frac{1}{2}(\mathbf{e}_i \mathbf{e}_j + \mathbf{e}_j \mathbf{e}_i) = \begin{cases} -1, 0, +1, & i = j \\ 0, & i \neq j \end{cases} \tag{1.2}$$

Clifford algebras are closed under multiplication. This constraint gives rise to a multivector space spanned by 2^k bases $\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, ..., \mathbf{e}_1\mathbf{e}_2, ..., \mathbf{e}_1\mathbf{e}_2\mathbf{e}_3, ...\}$, which are indeed the bases of a hypercomplex system. Unlike the bases $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_k\}$, which do not commute (i.e. $\mathbf{e}_1\mathbf{e}_2 \neq \mathbf{e}_2\mathbf{e}_1 \rightarrow \mathbf{e}_1\mathbf{e}_2 = -\mathbf{e}_2\mathbf{e}_1$), the remaining bases may or may not anti-commute, depending on how many times the elements swap in the product (e.g. $\mathbf{e}_1(\mathbf{e}_2\mathbf{e}_3) = (\mathbf{e}_2\mathbf{e}_3)\mathbf{e}_1$).

Clifford algebras (e.g. over the reals $\mathbb{R}$, that is the coefficients of the algebra are real numbers) are usually denoted by $\mathcal{C}\ell_{p,q}(\mathbb{R})$. Such a notation indicates that the algebra is generated by p bases with $e_i^2 = +1$ and q bases with $e_i^2 = -1$. Such Clifford algebras ignore those bases degenerating the quadratic form over which the vector space is defined, i.e. the directions along which $\mathbf{e}_i^2 = 0$. These algebras (also called geometric algebras) include some of the numerical fields used in this work, such as the complex numbers $\mathcal{C}\ell_{0,1}(\mathbb{R})$ and quaternions $\mathcal{C}\ell_{0,2}(\mathbb{R})$ and other remarkable sets which can be found in different fields of physics and engineering, where rotations and spins are involved. Fields belonging to this category are classical and quantum mechanics, relativity, electromagnetic theory, acoustics, navigation, imaging and beamforming applications.

1.2 Normed division algebras: complex numbers, quaternions, octonions

In division algebras, each non-zero element $a \in \mathbb{A}$ has one and only one inverse element $a^{-1} \in \mathbb{A}$. Only $\mathbb{R}$, $\mathbb{C}$, $\mathbb{H}, \mathbb{O}$ are division algebras over the field of the reals. These algebras are introduced just below.

1.2.1 Complex numbers

Complex numbers were discovered in Italy in the 15th century although their geometrical meaning was canonised in the 18th century. Their discovery was due to finding a general solution to cubic equations, such as

$$x^3 + ax^2 + bx + c = 0. \tag{1.3}$$

Complex numbers are defined as (in algebraic form)

$$a + b\mathbf{i} \tag{1.4}$$

where a and b are real numbers and they are called respectively the *real* and the *imaginary* parts of the complex number and $\mathbf{i} = \sqrt{-1}$ is the *imaginary unit*. The field of complex numbers $\mathbb{R}$ can be also interpreted as a 2-dimensional real space $\mathbb{R}^2$

$$a + b\mathbf{i} \to (a, b)\,. \tag{1.5}$$

The pair of numbers in brackets represents indeed a point on a complex plane defined as shown in Fig. 1.1.

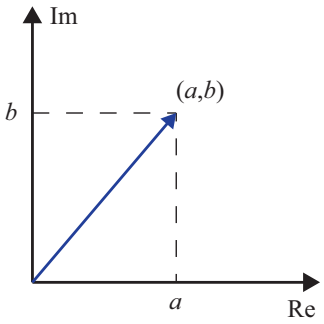


Figure 1.1. Complex plane.

In Fig. 1.1 the horizontal axis is for the real part and the vertical axis for the imaginary part. In $\mathbb{C}$, addition, multiplication by reals and multiplication and division between complex numbers are allowed:

- Addition: $(a + b\mathbf{i}) + (c + d\mathbf{i}) = (a + c) + (b + d)\,\mathbf{i}$
- Multiplication by reals: $r\,(a + b\mathbf{i}) = ra + rb\mathbf{i}$
- Complex multiplication: $(a + b\mathbf{i}) \cdot (c + d\mathbf{i}) = (ac - bd) + (bc + ad)\,\mathbf{i}$
- Conjugate: $\overline{a + b\mathbf{i}} = a - b\mathbf{i}$
- Norm (length): $\|a + b\mathbf{i}\| = a^2 + b^2 = (a + b\mathbf{i})\,\overline{(a + b\mathbf{i})}$
- Division between complex numbers: $\frac{(a+b\mathbf{i})}{(c+d\mathbf{i})} = (a + b\mathbf{i})\,\frac{\overline{(c+d\mathbf{i})}}{\|c+d\mathbf{i}\|}$.

In view of this short recall about complex numbers, it is plain to see that the complex field belongs to the family of *normed division algebras*. Such numerical fields are multidimensional real sets, $\mathbb{R}^n$, with division, that is, multiplication has to be defined so that division is allowed. On the other hand, addition and norm are defined in a familiar way. That said, real numbers are a 1-dimensional normed division algebra, complex numbers are a 2-dimensional normed division algebra. Is there a 3-dimensional normed division algebra? The answer is *no* and the reason for that will be clear in a while and it concerns the fact that multiplication and division cannot be defined. On the contrary, a 4-dimensional normed division algebra does exist. It is the quaternion field $\mathbb{H}$.

1.2.2 Quaternions

Quaternions were discovered by Sir. William Hamilton in 1843 in Ireland. Admittedly, Hamilton never knew his quaternions would have been used in 3D graphics, acoustics,

in avionics, navigation, mechanics, quantum physics and so on. Quaternions ($\mathbb{H}$) are a geometric algebra in the sense of Clifford algebras. Such an algebra is a *unital associative algebra*, i.e. it contains an *identity* element I, such that $Iq = q, \forall q$ in the algebra and it is endowed with the operations of associative multiplication of elements in the algebra and scalar multiplication. Quaternion algebra (and any Clifford algebra, in general) contains and is generated by an *inner product vector space*[1] V over a field K $(V, a \cdot b : a, b \in V \to \mathbb{R})$.

A quaternion is made up of 4 components: one real and 3 imaginary components:

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} = q_0 + \mathbf{q}. \tag{1.6}$$

A quaternion having real part $q_0 = 0$ 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 $\mathbb{R}^3$ and comply with the fundamental properties shown below (obtained by cyclic permutation):

$$\mathbf{ij} = \mathbf{i} \times \mathbf{j} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{j} \times \mathbf{k} = \mathbf{i}, \quad \mathbf{ki} = \mathbf{k} \times \mathbf{i} = \mathbf{j} \tag{1.7}$$

and

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1. \tag{1.8}$$

The sum of quaternions is defined in a way similar to the sum of complex numbers:

$$\begin{aligned} q \pm p &= (q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \pm (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \\ &= (q_0 \pm p_0) + (q_1 \pm p_1)\,\mathbf{i} + (q_2 \pm p_2)\,\mathbf{j} + (q_3 \pm p_3)\,\mathbf{k}. \end{aligned} \tag{1.9}$$

A quaternion vector space is a *skew field* or *division ring*. In skew fields, division is possible and such division rings are characterized by the fact that multiplication is not necessarily commutative. In fact, with quaternions we have $\mathbf{ij} \neq \mathbf{ji}$ and

$$\mathbf{ij} = -\mathbf{ji} \qquad \mathbf{jk} = -\mathbf{kj} \qquad \mathbf{ki} = -\mathbf{ik}. \tag{1.10}$$

The product between quaternions q_1 and q_2 is computed as

$$\begin{aligned} q_1 q_2 &= (a_0 + a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k})\,(b_0 + b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}) \\ &= (a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3) \\ &+ (a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2)\,\mathbf{i} \\ &+ (a_0 b_2 - a_1 b_3 + a_2 b_0 + a_3 b_1)\,\mathbf{j} \\ &+ (a_0 b_3 + a_1 b_2 - a_2 b_1 + a_3 b_0)\,\mathbf{k}. \end{aligned} \tag{1.11}$$

The quaternion product can be defined in a compact form. Given two quaternions $p = p_0 + \mathbf{p}$ and $q = q_0 + \mathbf{q}$, their product is

$$pq = (p_0 + \mathbf{p})(q_0 + \mathbf{q}) = p_0 q_0 - \mathbf{p} \cdot \mathbf{q} + p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \wedge \mathbf{q} \tag{1.12}$$

where $\mathbf{p} \wedge \mathbf{q}$ is the *cross* product between pure quaternions $\mathbf{p}$ and $\mathbf{q}$ and $\mathbf{p} \cdot \mathbf{q}$ is their *dot* product.

[1] An *inner product vector space* is a vector space V over $\mathbb{R}$ with an inner product map $\langle .\,, .\rangle : V \times V \to \mathbb{R}$.

The dot product of two quaternions p and q is defined as

$$p \cdot q = p_0 q_0 + p_1 q_1 + p_2 q_2 + p_3 q_3. \tag{1.13}$$

Finally, the conjugate of a quaternion $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ is defined as

$$q^* = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k} \tag{1.14}$$

and we have the following properties:

$$(p^*)^* = p, \quad (pq)^* = q^* p^* \tag{1.15}$$

The module (norm) of q is

$$|q| = \sqrt{w^2 + x^2 + y^2 + z^2} = |q^*|. \tag{1.16}$$

and satisfies the following properties:

$$|q^*| = |q|\,, \quad |pq| = |p|\,|q|\,. \tag{1.17}$$

The inverse of a quaternion $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ is

$$q^{-1} = \frac{q^*}{|q|^2} \tag{1.18}$$

The division by $|q|$ in (1.18) is meant to be a component-wise division. The following properties are applicable:

$$\left(q^{-1}\right)^{-1} = q, \quad (pq)^{-1} = q^{-1} p^{-1} \tag{1.19}$$

This introduction about quaternions will continue in the next sections (from Par. 1.3 on) with more detailed information and comments about quaternion matrix algebras, quaternion transforms and mathematical operations. Recommended readings about quaternions are [20, 37], where further properties and definitions can be found.

1.2.3 Octonions

What about higher dimensions? Octonions $\mathbb{O}$ are an 8-dimensional normed division algebra. They were invented by John Graves in 1843, after Hamilton's inspiration. Octonions are defined as

$$o = a_0\mathbf{e}_0 + a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3 + a_4\mathbf{e}_4 + a_5\mathbf{e}_5 + a_6\mathbf{e}_6 + a_7\mathbf{e}_7 \tag{1.20}$$

The versors $\{\mathbf{e}_0, \ldots, \mathbf{e}_7\}$ are square roots of -1. Multiplication can be carried out according to the rules in Table 1.1.

$\times$	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$
$\mathbf{e}_0$	$\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$	$\mathbf{e}_6$	$\mathbf{e}_7$
$\mathbf{e}_1$	$\mathbf{e}_1$	$-\mathbf{e}_0$	$\mathbf{e}_3$	$-\mathbf{e}_2$	$\mathbf{e}_5$	$-\mathbf{e}_4$	$-\mathbf{e}_7$	$\mathbf{e}_6$
$\mathbf{e}_2$	$\mathbf{e}_2$	$-\mathbf{e}_3$	$-\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_6$	$\mathbf{e}_7$	$-\mathbf{e}_4$	$-\mathbf{e}_5$
$\mathbf{e}_3$	$\mathbf{e}_3$	$\mathbf{e}_2$	$-\mathbf{e}_1$	$-\mathbf{e}_0$	$\mathbf{e}_7$	$-\mathbf{e}_6$	$\mathbf{e}_5$	$-\mathbf{e}_4$
$\mathbf{e}_4$	$\mathbf{e}_4$	$-\mathbf{e}_5$	$-\mathbf{e}_6$	$-\mathbf{e}_7$	$-\mathbf{e}_0$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$
$\mathbf{e}_5$	$\mathbf{e}_5$	$\mathbf{e}_4$	$-\mathbf{e}_7$	$\mathbf{e}_6$	$-\mathbf{e}_1$	$-\mathbf{e}_0$	$-\mathbf{e}_3$	$\mathbf{e}_2$
$\mathbf{e}_6$	$\mathbf{e}_6$	$\mathbf{e}_7$	$\mathbf{e}_4$	$-\mathbf{e}_5$	$-\mathbf{e}_2$	$\mathbf{e}_3$	$-\mathbf{e}_0$	$-\mathbf{e}_1$
$\mathbf{e}_7$	$\mathbf{e}_7$	$-\mathbf{e}_6$	$\mathbf{e}_5$	$\mathbf{e}_4$	$-\mathbf{e}_3$	$-\mathbf{e}_2$	$\mathbf{e}_1$	$-\mathbf{e}_0$

Table 1.1. Octonion multiplication rules.

Addition, norm and conjugation are defined as in quaternion algebra. It is important to highlight that octonion multiplication is not associative.

Recalling Hurwitz Theorem (1898):
The only normed division algebras are real numbers, complex numbers, quaternions and octonions.

Normed division algebras can be built from the existing ones by doubling, i.e. $b = a_1 + a_2\mathbf{i}_{n+1} \in \mathbb{B}$, with $a_1, a_2 \in \mathbb{A}$ and $\mathbf{i}_{n+1}^2 = -1$ is a new imaginary unit:

- $\mathbb{C} = \mathbb{R} + \mathbb{R}\mathbf{i}$
- $\mathbb{H} = \mathbb{C} + \mathbb{C}\mathbf{j}$
- $\mathbb{O} = \mathbb{H} + \mathbb{H}$ (i.e. combination of two $\mathbb{H}$: $o = q_1 + q_2$,
  where $q_1 = a\mathbf{e}_0 + b\mathbf{e}_1 + c\mathbf{e}_2 + d\mathbf{e}_3$ and $q_2 = \alpha\mathbf{e}_4 + \beta\mathbf{e}_5 + \gamma\mathbf{e}_6 + \delta\mathbf{e}_7$).

So, is there a 16-dimensional normed division algebra? *No*, because octonion algebra is not associative and a non-associative normed division algebra cannot be doubled.

1.3 Quaternion Algebra

We extensively introduced quaternion math in Par. 1.2.2 since in this work, we will focus on the properties of this particular algebra and discuss adaptive filters implemented in this algebra.

1.3.1 Quaternion matrices

Let $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{C}^{n\times m}$, it is possible to obtain a quaternion matrix by combining two complex-valued matrices as

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2\mathbf{j}. \tag{1.21}$$

This is exacly the Cayley-Dickson decomposition and it can applied to numbers, matrices and functions, as well. Cayley-Dickson decomposition is a powerful *trick* that we will often employ later on this work.

Given $\mathbf{A}, \mathbf{B} \in \mathbb{H}^{n\times m},\ \lambda \in \mathbb{H}$,

$$\lambda\mathbf{A} \neq \mathbf{A}\lambda \tag{1.22}$$

$$(\mathbf{AB})^T \neq \mathbf{B}^T\mathbf{A}^T \tag{1.23}$$

$$\mathbf{AB}^H \neq \mathbf{A}^H\mathbf{B}^H \tag{1.24}$$

$$(\mathbf{AB})^H = \mathbf{B}^H\mathbf{A}^H \tag{1.25}$$

where H denotes the conjugate transpose (self-adjoint) of a matrix ($\mathbf{A}^H = \overline{\mathbf{A}}$).

1.3.2 More about quaternion vector-matrix product

The implementation of the algorithms presented in this work requires the calculation of vector-matrix products. This paragraph is intended to enlighten the difference between the *Hadamard* product and the *diagonal* product, as shown below: Let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}^T$, $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_N \end{bmatrix}^T$, where $x_1, \dots, x_N$ and $w_1, \dots, w_N$ may be whatsoever type of number: reals, complex numbers, quaternions, etc.

Hadamard product

$$\mathbf{x} \otimes \mathbf{w} = \mathbf{x}^T\mathbf{w} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 \tag{1.26}$$

Diagonal product

$$\operatorname{diag}(\mathbf{x})\mathbf{w} = \begin{bmatrix} x_1 & 0 & 0 & 0 \\ 0 & x_2 & 0 & 0 \\ 0 & 0 & x_3 & 0 \\ 0 & 0 & 0 & x_4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} x_1w_1 \\ x_2w_2 \\ x_3w_3 \\ x_4w_4 \end{bmatrix} \tag{1.27}$$

1.3.3 Quaternion representations

Matrix representation

Alike complex numbers, so quaternions can be represented as matrices. A brief brush-up on complex algebra will expedite the understanding of the quaternionic case. Given a complex number $z = a + \mathbf{i}b$ it can also be represented by a 2×2 matrix, as

$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix} \tag{1.28}$$

Multiplying or summing two of such matrices returns matrices of this same form. Similarly, the sum and product of complex numbers can be converted into the sum and product of such matrices. The computation of the modulus of z elucidates the idea:

$$|z|^2 = \begin{vmatrix} a & -b \\ b & a \end{vmatrix} = a^2 + b^2 \tag{1.29}$$

In the quaternion field $\mathbb{H}$, the situation is akin to complex matrices. The objective is to find a form of matrices in a way that quaternion sum and multiplication correspond to the sum and multiplication of matrices, all of them having the same form, as it was in the case of complex numbers. This is achievable in at least two ways. One is to use 2×2 complex matrices and the other is to use 4×4 real matrices. In the abstract algebra jargon, these are injective *homomorphisms* from $\mathbb{H}$ to the matrix rings $\mathrm{M}(2, \mathbb{C})$ and $\mathrm{M}(4, \mathbb{R})$, respectively. Given a quaternion $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, we have the two possibilities:

A) 2×2 **matrix**

$$\begin{bmatrix} a + b\mathbf{i} & c + d\mathbf{i} \\ -c + d\mathbf{i} & a - b\mathbf{i} \end{bmatrix} \tag{1.30}$$

If we compute the determinant of matrix (1.30), we get

$$\begin{vmatrix} a + b\mathbf{i} & c + d\mathbf{i} \\ -c + d\mathbf{i} & a - b\mathbf{i} \end{vmatrix} = a^2 + b^2 + c^2 + d^2 \tag{1.31}$$

which is equal to the squared quaternion norm $|q|^2$. This is possible if the matrix is treated as such. A better definition for representation (1.30) is a map. In fact, matrix (1.30) maps quaternions into a 2×2 complex field. This representation embeds some important properties:

- Complex numbers ($c = d = 0$) correspond to diagonal quaternion matrices.
- Real numbers ($b = c = d = 0$) correspond to diagonal quaternion matrices.
- The norm of a quaternion is the square root of the determinant of the corresponding matrix.
- The conjugate of a quaternion corresponds to the conjugate transpose of the matrix.

B) 4×4 **matrix**

$$\begin{aligned} \begin{bmatrix} a & b & c & d \\ -b & a & -d & c \\ -c & d & a & -b \\ -d & -c & b & a \end{bmatrix} = a \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + b \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ + c \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} + d \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{1.32}$$

and we find the following properties:

- The conjugate of a quaternion corresponds to the transpose of the matrix.
- The fourth power of the norm of a quaternion is equal to the determinant of the corresponding matrix (not the squared norm as in 2-by-2 complex matrices).
- Complex numbers ($c = d = 0$) are block diagonal matrices with two 2×2 blocks.
- Real numbers ($b = c = d = 0$) correspond to diagonal quaternion matrices.

Polar representation

Similarly to complex numbers, quaternions can be represented in polar form. Given $q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} = a + \mathbf{v}$,

$$q = |q|\, e^{\mu\theta} = |q|\,(\cos\theta + \mu \sin\theta) \tag{1.33}$$

Correspondingly, $a = |q|\cos\theta$, $\mathbf{v} = \mu\,|\mathbf{v}| = \mu\,|q|\sin\theta$, where μ is a pure *unitary* quaternion.
Ensuing from (1.33), the α-th power of quaternion q is

$$q^\alpha = |q|^\alpha e^{\mu\alpha\theta} = |q|^\alpha\,[\cos(\alpha\theta) + \mu \sin(\alpha\theta)] \tag{1.34}$$

Polar coordinates Polar coordinates in the quaternion space $\mathbb{H}$ are defined as

$$\begin{cases} a = \|q\|\cos\theta \\ b = \|q\|\sin\theta\cos\varphi \\ c = \|q\|\sin\theta\sin\varphi\cos\psi \\ d = \|q\|\sin\theta\sin\varphi\sin\psi \end{cases} \tag{1.35}$$

In rotation terminology the angles (θ, φ, ψ) have names:

- θ altitude
- φ: latitude (or co-latitude)
- ψ: longitude

Polar representation provides the definition of the quaternionic phases. Equation (1.33) can be rewritten as

$$q = \|q\|\, e^{\mathbf{i}\theta} e^{\mathbf{j}\varphi} e^{\mathbf{k}\psi} \tag{1.36}$$

where (θ, φ, ψ) are the phase angles to extract. As reported in [110]:

$$\begin{cases} \theta = \text{atan2}\,(n_\theta, d_\theta) \\ \varphi = \text{atan2}\,(n_\varphi, d_\varphi) \\ \psi = \text{asin}\,(n_\psi) \end{cases} \tag{1.37}$$

where

$$\begin{cases} n_\theta = 2\,(c \cdot d + a \cdot b) \\ d_\theta = a^2 - b^2 + c^2 - d^2 \end{cases}$$
$$\begin{cases} n_\varphi = 2\,(b \cdot d + a \cdot c) \\ d_\varphi = a^2 + b^2 - c^2 - d^2 \end{cases} \tag{1.38}$$
$$\begin{cases} n_\psi = 2\,(b \cdot c + a \cdot d) \\ d_\psi = 1 \end{cases}$$

Let us consider the 1D and 2D scenarios ("easy" to compute):

1D $\varphi = \psi = 0$:

$$\begin{cases} a = \|q\| \cos\theta \\ b = \|q\| \sin\theta \\ c = 0 \\ d = 0 \end{cases} \tag{1.39}$$

Phase θ can be found by inverting the first Equation of system (1.39):

$$\theta = \operatorname{acos}\left(\frac{a}{\|q\|}\right) = \operatorname{acos}\left(\frac{q+q^*}{2\|q\|}\right) \tag{1.40}$$

1D $\psi = 0$:

$$\begin{cases} a = \|q\| \cos\theta \\ b = \|q\| \sin\theta \cos\varphi \\ c = \|q\| \sin\theta \sin\varphi \\ d = 0 \end{cases} \tag{1.41}$$

Phases φ and ψ can be found by solving of system (1.41):

$$\begin{cases} \varphi = \operatorname{atan}\left(\frac{c}{b}\right) \\ \theta = \operatorname{acos}\left(\frac{a}{\|q\|}\right) = \operatorname{acos}\left(\frac{q+q^*}{2\|q\|}\right) \end{cases} \tag{1.42}$$

1.3.4 Quaternion Eigenvalues

Due to the non-commutativity of quaternion product, a quaternionic matrix may have two types of eigenvalues: *left* and *right* [112].

Right Eigenvalues: given matrix $\mathbf{A} \in \mathbb{H}^{n\times n}$, $\lambda \in \mathbb{H}$ is called *right eigenvalue* of $\mathbf{A}$ if $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda$ for some nonzero $\mathbf{x} \in \mathbb{H}^n$. The set of distinct right eigenvalues is called *right spectrum* of $\mathbf{A}$, denoted $\sigma_R(\mathbf{A})$.

Left Eigenvalues: given matrix $\mathbf{A} \in \mathbb{H}^{n\times n}$, $\lambda \in \mathbb{H}$ is called *left eigenvalue* of $\mathbf{A}$ if $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ for some nonzero $\mathbf{x} \in \mathbb{H}^n$. The set of distinct left eigenvalues is called *left spectrum* of $\mathbf{A}$, denoted $\sigma_L(\mathbf{A})$.

Left eigenvalues are also called singular eigenvalues. Accordingly, the characteristic polynomial $\mathbf{A} - \lambda\mathbf{I}$ is singular if and only if $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ for some nonzero $\mathbf{x} \in \mathbb{H}^n$. On the other hand, right eigenvalues are also named standard eigenvalues, since any $n \times n$ quaternion matrix $\mathbf{A}$ has exactly n right eigenvalues [11]. In [114], Zhang asserts that these eigenvalues are complex numbers with nonnegative imaginary parts. Lee states in [53], that, given two $n \times n$ complex matrices $\mathbf{A}$ and $\mathbf{B}$, then every real eigenvalue (if any) of the complex adjoint matrix appears an even number of times, and the complex eigenvalues of that matrix appear in conjugate pairs.

Right eigenvalues are well-studied in literature, seemingly the right spectrum is always non-empty, so there is no problem concerning this. Unfortunately, the

algorithm for their computation is still in dispute. Concerning the left spectrum, we are interested in the existence of a nonzero $\mathbf{x} \in \mathbb{H}^n$ for which $\varphi(\mathbf{x}) = \lambda\mathbf{x}$, but the writs convey that there seems to be no obvious interpretation of this in terms of the endomorphism $\varphi(.)$ [7, 32, 42, 112, 114].

Right eigenvalues

The right spectrum is unitarily invariant:

$$\sigma_R\left(\mathbf{U}^H\mathbf{A}\mathbf{U}\right) = \sigma_R(\mathbf{A}) \tag{1.43}$$

where $\mathbf{U}$ is a unitary matrix, that is $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$.

Let $\mathbf{A} \in \mathbb{H}^{n\times n}$ and $\mathbf{x} \in \mathbb{H}^n$, we can decompose $\mathbf{A}$ and $\mathbf{x}$ into a sum of complex matrices and vectors, respectively:

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_1 + \mathbf{A}_2\mathbf{j} \\ \mathbf{x} &= \mathbf{x}_1 + \mathbf{x}_2\mathbf{j} \end{aligned} \tag{1.44}$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{C}^{n\times n}$ and $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{C}^n$. The characteristic equation for right eigenvalues $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda$ becomes:

$$\mathbf{A}^a\mathbf{x}^a = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ -\bar{\mathbf{A}}_2 & \bar{\mathbf{A}}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ -\bar{\mathbf{x}}_2 \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x}_1 \\ -\bar{\mathbf{x}}_2 \end{bmatrix}. \tag{1.45}$$

Since matrix $\mathbf{A}^a$ is a $2n \times 2n$ complex matrix, it has exactly $2n$ complex eigenvalues which appear in complex conjugate pairs. The final step is to choose those eigenvalues having non-negative imaginary part [114].

Left eigenvalues

The following two lemmas are given:

Lemma 1. For $p, q \in \mathbb{H}$

$$\sigma_L(p\mathbf{I} + q\mathbf{A}) = \{p + qt : t \in \sigma_L(\mathbf{A})\} \tag{1.46}$$

where $\mathbf{I}$ is the identity matrix.

Lemma 2. If $\mathbf{A}$ is a (lower or upper) triangular matrix, then the left spectrum $\sigma_L(\mathbf{A})$ is the set of the distinct diagonal elements of $\mathbf{A}$.

For example, given the matrix $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then:

1. if $bc = 0$, then $\mathbf{A}$ is a triangular matrix and $\sigma_L(\mathbf{A}) = \{a, d\}$, according to Lemma 2;
2. if $bc \neq 0$, using Lemma 1:

$$\sigma_L(\mathbf{A}) = a + b\sigma_L\left(\begin{bmatrix} 0 & 1 \\ b^{-1}c & b^{-1}(d-a) \end{bmatrix}\right). \tag{1.47}$$

The quantity λ is a left eigenvalue if and only if there exist a nonzero $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$ such that:

$$\begin{bmatrix} 0 & 1 \\ b^{-1}c & b^{-1}(d-a) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{1.48}$$

In other terms:

$$\begin{cases} \lambda x_1 = x_2 \\ \lambda x_2 = b^{-1}cx_1 + b^{-1}(d-a)\, x_2 \end{cases} \Rightarrow \lambda^2 - b^{-1}(d-a)\,\lambda - b^{-1}c = 0 \tag{1.49}$$

The eigenvalues in (1.47) are generally quaternion-valued. The difficulty here is that (1.49) is a quaternionic equation. Solving a 2nd order quaternionic equation is not a problem, as explained in [42]. On the contrary, higher order equations require advanced resolution methods.

Huang & So Theorem (2001)

Huang & So Theorem follows from Lemma 1 and equations (1.47)-(1.49):

Given the matrix $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, if $bc \neq 0$, then ℓ is a left eigenvalue of $\mathbf{A}$ if and only if

$$\ell = a + b\lambda \tag{1.50}$$

where λ satisfies

$$b\lambda^2 - (d-a)\,\lambda - c = 0 \tag{1.51}$$

Solutions of quaternionic second order equations

Given a quadratic equation $x^2 + bx + c = 0$, its solutions are obtainable in four different cases:

1. If $b, c \in \mathbb{R}$ and $b^2 < 4c$, then

$$x = \frac{1}{2}\left(-b + \beta\mathbf{i} + \gamma\mathbf{j} + \delta\mathbf{k}\right), \quad \forall \beta, \gamma, \delta \in \mathbb{R}$$

   with $\beta^2 + \gamma^2 + \delta^2 = 4c - b^2$.

2. If $b, c \in \mathbb{R}$ and $b^2 \geqslant 4c$, then

$$x = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

   .

3. If $b \in \mathbb{R}$ and $c \notin \mathbb{R}$, then

$$x = \frac{-b}{2} \pm \frac{\rho}{2} \mp \frac{c_1}{\rho}\mathbf{i} \mp \frac{c_2}{\rho}\mathbf{j} \mp \frac{c_3}{\rho}\mathbf{k}$$

   where $c = c_0 + c_1\mathbf{i} + c_2\mathbf{j} + c_3\mathbf{k}$, $c_i \in \mathbb{R}$ and $\rho = \sqrt{\frac{b^2 - 4c_0 + \sqrt{(b^2-4c_0)^2 + 16\left(c_1^2 + c_2^2 + c_3^2\right)}}{2}}$.

4. If $b \notin \mathbb{R}$, then

$$x = \frac{-\operatorname{Re}[b]}{2} - (b' + T)^{-1}(c' - N)$$

where $b' = b - \operatorname{Re}[b] = \operatorname{Im}[b]$, $c' = c - \frac{\operatorname{Re}[b]}{2}\left(b - \frac{\operatorname{Re}[b]}{2}\right)$ and (T, N) are chosen as

(a) $T = 0,\ N = \left(B \pm \sqrt{B^2 - 4E}\right)\Big/2$, provided that $D = 0,\ B^2 \geqslant 4E$.

(b) $T = \pm\sqrt{2\sqrt{E} - B},\ N = \sqrt{E}$, provided that $D = 0,\ B^2 < 4E$.

(c) $T = \pm\sqrt{z},\ N = (T^3 + BT + D)/2T$, provided that $D \neq 0$ and z is the only positive root of the real polynomial $z^3 + 2Bz^2 + (B^2 - 4E)\,z - D^2$, where $B = |b'|^2 + 2\operatorname{Re}[c']$,$E = |c'|^2$ and $D = 2\operatorname{Re}[b'c']$.

It was important to dedicate a sub-paragraph to quaternionic second order equations in order to give an idea of the entity of the problem.

1.3.5 Eigenvalues of $n \times n$ matrices – Distribution of left and right eigenvalues

Whether one does the utmost to get the left spectrum, taking into account that left and right spectra may be different, the handy proposal for convergence analysis is to try how the algorithms behave considering the right spectrum only. In fact, in $\mathbb{H}$, it may happen that:

- No left eigenvalue of $\mathbf{A}$ is a right eigenvalue.
- There exist finite left eigenspectrum and infinite right eigenspectrum.
- Similar matrices have different left eigenvalues.
- Matrix $\mathbf{A}$ and its transpose $\mathbf{A}^T$ have different left eigenvalues.

A significant theorem in [42] states that, given $\mathbf{A} \in \mathbb{H}^{n\times n}$, if both $\sigma_R(\mathbf{A})$ and $\sigma_L(\mathbf{A})$ are finite, then $\sigma_R(\mathbf{A}) = \sigma_L(\mathbf{A})$.

Zhang Theorem (2007)

Let $\mathbf{A} = (a_{ij}) \in \mathbb{H}^{n\times n}$ and $\lambda \in \mathbb{H}$ be a left or right eigenvalue of $\mathbf{A}$, then

$$|\lambda| \leqslant \max_i \sum_{j=1}^{n} |a_{ij}| \triangleq R \tag{1.52}$$

where R denotes the maximum radius and

$$\rho_L(\mathbf{A}), \rho_R(\mathbf{A}) \leqslant \max_{\|x\|=1} \|\mathbf{A}x\| \tag{1.53}$$

are respectively the radius of the left and right eigenspectrum.

Geršgorin Theorem

In the complex field $\mathbb{C}$ Geršgorin theorem ensures that all eigenvalues of a matrix $\mathbf{A}$ are enclosed in the so-called Geršgorin discs.

Let $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{n\times n}$, the radius of the i-th Geršgorin disc is defined by

$$R_i(\mathbf{A}) = \sum_{i=1, j\neq i}^{n} |a_{ij}|, \ 1 \leqslant i \leqslant n \tag{1.54}$$

A Geršgorin disc of matrix $\mathbf{A}$ is a set in the complex plane defined as

$$\{z \in \mathbb{C} : |z - a_{ii}| \leqslant R_i(\mathbf{A})\} \tag{1.55}$$

with radius $R_i(\mathbf{A})$ and center a_{ii}.

In $\mathbb{H}$, the disc becomes a *ball*:

$$\{q \in \mathbb{H} : |q - a_{ii}| \leqslant R_i(\mathbf{A})\} \tag{1.56}$$

Unfortunately, Geršgorin theorem is suitable for *left* eigenvalues only.

Geršgorin-Zhang Theorem for right eigenvalues

Let $\mathbf{A} = (a_{ij}) \in \mathbb{H}^{n\times n}$, for each right eigenvalue λ of $\mathbf{A}$ there exist a quaternion α such that $\alpha^{-1}\lambda\alpha$ (which is also a right eigenvalue) is enclosed in the union of the Geršgorin balls:

$$\{q \in \mathbb{H} : |q - a_{ii}| \leqslant R_i(\mathbf{A})\}. \tag{1.57}$$

For example,

$$\left\{z^{-1}\lambda z : 0 \neq z \in \mathbb{H}\right\} \cap \bigcup_{i=1}^{n} \{q \in \mathbb{H} : |q - a_{ii}| \leqslant R_i(\mathbf{A})\} \tag{1.58}$$

(when λ is real, it is contained in a Geršgorin ball).

In other words, notice that the existence of an eigenvector $\mathbf{x}$ related to eigenvalue λ is equivalent to the existence of a fixed point of matrix $\mathbf{A}$ acting on a quaternion projection space denoted $\mathbb{H}\mathrm{P}^{n-1}$, defined as $\mathbb{H}\mathrm{P}^{n-1} = \mathbb{H}_0^n/\mathbb{H}^\times$, where $\mathbb{H}^\times = \{\mathbf{x} \in \mathbb{H}^n : \mathbf{x} \neq 0\}$.
If $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda$, then $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda \Rightarrow \mathbf{A}\mathbf{x}\alpha = \mathbf{x}\alpha\left(\alpha^{-1}\lambda\alpha\right)$. More details are described in [7].

1.3.6 Determinant of a quaternion matrix

Let $\mathbf{A} \in \mathbb{H}^{n\times n}$, the determinant of the adjoint matrix of $\mathbf{A}$ is equivalent to

$$|\chi_{\mathbf{A}}| = |\mathbf{A}|_q \tag{1.59}$$

where $|\mathbf{A}|_q$ is named *q-determinant* of $\mathbf{A}$. When $\mathbf{A}$ is a complex matrix, Zhang asserts in [114] that it is immediate that

$$|\mathbf{A}|_q = |\mathbf{A}|\,|\mathbf{A}^*| = |\det \mathbf{A}|^2 \tag{1.60}$$

where $\mathbf{A}^*$ denotes the conjugate matrix of $\mathbf{A}$.

Let $\mathbf{A} \in \mathbb{H}^{n\times n}$, then these useful properties hold and are proved in [114]:

- If matrices $\mathbf{A}$ and $\mathbf{B}$ are similar, then $|\mathbf{A}|_q = |\mathbf{B}|_q$.
- $|\mathbf{AB}|_q = |\mathbf{A}|_q |\mathbf{B}|_q \Rightarrow \left|\mathbf{A}^{-1}\right|_q = |\mathbf{A}|_q^{-1}$ if $\mathbf{A}^{-1}$ exists.
- $|\mathbf{A}|_q = \prod_{i=1}^{n} |\lambda_i|^2 \geqslant 0$, where λ_i is the i-th standard eigenvalue of $\mathbf{A}$.
- $|\mathbf{PAQ}|_q = |\mathbf{A}|_q$ for any elementary matrices $\mathbf{P}$ and $\mathbf{Q}$.

1.3.7 Inverse of a quaternion matrix

Let $\mathbf{A} \in \mathbb{H}^{n\times n}$, then the following statements are equivalent [114]:

- $\mathbf{A}$ is invertible.
- $\mathbf{Ax} = \mathbf{0}$ has a unique solution $\mathbf{x} = \mathbf{0}$.
- $\det \chi_{\mathbf{A}} \triangleq |\chi_{\mathbf{A}}| \neq 0$, i.e. $\chi_{\mathbf{A}}$ is invertible.

The procedure presented here is a practical solution to find the inverse of a quaternion matrix $\mathbf{A}$. We call $\mathbf{B}$ the inverse matrix of $\mathbf{A}$, that is $\mathbf{AB} = \mathbf{I}$. The first step is to decompose both $\mathbf{B}$ and $\mathbf{A}$ into Cayley-Dickson form: $\begin{cases}\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2\mathbf{j} \\ \mathbf{B} = \mathbf{B}_1 + \mathbf{B}_2\mathbf{j}\end{cases}$. The expression $\mathbf{AB} = \mathbf{I}$ is expanded through the following steps [114]:

$$\begin{aligned}
&\Rightarrow \left(\mathbf{A}_1\mathbf{B}_1 - \mathbf{A}_2\bar{\mathbf{B}}_2\right) + \left(\mathbf{A}_1\mathbf{B}_2 - \mathbf{A}_2\bar{\mathbf{B}}_1\right)\mathbf{j} = \mathbf{I} \\
&\Rightarrow \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ -\bar{\mathbf{B}}_2 & \bar{\mathbf{B}}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \\
&\Rightarrow \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ -\bar{\mathbf{B}}_2 & \bar{\mathbf{B}}_1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ -\bar{\mathbf{A}}_2 & \bar{\mathbf{A}}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}
\end{aligned} \tag{1.61}$$

The results in (1.61) are equivalent to the system of matrix equations below:

$$\begin{cases}\mathbf{B}_1\mathbf{A}_1 - \mathbf{B}_2\bar{\mathbf{A}}_2 = \mathbf{I} \\ \mathbf{B}_1\mathbf{A}_2 + \mathbf{B}_2\bar{\mathbf{A}}_1 = \mathbf{0}\end{cases} \tag{1.62}$$

which, in turn, is equivalent to

$$\left(\mathbf{B}_1\mathbf{A}_1 - \mathbf{B}_2\bar{\mathbf{A}}_2\right) + \left(\mathbf{B}_1\mathbf{A}_2 + \mathbf{B}_2\bar{\mathbf{A}}_1\right)\mathbf{j} = \mathbf{I} \Rightarrow \mathbf{BA} = \mathbf{I} \tag{1.63}$$

Post-multiplying the second equation of (1.63) by $\mathbf{A}_2^{-1}$, we obtain

$$\mathbf{B}_1\mathbf{A}_2\mathbf{A}_2^{-1} + \mathbf{B}_2\bar{\mathbf{A}}_1\mathbf{A}_2^{-1} = \mathbf{0} \;\Rightarrow\; \mathbf{B}_1 = -\mathbf{B}_2\bar{\mathbf{A}}_1\mathbf{A}_2^{-1} \tag{1.64}$$

Substituting (1.64) into the first equation of (1.63):

$$\begin{aligned}
-\mathbf{B}_2\bar{\mathbf{A}}_1\mathbf{A}_2^{-1}\mathbf{A}_1 - \mathbf{B}_2\bar{\mathbf{A}}_2 = \mathbf{I} &\Rightarrow \mathbf{B}_2\left(-\bar{\mathbf{A}}_1\mathbf{A}_2^{-1}\mathbf{A}_1 - \bar{\mathbf{A}}_2\right) = \mathbf{I} \\
&\Rightarrow \mathbf{B}_2 = \left(-\bar{\mathbf{A}}_1\mathbf{A}_2^{-1}\mathbf{A}_1 - \bar{\mathbf{A}}_2\right)^{-1}
\end{aligned} \tag{1.65}$$

Definitively, we have to find

$$\begin{cases} \mathbf{B}_1 = -\left(-\bar{\mathbf{A}}_1\mathbf{A}_2^{-1}\mathbf{A}_1 - \bar{\mathbf{A}}_2\right)^{-1}\bar{\mathbf{A}}_1\mathbf{A}_2^{-1} \\ \mathbf{B}_2 = \left(-\bar{\mathbf{A}}_1\mathbf{A}_2^{-1}\mathbf{A}_1 - \bar{\mathbf{A}}_2\right)^{-1} \end{cases} \tag{1.66}$$

and recompose the inverse quaternion matrix:

$$\mathbf{B} = \mathbf{B}_1 + \mathbf{B}_2\mathbf{j} = \mathbf{B}_a + \mathbf{B}_b\mathbf{i} + \mathbf{B}_c\mathbf{j} + \mathbf{B}_d\mathbf{k}. \tag{1.67}$$

1.3.8 Norm of a quaternion matrix

A practical way to define the norm of a quaternion matrix $\mathbf{A}$ is

$$\|\mathbf{A}\|_F = \sqrt{Tr\left(\mathbf{A}\mathbf{A}^H\right)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{1.68}$$

where subscript F denotes *Frobenius* norm.

The trace of a square quaternion matrix $\mathbf{K} \in \mathbb{H}^{n\times n}$ is defined the same way as for matrices in $\mathbb{R}^{n\times n}$ or $\mathbb{C}^{n\times n}$ [41]:

$$Tr\left(\mathbf{K}\right) = \sum_{i=1}^{n} a_{ii} \tag{1.69}$$

that is, the sum of the elements of the principal diagonal of $\mathbf{K}$.

The Frobenius norm is a particular case of p-norms (where $p = 2$). In general, the p-norm is defined as

$$\|\mathbf{A}\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^p\right)^{\frac{1}{p}} \tag{1.70}$$

An alternative way to compute the Frobenius norm in $\mathbb{R}$ and $\mathbb{C}$ uses the spectral radius ρ of $\mathbf{A}\mathbf{A}^H$:

$$\|\mathbf{A}\|_2 = \sqrt{\rho\left(\mathbf{A}\mathbf{A}^H\right)} \tag{1.71}$$

The spectral radius is the largest eigenvalue of $\mathbf{A}\mathbf{A}^H$ in absolute value. However, the extension of norm (1.71) to quaternions is still under debate.

1.3.9 Quaternion unitary matrices

Unitary matrices are useful to compute the eigenvalues of a given matrix easily. In fact, it is possible to convert a matrix $\mathbf{A} \in (\mathbb{R}, \mathbb{C}, \mathbb{H})^{n\times n}$ into a diagonal matrix $\mathbf{A}_D \in (\mathbb{R}, \mathbb{C})^{n\times n}$ having the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ (recall that the eigenvalues of a quaternion matrix are complex numbers). This is achievable by means of a unitary transformation. We can take advantage of Cayley-Dickson algebra once again and express a quaternion unitary matrix $\mathbf{P}$ in the form

$$\mathbf{P} = \mathbf{U}\mathbf{D}\mathbf{W} \tag{1.72}$$

where $\mathbf{U}$ and $\mathbf{W}$ are two *complex* unitary matrices and can be found in a conventional way. As expected, matrix $\mathbf{D}$ is a quaternion-valued diagonal matrix. A full

explanation of this and further information about unitary and orthogonal quaternion matrices were presented in [109].

Example

We aim at finding a matrix $\mathbf{U} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \mathbb{C}^{2\times 2}$ such that $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$ and $|\det(\mathbf{U})| = 1$, that is

$$\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I} \Rightarrow \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1.73}$$

In order to find the elements of $\mathbf{U}$, we solve the following system in the unknowns a, b, c, d:

$$\begin{cases} aa^* + bb^* = 1 \\ ac^* + bd^* = 0 \\ ca^* + db^* = 0 \\ cc^* + dd^* = 1 \end{cases} \tag{1.74}$$

thus obtaining the matrix

$$\mathbf{U} = \frac{1}{b^*c^* - d^*a^*} \begin{bmatrix} -d^* & c^* \\ b^* & -a^* \end{bmatrix} = \frac{1}{\det(\mathbf{U})} \begin{bmatrix} -d^* & c^* \\ b^* & -a^* \end{bmatrix}. \tag{1.75}$$

So, we decide for $|\det(\mathbf{U})| = |ad - cb| = 1$.

This is true in two cases: $\begin{cases} ad - cb = 1 \\ ad - cb = -1 \end{cases}$. For example, $ad - cb = 1$ and we have:

$$\begin{cases} a = -d^* \\ b = c^* \\ c = b^* \\ d = -a^* \end{cases} \tag{1.76}$$

(the third and fourth equations in (1.76) can be erased since they are equivalent to the first and the second respectively). Finally, the system to solve becomes

$$\begin{cases} a = -d^* \\ b = c^* \\ ad - cb = 1 \end{cases} \tag{1.77}$$

We have obtained 3 equations and 4 unknowns: this means that one parameter is free. Substituting the first and the second equation into the third in (1.77):

$$-aa^* + bb^* = 1 \Rightarrow bb^* = 1 + aa^* \tag{1.78}$$

We need explicating b and its conjugate b^* in its real and imaginary parts:

$$\begin{cases} b = \mathrm{Re}(b) + j\,\mathrm{Im}(b) \\ b^* = \mathrm{Re}(b) - j\,\mathrm{Im}(b) \end{cases} \tag{1.79}$$

So, finally

$$\begin{cases} bb^* = \mathrm{Re}^2(b) + \mathrm{Im}^2(b) \\ aa^* = \mathrm{Re}^2(a) + \mathrm{Im}^2(a) \end{cases} \tag{1.80}$$

Substituting (1.80) into (1.78):

$$\mathrm{Re}^2(b) + \mathrm{Im}^2(b) = 1 + \mathrm{Re}^2(a) + \mathrm{Im}^2(a) \tag{1.81}$$

Equation (1.81) can be split into two equations:

$$\begin{cases} \mathrm{Re}^2(b) = 1 + \mathrm{Re}^2(a) \\ \mathrm{Im}^2(b) = \mathrm{Im}^2(a) \end{cases}. \tag{1.82}$$

We finally choose the value of b or a and solve for all the elements of $\mathbf{U}$.

Chapter 2

Hypercomplex Signal Processing

Contents

Most operations in digital signal processing are based on filtering. The reasons for transforming a signal may be merely aesthetic (e.g. image and sound coloring, equalization, correction) or functional (e.g. spectral analysis, noise reduction). Convolution is the basic operation in linear filtering and it is known that convolution in the frequency domain allows a faster execution due to a reduced computational cost. In hypercomplex algebras, because of the high-dimensional nature of the signals, time-domain convolution would be excessively burdensome. With this aim, we focused on the development of hypercomplex (quaternion) adaptive filters in the frequency domain. This short chapter is dedicated to the principal operation blocks in quaternion digital signal processing. Firstly, the quaternion discrete Fourier transform (QDFT) is presented in all its forms. Thanks to QDFT, it is possible to take advantage of the convolution and crosscorrelation theorems in the quaternion domain. It will be shown that generally, the *classic* theorems known in real and complex algebras are no longer valid in quaternion signal processing. The operations defined in this chapter will be employed in the algorithms presented later in the next chapters.

2.1 Quaternion-valued transforms

2.1.1 Quaternion-valued Discrete Fourier Transform

Transform domain algorithms require mathematical transformations in order to (pre-)process input and output signals. Such transformations, e.g. DFT and FFT, are quite uncommon in a quaternionic format. General information about quaternion-valued transformations can be found in [21, 22, 38]. Using hypercomplex algebra to build a quaternion DFT/FFT function from scratch is quite an arduous endeavour. Fortunately, there is a quick and easy method that exploits the conventional DFT/FFT functions available in several programming environments by decomposing the quaternion-valued FFT (QFFT) into complex FFTs. This method was formerly developed in image processing and presented in [21, 23, 74, 82, 83]. Originally, the need for the development of a QFFT arose from the idea of collecting colour (RGB) or luminance/chrominance signals in one quaternion, rather than treating them as independent vectors [49, 82, 83, 106], thus permitting the generalization of several techniques in image processing depending on the Fourier transform of the color.

The non-commutativity property of the quaternionic product gives rise to a two-sided mono-dimensional 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 2.1.

Table 2.1. Kernel definitions for monodimensional QDFT

	Left	**Right**
Axis $\boldsymbol{\nu}$	$e^{-\boldsymbol{\nu}\omega n} f(\cdot)$	$f(\cdot)\, e^{-\boldsymbol{\nu}\omega n}$

Equations (2.1a) and (2.1b) represent the quaternionic Fourier monodimensional left-handed transform (the exponential function is on the left) and its inverse [22]:

$$F(u) = \sum_{n=0}^{N-1} \exp\left(-2\pi\boldsymbol{\nu}\frac{nu}{N}\right) f(n) \tag{2.1a}$$

$$f(n) = \frac{1}{N}\sum_{u=0}^{N-1} \exp\left(2\pi\boldsymbol{\nu}\frac{nu}{N}\right) F(u) \tag{2.1b}$$

where $F(u)$ is the spectrum of $f(n)$. 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 $\boldsymbol{\nu}$ is an arbitrarily chosen pure unitary quaternion versor and can be expressed as

$$\boldsymbol{\nu} = \frac{x\mathbf{i}}{\sqrt{x^2+y^2+z^2}} + \frac{y\mathbf{j}}{\sqrt{x^2+y^2+z^2}} + \frac{z\mathbf{k}}{\sqrt{x^2+y^2+z^2}}. \tag{2.2}$$

Fourier transform analyzes a signal according to sinusoidal components and de Moivre's formula ($e^{\mathbf{i}\theta} = \cos\theta + \mathbf{i}\sin\theta$) generalizes all kinds of algebra where roots of -1 are definable. For instance, in the case of the complex quaternion Fourier transform, a complex quaternion root of -1 ($\boldsymbol{\nu}^2 = -1$) is required:

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = |q|\, e^{\boldsymbol{\nu}\theta} = |q|\,(\cos\theta + \boldsymbol{\nu}\sin\theta)\,. \tag{2.3}$$

Versor $\boldsymbol{\nu}$ defined in (2.2) describes the spatial direction of the imaginary part of quaternion q. The angle named with letter θ denotes the rotation angle about the axis $\boldsymbol{\nu}$. As one can see, if $\boldsymbol{\nu} = \mathbf{i}$ and the input function is complex, equations (2.1a) and (2.1b) reduce to the conventional complex Fourier transform.

Quaternions can be thought of as complex numbers whose real and imaginary parts are complex numbers in turn (Cayley-Dickson form):

$$q = a + b\boldsymbol{\nu_2} \tag{2.4}$$

where $a = w_1 + x_1\boldsymbol{\nu_1}$ and $b = y_1 + z_1\boldsymbol{\nu_1}$. In this manner a quaternion q can be decomposed into

$$\begin{aligned} q &= \underbrace{(w_1 + x_1\boldsymbol{\nu_1})}_{simplex} + \underbrace{(y_1 + z_1\boldsymbol{\nu_1})}_{perplex}\boldsymbol{\nu_2} \\ &= w_1 + x_1\boldsymbol{\nu_1} + y_1\boldsymbol{\nu_2} + z_1\boldsymbol{\nu_3}. \end{aligned} \tag{2.5}$$

The highlighted parts in equation (2.5) are named respectively *simplex* and *perplex* parts of quaternion q, both isomorphic to the conventional field $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, since they are both defined in the same space of the complex operator $\boldsymbol{\nu_1}$. In addition, we denote with $\mathcal{S}\{\cdot\}$ and $\mathcal{P}\{\cdot\}$ the operators that extract the simplex and perplex parts of a quaternion, respectively. Using these operators, it is possible to write $q = \mathcal{S}\{q\} + \mathcal{P}\{q\}\boldsymbol{\nu_2}$.

Hence, each quaternion function $f(n)$ can be formulated in terms of an orthonormal basis. The versors $(\boldsymbol{\nu_2}, \boldsymbol{\nu_3})$ must be chosen in a way that $\boldsymbol{\nu_1} \perp \boldsymbol{\nu_2} \perp \boldsymbol{\nu_3}$, $\boldsymbol{\nu_1}\boldsymbol{\nu_2} = \boldsymbol{\nu_3}$ and $\boldsymbol{\nu_1}\boldsymbol{\nu_2}\boldsymbol{\nu_3} = -1$. The advantage of using the basis $(\boldsymbol{\nu_1}, \boldsymbol{\nu_2}, \boldsymbol{\nu_3})$ is that such a system of operators, isomorphic to system $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, is not bound to the axes $(\mathbf{i}, \mathbf{j}, \mathbf{k})$.

The orthonormal basis can be represented in a compact matrix form (matrix of change of basis):

$$\mathbf{B} = \begin{pmatrix} \nu_{1x} & \nu_{1y} & \nu_{1z} \\ \nu_{2x} & \nu_{2y} & \nu_{2z} \\ \nu_{3x} & \nu_{3y} & \nu_{3z} \end{pmatrix} \tag{2.6}$$

where

$$\begin{cases} \boldsymbol{\nu_1} = \nu_{1x}\mathbf{i} + \nu_{1y}\mathbf{j} + \nu_{1z}\mathbf{k} \\ \boldsymbol{\nu_2} = \nu_{2x}\mathbf{i} + \nu_{2y}\mathbf{j} + \nu_{2z}\mathbf{k} \\ \boldsymbol{\nu_3} = \nu_{3x}\mathbf{i} + \nu_{3y}\mathbf{j} + \nu_{3z}\mathbf{k} \end{cases} \tag{2.7}$$

Then it is possible to extract the component functions of $f(n)$ in the new basis:

$$\begin{aligned} f(n) &= w_1(n) + x_1(n)\boldsymbol{\nu_1} + y_1(n)\boldsymbol{\nu_2} + z_1(n)\boldsymbol{\nu_3} \\ &= [w_1(n) + x_1(n)\boldsymbol{\nu_1}] + [y_1(n) + z_1(n)\boldsymbol{\nu_1}]\boldsymbol{\nu_2}. \end{aligned} \tag{2.8}$$

Each component function is obtained by the dot products below:

$$\begin{cases} w_1(n) = w(n) \\ x_1(n) = \langle \boldsymbol{\nu_1}, x(n)\mathbf{i} + y(n)\mathbf{j} + z(n)\mathbf{k} \rangle \\ y_1(n) = \langle \boldsymbol{\nu_2}, x(n)\mathbf{i} + y(n)\mathbf{j} + z(n)\mathbf{k} \rangle \\ z_1(n) = \langle \boldsymbol{\nu_3}, x(n)\mathbf{i} + y(n)\mathbf{j} + z(n)\mathbf{k} \rangle \end{cases} \tag{2.9}$$

After changing the basis, it is possible to separate the quaternion Fourier transform into the sum of two transforms, thanks to the linearity property of the Fourier transform:

$$\begin{aligned} F(u) = &\sum_{n=0}^{N-1} \mathrm{e}^{\left(-2\pi\boldsymbol{\nu}\frac{nu}{N}\right)} \left(w_1(n) + x_1(n)\,\boldsymbol{\nu_1}\right) \\ &+ \sum_{n=0}^{N-1} e^{\left(-2\pi\boldsymbol{\nu}\frac{nu}{N}\right)} \left(y_1(n) + z_1(n)\,\boldsymbol{\nu_1}\right)\boldsymbol{\nu_2}. \end{aligned} \tag{2.10}$$

Once the quaternion transform is executed, the next step is to reassemble the transformed quaternion and change back to the original basis by means of inverse change of basis (it is sufficient to transpose each element of matrix $\mathbf{B}$).

We wonder what implications the existence of a two-sided transform has on filtering applications. Tests conducted during the development of the algorithms presented in this thesis revealed that, in order to avoid flawed algorithms, once a direction of rotation is chosen, this has to be kept unchanged.

2.1.2 QDFT is a unitary transformation

We will prove here that the QDFT si a unitary transformation. A transformation $\mathbf{F} \in \mathbb{H}^{N\times N}$ is unitary if the following relations are satisfied:

$$\mathbf{F}^{-1} = \mathbf{F}^{H} \Leftrightarrow \mathbf{F}\mathbf{F}^{H} = \mathbf{I}. \tag{2.11}$$

Considering $\boldsymbol{\nu_1} = \mathbf{i}$ in a Cayley-Dickson decomposition, the QDFT transformation can be rewritten in a matrix form as

$$\mathbf{F}_{QDFT} \triangleq K \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\mathbf{i}2\pi/N} & e^{-\mathbf{i}4\pi/N} & \cdots & e^{-\mathbf{i}2\pi(N-1)/N} \\ 1 & e^{-\mathbf{i}4\pi/N} & e^{-\mathbf{i}8\pi/N} & \cdots & e^{-\mathbf{i}4\pi(N-1)/N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\mathbf{i}2\pi(N-1)/N} & e^{-\mathbf{i}4\pi(N-1)/N} & \cdots & e^{-\mathbf{i}2\pi(N-1)^2/N} \end{bmatrix} \tag{2.12}$$

where N is the QDFT length and $K = \frac{1}{\sqrt{N}}$ is a term that makes the transformation unitary.
The Hermitian of $\mathbf{F}_{QDFT}$ (2.12) is defined as

$$\mathbf{F}^{H}_{QDFT} \triangleq K \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{\mathbf{i}2\pi/N} & e^{\mathbf{i}4\pi/N} & \cdots & e^{\mathbf{i}2\pi(N-1)/N} \\ 1 & e^{-\mathbf{i}4\pi/N} & e^{\mathbf{i}8\pi/N} & \cdots & e^{\mathbf{i}4\pi(N-1)/N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\mathbf{i}2\pi(N-1)/N} & e^{\mathbf{i}4\pi(N-1)/N} & \cdots & e^{\mathbf{i}2\pi(N-1)^2/N} \end{bmatrix} \tag{2.13}$$

and it is equal to the *inverse* discrete Fourier transform $\mathbf{F}^{-1}_{QDFT}$. We can verify by quaternion multiplication that $\mathbf{F}_{QDFT}\mathbf{F}^{H}_{QDFT} = \mathbf{I}$.

2.2 Quaternion convolution

In order to understand how quaternion-valued filtering works, it is helpful to define the quaternion-valued convolution. In a way similar to the complex case, the *quaternion-valued impulse response* is defined as

$$h[n] = h_A[n] + h_B[n]\mathbf{i} + h_C[n]\mathbf{j} + h_D[n]\mathbf{k} \tag{2.14}$$

and the convolution operation produces the output signal $y[n]$ as

$$\begin{aligned} y[n] &= x[n] * h[n] \\ &= [x_A[n] + x_B[n]\mathbf{i} + x_C[n]\mathbf{j} + x_D[n]\mathbf{k}] * [h_A[n] + h_B[n]\mathbf{i} + h_C[n]\mathbf{j} + h_D[n]\mathbf{k}] \\ &= [x_A[n] * h_A[n] - x_B[n] * h_B[n] - x_C[n] * h_C[n] - x_D[n] * h_D[n]] \\ &+ [x_A[n] * h_B[n] + x_B[n] * h_A[n] + x_C[n] * h_D[n] - x_D[n] * h_C[n]]\,\mathbf{i} \\ &+ [x_A[n] * h_C[n] - x_B[n] * h_D(t) + x_C[n] * h_A[n] + x_D(t) * h_B(t)]\,\mathbf{j} \\ &+ [x_A[n] * h_D(t) + x_B[n] * h_C(t) - x_C[n] * h_B[n] + x_D(t) * h_A[n]]\,\mathbf{k} \\ &= y_A[n] + y_B[n]\mathbf{i} + y_C[n]\mathbf{j} + y_D[n]\mathbf{k}. \end{aligned} \tag{2.15}$$

Equivalently, quaternion convolution can be expressed in terms of scalar product (dot product), i.e. we compute the convolution between two quaternion-valued sequences as

$$y[n] = \mathbf{h}^T\mathbf{x} \tag{2.16}$$

The operation can be summarized in a block diagram as in Fig. 2.1. Such a scheme suggests that quaternion convolution follows the same rules of quaternion multiplication and can be decomposed into the combination of 16 real-valued convolutions.

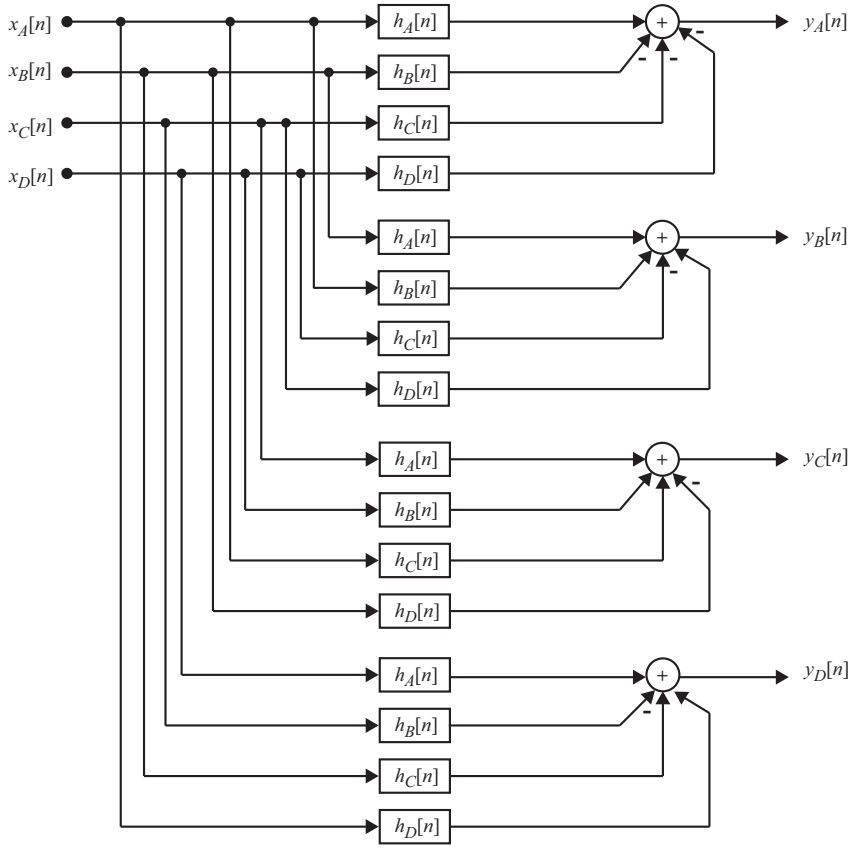


Figure 2.1. Quaternion convolution block diagram.

2.3 Quaternion convolution theorem

In the complex domain, given two functions $x[n], h[n] \in \mathbb{R}, \mathbb{C}$, we can compute their convolution efficiently by multiplying their frequency spectra $X(\omega), H(\omega) \in \mathbb{C}$: $y[n] = h[n] * x[n] \longleftrightarrow Y(\omega) = H(\omega)X(\omega)$ (*convolution theorem*).
In the quaternion domain, the convolution theorem has to be reformulated and the decomposition of a quaternion-valued function into its simplex and perplex parts has to be considered. Moreover, the existence of the left and the right Fourier transforms determines two possible formulations for both convolution $Y(\omega)$ and cross-correlation $C(\omega)$ [22, 74]. For the monodimensional case, we have:

Left transform

$$Y(\omega) = H^a(\omega)X(\omega) + H^b(\omega)\mathbf{j}X(-\omega) \tag{2.17}$$

$$C(\omega) = H^a(\omega)\left[X_a^H(\omega) - \mathbf{j}X_b(-\omega)\right] + H^b(\omega)\mathbf{j}\left[X_a^H(-\omega) - \mathbf{j}X_b(\omega)\right] \tag{2.18}$$

Right transform

$$Y(\omega) = H(\omega)X^a(\omega) + H(-\omega)\mathbf{j}X^d(\omega) \tag{2.19}$$

$$C(\omega) = H(\omega)X^{aH}(\omega) - H(-\omega)\mathbf{j}X^b(-\omega) \tag{2.20}$$

where, given $H(\omega) = H_1(\omega) + H_2(\omega)\mathbf{i} + H_3(\omega)\mathbf{j} + H_4(\omega)\mathbf{k}$, we define $H^a(\omega) = H_1(\omega) + H_2(\omega)\mathbf{i}$ and $H^b(\omega) = H_3(\omega) + H_4(\omega)\mathbf{i}$ the simplex and perplex parts of $H(\omega)$, respectively, and $H^d(\omega) = H_3(\omega) - \mathbf{i}H_4(\omega)$. A similar notation holds for $X(\omega)$. The symbol $H(-\omega)$ denotes the frequency reversal for the transform $H(\omega)$: the swapping can be practically achieved by simply reversing the order of the whole frequency array, excluding the zero-frequency coefficient. The swap is shown in Fig. 2.2. In the picture, *DC* and Nyquist frequency *Ny* do not move during the swapping.

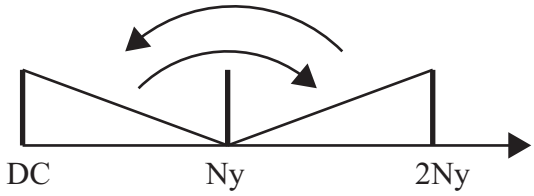


Figure 2.2. Frequency Swap.

Some simplifications can be applied in case the signal $x[n]$ has some special simmetric properties. If either $x[n]$ or $h[n]$ has even simmetry, i.e. $x[n] = x[-n]$, $h[n] = h[-n]$, then $X(\omega) = X(-\omega)$ or $H(\omega) = H(-\omega)$ and $Y(\omega) = H(\omega)X(\omega)$ with the left transform or the right transform, respectively. If $x[n]$ has odd simmetry, i.e. $x[n] = -x[-n]$, then $X(\omega) = -X(-\omega)$ and $Y(\omega) = [H_a(\omega) - \mathbf{j}H_b(\omega)]\, X(\omega)$ with the left transform and $Y(\omega) = H(\omega)\,[X_a(\omega) - \mathbf{j}X_b(\omega)]$ with the right transform.

A rule of thumb for checking whether the QDFT is well-implemented is to follow the scheme in Fig. 2.3. After computing the convolution in the frequency domain, the inverse transform of the output should be equal to the output of the time-domain convolution. Figure 2.4 shows how the classic convolution theorem adopted in a quaternion context produces an erroneous result, since it is not coicident with the result of quaternion time-domain convolution. On the other hand, the split convolution method proposed in (2.18) and in (2.20) does provide the expected result as shown in Fig. 2.5.

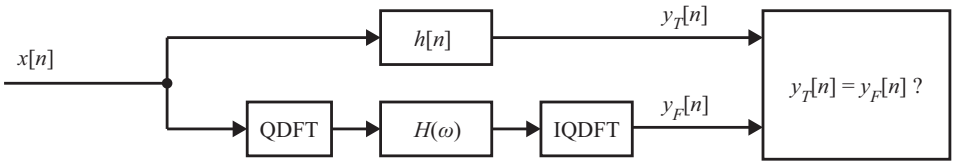


Figure 2.3. Time-domain and frequency-domain comparison scheme.

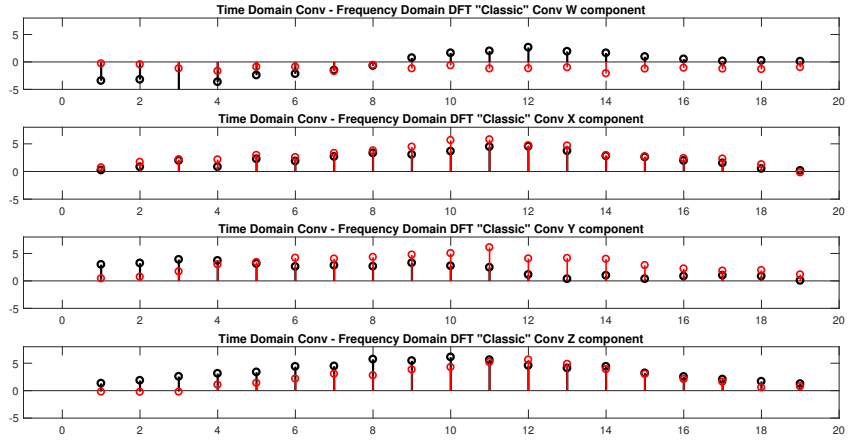


Figure 2.4. The classic convolution theorem adopted in a quaternion context produces an erroneous result, since it is not coicident with the result of quaternion time-domain convolution.

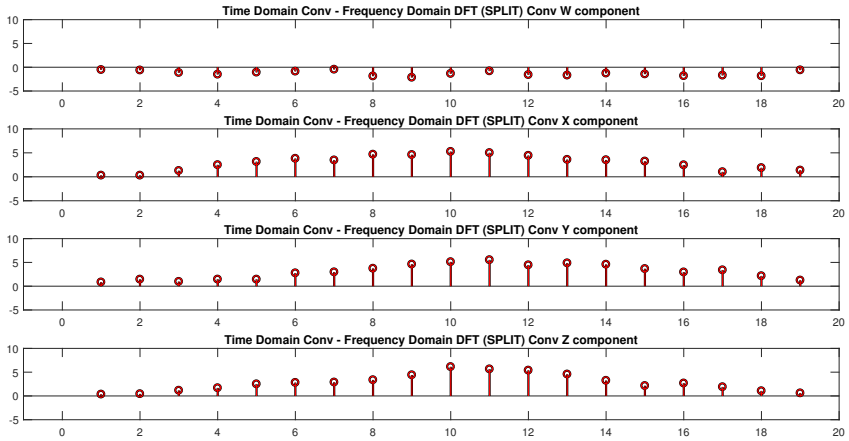


Figure 2.5. The split quaternion convolution theorem provides an output which is coicident with the result of quaternion time-domain convolution.

2.4 Relations between LEFT and RIGHT transforms

There exist relations between Left and Right quaternion transforms. We have defined the *symplectic* decomposition as

$$\begin{aligned} F^L(\omega) &= F_s(\omega) + F_p(\omega)\nu_2 \\ F^R(\omega) &= F_s(\omega) + F_p(-\omega)\nu_2 \end{aligned} \tag{2.21}$$

and we can find the relations

$$\begin{aligned} F^R_{\pm\nu}(\overline{\omega}) &= \overline{F^L_{\mp\nu}(\omega)} = -F^L_{\mp\nu}(\omega) \\ F^L_{\pm\nu}(\overline{\omega}) &= \overline{F^R_{\mp\nu}(\omega)} = -F^R_{\mp\nu}(\omega) \end{aligned}. \tag{2.22}$$

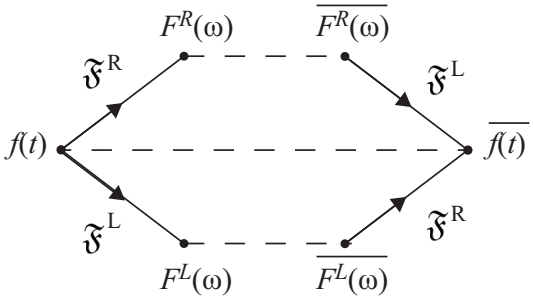


Figure 2.6. Relations between left and right transforms [22].

2.5 Time reversal in $\mathbb{H}$

Of course, frequency swap can be related to time reversal [22].

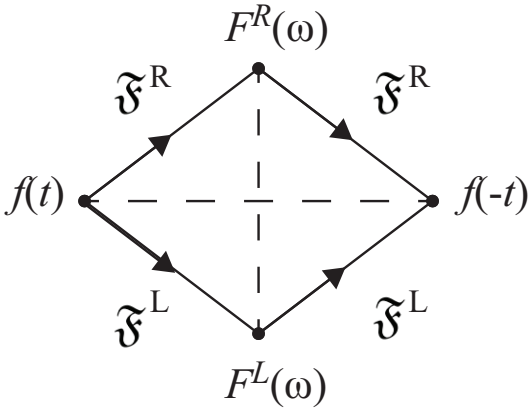


Figure 2.7. Quaternion Fourier transform and time reversal relations [22].

In the discrete time, the operation of time reversal is not simply a time reversal: it is a time reversal and a shift (Heaviside theorem/Time shifting theorem). The application of the same transform to a signal $f(t)$ twice results in a time-reversed signal $f(-t)$. Scale factors are ignored. In Fig. 2.7 the dashed connection between $F^L(\omega)$ and $F^R(\omega)$ represents a partial frequency reversal.

Chapter 3

Quaternion Adaptive Filters

Contents

The algorithms developed in this work have Widrow and Hoff's Least Mean Square (LMS) algorithm (1960) as a common root [108]. The LMS algorithm is definitely a milestone in adaptive filtering and founded the bases for the new generations of algorithms and advanced techniques in this field. The derivation of the Quaternion LMS (QLMS) traces the starting line for the development of a new family of algorithms. As introduced, ordinarily working with 3D audio, we have searched for algorithms compliant with the specifications of fast computation and proper algebraic structure, in order to deal with very long impulse responses and exploit the physical and statistical properties of the acoustic field. Nevertheless, the algorithms suggested in this Chapter and tested in audio signal processing can be transferred to other research areas, e.g. mechanics, electromagnetism, etc.

This chapter is divided into two main sections: Paragraph 3.1 recalls the classic QLMS algorithm by Mandic and Took [98] as a delegate of algorithms in the time

domain. A full derivation of the QLMS is also provided, since it supplies a method for the development of algorithms in quaternion algebra. Paragraph 3.2 presents our new class of quaternion frequency-domain algorithms, with particular regard to the Overlap-Save Quaternion Frequency Domain Adaptive Filter (OS-QFDAF). Special attention will be given to the convergence properties of this family of filters. Finally, reports from simulations are shown in Par. 3.3.

3.1 Time Domain Quaternion Adaptive Filters

QLMS was originally introduced by Mandic and Took in [98] and later refined by Barthélemy *et al.* in [8], where other derivations were also proposed. We will shortly explain why the QLMS is not a four-dimensional extension of the complex LMS algorithm. In fact, we will demonstrate that, if two of four components vanish, QLMS does not degenerate into CLMS.

The algorithm learning equation in its general formulation updates the filter weights as follows:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \frac{1}{2} \left[-\nabla \hat{J} \left(\mathbf{w}_{n-1} \right) \right]. \tag{3.1}$$

where $\nabla \hat{J} \left(\mathbf{w}_{n-1} \right)$ is the gradient of a cost function $\hat{J} \left(\mathbf{w}_{n-1} \right)$ to be defined and minimized and represents the *innovation* term in the update law (3.1). The QLMS filter is an error-correction based algorithm and the cost function can be chosen, for example, as the Mean Square Error (MSE):

$$\hat{J} \left(\mathbf{w}_{n-1} \right) = e[n] e^*[n] = e_a^2[n] + e_b^2[n] + e_c^2[n] + e_d^2[n] \tag{3.2}$$

where the error is computed as the difference between a desired signal $d[n]$ and the adaptive filter output $y[n]$: $e[n] = d[n] - y[n]$. The minima and maxima of $\hat{J} \left(\mathbf{w}_{n-1} \right)$ are found by computing the quaternionic gradient:

$$\begin{aligned} \nabla_{\mathbf{w}} \left(e[n] e^*[n] \right) &= \nabla_{\mathbf{w}_a} \left(e[n] e^*[n] \right) + \nabla_{\mathbf{w}_b} \left(e[n] e^*[n] \right) \mathbf{i} \\ &\quad + \nabla_{\mathbf{w}_c} \left(e[n] e^*[n] \right) \mathbf{j} + \nabla_{\mathbf{w}_d} \left(e[n] e^*[n] \right) \mathbf{k} \\ &= \frac{\partial \hat{J} \left(\mathbf{w} \right)}{\partial \mathbf{w}_a} + \frac{\partial \hat{J} \left(\mathbf{w} \right)}{\partial \mathbf{w}_b} \mathbf{i} + \frac{\partial \hat{J} \left(\mathbf{w} \right)}{\partial \mathbf{w}_c} \mathbf{j} + \frac{\partial \hat{J} \left(\mathbf{w} \right)}{\partial \mathbf{w}_d} \mathbf{k} \end{aligned} \tag{3.3}$$

where $\mathbf{w} = \mathbf{w}_a + \mathbf{w}_b \mathbf{i} + \mathbf{w}_c \mathbf{j} + \mathbf{w}_d \mathbf{k}$. The four components in (3.3) are calculated separately (according to the product rule of derivatives):

$$\begin{aligned} \nabla_{\mathbf{w}_a} \left(e[n] e^*[n] \right) &= e[n] \nabla_{\mathbf{w}_a} \left(e^*[n] \right) + \nabla_{\mathbf{w}_a} \left(e[n] \right) e^*[n] \\ \nabla_{\mathbf{w}_b} \left(e[n] e^*[n] \right) \mathbf{i} &= e[n] \nabla_{\mathbf{w}_b} \left(e^*[n] \right) \mathbf{i} + \nabla_{\mathbf{w}_b} \left(e[n] \right) e^*[n] \mathbf{i} \\ \nabla_{\mathbf{w}_c} \left(e[n] e^*[n] \right) \mathbf{j} &= e[n] \nabla_{\mathbf{w}_c} \left(e^*[n] \right) \mathbf{j} + \nabla_{\mathbf{w}_c} \left(e[n] \right) e^*[n] \mathbf{j} \\ \nabla_{\mathbf{w}_d} \left(e[n] e^*[n] \right) \mathbf{k} &= e[n] \nabla_{\mathbf{w}_d} \left(e^*[n] \right) \mathbf{k} + \nabla_{\mathbf{w}_d} \left(e[n] \right) e^*[n] \mathbf{k} \end{aligned} \tag{3.4}$$

The adaptive filter output can be defined in several forms (the update equation will change accordingly), e.g. it can be expressed as

$$y[n] = \mathbf{w}_{n-1}^T \mathbf{x}_n = \begin{bmatrix} \mathbf{w}_a^T \mathbf{x}_a - \mathbf{w}_b^T \mathbf{x}_b - \mathbf{w}_c^T \mathbf{x}_c - \mathbf{w}_d^T \mathbf{x}_d \\ \mathbf{w}_a^T \mathbf{x}_b + \mathbf{w}_b^T \mathbf{x}_a + \mathbf{w}_c^T \mathbf{x}_d - \mathbf{w}_d^T \mathbf{x}_c \\ \mathbf{w}_a^T \mathbf{x}_c + \mathbf{w}_c^T \mathbf{x}_a + \mathbf{w}_d^T \mathbf{x}_b - \mathbf{w}_b^T \mathbf{x}_d \\ \mathbf{w}_a^T \mathbf{x}_d + \mathbf{w}_d^T \mathbf{x}_a + \mathbf{w}_b^T \mathbf{x}_c - \mathbf{w}_c^T \mathbf{x}_b \end{bmatrix} \tag{3.5}$$

or equivalently:

$$y[n] = \mathbf{x}_n^H \mathbf{w}_{n-1}^* = \begin{bmatrix} \mathbf{w}_a^T\mathbf{x}_a - \mathbf{w}_b^T\mathbf{x}_b - \mathbf{w}_c^T\mathbf{x}_c - \mathbf{w}_d^T\mathbf{x}_d \\ -\mathbf{w}_a^T\mathbf{x}_b - \mathbf{w}_b^T\mathbf{x}_a - \mathbf{w}_c^T\mathbf{x}_d + \mathbf{w}_d^T\mathbf{x}_c \\ -\mathbf{w}_a^T\mathbf{x}_c - \mathbf{w}_c^T\mathbf{x}_a - \mathbf{w}_d^T\mathbf{x}_b + \mathbf{w}_b^T\mathbf{x}_d \\ -\mathbf{w}_a^T\mathbf{x}_d - \mathbf{w}_d^T\mathbf{x}_a - \mathbf{w}_b^T\mathbf{x}_c + \mathbf{w}_c^T\mathbf{x}_b \end{bmatrix} \tag{3.6}$$

Notation was simplified by omitting the subscript n. Going through all the calculations, with $y[n]$ defined as in (3.5), and substituting the partial gradients into (3.3), we have:

$$\begin{aligned} \nabla_{\mathbf{w}}\left(e[n]e^*[n]\right) &= -4e[n]\mathbf{x}_n^* - \mathbf{x}_n e^*[n] \\ &\quad + 2\left(e_a + e_b\mathbf{i} + e_c\mathbf{j} + e_d\mathbf{k}\right)\left(x_a - x_b\mathbf{i} - x_c\mathbf{j} - x_d\mathbf{k}\right) \\ &\quad + \left(x_a + x_b\mathbf{i} + x_c\mathbf{j} + x_d\mathbf{k}\right)\left(e_a - e_b\mathbf{i} - e_c\mathbf{j} - e_d\mathbf{k}\right) \\ &= -4e[n]\mathbf{x}_n^* - \mathbf{x}_n e^*[n] + 2e[n]\mathbf{x}_n^* + \mathbf{x}_n e^*[n] \\ &= -2e[n]\mathbf{x}_n^*. \end{aligned} \tag{3.7}$$

In conclusion, the QLMS update equation is

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e\left[n\right]\mathbf{x}_n^*. \tag{3.8}$$

Note: due to the non-commutativity of the quaternionic product, the order of the factors does affect the result. When the cost function is chosen as

$$\hat{J}\left(\mathbf{w}_{n-1}\right) = e^*[n]e[n] \tag{3.9}$$

the QLMS update equation becomes

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e^*[n]\mathbf{x}_n^*. \tag{3.10}$$

3.1.1 Differences with CLMS

Forcing two quaternion components to be zero, it is possible to prove that the QLMS does not generalize CLMS and LMS. Let $e_3, e_4 = 0, \mathbf{x}_3, \mathbf{x}_4 = \mathbf{0}$, then

$$e[n]e^*[n] = (e_1\mathbf{x}_1 + e_2\mathbf{x}_2) + (-e_1\mathbf{x}_2 + e_2\mathbf{x}_1)\,\mathbf{i} \tag{3.11}$$

In CLMS we have

$$e[n]e^*[n] = (e_1\mathbf{x}_1 + e_2\mathbf{x}_2) + (e_1\mathbf{x}_2 - e_2\mathbf{x}_1)\,\mathbf{i} \tag{3.12}$$

The two expressions (3.11) and (3.12) do not coincide. In the special case the input data are *isomorphic*, that is $q = q_a + Q\iota_r$, where $Q = \sqrt{q_b^2 + q_c^2 + q_d^2}$ and $\iota_r = \frac{(q_b\mathbf{i} + q_c\mathbf{j} + q_d\mathbf{k})}{Q}$, the derivations of the QLMS and CLMS algorithms are just alike [98].

3.1.2 Convergence properties of QLMS

In this paragraph, a note about the relation between the QLMS algorithm convergence and the choice of the step size μ is given. This is not the full study of convergence, but a simplified approach aimed at giving some initial information to the reader. The result to be obtained is the range of values of μ that guarantees the filter stability. This demonstration was proposed in [95] in the case of the earlier version of QLMS [98]. Here, we retrace the demonstration in the case of the correct QLMS algorithm [8].

Let us define the *a-priori* error $\tilde{e}[n]$ and the *a-posteriori* error $\overline{e}[n]$ as

$$\begin{aligned} \tilde{e}[n] &= d[n] - \mathbf{w}_n^T \mathbf{x}_n \\ \overline{e}[n] &= d[n] - \mathbf{w}_{n-1}^T \mathbf{x}_n \end{aligned}. \tag{3.13}$$

The goal is to estimate the range of values of the step size μ such that the *a-posteriori* error never exceeds the *a-priori* error: $\|\overline{e}[n]\| < \|\tilde{e}[n]\|$. The squared error $\|\overline{e}[n]\|^2$ can be expanded into a Taylor series as

$$\|\overline{e}[n]\|^2 = \|\tilde{e}[n]\|^2 + \Delta\mathbf{w}_n^H \frac{\partial \|\tilde{e}[n]\|^2}{\partial \mathbf{w}_n}. \tag{3.14}$$

Substituting $\Delta\mathbf{w}_n = \mu\tilde{e}[n]\mathbf{x}_n^*$ and $\frac{\partial\|\tilde{e}[n]\|^2}{\partial\mathbf{w}_n} = -2(\tilde{e}[n]\mathbf{x}_n^*)$ into (3.14), we have

$$\begin{aligned} \|\overline{e}[n]\|^2 &= \|\tilde{e}[n]\|^2 - 2\mu\mathbf{x}_n^T\tilde{e}^*[n]\tilde{e}[n]\mathbf{x}_n^* \\ &= \|\tilde{e}[n]\|^2 - 2\mu\,\|\tilde{e}[n]\|^2\,\|\mathbf{x}_n\|^2. \\ &= \|\tilde{e}[n]\|^2\,(1 - 2\mu\,\|\mathbf{x}_n\|^2) \end{aligned} \tag{3.15}$$

In order to guarantee convergence, it must be

$$\left|1 - 2\mu\,\|\mathbf{x}_n\|^2\right| < 1 \tag{3.16}$$

So the values of μ will fall in the estimated range

$$0 < \mu < \frac{1}{\|\mathbf{x}_n\|^2}. \tag{3.17}$$

3.2 Frequency Domain Quaternion Adaptive Filters

With regard to frequency-domain filtering, we present here the Overlap-Save Quaternion Frequency Domain Adaptive Filter (OS-QFDAF). The OS-QFDAF algorithm is a block algorithm. Block algorithms are defined by a periodic update equation, i.e. the filter coefficients are updated at each block iteration. In general, such algorithms differ in the length of the input block, the number of the overlapping samples, the type of transform used (for those algorithms working in a transform domain). In the OS-QFDAF algorithm, the number of the overlapping samples is denoted by M. Usually, the overlap length M is chosen equal to the filter length in the time domain, so we can conventionally refer to M as the filter length. In order to simplify the implementation, the transform length is chosen as $N = M + L$, where L is the block length.

3.2.1 Introduction to the OS-QFDAF algorithm

In complex algebra, the inverse transform of the product of two DFT sequences provides a circular convolution in the time domain, while filtering operations are implemented with linear convolution. In order to obtain the linear convolution from the circular convolution, proper window constraints on data are needed. If these constraints are not taken into account, with the idea of designing an algorithm with reduced computational cost, the algorithm may not exactly converge to Wiener optimum solution. Fast convolution may be performed using two different methods: overlap-add (OA) and overlap-save (OS). The former requires much more computation than needed [104], so the OS method with 50% overlap turns out to be the best performing choice. Therefore, the OS-QFDAF algorithm presented here embeds the OS method. The algorithm comes with power normalization, a strategy intended to improve convergence to optimum and fully described later on in this chapter. Differently from the complex-valued OS-FDAF, in the quaternion domain some modifications are need, due to the fact that the convolution theorem is not valid in the standard formulation (the product of the DFT sequences), but it is slightly more complicated [22, 74]. We refer to 2.3 for a brief introduction to this topic. Considering the mathematical formulation of the convolution and correlation operations as deeply described in [74], the proposed OS-QFDAF from the Block QLMS algorithm as formulated in 3A. It is possible to store the input samples into a block matrix $\mathbf{X}_k$ and obtain a similar formulation for all block algorithms in both time and transform domain. The final form of the OS-QFDAF algorithm results from the transformation into the frequency domain of the equations of Block QLMS, being aware of the properties of convolution and correlation in the quaternion domain [74]. The Quaternion Discrete Fourier Transform was introduced in Par. 2.1.1 and a practical method for computing the Quaternion Fast Fourier Transform was suggested. This method is based on the Cayley-Dickson symplectic decomposition for quaternions. A brief overview of the OS-QFDAF algorithm and comments about it are given just below. A block diagram of the algorithm is illustrated in Fig. 3.1.

3.2.2 OS-QFDAF algorithm overview

Initialize the algorithm with

$$\begin{aligned} &\mathbf{W}_{init} = \mathbf{0} \text{ (2M-by-1 null vector)} \\ &\mu = \mu_0, \quad P_0(m) = \delta, \quad m = 0, 1, ..., N-1 \end{aligned} \tag{3.18}$$

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

The following steps are to be executed for each new input block k.

Compute the QFFT of the filter input samples as

$$\mathbf{X}_k = \text{diag}\left[\text{QFFT}\left[\mathbf{x}_{old}^M \quad \mathbf{x}_k^L\right]^T\right] \tag{3.19}$$

where the input block consists of $\mathbf{x}_{old}^M$ and $\mathbf{x}_k^L$, defined as

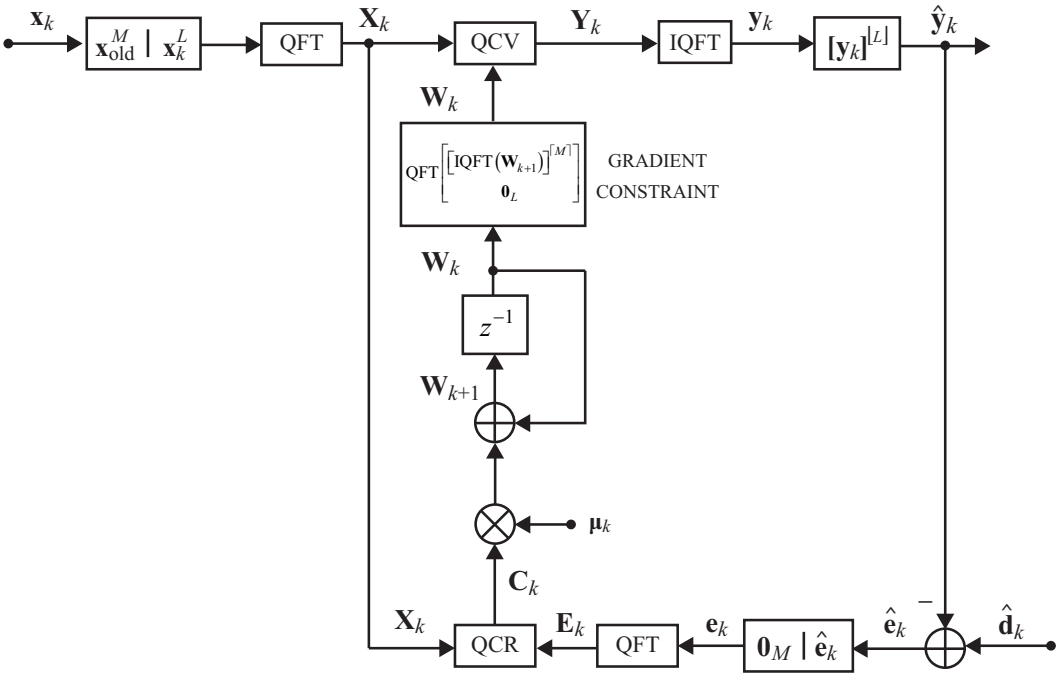


Figure 3.1. OS-QFDAF block diagram.

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

Compute the filter output in the frequency domain, by using the convolution theorem [74] in (2.17) as

$$\mathbf{Y}_k = \mathbf{W}_k^a \mathbf{X}_k + \mathbf{W}_k^b \boldsymbol{\nu}_2 \mathbf{X}_{-k}, \tag{3.20}$$

where $\mathbf{X}_{-k} \equiv \mathbf{X}_k(-\omega)$ is the frequency reversed signal of $\mathbf{X}_k(\omega)$, while $\mathbf{W}_k^a$ and $\mathbf{W}_k^b$ are the simplex and perplex parts such that $\mathbf{W}_k = \mathbf{W}_k^a + \mathbf{W}_k^b \boldsymbol{\nu}_2$. The quaternion convolution in (6.27) is implemented by the block labeled as QCV in Fig. 3.1 and detailed in Fig. 3.2. Then we return to time domain

$$\hat{\mathbf{y}}_k = [\text{IQFFT}(\mathbf{Y}_k)]^{\lfloor L \rfloor} \tag{3.21}$$

where $\mathbf{Y}_k$ is the filter output in the frequency domain[1], the symbol $\lfloor L \rfloor$, means the last L samples of vector $\mathbf{y}_k$ and $\hat{\mathbf{y}}_k$ is the windowed filter output in the time domain[2].

Let $\hat{\mathbf{d}}_k$ be the desired output vector in the time domain at block k:

$$\hat{\mathbf{d}}_k = \begin{bmatrix} d(kL) & d(kL+1) & \cdots & d(kL+L-1) \end{bmatrix}^T \tag{3.22}$$

the error vector in the time domain is defined as

$$\hat{\mathbf{e}}_k = \hat{\mathbf{d}}_k - \hat{\mathbf{y}}_k \tag{3.23}$$

[1]Diagonalization in (3.19) allows to express the filter output signal in a formalism similar to that of time-domain Block LMS [104].

[2]In OS only the last L samples of the anti-transformed output block are useful in the time domain. In fact, it is not assured that the first M samples are zero, so windowing is necessary.

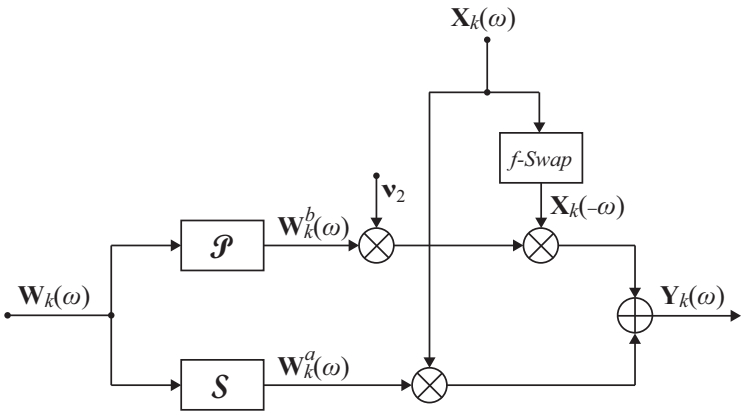


Figure 3.2. Block diagram of the quaternion convolution (QCV). In quaternion domain the standard convolution theorem is not valid and the frequency domain multiplication is performed separately for the simplex and perplex parts.

and transformed[3] into the frequency domain as

$$\mathbf{E}_k = \mathrm{QFFT}\left(\begin{bmatrix} 0_M & \hat{\mathbf{e}}_k \end{bmatrix}^T\right). \tag{3.24}$$

This algorithm updates the learning rates by means of power normalization, fully explained in Par. 3.2.3:

$$\boldsymbol{\mu}_k = \mu_0 \cdot \mathrm{diag}\left[P_k^{-1}(0), \ldots, P_k^{-1}(N-1)\right]. \tag{3.25}$$

Finally, we can update the filter weights as

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \boldsymbol{\mu}_k \mathbf{C}_k \tag{3.26}$$

where $\mathbf{C}_k$ is the quaternion correlation [74] in (2.18) between the input $\mathbf{X}_k$ and the error $\mathbf{E}_k$

$$\mathbf{C}_k = E_k^a(\omega)\left[X_k^{aH}(\omega) - \boldsymbol{\nu_2} X_k^b(-\omega)\right] + E_k^b(\omega)\boldsymbol{\nu_2}\left[X_k^{aH}(-\omega) - \boldsymbol{\nu_2} X_k^b(\omega)\right] \tag{3.27}$$

where $\mathbf{X}_{-k} \equiv \mathbf{X}_k(-\omega)$, while $\mathbf{E}_k^a$ and $\mathbf{E}_k^b$ are the simplex and perplex parts of $\mathbf{E}_k = \mathbf{E}_k^a + \mathbf{E}_k^b \boldsymbol{\nu_2}$, respectively. This operation is represented by the box labeled as QCR in Fig. 3.1 and detailed in Fig. 3.3.

The classic version of the OS-QFDAF algorithm considers constraining the gradient as follows:

$$\mathbf{W}_{k+1} = \mathrm{QFFT}\begin{pmatrix} [\mathrm{IQFFT}(\mathbf{W}_{k+1})]^{\lceil M \rceil} \\ \mathbf{0}_L \end{pmatrix}. \tag{3.28}$$

The stochastic gradient in the time domain is defined as
$\nabla \hat{J}_k = [\mathrm{IQFFT}(\mathbf{C}_k)]^{\lceil M \rceil}$, i.e. only the first M samples are useful, since the last L samples are those relative to circular correlation and it is not guaranteed they are

[3] In order to transform the error vector, zero-padding is needed.

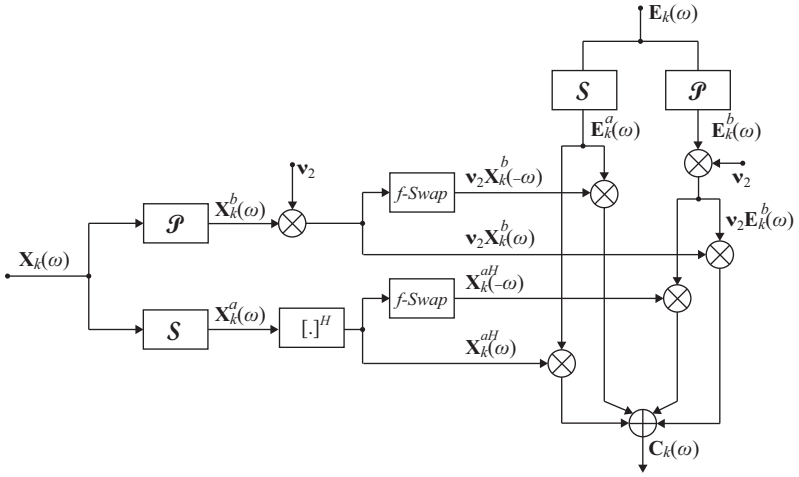


Figure 3.3. Block diagram of the quaternion correlation (QCR).

zero. Equation (3.28) applies the gradient constraint after updating the filter weights in the frequency domain. If the gradient constraint is neutralized, the term $\mathbf{C}_k$ corresponds to a circular correlation in the time domain and this modified version of the QFDAF algorithm is denoted as *Unconstrained* QFDAF. Usually, unconstrained algorithms exhibit a polarized convergence. In order to get convergence to optimum, the filter length M should be increased, but this is not recommended.

A second version of the OS-QFDAF algorithms is the Sliding Window transform domain algorithms concisely presented in 3A.

3.2.3 Power Normalization

Power normalization makes it possible to have all modes converging at the same rate. This is achieved, as in (3.25), by assigning to each weight an individual step size of its own, $\mu(m) = \mu/P(m)$, where $P(m)$ is an estimation of the average power relative to the m-th frequency bin. The elements of vector $\boldsymbol{\mu}_k$ have the property of equalizing the convergence modes by whitening the input signal. When power estimation $P(m)$ is not available, as it usually happens, especially when the input data are non-stationary, it has to be computed differently. The update for the m-th power in the m-th frequency bin at step k (block k) is carried out by recursion as follows:

$$P_k(m) = \lambda P_{k-1}(m) + (1-\lambda)\left|X_k(m)\right|^2 \tag{3.29}$$

with $m = 0, ..., N-1$. The parameter denoted by $\lambda \in [0,1]$ represents a forgetting factor and (3.29) is the expression of a low-pass filter. If $\lambda = 1$, the term $|X_k(m)|^2$ (punctual energy of the m-th frequency bin) is ignored.

At step k, the m-th step-size can be defined as

$$\mu_k(m) = \frac{\mu}{P_k(m)} \tag{3.30}$$

where $m = 0, ..., N-1$ and μ is a scalar chosen empirically.

Table 3.1. Computational Cost of Quaternion Adaptive Algorithms.

Algorithm	Computational Cost
QLMS	$32 \cdot M \cdot n_{samples}$
BQLMS	$16 \cdot M + 16 \cdot L \cdot n_{blocks}$
OS-QFDAF	$5 \cdot 2N\log_2 N + 4 \cdot 16N$

3.2.4 Computational cost of OS-QFDAF

An approximation of the computational cost is given by taking into consideration the critical paths of the algorithm, i.e. multiplications and computation of FFTs. 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, $2 \cdot 16N$ multiplications to compute the filter output and $2 \cdot 16N$ multiplications to update the filter weights. Considered that a block has $L = M$ samples, the computational cost for OS-QFDAF is approximately:

$$C_{OS-QFDAF} = 5 \cdot 2N\log_2 N + 4 \cdot 16N = 20M\log_2 2M + 128M, \tag{3.31}$$

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

In the QLMS algorithm the computation of the filter output requires $4{\cdot}4M = 16M$ multiplications for each sample and so does the computation of the cross-correlation in the update equation. Overall, for M samples, the computational cost for QLMS is approximately $C_{QLMS} = 32M \cdot M = 32M^2$. The complexity ratio between OS-QFDAF and QLMS reveals that the former is several times faster than its time-domain ancestor:

$$\frac{C_{OS-QFDAF}}{C_{QLMS}} = \frac{5\log_2 2M + 32}{8M}. \tag{3.32}$$

For example, for $M = 64$, the OS-QFDAF is about 7 times faster than the QLMS algorithm. A comparison among algorithms is given in Table 3.1.

3.2.5 Convergence Properties

The study of the convergence properties of an algorithm gives clues about how the algorithm behavior is conditioned by the value of the $\mathbf{R_{xx}}$ eigenvalues, the filter length, the block length and other parameters.

In order to simplify the theoretical derivation of the convergence and steady-state behavior, without loss of generality the following analyses will be performed assuming a real-valued input signal $\mathbf{x}_k$. In this case, the spectrum $\mathbf{X}_k$ is symmetric, i.e. $\mathbf{X}_{-k} = \mathbf{X}_k$, and the convolution in (6.27) simply returns in the traditional product of the sequence transforms. A similar result can be obtained for the correlation (3.27). In case of quaternion input, the analyses are simply replicated for the simplex and perplex parts, respectively, obtaining similar results.

Step size stability range

As a matter of simplicity, we analyze the convergence behavior of the *Unconstrained* version of the OS-QFDAF algorithm for a real-valued input. The analysis in the

case of constrained algorithm can be conducted the same way by adding the gradient constraint [27]. Considering the test circuit in Fig. 3.4 (Par. **??**), the desired output in the frequency domain can be expressed as

$$\mathbf{D}_k = \mathbf{G}_{M,0}\left(\mathbf{X}_k\mathbf{W}_0 + \mathbf{V}_k\right) \tag{3.33}$$

where $\mathbf{W}_0$ is the optimum solution, $\mathbf{G}_{M,0} = \mathbf{F}\mathbf{g}_{M,0}\mathbf{F}^{-1}$ is a window constraint (necessary to get linear convolution from circular convolution). The transform denoted by $\mathbf{F}$ is the QFFT and matrix $\mathbf{g}_{M,0}$ is defined as

$$\mathbf{g}_{M,0} = \begin{bmatrix} \mathbf{I}_{M,M} & \mathbf{0}_{M,L} \\ \mathbf{0}_{L,M} & \mathbf{0}_{L,L} \end{bmatrix} \tag{3.34}$$

$\mathbf{V}_k$ denotes the additive noise in the frequency domain, as shown in Fig. 3.4. In other words, $\mathbf{E}_{0k}$ is the error in case $\mathbf{W}_k = \mathbf{W}_0 \Rightarrow \mathbf{E}_{0k} = \mathbf{V}_k$. The error in the frequency domain is defined as

$$\mathbf{E}_k = \mathbf{D}_k - \mathbf{Y}_k = \mathbf{G}_{M,0}\left(\mathbf{X}_k\mathbf{W}_0 + \mathbf{V}_k - \mathbf{X}_k\mathbf{W}_k\right). \tag{3.35}$$

Substituting (3.35) into the unconstrained adaptation law (3.26), where $\mathbf{G}_{M,0} = \mathbf{I}$, we obtain:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \boldsymbol{\mu}_k\mathbf{X}_k^H\mathbf{G}_{M,0}\left(\mathbf{X}_k\mathbf{W}_0 + \mathbf{V}_k - \mathbf{X}_k\mathbf{W}_k\right). \tag{3.36}$$

We define the weight error vector in the frequency domain:

$$\mathbf{U}_k = \mathbf{W}_k - \mathbf{W}_0 \tag{3.37}$$

thus resulting:

$$\begin{aligned} \mathbf{U}_{k+1} &= \mathbf{U}_k + \boldsymbol{\mu}_k\mathbf{X}_k^H\mathbf{G}_{M,0}\left(\mathbf{X}_k\mathbf{W}_0 + \mathbf{V}_k - \mathbf{X}_k\mathbf{W}_k\right) \\ &= \left(\mathbf{I} - \boldsymbol{\mu}_k\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k\right)\mathbf{U}_k + \boldsymbol{\mu}_k\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{V}_k. \end{aligned} \tag{3.38}$$

Taking the expectation of each term in (3.38), we assume that $\mathbf{X}_k$ and $\mathbf{V}_k$ are independent. Under this assumption we can apply the orthogonality principle according to which $E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{V}_k\right\} = 0$:

$$\begin{aligned} E\left\{\mathbf{U}_{k+1}\right\} &= \left(\mathbf{I} - \boldsymbol{\mu}_k E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k\right\}\right) E\left\{\mathbf{U}_k\right\} + \boldsymbol{\mu}_k E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{V}_k\right\} \\ &= \left(\mathbf{I} - \boldsymbol{\mu}_k\mathbf{R}_{xx}^u\right) E\left\{\mathbf{U}_k\right\} \end{aligned} \tag{3.39}$$

where the autocorrelation matrix $\mathbf{R}_{xx}^u$ was defined as

$$\mathbf{R}_{xx}^u = E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k\right\}. \tag{3.40}$$

We see that the convergence behavior of the step-normalized Unconstrained QFDAF is controlled by the eigenvalues of the power-normalized autocorrelation matrix $\boldsymbol{\mu}_k\mathbf{R}_{xx}^u$.

Matrix diagonalization within the right eigenvalue spectrum[4], $\sigma_R(\mathbf{A})$, is possible with the use of a unitary matrix $\mathbf{Q}$ as stated in [114]:

$$\sigma_R\left(\mathbf{Q}^H\mathbf{A}\mathbf{Q}\right) = \sigma_R\left(\mathbf{A}\right). \tag{3.41}$$

[4] λ is a right eigenvalue if there exist a non-zero $\mathbf{x} \in \mathbb{H}$ such that $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda$ and λ is a left eigenvalue if there exist a non-zero $\mathbf{x} \in \mathbb{H}$ such that $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$.

The diagonal matrix having the right eigenvalues of $\mathbf{R}_{xx}^{u}$ on the principal diagonal is denoted by $\mathbf{\Lambda}$:

$$\mathbf{\Lambda} = \mathbf{Q}^{H}\mathbf{R}_{xx}^{u}\mathbf{Q}. \tag{3.42}$$

Being $\hat{\mathbf{U}}_k = \mathbf{Q}^H\mathbf{U}_k$ the transformed error vector, (3.39) can be rewritten using the new vectors (we do not consider power normalization here). We take the expectation of each element of (3.39):

$$E\left\{\hat{\mathbf{U}}_{k+1}\right\} = (\mathbf{I} - \mu\mathbf{\Lambda})\, E\left\{\hat{\mathbf{U}}_k\right\}. \tag{3.43}$$

Equation (3.43) can be decomposed into a finite-difference-equation system as

$$E\left\{\hat{U}_{k+1}(m)\right\} = (1 - \mu\lambda_m)\, E\left\{\hat{U}_k(m)\right\}. \tag{3.44}$$

Back-substitution leads to

$$E\left\{\hat{U}_{k+1}(m)\right\} = (1 - \mu\lambda_m)^k E\left\{\hat{U}_{-1}(m)\right\}. \tag{3.45}$$

Convergence occurs if $|1 - \mu\lambda_m| < 1$.

The autocorrelation matrix is a Hermitian matrix, i.e. $\mathbf{R}_{xx} = \mathbf{R}_{xx}^{H}$ and the eigenvalues of a Hermitian matrix are all real. This property was formerly demonstrated by Charles Hermite for complex Hermitian matrices and it can be expanded to quaternions at least for right eigenvalues. In effect, right eigenvalues can be computed by Cayley-Dickson decomposition and each complex sub-matrix is Hermitian, as well.

If we consider the left eigenspectrum, the convergence properties are not elementary: whereas the right eigenspectrum is a quaternionic analogue of the complex eigenvalues [87], the left eigenspectrum is not and it is not even unitarily invariant, so the autocorrelation matrix $\mathbf{R}_{xx}$ cannot be diagonalized as in (3.42). However, if both the eigenspectra $\sigma_R(\mathbf{A})$ and $\sigma_L(\mathbf{A})$ are finite, then $\sigma_R(\mathbf{A}) = \sigma_L(\mathbf{A})$. A proof of that is given in [42, 87]. In our case $\sigma_R(\mathbf{A})$ is indeed finite: we can read in [42, 114] that $\sigma_R(\mathbf{A})$ is infinite if and only if $\sigma_R(\mathbf{A}) \not\subset \mathbb{R}$. Nothing can be said about $\sigma_L(\mathbf{A})$ a priori. In fact, examples showed that a Hermitian matrix does not necessarily produce a finite left eigenspectrum. So, further investigation is essential. Detailed information about the computation of the quaternion eigenspectra is given in [42, 114] and a summary is reported in 1.3.4.

The step size convergence range, working with the right eigenspectrum, can be found by solving the inequality

$$|1 - \mu\lambda_m| < 1. \tag{3.46}$$

Definitively, the step size μ must be chosen such that

$$0 < \mu < \frac{1}{\lambda_{\max}}. \tag{3.47}$$

Alternatively, given (3.39) and considering that

$$\begin{aligned} E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k\right\} &= E\left\{\mathrm{tr}(\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k)\right\} \\ &= \mathrm{tr}(E\left\{\mathbf{X}_k^H\mathbf{G}_{M,0}\mathbf{X}_k\right\}) \\ &= \mathrm{tr}(\mathbf{R}_{xx}) \end{aligned} \tag{3.48}$$

a conservative relation for choosing the step size stability range is suggested below:

$$0 < \mu < \frac{1}{\mathrm{tr}(\mathbf{R}_{xx})}. \tag{3.49}$$

Weight adaptation in OS-QFDAF is independent for each filter weight, since each weight is associated with one mode of the adaptive filtering system. On the contrary, in the standard original time domain LMS/QLMS algorithm each weight is accountable for multiple modes summed up and therefore the convergence rate cannot be optimized for any specific mode of the process.

From (3.39) we see that the time constant τ_m for the m-th mode of the algorithm can be found by considering the time it takes for the error vector to decay as $1/e$ of its initial value:

$$\begin{aligned} \left|\hat{U}_{\tau_m}(m)\right| &= \left|\hat{U}_{-1}(m)\right| e^{-1} \\ &= \left|(1-\mu\lambda_m)^{\tau_m}\right| \left|\hat{U}_{-1}(m)\right| \\ &\rightarrow \tau_m \left|\ln(1-\mu\lambda_m)\right| = -1. \end{aligned} \tag{3.50}$$

Definitively, the time constant for the m-th mode is

$$\tau_m = -\frac{1}{\left|\ln(1-\mu\lambda_m)\right|} \tag{3.51}$$

where λ_m is the m-th eigenvalue of the autocorrelation matrix $\mathbf{R}_{xx}^{u}$ of the input data $\mathbf{X}_k$. Anyway, taking advantage of power normalization, in a wide-sense stationary environment, the weights tend to converge at the same rate.

Excess Mean Square Error

When the OS-QFDAF algorithm operates without power normalization, the Excess Mean Square Error (EMSE) can be computed in a straightforward way. With the definition in (3.37), the error at block k is

$$\begin{aligned} \mathbf{E}_k &= \mathbf{D}_k - \mathbf{Y}_k = \mathbf{D}_k - \mathbf{X}_k\mathbf{W}_k - \mathbf{X}_k\mathbf{W}_0 + \mathbf{X}_k\mathbf{W}_0 \\ &= \mathbf{V}_k - \mathbf{X}_k\mathbf{U}_k. \end{aligned} \tag{3.52}$$

Substituting (3.52) into the MSE curve, with the assumption that $\mathbf{V}_k$ and $\mathbf{X}_k$ are independent, the MSE at block k is

$$\begin{aligned} J_k &= E\left\{|\mathbf{E}_k|^2\right\} = E\left\{(\mathbf{V}_k - \mathbf{X}_k\mathbf{U}_k)\left(\mathbf{V}_k^H - \mathbf{U}_k^H\mathbf{X}_k^H\right)\right\} \\ &= J_{\min} + E\left\{\mathbf{X}_k\mathbf{U}_k\mathbf{U}_k^H\mathbf{X}_k^H\right\}. \end{aligned} \tag{3.53}$$

We define the weight error vector correlation as

$$\mathbf{K}_k = E\left\{\mathbf{U}_k\mathbf{U}_k^H\right\} \in \mathbb{H}^{M\times M} \tag{3.54}$$

which can be conveniently diagonalized as

$$\hat{\mathbf{K}}_k = \mathbf{Q}^H\mathbf{K}_k\mathbf{Q} = \mathrm{diag}\left[\hat{k}_k(i)\right] \in \mathbb{R}^{M\times M}. \tag{3.55}$$

Since $\hat{\mathbf{K}}_k$ is a diagonal matrix, the elements forming its principal diagonal are its eigenvalues. With the unitary similarity transformation in (3.55), we are assuming that we are using the right eigenspectrum and since $\mathbf{K}_k$ is Hermitian, $\hat{\mathbf{K}}_k$ is real-valued. In view of the above, (3.53) can be rewritten as

$$\begin{aligned} J_k &= J_{\min} + E\left\{\mathbf{X}_k\mathbf{U}_k\mathbf{U}_k^H\mathbf{X}_k^H\right\} = J_{\min} + E\left\{\mathbf{X}_k\hat{\mathbf{K}}_k\mathbf{X}_k^H\right\} \\ &= J_{\min} + E\left\{\hat{\mathbf{K}}_k\mathbf{X}_k\mathbf{X}_k^H\right\} = J_{\min} + E\left\{\mathrm{tr}\left(\hat{\mathbf{K}}_k\mathbf{X}_k\mathbf{X}_k^H\right)\right\} \\ &= J_{\min} + \mathrm{tr}\left(\hat{\mathbf{K}}_k\mathbf{R}_{xx}\right) = J_{\min} + \mathrm{tr}\left(\hat{\mathbf{K}}_k\mathbf{\Lambda}\right) \end{aligned} \tag{3.56}$$

where $\mathbf{\Lambda} = \mathbf{Q}^H\mathbf{R}_{xx}\mathbf{Q}$.

The EMSE is defined as

$$J_{EMSE} = J_k - J_{\min} = \mathrm{tr}\left(\hat{\mathbf{K}}_k\mathbf{\Lambda}\right) = \sum_{i=0}^{M-1}\hat{k}_k(i)\,\lambda_i. \tag{3.57}$$

The computation of $\hat{k}_k(i)$ can be conducted as follows. Multiplying the error vector $\mathbf{U}_k = \mathbf{U}_{k-1} + \mu\mathbf{X}_k^H\mathbf{E}_k$ by its Hermitian vector and taking the expectation of each term, we have:

$$E\left\{\mathbf{U}_k^H\mathbf{U}_k\right\} = E\left\{\left(\mathbf{I} - \mu\mathbf{X}_k^H\mathbf{X}_k\right)\mathbf{U}_{k-1}\mathbf{U}_{k-1}^H\left(\mathbf{I} - \mu\mathbf{X}_k^H\mathbf{X}_k\right)^H\right\} + \mu^2 J_{\min}\mathbf{R}_{xx} \tag{3.58}$$

where $J_{\min} = E\left\{\mathbf{V}_k\mathbf{V}_k^H\right\} \in \mathbb{R}$.

The weight error vector correlation can be expressed as

$$\mathbf{K}_k = \left(\mathbf{I} - \mu\mathbf{X}_k^H\mathbf{X}_k\right)\mathbf{K}_{k-1}\left(\mathbf{I} - \mu\mathbf{X}_k^H\mathbf{X}_k\right)^H + \mu^2 J_{\min}\mathbf{R}_{xx} \tag{3.59}$$

and diagonalizing as in (3.52) and (3.53)

$$\hat{\mathbf{K}}_k = \left(\mathbf{I} - \mu\mathbf{\Lambda}\right)\hat{\mathbf{K}}_{k-1}\left(\mathbf{I} - \mu\mathbf{\Lambda}\right)^H + \mu^2 J_{\min}\mathbf{\Lambda} \tag{3.60}$$

we obtain a system of M finite difference equations:

$$\hat{k}_k(i) = (1 - \mu\lambda_i)^2\hat{k}_{k-1}(i) + \mu^2 J_{\min}\lambda_i \tag{3.61}$$

for $i = 0, 1, \ldots M-1$. Back-substitution leads to

$$\hat{k}_k(i) = (1 - \mu\lambda_i)^{2k}\hat{k}_{-1}(i) + \mu^2\sum_{i=0}^{k/2}(1 - \mu\lambda_i)\,J_{\min}\lambda_i. \tag{3.62}$$

As $k \to \infty$

$$\begin{aligned} \lim_{k\to\infty}\hat{k}_k(i) &= \mu^2\sum_{i=0}^{k/2}(1 - \mu\lambda_i)\,J_{\min}\lambda_i \\ &= \mu^2\lambda J_{\min}\frac{1}{1 - (1 - \mu\lambda_i)^2} = \frac{\mu J_{\min}}{2 - \mu\lambda_i}. \end{aligned} \tag{3.63}$$

Substituting (3.63) into (3.57):

$$\begin{aligned} J_\infty &= J_{\min} + J_{EMSE} \\ &= J_{\min} + \sum_{i=0}^{M-1}\hat{k}_k(i)\,\lambda_i = J_{\min} + J_{\min}\sum_{i=0}^{M-1}\frac{\mu\lambda_i}{2 - \mu\lambda_i}. \end{aligned} \tag{3.64}$$

Equation (3.64) can be expanded into a Taylor series:

$$J_\infty = J_{\min} + J_{\min}\frac{\mu}{2}\sum_{i=0}^{M-1}\lambda_i\left(1+\frac{\mu}{2}\lambda_i+\frac{\mu^2}{2}\lambda_i^2+\ldots\right). \tag{3.65}$$

Recalling that the autocorrelation matrix $\mathbf{R}_{xx}$ is Hermitian, it has right eigenvalues belonging to the reals and, for $\mu << 2/\lambda_{\max}$, we have

$$J_\infty \approx J_{\min} + J_{\min}\frac{\mu}{2}\sum_{i=0}^{M-1}\lambda_i = J_{\min}\left(1+\frac{\mu}{2}\mathrm{tr}\left(\mathbf{R}_{xx}\right)\right). \tag{3.66}$$

Definitively, the EMSE is equal to

$$\begin{aligned} J_{EMSE} &\simeq \frac{\mu}{2}\mathrm{tr}\left(\mathbf{R}_{xx}\right) = \frac{\mu}{2}M\cdot r_{xx}\left[0\right] \\ &= \frac{\mu}{2}M\cdot\left\{\text{input power}\right\}. \end{aligned} \tag{3.67}$$

QDFT with Power Normalization

The combination of DFT and Power Normalization is a powerful gimmick to obtain fast convergence in an adaptive algorithm. One of the effects of the DFT, the *orthogonalizing property*, is to asymptotically make the autocorrelation matrix a diagonal matrix with increasing the DFT length N. This can be seen by finding a convenient expression of the autocorrelation matrix $\mathbf{R}_{xx}$. The expectation of the squared magnitude of $\mathbf{X}(\omega)$ is equal to

$$E\left[\left|X\left(\omega\right)\right|^2\right] = \sum_{n=0}^{N-1}\sum_{k=0}^{N-1}E\left[x\left(n\right)x^*\left(k\right)\right]e^{-\boldsymbol{\mu}\omega(n-k)/N} \tag{3.68}$$

where $\phi_{xx}\left(n-k\right) = E\left[x\left(n\right)x^*\left(k\right)\right]$ is the autocorrelation function of $x\left(n\right)$.

Equation (3.68) is conveniently divided by N and rewritten as

$$\frac{E\left[\left|X\left(\omega\right)\right|^2\right]}{N} = \sum_{l=-N+1}^{N-1}\left(1-\frac{|l|}{N}\right)\phi_{xx}\left(l\right)e^{-\boldsymbol{\mu}\omega l/N} \tag{3.69}$$

where the variable running over the summation has been changed in $l = n-k$ and the function is computed in a double (see $|l|$) triangular domain by multiplying the autocorrelation coefficient $\phi_{xx}\left(n-k\right)$ by Bartlett's window.

The autocorrelation coefficients ϕ_{xx} of $\mathbf{x}$ can be considered as an ARMA (AutoRegressive Moving Average) process. Its expression in the time domain is

$$\phi_{xx}\left(l\right) = E\left\{x\left(n\right)x^*\left(n-l\right)\right\} = \sum_{p=0}^{P-1}a_P c_P^{|l|} \tag{3.70}$$

where $l = \ldots,-1,0,1,\ldots$, c_P, a_P and P denote the process poles, the process coefficients and the order of the process, respectively.

In the frequency domain the autocorrelation coefficients are transformed into

$$\Phi_{xx}\left(e^{-\boldsymbol{\mu}2\pi n/N}\right) = \sum_{l=-\infty}^{\infty}\phi_{xx}\left(l\right)e^{-\boldsymbol{\mu}2\pi n/N} \tag{3.71}$$

that is the expression of the Power Spectral Density (PSD) of the input process.

As $l \to \infty$, we can see that $\phi_{xx}(l) \to 0$ and the diagonal elements of the autocorrelation matrix (divided by N for convience) tend to the PSD samples asymptotically.

Recalling the Fourier transform of the exponential series, the autocorrelation matrix can be rewritten by decomposing the coefficients as

$$\frac{\mathbf{R}_{xx}}{N} = \begin{cases} \sum_{l=-N+1}^{N+1} \left(1 - \frac{|l|}{N}\right) \phi_{xx}(l)\, e^{-\boldsymbol{\mu} 2\pi n/N} & m = n \\ \frac{1}{N} \sum_{p=0}^{P-1} a_P \left[\frac{1}{c_P - e^{+\boldsymbol{\mu} 2\pi m/N}} \frac{1}{c_P^{-1} - e^{-\boldsymbol{\mu} 2\pi n/N}} + c.c. \right] & m \neq n \end{cases} \tag{3.72}$$

where n,m denote the matrix indexes.

As the transform length N tends to infinity, the non-diagonal elements of the autocorrelation matrix tend to 0.

As can be read from (3.72), when power normalization is applied, the elements of the principal diagonal of $\mathbf{R}_{xx}$, multiplied by the normalization factors, all tend to have the same value, thus leading to a single convergence mode. As N increases, the diagonal matrix just obtained tends to the identity matrix $\mathbf{I}$, i.e. $\mathbf{R}_{xx} \to \boldsymbol{\Lambda}$, $\boldsymbol{\Lambda}^{-1}\mathbf{R}_{xx} = \mathbf{I}$.

3.3 Simulations

3.3.1 OS-QFDAF simulations

The results from the simulations presented in this paragraph were obtained by applying the transform domain algorithms to a system identification problem (Fig. 3.4). The system to be identified is characterized by a set of random weights $\mathbf{w}_0$ (in the time domain), uniformly distributed in the range $[-1, 1]$. The quaternion filter input signal $x[n]$ is a unit variance colored noise with length of 20,000 samples, obtained by filtering the white Gaussian noise $\eta[n]$ as

$$x[n] = bx[n-1] + \frac{\sqrt{1-b^2}}{\sqrt{4}} \eta[n] \tag{3.73}$$

where b determines the bandwidth of the signal. The additive signal $v[n]$ is defined the same way as $x[n]$, but it has a different bandwidth. In the following experiments an SNR of 40 dB is used. The choice of the overlap length and the block length ($M = L$) determines the FFT length ($N = M + L$).

The OS-QFDAF algorithm and its modifications exploit the *power normalization* technique in order to have all filter weights converging at the same rate. Figure 3.5 shows how power normalization affects the filter performance by whitening the input signal. The test is repeated with $b = 0$ (first 10,000 samples) and $b = 0.95$ (last 10,000 samples). The other parameters are set as $M = L = 32$, $\mu_0 = 0.002$, $\delta = 0.001$ and $\lambda = 0.5$. Whereas both the OS-QFDAF and the Sliding QFFT-QLMS algorithm converge nearly at the same rate when b is changed from 0 to 0.95, the QLMS algorithm converges very slowly if $b = 0.95$.

The step size μ_0 in a power-normalized algorithm is an overall parameter governing the diagonal inverse power matrix. In both cases power normalization is present

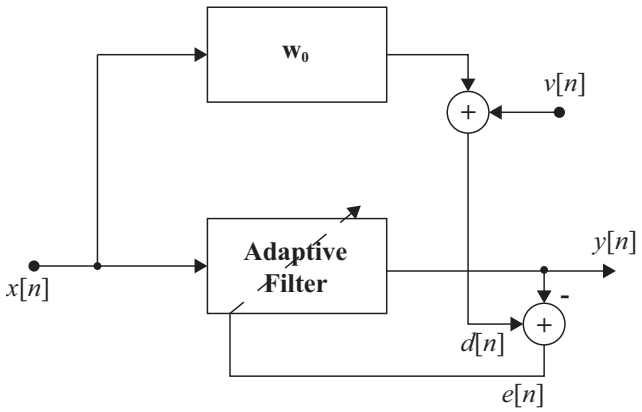


Figure 3.4. Algorithm test circuit.

or not, the step size modifies the algorithm behavior in the same way as in QLMS: if the step size is too large, the filter tends to instability. On the contrary, if the step size is too small, further iterations are needed for the filter to reach the steady state. Somewhere in the middle, when the step size decreases, the learning function drops slower, but at the steady state the mean square error is smaller. An example of how the step size modifies the filter behaviour is given in Fig. 3.6, by using $b = 0.7$ and the other parameters as before.

The second experiment compares the OS-QFDAF and its *unconstrained* counterpart. The scenario is the same depicted in Fig. 3.4. As can be seen from Fig. 3.7, all parameters being equal ($b = 0.7, M = L = 32, N = 64, \mu_0 = 0.0005, \delta = 0.001, \lambda = 0.7$), the OS-QFDAF convergences to optimum with a lesser excess MSE in the learning curve, in comparison with the Unconstrained OS-QFDAF that, instead, shows some bias. However, the execution in the Unconstrained OS-QFDAF is faster. The Excess MSE is a gap between the steady state MSE curve and the theoretical MSE bound.

3.3.2 Evaluation of the Excess Mean-Square Error

As suggested by (3.67), the Excess MSE (EMSE), defined by (3.57), increases with the filter length M and/or the step size μ. An experimental proof of that was reported in Fig. 3.6 with varying μ_0. In the following test, we check the validity of (3.67). The OS-QFDAF algorithm was applied to the adaptive block in Fig. 3.4 and a comparison between the real J_∞ (learning curve value at the steady state) and J_∞(formula) computed by means of (3.66) is given in Fig. 3.8. The EMSE has been defined as $J_{EMSE\infty} = J_\infty - J_{MSE\ Bound}$. The experiment reveals that the value of J_∞(formula), computed with (3.66), approximates quite well the learning curve at the steady state. The algorithm parameters were chosen as $M = L = 32$, $N = 64$, with $\mu = \{0.001, 0.008\}$ (left side of Fig. 3.8) and $M = L = 64$, $N = 128$, $\mu = 0.005$ (right side of Fig. 3.8). In both cases we set $b = 0$.

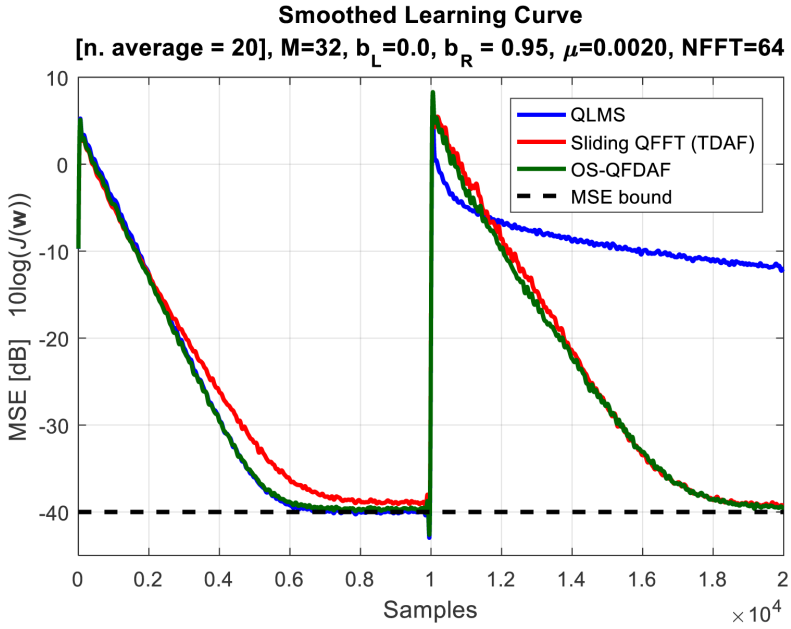


Figure 3.5. Power normalization effect on the filter performance.

3.3.3 Performance evaluation in changing scenario

Given the experiment in Fig. 3.4, the weights of the system to be identified are changed in value twice during the simulation. The filter response (in terms of adaptation) varies in speed accordingly with the amount of change. A greater change produces a slower adaptation, i.e. the algorithms need a higher number of iterations in order to reach the steady state after the alteration. The plots in Fig. 3.9 show results in terms of weights update. The tracking capability was tested by the OS-QFDAF algorithm with and without power normalization, by using the following parameters setting: $M = 4$, $L = 32$, $N = 64$, $b = 0.9$, $\mu_0 = 0.008$, $\lambda = 0.5$ and $\delta = 0.0005$.

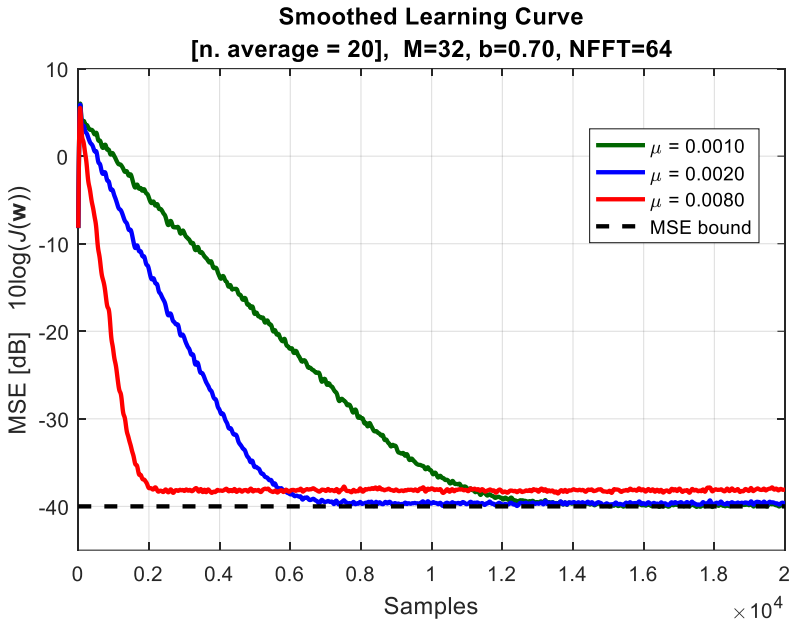


Figure 3.6. Step size effect on the filter performance.

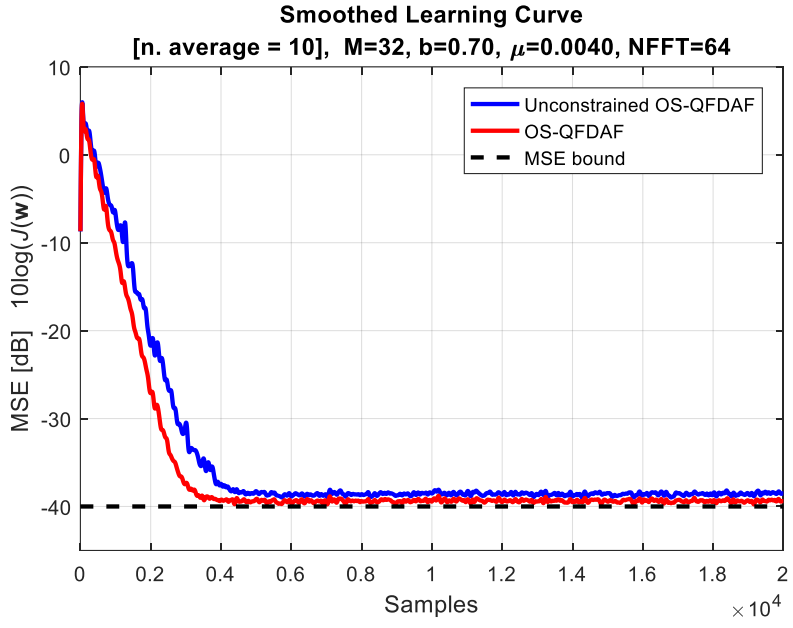


Figure 3.7. OS-QFDAF vs Unconstrained OS-QFDAF.

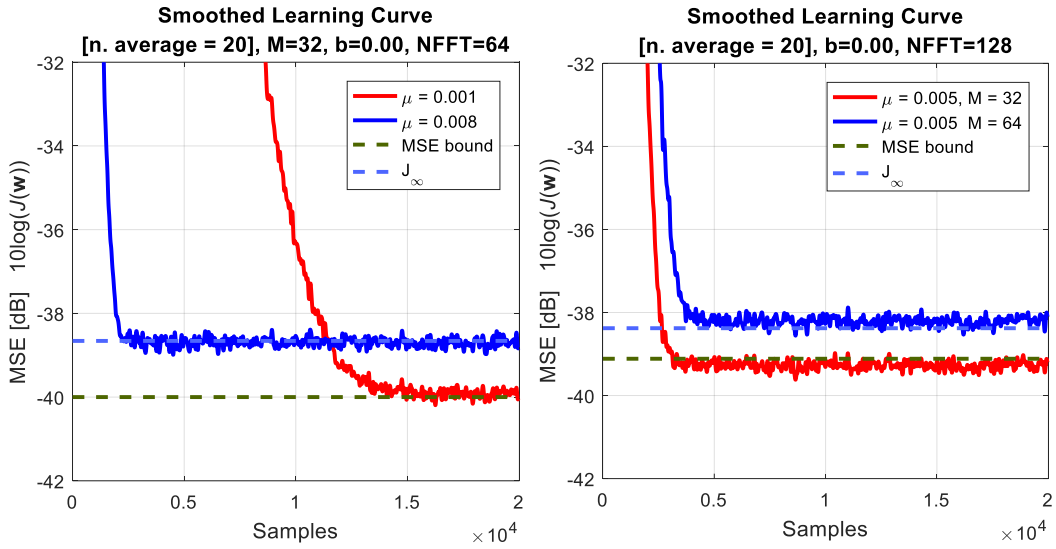


Figure 3.8. Excess MSE in OS-QFDAF.

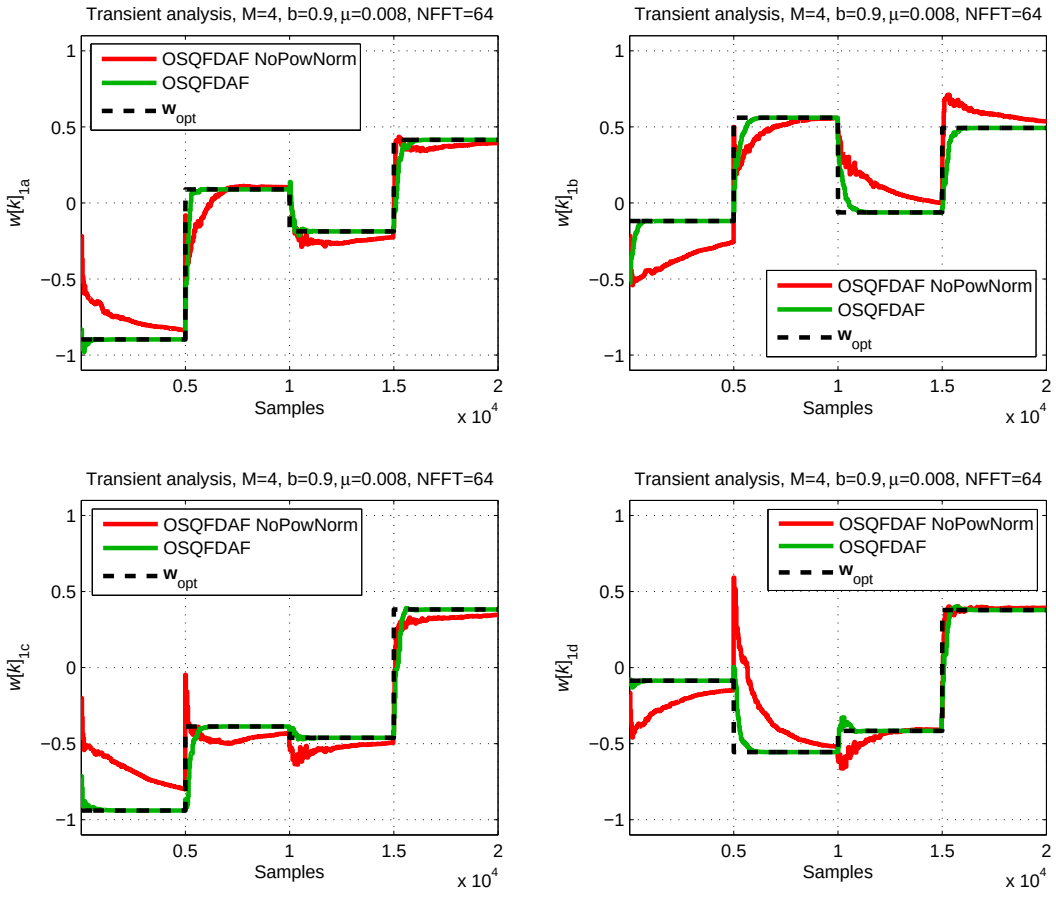


Figure 3.9. Tracking capability - Weights: OS-QFDAF, CC-QFDAF.

Appendix 3A

Algorithms

In this Appendix, modifications of the OS-QFDAF algorithm are presented. In the beginning, some vector definitions can be found in Table 3A.1.

Table 3A.1. Algorithm vector definitions (in the time domain)

$\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & \dots & x[n-M+1] \end{bmatrix}^T$	filter input vector at step n
$\mathbf{x}_k = \begin{bmatrix} x[kL] & x[kL-1] & \dots & x[kL-L+1] \end{bmatrix}^T$	filter input vector at step k
$\mathbf{y}_k = \begin{bmatrix} y[kL] & y[kL-1] & \dots & y[kL-L+1] \end{bmatrix}^T$	filter output
$\mathbf{e}_k = \begin{bmatrix} e[kL] & e[kL-1] & \dots & e[kL-L+1] \end{bmatrix}^T$	error signal
$\mathbf{w}_k = \begin{bmatrix} w_0[k] & w_1[k] & \dots & w_M[k] \end{bmatrix}^T$	filter weights
$n = kL + i$: sample index $k = 0, 1, 2, \dots$: block index $i = 0, 1, \dots, M-1$: internal block index	

3A.1 Block QLMS

Algorithm initialization: $\mathbf{w}_{init} = \textit{random values}$.
At each new input block k do:

$$y[n] = \mathbf{x}_n^T \mathbf{w}_k = \sum_{l=0}^{M-1} w_k[l]\, x[kL + i - l] \tag{3A.1}$$

$$e[n] = d[n] - y[n] \tag{3A.2}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + 2\frac{\mu_B}{L} \sum_{i=0}^{L-1} e[kL + i]\, \mathbf{x}_{kL+i}^* \tag{3A.3}$$

where μ_B is the block step size.

Alternatively, the Block QLMS algorithm can be expressed by defining the *block matrix* $\mathbf{X}_k$ by rows (R) and columns (C) as follows:

$$\begin{aligned} \mathbf{X}_k^R &= \begin{bmatrix} \mathbf{x}_{kL} & \mathbf{x}_{kL-1} & \dots & \mathbf{x}_{kL-L+1} \end{bmatrix}^T \\ \mathbf{X}_k^C &= \begin{bmatrix} \mathbf{x}[kL] & \mathbf{x}[kL-1] & \dots & \mathbf{x}[kL-M+1] \end{bmatrix}^T \end{aligned} \tag{3A.4}$$

where $\mathbf{x}_{kL} = \begin{bmatrix} x[kL] & x[kL-1] & \dots & x[kL-M+1] \end{bmatrix}^T$ and
$\mathbf{x}[kL] = \begin{bmatrix} x[kL] & x[kL-1] & \dots & x[kL-L+1] \end{bmatrix}^T$.

With the definition given in (3A.4), at each input block k do:

$$\mathbf{y}_k = \mathbf{X}_k \mathbf{w}_k \tag{3A.5}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + 2\frac{\mu_B}{L}\mathbf{X}_k^H \mathbf{e}_k. \tag{3A.6}$$

3A.2 Sliding Window Algorithms - Sliding QFFT-QLMS

The input block is pre-processed and decomposed into orthogonal components, each feeding its own adaptive filter in a parallel filter bank. Adaptation takes place in the transform domain.

Algorithm initialization:

$$\mathbf{W}_{init} = \mathbf{0},\ P_0(m) = \delta_m,\ \text{for } m = 0, 1, \dots, M-1. \tag{3A.7}$$

Let $L = 1$, at each new input sample for $n = 0, 1, \dots$ do:

$$\mathbf{X}_n = \text{diag}\{\text{QFFT}(\mathbf{x}_n)\} \tag{3A.8}$$

$$\mathbf{W}_n = \text{QFFT}(\mathbf{w}_n) \tag{3A.9}$$

$$\mathbf{Y}_n = \mathbf{W}_n^a \mathbf{X}_n + \mathbf{W}_n^b \boldsymbol{\nu}_\mathbf{2} \mathbf{X}_{-n} \tag{3A.10}$$

$$e(n) = d(n) - \mathbf{1}^T \mathbf{Y}_n \tag{3A.11}$$

$$\mathbf{E}_n = \mathbf{1}e(n) \tag{3A.12}$$

$$\mathbf{C}_k = E^a(\omega)\left[X_a^H(\omega) - \boldsymbol{\nu}_\mathbf{2} X_b(-\omega)\right] + E^b(\omega)\boldsymbol{\nu}_\mathbf{2}\left[X_a^H(-\omega) - \boldsymbol{\nu}_\mathbf{2} X_b(\omega)\right] \tag{3A.13}$$

$$\boldsymbol{\mu}_n = \mu \cdot diag\left\{P_n^{-1}(0), \dots, P_n^{-1}(M-1)\right\} \tag{3A.14}$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \boldsymbol{\mu}_k \mathbf{C}_n, \tag{3A.15}$$

where $e(n)$ is the error in the time domain, $\mathbf{E}_n$ the error in the frequency domain, and $\mathbf{1}$ is a column vector of ones.

The QFFT in the algorithm can be replaced by any other orthogonal unitary transform $\mathbf{F}$, i.e. $\mathbf{F}^{-1}\mathbf{F} = \mathbf{I}$. The Sliding Window algorithms can be considered as a boundary case of OS-QFDAF where $L = 1$.

Chapter 4

Quaternion Sound Space

Contents

The representation of 3-dimensional (3D) audio signals in a quaternion formalism is not merely a matter of compactness, but a bond between the physics and a proper mathematical expression. Quaternion spaces offer a powerful choice to represent the physics and its laws. In a simple way, the physical 3D space can be described as a quaternion structure, where $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ represent the space coordinate axes and the real quaternion component represents time. This formalism allows to simplify many physical problems. For instance, one of the most remarkable results is the simplification of the relativistic invariance and the reduction of Maxwell's equations to a single equation thanks to biquaternions (quaternions with all complex-valued components) [15, 33, 107].

There are other ways to obtain physics from quaternions. The method we propose here can be applied to 3D acoustics. A way to describe a sound field uses the combination of an orthonormal basis of functions. For instance, the 3D audio technique called Ambisonics decomposes the sound pressure field $p(\vec{r})$ into a set of *spherical harmonics* as described below. This 3D audio technique was introduced in the 70s by a team led by M. Gerzon, P. Fellgett and G. Barton [28–30] and it is renowned because of its versatility and ease to record and reproduce the reconstructed sound field. Besides that, Ambisonics describes the 3D sound space in a way that the transformation into a quaternion format is straightforward and physically consistent.

The description of the 3D sound space in a quaterion formalism is aimed at testing and integrating hypercomplex adaptive filtering into a 3D audio context. Immersive audio is indeed a useful test bench for the quaternion-valued filters that introduced in Chapter 3. The current chapter explains the Ambisonics technique and the way a transformation of the sound field into a quaternion format is feasible. In addition, we focus on the virtual miking potentialities of Ambisonics and the advantages in using quaternion signals in virtual rotations.

4.1 Introduction to Ambisonics

Source characterization is one of the most important tasks in sound field description. Some classic 3D audio techniques such as *Wave Field Synthesis* (WFS) or *Holophony*, represent the sound field by considering a wavefront as a secondary distribution of sources, namely *Huygens' principle*. According to source distribution, a specific spatial radiation pattern is created depending on the position and the distance of the sources. In mathematical terms, this is equivalent to saying that we can obtain the sound pressure on the area A, knowing the sound pressure p_0 and its gradient $\vec{\nabla} p_0$ on the boundary of A, by calculating the Kirchhoff-Helmholtz integral:

$$p\left(\overrightarrow{r}\right) = \iint_{\partial A} \left[\vec{\nabla} p_0 \cdot \overrightarrow{n} - \frac{\vec{R}}{R} \cdot \overrightarrow{n} \left(1 + jkR\right) \frac{p_0}{R} \right] \frac{e^{-jkR}}{4\pi R} dS_0, \qquad \forall \vec{r} \in A \tag{4.1}$$

where k is the wave number and $\vec{R}$ is the vector connecting the source with the listening point [61]. A detailed analysis of integral (4.1) shows how each secondary source is composed of a monopole (relative to the pressure gradient signal) and a dipole (relative to the pressure signal). However, there are slight conceptual differences in the formulations by Kirchhoff-Helmholtz and Huygens. The former is more general. The shape of the boundary does not depend on the wavefront, in addition the Kirchhoff-Helmholtz integral itself carries information relating to both amplitude and phase of the acoustic signal, whereas in Huygens' principle it is assumed that the secondary sources are located on equiphase surfaces. In practice, we can conclude that the Kirchhoff-Helmholtz integral generalizes Huygens' principle.

Another way to describe a sound field is based on the decomposition of the sound field into *spherical harmonics*. Ambisonics is founded on this second descriptive approach. Spherical harmonics are also used in issues concerning quantum mechanics, gravitational fields and can be found in 3D graphics applications and lighting engineering.

We start from writing the acoustic wave equation, i.e. the acoustic pressure, in spherical coordinates (r, θ, φ), where r is the radius, θ is the azimuth and φ is the

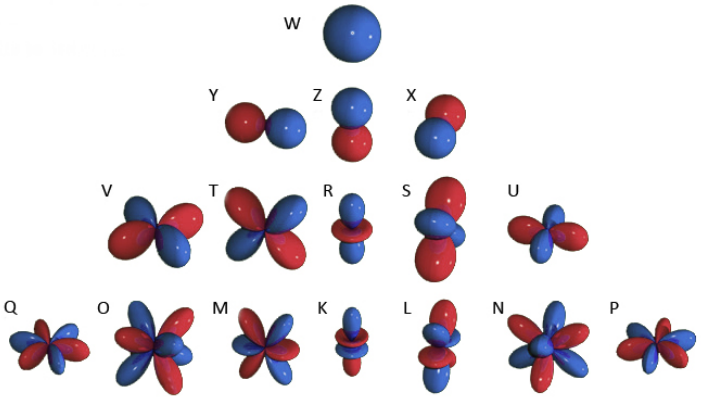


Figure 4.1. Spherical harmonics up to 3rd order.

elevation. In the time domain, the wave equation is

$$\nabla^2 p\left(r,\theta,\varphi,t\right) - \frac{1}{c^2}\frac{\partial^2 p\left(r,\theta,\varphi,t\right)}{\partial t^2} = 0 \tag{4.2}$$

where c is the speed of sound.

The acoustic pressure field, due to external sources, can be developed into a Fourier-Bessel series, whose terms are weighted products of the directional functions $Y_{mn}^{\sigma}\left(\theta,\varphi\right)$, called *spherical harmonics*, with the radial functions $J_m\left(kr\right)$, called *spherical Bessel functions of the first kind*:

$$p\left(\vec{r}\right) = \sum_{m=0}^{\infty}\left(2m+1\right)j^m J_m\left(kr\right)\sum_{0\leq n\leq m,\sigma=\pm 1} B_{mn}^{\sigma} Y_{mn}^{\sigma}\left(\theta,\varphi\right) \tag{4.3}$$

with definitions m for the *degree* and n for the *order*. The meaning of σ is the spin and k is the wave number ($k = 2\pi f/c$). Equation (4.3) represents the solution of the wave equation (4.2) in the special case of plane wave. The coefficients denoted by B_{mn}^{σ} represent, as shown later, the ambisonic signals in the transform domain [19] and behave like Fourier coefficients in a Fourier series. Note that, unlike WFS or Holophony, the sampling and the reconstruction of the sound field in Ambisonics are executed pointwise, rather than on an area. It follows that the number of channels needed to reconstruct the field will be much reduced compared to the other techniques mentioned above. The information relative to sound direction is coded precisely into the coefficients B_{mn}^{σ} just introduced. Ambisonics produces – in theory – a coherent and homogeneous reconstruction of the field for all frequencies and directions in the *sweet spot*, the optimal listening point. We will see that the area affected by a problem of incoherence gets smaller with increasing the order of Ambisonics. Similarly, there exists a frequency limit, beyond which the error exceeds a certain level that grows with the order. In other words, Ambisonics performs well in terms of coherence and homogeneity only in the sweet spot and only for low frequencies.

Let us see the spherical harmonic functions in detail, analysing how ambisonic signals are obtained from these functions. Spherical harmonics (see Fig. 4.1) are defined as

$$\begin{aligned} Y_{mn}^{\sigma}\left(\theta,\varphi\right) = \sqrt{2m+1}\sqrt{\left(2-\delta_{0,n}\right)\frac{(m-n)!}{(m+n)!}}P_{mn}\sin\varphi \\ \times\begin{cases}\cos n\theta & \text{if } \sigma = +1\\ \sin n\theta & \text{if } \sigma = -1 \text{ (ignore } n=0)\end{cases} \end{aligned} \tag{4.4}$$

where $P_{mn}\left(\xi\right)$ is the associated Legendre function of degree m and order n, δ_{pq} represents Kronecker delta and it is equal to 1 if $p = q$, else it is equal to 0. The associated Legendre function is defined as

$$\begin{aligned} P_{mn}\left(\xi\right) &= \left(1-\xi^2\right)^{\frac{n}{2}}\frac{d^n}{d\xi^n}P_m\left(\xi\right) \\ &= \frac{(-1)^m}{2^m m!}\left(1-\xi^2\right)^{\frac{n}{2}}\frac{d^{m+n}}{d\xi^{m+n}}\left(1-\xi^2\right)^m \end{aligned} \tag{4.5}$$

where $\xi = \cos\varphi$. In Ambisonics, some kind of normalization of the Legendre functions often takes place [18]. For example, Schmidt semi-normalization is defined by

$$\begin{cases} N_{mn} = \sqrt{2m+1}\sqrt{(2-\delta_{0,n})\,\frac{(m-n)!}{(m+n)!}} = \sqrt{e_n}\sqrt{\frac{(m-n)!}{(m+n)!}} \\ e_0 = 1 \text{ if } n = 0 \\ e_n = 2 \text{ if } n \geq 1 \end{cases} \tag{4.6}$$

The harmonic functions can be rewritten in *Schmidt* semi-normalized form (SN3D) by substituting (4.6) into (4.4):

$$Y_{mn}^{\sigma}(\theta,\varphi) = \tilde{P}_{mn}\sin\varphi \times \begin{cases} \cos n\theta & \text{if } \sigma = +1 \\ \sin n\theta & \text{if } \sigma = -1 \text{ (ignore } n = 0) \end{cases} \tag{4.7}$$

The set of spherical harmonics forms an orthonormal basis in the sense of the spherical scalar product. So, spherical harmonics can be linearly combined in order to define functions on the surface of a sphere. Actually, Ambisonics is not limited to a particular number of channels. That said, as can be immediately seen from the 3D illustration of the spherical harmonics (Fig. 4.1), in order to achieve a higher directional resolution, the order of Ambisonics must increase. In other words, a greater number of channels provides a higher directional resolution. Because of manageability, (4.3) has to be arrested to a certain order M, also known as *order of Ambisonics*. So as to avoid confusion, it should be noted that the order M of Ambisonics is different from the order n defined in Legendre functions. We can rather say that it refers to the ambisonic order in terms of degree m in Legendre functions.

4.2 Ambisonic format overview

The Ambisonic technique comes with several formats for microphone recording, broadcasting and reproduction of recorded signals [65, 81]. In addition, Ambisonics is compatible with a wide variety of speaker array configurations and can coexist with stereo and surround sound systems such as 5.1 or 7.1, etc.

- A-Format: suitable for miking with specific microphone (e.g. Soundfield mic);
- B-Format: suitable for miking and processing with studio equipment;
- C-Format/UHJ: suitable for mono, stereo, 3-channel systems and broadcasting;
- D-Format: suitable for decoding and playback through array of speakers;
- G-Format: alike D-Format, but the decoder is not required;

4.2.1 B-Format

If the expansion in (4.3) is arrested to degree $m = 1$, the sound space build-up includes only spherical harmonics of order zero (W) and one (X,Y,Z) and four microphones are required to pick up the 3D sound field. Accordingly, the 1st-order B-Format technique is defined by four signals relative to the pressure component of

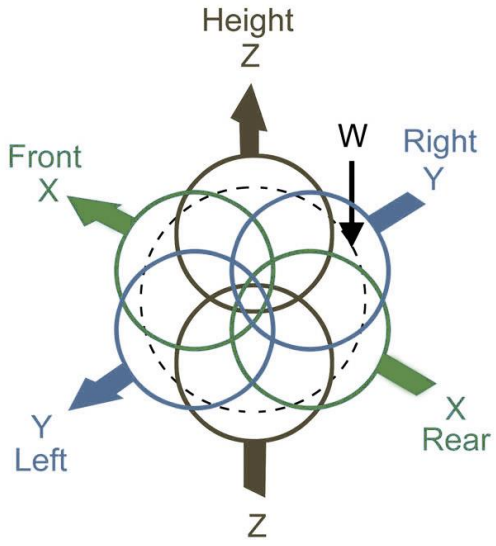


Figure 4.2. B-Format: microphone orientation.

the sound field in all directions (omnidirectional microphone, W) and the horizontal and vertical components (figure-of-eight microphones, X,Y,Z) of velocity. The four microphones are coincident and orthogonal to one another. Their capsules are pointed as shown in Fig. 4.2. The Y microphone is rotated by 90° (leftwards) with respect to X. Microphone Z is oriented along the orthogonal plane with respect to the plane described by the axes X and Y (the 0° axis points upwards). However, once the B-Format signals are recorded, it is possible to rotate the microphone array virtually, by means of a rotation matrix or a quaternion rotation operator, as explained in Par. 4A.1.

The four B-Format polar patterns are obtained from (4.3) and they are expressed as (in a normalized form)

$$\begin{cases} W = S \\ X = S\sqrt{2}\cos\theta\cos\varphi \\ Y = S\sqrt{2}\sin\theta\cos\varphi \\ Z = S\sqrt{2}\sin\varphi \end{cases} \tag{4.8}$$

where S is the recorded source [81]. The reader is referred to [65] for further explanations about the normalization factor $\sqrt{2}$.

See Appendix 4A for information about microphone polar patterns and functioning principles.

4.2.2 Extension of B-Format to quaternions

The use of microphones of the family of cardioids simplifies (4.3) in a way that, separating spatial dependence from frequency dependence, we can write the pressure

Table 4.1. Analogy among structures: euclidean vectors, Fourier transforms, spherical harmonics [36]

	Euclidean Vectors	**Fourier Transforms**	**Spherical Harmonics**
Group	*Orthogonal*	*Translations*	*3D Rotations*
Objects	$\mathbf{x}$	$f(x)$	$f(\theta,\varphi)$
Basis	$\hat{\mathbf{e}}_i$	e^{ikx}	$Y_{mn}(\theta,\varphi)$
Projection	*Dot product*	$\int_{-\infty}^{+\infty} dx$	$\int_0^{2\pi} d\varphi \int_{-1}^{+1} d\cos\theta$
Coefficients	x_i	$F(k)$	a_{mn}

field in a more readable form [18]:

$$p(\theta) = \sum_{m=0}^{\infty} W_m(\omega) \sum_{0\le n\le m,\sigma=\pm 1} B_{mn}^{\sigma} Y_{mn}^{\sigma}(\theta) \tag{4.9}$$

where $W_m(\omega)$ is a weighting factor defined as

$$W_m(\omega) = j^m(\alpha J_m(kr_{mic})) - j(1-\alpha) J'_m(kr_{mic}). \tag{4.10}$$

Equation (4.10) highlights how the recording field is dependent on the frequency (in fact, k is the wave number $k = 2\pi/\lambda$, where λ is the wavelength $\lambda = c/f$, c is the speed of sound and f is the frequency). Basically, when miking a source, besides considering source directivity, we must consider the microphone polar characteristic with respect to the frequency. Equation (4.10) was obtained by weighting (4.3) with a cardioid characteristic function of the kind $G(\theta) = \alpha + (1-\alpha)\cos\theta$. In effect, a cardioid microphone is generated by the superimposition of an omnidirectional microphone (responsive to the pressure) and a figure-of-8 microphone (responsive to the pressure gradient and so, to the derivative of the pressure). In light of the above, the ambisonic signals, recorded by means of microphones from the family of cardioids, correspond to the coefficients B_{mn}^{σ} weighted by $W_m(\omega)$, which depends on the frequency.

At this point, we merely need to change from Euler to quaternionic representation. The transformation of the spherical harmonics is solely sufficient in order to redefine the sound field in the new form. Table 4.1 reports a brilliant parallel comparison among structures founded respectively on Euclidean vectors, Fourier transforms and spherical harmonics. This table artfully explains the matter behind formulas. The analogy suggested in the table is

$$\begin{cases} \mathbf{x}\cdot\hat{\mathbf{e}}_i = x_i \\ f(x) = \int_{-\infty}^{+\infty} \underbrace{F(k)}_{coeff} \underbrace{e^{2\pi ikx}}_{basis} dk \\ f(\theta,\varphi) = \sum_{m=0}^{\infty} \sum_{n=-m}^{+m} a_{mn} Y_{mn}(\theta,\varphi) \end{cases} \tag{4.11}$$

Table 4.2. Spherical Harmonics: Euler to Quaternion

Order	m,n,σ	Ch.	Euler Spherical Harmonics (SN3D)	Quaternion Spherical Harmonics
0	0,0,1	W	1	$1\mathbf{e}$
1	1,1,1	X	$\cos\theta\cos\varphi$	$x\mathbf{i}$
1	1,0,1	Z	$\sin\varphi$	$z\mathbf{k}$
1	1,1,-1	Y	$\sin\theta\cos\varphi$	$y\mathbf{j}$
2	2,2,1	U	$\left(\sqrt{3}/2\right)\cos\left(2\theta\right)\cos^2\varphi$	$\left(\sqrt{3}/2\right)\left(x^2-y^2\right)\mathbf{e}$
2	2,1,1	S	$\left(\sqrt{3}/2\right)\cos\theta\sin\left(2\varphi\right)$	$\sqrt{3}xz\mathbf{j}$
2	2,0,1	R	$\left(3\cos^2\varphi-1\right)/2$	$\left(3z^2-1\right)\mathbf{e}$
2	2,1,-1	T	$\left(\sqrt{3}/2\right)\sin\theta\sin\left(2\varphi\right)$	$\sqrt{3}yz\mathbf{i}$
2	2,2,-1	V	$\left(\sqrt{3}/2\right)\sin\left(2\theta\right)\cos^2\varphi$	$\left(\sqrt{3}/2\right)xy\mathbf{k}$

Ambisonics uses the set of real spherical harmonics. In order to get a quaternionic form for real spherical harmonics, the Euler harmonics have to be converted into Cartesian coordinates and each Cartesian axis will be assigned to a quaternionic versor ($\mathbf{e}$,$\mathbf{i}$,$\mathbf{j}$,$\mathbf{k}$), where the real axis versor has been called $\mathbf{e}$ as a matter of notation homogeneity. The transformation up to 2nd order is tabulated in Table 4.2. Note that, in the original Ambisonics, angles go clockwise and the Cartesian coordinates are defined as

$$\begin{cases} x = \cos\theta\cos\varphi \\ y = \sin\theta\cos\varphi \\ z = \sin\varphi \end{cases} \tag{4.12}$$

For instance, the B-Format (up to 1st order) can be rewritten in quaternionic form:

$$\begin{cases} Y_{0,0}^{1Q} = Y_{0,0}^{1}\left(\theta,\varphi\right)\mathbf{e} \\ Y_{1,1}^{1Q} = Y_{1,1}^{1}\left(\theta,\varphi\right)\mathbf{i} \\ Y_{1,1}^{-1Q} = Y_{1,1}^{-1}\left(\theta,\varphi\right)\mathbf{j} \\ Y_{1,0}^{1Q} = Y_{1,0}^{1}\left(\theta,\varphi\right)\mathbf{k} \end{cases} \tag{4.13}$$

Now the sound pressure field can be expressed with the new quaternionic spherical harmonics in (4.13):

$$\begin{aligned} p\left(\theta\right) &= \sum_{m=0}^{1} W_m\left(\omega\right) \sum_{0\le n\le 1,\sigma=\pm 1} B_{mn}^{\sigma} Y_{mn}^{\sigma}\left(\theta\right) \\ &= \sum_{m=0}^{1} W_m\left(\omega\right)\left(B_W Y_{0,0}^{1Q}\mathbf{e} - B_X Y_{1,1}^{1Q}\mathbf{i} - B_Y Y_{1,1}^{-1Q}\mathbf{j} - B_Z Y_{1,0}^{1Q}\mathbf{k}\right) \end{aligned} . \tag{4.14}$$

Since the four spherical harmonics ($Y_{0,0}^{1Q}$, $Y_{1,1}^{1Q}$, $Y_{1,1}^{-1Q}$, $Y_{1,0}^{1Q}$) are othogonal, we can process the four B-Format coefficients packed into a single quaternion function:

$$B[n] = B_W[n] + B_X[n]\mathbf{i} + B_Y[n]\mathbf{j} + B_Z[n]\mathbf{k}. \tag{4.15}$$

The usage of a quaternion Ambisonic representation was widely experimented by the author of this thesis. Precisely, the author commented the possibility of conditioning the performance of adaptive filters by choosing a certain algebraic format for signals and systems. Ambisonics and 3D sound have been extensively employed as a testbed with this aim [66, 69, 71].

Appendix 4A

Virtual Miking and Rotations

4A.1 Virtual Miking and Rotations

One of the advantages in exploiting a quaternionic formulation is the possibility to achieve very efficient rotations. The Ambisonic technique permits to rotate the microphone array electrically (*virtually*) without the need to rotate the array mechanically. The possibile rotations are about the axes **x**, **y**, **z**. Virtual rotations and zooming can be attained by operating matrix transformations of the ambisonic signals.

The microphone rotation by an angle θ about its vertical axis **z** is called *yaw/rotate* and corresponds to the control of the azimuth angle. The following transformations are applied to the ambisonic signals up to 1st order:

$$\mathbf{B}' = \mathbf{R}_z(\theta)\,\mathbf{B} \rightarrow \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{4A.1}$$

The microphone rotation by an angle γ about its horizontal axis **x** is called *roll/tilt*:

$$\mathbf{B}' = \mathbf{R}_x(\gamma)\,\mathbf{B} \rightarrow \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{4A.2}$$

The microphone rotation by an angle φ about its horizontal axis **y** is called *pitch/tumble* and corresponds to the control of the elevation angle:

$$\mathbf{B}' = \mathbf{R}_y(\varphi)\,\mathbf{B} \rightarrow \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{4A.3}$$

Euler's theorem asserts that every elementary rotation (*principal rotation*) can be represented as a rotation by an angle α about an axis **n**. In general, rigid body motion has 6 degrees of freedom (3 degrees describe a translation and 3 degrees describe a rotation). So, only 3 scalar numbers are sufficient to specify rotations. The rotation axis can be represented by 3 scalar components $\mathbf{n} = [n_1, n_2, n_3]^T$ with the constraint $\|\mathbf{n}\| = 1$. Unfortunately, this representation has no unique solution. In fact, the axis-angle pairs $(\mathbf{n}, \alpha)$ and $(-\mathbf{n}, -\alpha)$ identify the same rotation. Furthermore, when $\alpha = 0$, **n** may be any axis, represented by any versor (*unitary*

vector). For example, in a 3D Euclidean space, an orthonormal basis describing the 3 axes can be expressed by the 3-tuples: $\mathbf{x} = [1, 0, 0,]^T, \mathbf{y} = [0, 1, 0,]^T, \mathbf{z} = [0, 0, 1]^T$. Euler angles specify a rotation by operating 3 consecutive principal rotations. In (4A.1), (4A.2) and (4A.3) this operation was described by the rotation matrices $\mathbf{R}_z(\theta), \mathbf{R}_x(\gamma), \mathbf{R}_y(\varphi)$. There exist 24 distinct ways to combine a sequence of 3 principal axes rotations: 12 of them follow the right-hand rule and 12 follow the left-hand rule. In conclusion, we have 24 different Euler angle conventions. The one we generally use in ambisonic virtual miking is the *zyx Tait-Bryan* convention (or *Cardano nautical angles*), most commonly used in aero-navigation. See also [9]. In Ambisonics, the Tait-Bryan angles are denoted by:

$$\begin{cases} \theta = \text{yaw, azimuth} \\ \varphi = \text{pitch, elevation} \\ \gamma = \text{bank, roll} \end{cases} \tag{4A.4}$$

So, why should we represent rotations with quaternions? Given the coordinates and a gyroscope mounted in 3 nested gimbals, we define two coordinate systems:

- $(\mathbf{x}, \mathbf{y}, \mathbf{z})$: space-fixed coordinate system,
- $(\mathbf{x}', \mathbf{y}', \mathbf{z}')$: body-fixed coordinate system.

Rotations in a *zyx* rotation convention can be operated if the *xy*-plane and the *y'z'*-plane intersect in a line of nodes. This can be achieved if $\varphi \neq \pm\frac{\pi}{2} + 2k\pi$, with $k = 0, 1, 2...$ On the contrary, when $\varphi = \pm\frac{\pi}{2} + 2k\pi$, the $\mathbf{x}'$ axis of the body-fixed reference system and the $\mathbf{z}$ axis of the space-fixed reference system become parallel (the *xy*-plane and the *y'z'*-plane become identical). In such a situation, a degree of freedom is lost and the 3D system degenerates into a 2D system [9]. Consequently, the sequence of rotations collapse down to one single principal rotation defined by:

$$\begin{aligned} \varphi = +\frac{\pi}{2}: \quad & \mathbf{R}_z(\theta)\,\mathbf{R}_y\left(+\frac{\pi}{2}\right)\mathbf{R}_x(\gamma) = \mathbf{R}_z(\theta+\gamma) \\ \varphi = -\frac{\pi}{2}: \quad & \mathbf{R}_z(\theta)\,\mathbf{R}_y\left(-\frac{\pi}{2}\right)\mathbf{R}_x(\gamma) = \mathbf{R}_z(\theta-\gamma) \end{aligned} \tag{4A.5}$$

At such critical angles $\varphi \neq \pm\frac{\pi}{2}$, roll rotations cannot be distinguished from yaw rotations. In other words, the individual Euler angles $(\theta, \varphi, \gamma)$ are no longer uniquely determined and only the sum/difference $(\theta \pm \gamma)$ can be uniquely defined (see Appendix 4A.2). So, when the moving device (e.g. an aircraft or the ambisonic microphone array) either climbs or dives vertically, its rotation system gets stuck. This phenomenon is called *Gimbal Lock*. The same occurs on the other planes. When the gimbals are parallel and lie in a single plane, the rotation within that plane is locked. This trouble can be avoided if quaternions are used in place of Euler angles.

A quaternion can be expressed in three equivalent forms:

$$\begin{aligned} q = q_a + q_b\mathbf{i} + q_c\mathbf{j} + q_d\mathbf{k} \quad & \text{HAMILTON FORM} \\ q = (q_a, \mathbf{q}) \quad & \text{SCALAR/VECTOR FORM} \\ q = (q_a, q_b, q_c, q_d) \quad & 4-\text{TUPLE FORM} \end{aligned}$$

In general, we can represent a rotation by the angle α, about the axis $\mathbf{n}$, in the compact quaternionic scalar/vector form:

$$q = \left(\cos\left(\frac{\alpha}{2}\right), \sin\left(\frac{\alpha}{2}\right)\mathbf{n}\right). \tag{4A.6}$$

With such a notation, the principal rotations in Tait-Bryan convention are transformed into the following quaternion operators:

$$\begin{aligned}
\mathbf{R}_z(\theta) \rightarrow q_z(\theta) &= \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\mathbf{z}\right) \\
\mathbf{R}_y(\varphi) \rightarrow q_y(\varphi) &= \left(\cos\left(\frac{\varphi}{2}\right), \sin\left(\frac{\varphi}{2}\right)\mathbf{y}\right) \\
\mathbf{R}_x(\gamma) \rightarrow q_x(\gamma) &= \left(\cos\left(\frac{\gamma}{2}\right), \sin\left(\frac{\gamma}{2}\right)\mathbf{x}\right)
\end{aligned} \tag{4A.7}$$

The consecutive rotation operators are multiplied and the definitive rotation quaternion results in

$$q_{zyx} = q_z(\theta)q_y(\varphi)q_x(\gamma). \tag{4A.8}$$

Substituting (4A.7) into (4A.8), we have:

$$q_{zyx} = (\cos(\theta/2), \sin(\theta/2)\,\mathbf{z}) \cdot (\cos(\varphi/2), \sin(\varphi/2)\,\mathbf{y}) \cdot (\cos(\gamma/2), \sin(\gamma/2)\,\mathbf{x}) \tag{4A.9}$$

which can be decomposed into the 4-tuple:

$$\begin{cases}
q_a = \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\gamma}{2}\right) + \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\gamma}{2}\right) \\
q_b = \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\gamma}{2}\right) - \sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\gamma}{2}\right) \\
q_c = \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\gamma}{2}\right) + \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\gamma}{2}\right) \\
q_d = \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\gamma}{2}\right) - \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\gamma}{2}\right)
\end{cases} \tag{4A.10}$$

The relations to change back from quaternions to Euler angles are given in [9]. The set of equations in the Tait-Bryan form are reported below:

$$\begin{cases}
\gamma = \arctan 2\left[(q_c q_d + q_a q_b), \frac{1}{2} - (q_b^2 + q_c^2)\right] \\
\varphi = \arcsin\left[-2(q_b q_d - q_a q_c)\right], \qquad p.v. \\
\theta = \arctan 2\left[(q_b q_c + q_a q_d), \frac{1}{2} - (q_c^2 + q_d^2)\right]
\end{cases} \tag{4A.11}$$

where the principal value *p.v.* is in the range $-\pi/2 < \varphi < \pi/2$.
Recalling what we said above, if $\mathbf{n}$ is a unit pure quaternion (that is, $\mathbf{n}$ meets the constraint $\|\mathbf{n}\| = 1$), any 3-dimensional rotation about the axis $\mathbf{n}$ can be obtained. For instance, given

$$v = [a, \mathbf{r}] \in \mathbb{H} \tag{4A.12}$$

with $\mathbf{r} = (x, y, z) \in \mathbb{R}^3$, and chosen a quaternion q (rotation operator) [17], e.g.

$$q = \left(\cos\left(\frac{\alpha}{2}\right), \sin\left(\frac{\alpha}{2}\right)\mathbf{n}\right) \tag{4A.13}$$

we can rotate v by the operator q and get

$$w = qvq^* \tag{4A.14}$$

where w is v rotated by the operator q.
Intuitively, we can see that, if $q = [s, \mathbf{v}] \in \mathbb{H}$, then $q^* = [s, -\mathbf{v}] \in \mathbb{H}$ and

$$q^* = [s, -\mathbf{v}] = [s, \mathbf{v}]^{-1} = q^{-1}. \tag{4A.15}$$

So, a rotation can be reversed by inverting the axis of rotation.

Rotations with quaternions are handy indeed. In Euler notation, subsequent rotations would require matrix products. Computations with quaternions are much quicker and notation is easier. Let $q_1, q_2 \in \mathbb{H}$ be two quaternion rotation operators and a rotation by q_1 is followed by a rotation by q_2. The resulting rotation is equivalent to a rotation by $q_2 q_1$. So, successive rotations can be expressed as one. Remember that quaternion product is not commutative, so the order of the factors matters. Another advantage in using quaternions is that rotations do not depend on the coordinate system. Rotations are not influenced by the conventional order of rotation about the axes. In addition it is possible to implement simple interpolation methods using quaternions. This makes the use of quaternions the best choice in animations, virtual reality applications, augmented reality and so on.
Do quaternions beat out the traditional Euler matrix notation on all points? Quaternions undoubtedly excel with rotations, but Euler matrices allow representing all the other transformations easily: translation, scaling, shearing (i.e. sliding deformation of parallel surfaces past one another), projections, etc.
For what concerns translation, it can be implemented using quaternions as well. This kind of transformation takes a point P to the new position P' by a simple addition: $P' = (x + \Delta x, y + \Delta y, z + \Delta z)$.

4A.1.1 Interpolation with Quaternions - Linear Quaternion Interpolation (LERP)

Applications like virtual reality and 3D audio require some kind of automation to rotate the sound sources around the listener. Although, as a matter of fact, it is not possible to obtain a continuous interpolation line because of the digital nature of processing, it is possible to get high resolution lines and different shapes by means of different interpolation methods. We only give an example to see how interpolations of rotations work. Two comprehensive readings about quaternions, rotations and interpolations can be found in [17] and [86].

Let $q_0, q_1 \in \mathbb{H}$. We operate a linear interpolation between q_0 and q_1. The interpolation curve is parameterized by $h \in [0, 1]$ and meets the constraints:

$$\begin{cases} \Gamma : \mathbb{H} \times \mathbb{H} \times [0,1] \to \mathbb{H} \\ \Gamma(q_0, q_1, 0) = q_0 \\ \Gamma(q_0, q_1, 1) = q_1 \end{cases} \tag{4A.16}$$

We define the linear interpolation curve (LERP) as follows:

$$l = \mathrm{Lerp}(q_0, q_1, h) = q_0(1 - h) + q_1 h \tag{4A.17}$$

As suggested by the name, the linear interpolation curve is a straight line in the quaternion space $\mathbb{H}$. As usual, the final rotation of a quaternion q by the quaternion operator l will be

$$w = lvl^*. \tag{4A.18}$$

4A.1.2 Computational Cost

Working with quaternions has a remarkable impact on the computational effort reduction. If Euler matrices are used in order to achieve a full 3D rotation, as in

$$\mathbf{R}_{zyx} = \mathbf{R}_z(\theta)\,\mathbf{R}_y(\varphi)\,\mathbf{R}_x(\gamma) \tag{4A.19}$$

where matrices $\mathbf{R}_z, \mathbf{R}_y, \mathbf{R}_x \in \mathbb{R}^{3\times 3}$ are defined as (4A.1),(4A.2),(4A.3), 54 multiplications are required:

$$\mathbf{R}_{zyx} = \overbrace{\underbrace{\mathbf{R}_z(\theta)\,\mathbf{R}_y(\varphi)}_{3\times3\times3\text{ mult.}}\mathbf{R}_x(\gamma)}^{3\times3\times3\text{ mult.}} \rightarrow 54 \text{ multiplications.} \tag{4A.20}$$

By contrast, if quaternion operators are used in place of Euler matrices, as in

$$q_{zyx} = q_z(\theta)q_y(\varphi)q_x(\gamma) \tag{4A.21}$$

only 32 multiplications are needed:

$$q_{zyx} = \overbrace{\underbrace{q_z(\theta)q_y(\varphi)}_{4\times4\ mult.}q_x(\gamma)}^{4\times4\ mult.} \rightarrow 32 \text{ multiplications.} \tag{4A.22}$$

Effectively, the elementary rotations $q_z(\theta)$, $q_y(\varphi)$, $q_x(\gamma)$ have two of three vanishing imaginary components, so the the number of multiplications required is reduced to 16, as can be seen from (4A.10).
When the final rotation is operated, matrix algebra requires only $9\times N$ multiplications, whereas quaternions involve $32 \times N$ real-valued multiplications:

$$v_{zyx} = \overbrace{\underbrace{q_{zyx}v}_{4\times4\ mult.}\ q^*_{zyx}}^{4\times4\ mult.} \rightarrow 32 \text{ multiplications} \tag{4A.23}$$

where N is the signal length. However, because of (4A.20), quaternion rotations proved to be more efficient from the computational point of view.

4A.2 Gimbal Lock System Degeneration

Given the rotation matrices in (4A.1), (4A.2), (4A.3), the overall rotation matrix is obtained as

$$\begin{aligned}\mathbf{R}_{zyx} &= \begin{bmatrix} \cos\theta\cos\varphi & -\sin\theta\cos\gamma+\cos\theta\sin\gamma\sin\varphi & \sin\theta\sin\gamma+\cos\theta\cos\gamma\sin\varphi \\ \sin\theta\cos\varphi & \cos\theta\cos\gamma+\sin\theta\sin\gamma\sin\varphi & -\cos\theta\sin\gamma+\sin\theta\cos\gamma\sin\varphi \\ -\sin\varphi & \sin\gamma\cos\varphi & \cos\gamma\cos\varphi \end{bmatrix} \\ &= \mathbf{R}_z(\theta)\mathbf{R}_y(\varphi)\mathbf{R}_x(\gamma).\end{aligned} \tag{4A.24}$$

If $\varphi = \pm\pi/2$, the rotation matrix (4A.24) becomes

$$\begin{aligned}\mathbf{R}_{zyx} &= \begin{bmatrix} 0 & -(\sin\theta\cos\gamma \mp \cos\theta\sin\gamma) & (\cos\theta\cos\gamma \pm \sin\theta\sin\gamma) \\ 0 & (\cos\theta\cos\gamma \pm \sin\theta\sin\gamma) & -(\sin\theta\cos\gamma \mp \cos\theta\sin\gamma) \\ \mp1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -\sin(\theta\mp\gamma) & \cos(\theta\mp\gamma) \\ 0 & \cos(\theta\mp\gamma) & -\sin(\theta\mp\gamma) \\ \mp1 & 0 & 0 \end{bmatrix}.\end{aligned} \tag{4A.25}$$

Matrix (4A.25) is equivalent to $\mathbf{R}_z(\theta \mp \gamma)$:

$$\mathbf{R}_z(\theta \mp \gamma) = \begin{bmatrix} \cos(\theta \mp \gamma) & -\sin(\theta \mp \gamma) & 0 \\ -\sin(\theta \mp \gamma) & \cos(\theta \mp \gamma) & 0 \\ 0 & 0 & \mp 1 \end{bmatrix} \tag{4A.26}$$

since it is obtained from (4A.25) by operations such as swapping columns and changing the sign of a column. From (4A.10) we can see that if $\varphi = \pm\pi/2$ a quaternion operator is always uniquely defined:

$$\begin{cases} q_a = \frac{\sqrt{2}}{2}\left[\cos(\frac{\theta}{2})\cos(\frac{\gamma}{2}) \pm \sin(\frac{\theta}{2})\sin(\frac{\gamma}{2})\right] \\ q_b = \frac{\sqrt{2}}{2}\left[\cos(\frac{\theta}{2})\sin(\frac{\gamma}{2}) \mp \sin(\frac{\theta}{2})\cos(\frac{\gamma}{2})\right] \\ q_c = \frac{\sqrt{2}}{2}\left[\sin(\frac{\theta}{2})\sin(\frac{\gamma}{2}) \pm \cos(\frac{\theta}{2})\cos(\frac{\gamma}{2})\right] \\ q_d = \frac{\sqrt{2}}{2}\left[\sin(\frac{\theta}{2})\cos(\frac{\gamma}{2}) \mp \cos(\frac{\theta}{2})\sin(\frac{\gamma}{2})\right] \end{cases} \tag{4A.27}$$

Appendix 4A

Microphone characteristics

4A.1 Polar pattern

The polar pattern of a microphone is a directionality curve describing the sensitivity of the microphone with respect to the sound angle of incidence θ. The polar pattern is drawn by reproducing a sound of the same signal level at a certain distance from the microphone at different angles θ. The microphone responds differently according to the angle of incidence.

We are primarily interested in the characteristics of the microphones of the *family of cardioids*, especially of *1st order* cardioids. The family of cardioids includes the range of microphones from *omnidirectional* to *figure-of-8*. Table 4A.1 below shows the polar patterns of these microphones and the equations describing them.
Cardioid polar characteristics can be achieved in different ways:

- Superimposition of a figure-of-8 and an omnidirectional mic.
- Microphone composed of a part of the diaphragm having only the front side exposed to the sound field and another part having both sides exposed to the field.
- It is possible to build a microphone in which the sound gets to its rear side passing through a delay element.

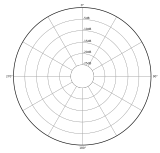
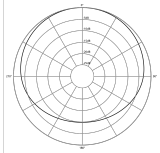
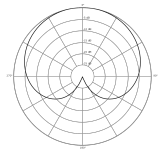
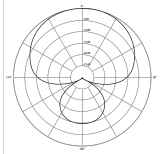
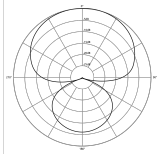
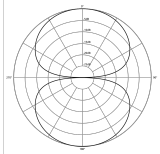
4A.2 Pressure microphones and pressure gradient microphones

We conclude this chapter with some useful smattering about the microphones that are commonly used in ambisonic arrays [10].

4A.2.1 Pressure Microphones

These microphones show only the front face to the sound field and respond in the same way to changes in the acoustic pressure for all the directions of the incident sound. In effect, pressure microphones have no directional characteristic and they are also known as *omnidirectional* microphones. Actually, the microphone body

Table 4A.1. Polar diagrams and equations for the microphone from the family of cardioids

POLAR PATTERN	MICROPHONE TYPE	EQUATION
	Family of cardioids (general equation)	$\rho(\theta) = \alpha + (1-\alpha)\cos\theta$
	OMNIDIRECTIONAL	$\rho(\theta) = 1$
	SUB-CARDIOID	$\rho(\theta) = 0.7 + 0.3\cos\theta$
	CARDIOID	$\rho(\theta) = 0.5 + 0.5\cos\theta$
	SUPERCARDIOID	$\rho(\theta) = 0.37 + 0.63\cos\theta$
	HYPERCARDIOID	$\rho(\theta) = 0.25 + 0.75\cos\theta$
	FIGURE-OF-8	$\rho(\theta) = \cos\theta$

will cause its response to tend to become directional with increasing frequency, because its size becomes comparable with the wavelength of the incident sound on the diaphragm (this is true, in general, for all microphones, which effectively tend to become hypercardioid with increasing frequency).

4A.2.2 Pressure Gradient Microphones

These microphones have a *figure-of-8* polar diagram along the longitudinal axis. They respond to the pressure difference between two points A and B, shown in Fig. 4A.1, close together and immersed in the sound field. The greatest pressure difference occurs at 0° and 180°, whereas for a sound coming from 90° with respect to the axis of the microphone, the sound is received with the same sensitivity from both points A and B. In fact, naming T_F the transmission factor of the sound field

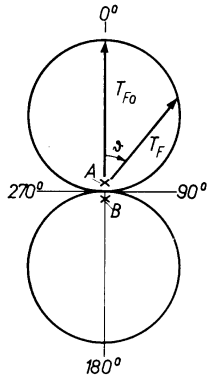


Figure 4A.1. Polar pattern of a pressure gradient microphone [10].

(or *sensitivity*), the following relation exists

$$T_F = T_{F0} \cos \vartheta \tag{4A.1}$$

where T_{F0} is the transmission factor when the sound is impinging from direction 0° (microphone axis) and ϑ is the angle of incidence of the acoustic wave.
If $\vartheta = 90° \rightarrow T_F = 0$. The incident pressure is compared at points A and B. This can be achieved electrically by using two identical adjacent capsules having opposite faces and measuring the output voltages connected with reverse polarity. Alternatively, the comparison is done mechanically in case of microphones having both the front and the rear sides of the diaphragm exposed to the sound field. In this second case, only the instant differences of the forces acting on the front and the rear result in a movement of the diaphragm. The pressure difference is due to the velocity of the particles of the medium in which sound propagates. Since the microphone output voltage is proportional to the pressure difference, it is also proportional to particle velocity, hence the name *velocity microphones*.

At the very beginning of this paragraph, we said that pressure gradient microphones have a *figure-of-8* polar diagram. This is worth pointing out because when it comes to microphones belonging to the family of cardioids, we refer to pressure gradient microphones, as well.

4A.3 Microphone behaviour in the presence of plane waves

Figure 4A.2 shows how, in the presence of a plane wave front, sound affects points A and B with the same strength, but with a difference in phase. With a constant sound pressure, the angle swept by the sound and the pressure gradient increase with frequency. In Fig. 4A.2b the acoustic wave has a frequency approximately twice that of Fig. 4A.2a and the same pressure. As you can see, approximately, a doubling of the pressure gradient occurs. Reading the technical specifications of the microphones that can be normally found on the market, specifically the frequency response graph, we note that all microphones have - sooner or later - a "hole" next to the so-called *characteristic frequency* of the microphone as shown in Fig. 4A.3. Of course, all

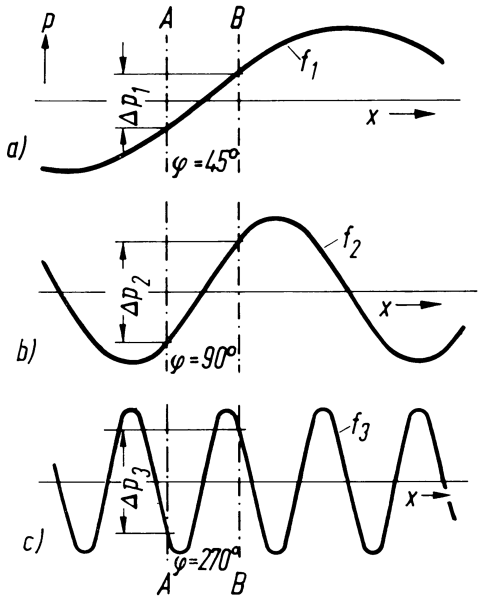


Figure 4A.2. Microphone behaviour with plane waves [10].

the technical specifications should always be taken into consideration, especially when building a microphone array for such a delicate system like Ambisonics, which exhibits most of its problems at high frequencies. Usually, the distance between

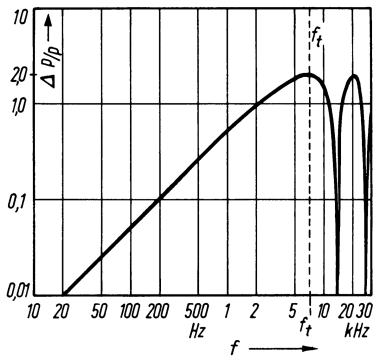


Figure 4A.3. Frequency response of a pressure gradient microphone: characteristic frequency effect [10].

points A and B is very tiny in microphones. There exists a limit beyond which the microphone does not respond efficiently to very high frequencies. If the distance A-B is shorter than half the wavelength we want to reproduce, this limit distance refers to the characteristic frequency f_t. For such a limit distance we have phase $\varphi = 180°$. Beyond the characteristic frequency, the pressure gradient diminishes abruptly.

4A.4 Microphone behaviour in the presence of spherical waves

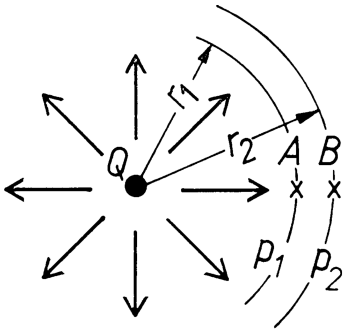


Figure 4A.4. Pressure gradient microphone behaviour with spherical front.

In the case of spherical waves (Fig. 4A.4), the pressure gradient at points A and B depends not only on the phase difference, but also on the distance between source and microphone. In that case, for a point source radiating a spherical front, pressure decreases with increasing distance from the source (that is, the pressure is proportional to $1/r$). Students at singing schools are taught that, approaching a microphone to the mouth, it is possible to enhance low frequencies. This is called *proximity effect* and is explained by the fact that the effect is noticeable especially at low frequencies for which the forces acting on the diaphragm are weaker, because the phase shift is smaller than in the case of high frequencies.
The boost at frequency f is calculated as follows:

$$\begin{cases} \frac{v_8}{v_0} = \frac{1}{\cos\alpha} \\ \tan\alpha = \frac{\lambda}{2\pi r} = \frac{54.14}{f \cdot r} \end{cases} \tag{4A.2}$$

where r is the distance between the microphone and the source, $\frac{v_8}{v_0}$ represents the boost at frequency f (wavelength λ), v_8 is the output voltage of a pressure gradient microphone having a directivity pattern such as figure-of-8 and v_0 is the output voltage of an omnidirectional microphone with same sensitivity at 0° [10].

Chapter 5

Quaternion augmented statistics

Contents

In Chapter 3, we discussed how in multidimensional adaptive filtering applications, like weather prediction [101, 103], 3D audio [71] or orientation tracking [50], very interesting results have been observed when quaternion-valued filters are used. Because of the peculiar properties of quaternion algebra, it is possible to exploit the correlation among all channels (quaternion dimensions), thus obtaining improved filter performance.

The latest efforts in quaternion-valued adaptive filtering have been oriented towards the study of the full second order statistics of the signals [3,60,100,105]. This trend was steered by the need of mathematical rigor in representing physical problems and phenomena with the aim of implementing accurate and efficient digital systems. In fact, simulations may fail if the models of the actors participating in the system are inadequate and, in adaptive filtering, performance may vary notably with the nature

of the input signal. Moreover, it was found that the early quaternion-valued filters, such as those derived directly from the Quaternion Least Mean Square algorithm (QLMS) [98], have no general validity [100]. It was demonstrated [60, 100, 105] that a complete insight of quaternion second order statistics is necessary in order to fully exploit the information coupling within all quaternion channels. Given a complex or hypercomplex signal $x[n]$, its second order statistics are typically described by the covariance function $\mathcal{C}_x(n_1, n_2) = E\{x[n_1]x^*[n_2]\}$ and the relation function $\mathcal{R}_x(n_1, n_2) = E\{x[n_1]x[n_2]\}$. In [99], a *Widely Linear* Quaternion Least Mean Square (WL-QLMS) filter was presented. This algorithm considers and incorporates complementary covariance matrices into quaternion augmented statistics by means of quaternion involutions. Such an algorithm showed improved performance with the processing of signals having a rotation dependent distribution. Such signals are called *non-circular* or *improper* and they can be encountered wherever signal components have disparities in the dynamics and they are correlated. Unfortunately, most real-world signals are not circular (*non-circular, improper*) [60, 100, 105]: examples were found in communications, where imbalance between in-phase and quadrature components gives rise to improper baseband signals [4]. Improper signals, in general, result from multichannel systems having channel gain disparities and correlated components, such as multi-sensor applications [12, 14, 57, 84]. In typical array processing applications, if signals (either signal of interest or interference) are improper, it is possible to achieve a high DOA resolution [57]. In [12, 14, 57] the *widely linear* model of the signals was exploited. The model takes into account the signal full second order statistics by incorporating complementary covariance matrices into quaternion math by means of quaternion involutions.

The shortcoming of WL-QLMS is the computational cost. In fact, the algorithm architecture includes the computation of quaternion-valued convolutions and cross-correlations for all signal involutions. In this work, besides recalling the main features of WL-QLMS, we propose a solution to this problem. We developed a block version of the WL-QLMS, i.e. it updates the filter weights periodically (Widely Linear Block Quaternion Least Mean Square, WL-BQLMS). We applied the algorithm to a 3D audio system, where improper signals are usually encountered. In acoustic applications, we often have to handle long impulse responses, e.g. long reverberation tails, and block algorithms represent a solution in most cases. Moreover, block time-domain algorithms pave the way for the implementation of frequency-domain algorithms, which compute transformations over blocks of samples. As known from the properties of the Fourier transform, this mathematical tool allows a fast execution of convolutions and cross-correlations, thus speeding up the processing.

This chapter shortly introduces quaternion augmented statistics and recalls the main facts about quaternion *properness*. Later, after reporting the basic Widely Linear QLMS algorithm, our Widely Linear Block QLMS is presented with simulations. A modification of the block algorithm in the frequency domain is also proposed.

5.1 Introduction to quaternion augmented statistics and quaternion properness

5.1.1 Quaternion augmented statistics

In complex numbers, we need both $z = z_a + \mathbf{i}z_b$ and its conjugate $z^* = z_a - \mathbf{i}z_b$ in order to write both its real and complex components as $z_a = \frac{1}{2}(z + z^*)$ and $z_b = \frac{1}{2\mathbf{i}}(z - z^*)$. In the quaternion domain, the same task is a little bit harder to achieve. We actually need to define and exploit the quaternion involutions:

$$\begin{aligned} q^{\mathbf{i}} &= -\mathbf{i}q\mathbf{i} = q_a + \mathbf{i}q_b - \mathbf{j}q_c - \mathbf{k}q_d \\ q^{\mathbf{j}} &= -\mathbf{j}q\mathbf{j} = q_a - \mathbf{i}q_b + \mathbf{j}q_c - \mathbf{k}q_d \\ q^{\mathbf{k}} &= -\mathbf{k}q\mathbf{k} = q_a - \mathbf{i}q_b - \mathbf{j}q_c + \mathbf{k}q_d. \end{aligned} \tag{5.1}$$

Given the definitions above in (5.1), we can now express each quaternionic component as $q_a = \frac{1}{2}(q + q^*)$, $q_b = \frac{1}{2\mathbf{i}}(q - q^{\mathbf{i}*})$, $q_c = \frac{1}{2\mathbf{j}}(q - q^{\mathbf{j}*})$, $q_d = \frac{1}{2\mathbf{k}}(q - q^{\mathbf{k}*})$ and establish a mapping between quaternion components and involutions.

In order to learn about the *properness* of a signal (Par. 5.1.2), it is helpful to define the complementary covariance matrices:

$$\mathcal{C}_{\mathbf{q}}^{\mathbf{i}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{i}H}\right\}, \mathcal{C}_{\mathbf{q}}^{\mathbf{j}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{j}H}\right\}, \mathcal{C}_{\mathbf{q}}^{\mathbf{k}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{k}H}\right\}. \tag{5.2}$$

Further details and definitions about second order statistics can be found in [60, 100, 105].

5.1.2 On the properness of quaternion-valued signals

Frequently, improper signals are found in real-world applications. One of the goals of research in adaptive filtering is to find a solution of general validity for both proper and improper signals. The *widely linear* model for quaternion-valued signals was introduced to achieve this target [100, 102]. Consequently to simulations, it was demonstrated [100, 102] that the WL-QLMS algorithm outperforms the QLMS when the filter input signal is improper.

Properness, or second order circularity, features in those variables having rotation-invariant probability distribution with respect to all six pairs of rotation axes $(\mathbf{1}, \mathbf{i})$, $(\mathbf{1}, \mathbf{j})$, $(\mathbf{1}, \mathbf{k})$, $(\mathbf{i}, \mathbf{j})$, $(\mathbf{k}, \mathbf{j})$, $(\mathbf{k}, \mathbf{i})$. A straightforward way to check the properness of a quaternion random variable $q = q_a + q_b\mathbf{i} + q_c\mathbf{j} + q_d\mathbf{k}$ is to test the following characterizing properties [100]:

1. $E\{q_m^2\} = \sigma^2, \quad \forall m = a, b, c, d$ (i.e. all four components of q have equal power).
2. $E\{q_m q_n\} = 0, \quad \forall m, n = a, b, c, d$ and $m \neq n$ (i.e. all four components of q are uncorrelated).
3. $E\{qq\} = -2E\{q_m^2\} = -2\sigma^2, \quad \forall m = a, b, c, d$ (i.e. the pseudocovariance matrix does not vanish)
4. $E\left\{|q|^2\right\} = 4E\{q_m^2\} = 4\sigma^2, \quad \forall m = a, b, c, d$ (i.e. the covariance of a quaternion variable is the sum of the covariances of all components).

A corollary of the above properties is that a Q-proper vector is not correlated with its involutions $(\mathbf{q}^{\mathbf{i}}, \mathbf{q}^{\mathbf{j}}, \mathbf{q}^{\mathbf{k}})$ [3], thus resulting

$$E\left\{\mathbf{q}\mathbf{q}^{\mathbf{i}H}\right\} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{j}H}\right\} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{k}H}\right\} = \mathbf{0}. \tag{5.3}$$

As a consequence, we have that Q-proper signals exhibit vanishing cross-correlation matrices for all components $(\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c, \mathbf{q}_d)$ and the augmented covariance matrix $E\left\{\mathbf{q}^a\mathbf{q}^{aH}\right\} = 4\sigma^2\mathbf{I}$ is real-valued, positive definite and symmetric ($\mathbf{q}^a = [\mathbf{q}^T\mathbf{q}^{\mathbf{i}T}\mathbf{q}^{\mathbf{j}T}\mathbf{q}^{\mathbf{k}T}]^T$). In addition, all the complementary matrices are real-valued and diagonal. More properties and detailed information about this topic can be found in [3, 100].

5.2 Motivation and theoretical foundation for Quaternion Widely Linear Processing

The first intuition about the advantages of widely linear signal processing is ascribable to Picinbono *et al.* who proved in [77] that Widely Linear Mean Square Estimation (WLMSE) supplies a reduced estimation error in comparison with Linear Mean Square Estimation (LMSE). In $\mathbb{C}$, we define the *linear* estimator $\hat{y}_L$ of y as

$$\hat{y}_L = \mathbf{h}^H\mathbf{x} \tag{5.4}$$

where $\mathbf{h}, \mathbf{x} \in \mathbb{C}^N$. The goal here is to find $\mathbf{h}$ that minimizes the estimation error $\varepsilon_L^2 = E\left[|y - \hat{y}_L|^2\right]$. The *widely linear* estimator $\hat{y}_{WL}$ of y, instead, is defined as

$$\hat{y}_{WL} = \mathbf{h}^H\mathbf{x} + \mathbf{g}^H\mathbf{x}^*. \tag{5.5}$$

From (5.5), it is clear that $\hat{y}_{WL}$ is not a *linear* function of $\mathbf{x}$, but a *widely linear* function of $\mathbf{x}$ and $\mathbf{x}^*$. Here we have to find $\mathbf{g}, \mathbf{h}$ that minimize the estimation error $\varepsilon_{WL}^2 = E\left[|y - \hat{y}_{WL}|^2\right]$. Picinbono *et al.* demonstrated that widely linear processing is generally superior than linear processing in terms of the mean square error:

$$\begin{aligned} \varepsilon_{WL}^2 &\leq \varepsilon_L^2 \\ E\left[|y - \hat{y}_{WL}|^2\right] &\leq E\left[|y - \hat{y}_L|^2\right]. \end{aligned} \tag{5.6}$$

In fact, given the definitions $\mathbf{r} = E[y^*\mathbf{x}]$, $\mathbf{s} = E[y\mathbf{x}]$, we have

$$\varepsilon_{WL}^2 = E[|y|^2] - (\mathbf{g}^H\mathbf{r} + \mathbf{h}^H\mathbf{s}^*) \leq E[|y|^2] - \mathbf{g}^H\mathbf{r} = \varepsilon_L^2. \tag{5.7}$$

Full details about this proof can be found in [77].

Tohru Nitta in [62] gave a similar explanation about the reasons why and when quaternion widely linear processing is convenient in place of linear processing. Nitta's proof concerns *augmented* quaternion LMSE, i.e. the estimator $\hat{y}_A$ includes the only involution term relative to signal conjugate ($\mathbf{x}^*$ actually is an involution of $\mathbf{x}$) and is defined in a way similar to (5.5) for complex data:

$$\hat{y}_A = \mathbf{h}^H\mathbf{x} + \mathbf{g}^H\mathbf{x}^*. \tag{5.8}$$

where $\mathbf{g}, \mathbf{h} \in \mathbb{H}^N$. We give the definitions $\mathbf{C} = E[\mathbf{x}\mathbf{x}^T]$, $\mathbf{r} = E[\mathbf{x}y^*]$, $\mathbf{s} = E[y\mathbf{x}]$, $\mathbf{\Gamma}_1 = E[\mathbf{x}\mathbf{x}^H]$, $\mathbf{\Gamma}_2 = E[\mathbf{x}^*\mathbf{x}^T]$ and write the following equations:

$$\begin{aligned} \mathbf{\Gamma_1 h} + \mathbf{C g} &= \mathbf{r} \\ \mathbf{C}^H \mathbf{h} + \mathbf{\Gamma}_2 \mathbf{g} &= \mathbf{s}^* \end{aligned} \tag{5.9}$$

We can finally get $\mathbf{g}, \mathbf{h}$ from (5.9):

$$\begin{aligned} \mathbf{g} &= \left(\mathbf{\Gamma}_2 - \mathbf{C}^H \mathbf{\Gamma}_1^{-1} \mathbf{C}\right)^{-1} \left(\mathbf{s}^* - \mathbf{C}^H \mathbf{\Gamma}_1^{-1} \mathbf{r}\right) \\ \mathbf{h} &= \left(\mathbf{\Gamma}_1 - \mathbf{C} \mathbf{\Gamma}_2^{-1} \mathbf{C}^H\right)^{-1} \left(\mathbf{r} - \mathbf{C} \mathbf{\Gamma}_2^{-1} \mathbf{s}^*\right) \end{aligned} . \tag{5.10}$$

If we substitute (5.10) into (5.8), we find that

$$\varepsilon_A^2 \leq \varepsilon_L^2 \tag{5.11}$$

where

$$\begin{aligned} \varepsilon_A^2 &= E[|y - \hat{y}_A|^2] = E[|y|^2] - \left(\mathbf{r}^H \mathbf{h} + \mathbf{s}^T \mathbf{g}\right) \\ \varepsilon_L^2 &= E[|y - \hat{y}_L|^2] = E[|y|^2] - \left(\mathbf{r}^H \mathbf{\Gamma}_1^{-1} \mathbf{r}\right) \end{aligned} . \tag{5.12}$$

Further details are given in [62]. If we consider the full second order statistics and include the terms relative to involutions into the widely linear estimator

$$\hat{y}_{WL} = \mathbf{w}^T \mathbf{x} + \mathbf{h}^T \mathbf{x^i} + \mathbf{u}^T \mathbf{x^j} + \mathbf{v}^T \mathbf{x^k} \tag{5.13}$$

it is clear from (5.12), that the estimation error will be even smaller:

$$\varepsilon_{WL}^2 \leq \varepsilon_A^2 \leq \varepsilon_L^2. \tag{5.14}$$

We omit the demonstration here, since it is not different from the *augmented* case with $\hat{y}_A = \mathbf{h}^H \mathbf{x} + \mathbf{g}^H \mathbf{x}^*$.

5.3 Widely Linear QLMS

Both the Augmented QLMS and the Widely Linear QLMS algorithms have been studied by Mandic *et al.* extensively in [99, 100]. The difference between the two algorithms is that the latter includes the *full* second order statistics of the quaternion signals as introduced in Par. 5.2. As can be foreseen by (5.13), the WL-QLMS algorithm updates four vectors of filter weights: $\mathbf{w}$, $\mathbf{h}$, $\mathbf{u}$, $\mathbf{v} \in \mathbb{H}^{M \times 1}$, where M is the filter length. Accordingly, each filter output component is the convolution of each weight vector with its corresponding input involution:

$$y_w[n] = \mathbf{w}_{n-1}^T \mathbf{x}_n, \quad y_h[n] = \mathbf{h}_{n-1}^T \mathbf{x^i}_n, \quad y_u[n] = \mathbf{u}_{n-1}^T \mathbf{x^j}_n, \quad y_v[n] = \mathbf{v}_{n-1}^T \mathbf{x^k}_n. \tag{5.15}$$

The overall filter output is the sum of all contributions:

$$y[n] = y_w[n] + y_h[n] + y_u[n] + y_v[n]. \tag{5.16}$$

The output in (5.16) is compared with a desired signal $d[n]$ and the filter error is calculated as $e[n] = d[n] - y[n]$, as usual. In conclusion, the four adaptation equations are

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}_{n-1} + \mu e\,[n]\,\mathbf{x}_n^*, \quad & \mathbf{h}_n &= \mathbf{h}_{n-1} + \mu e\,[n]\,\mathbf{x}_n^{\mathbf{i}*} \\ \mathbf{u}_n &= \mathbf{u}_{n-1} + \mu e\,[n]\,\mathbf{x}_n^{\mathbf{j}*}, \quad & \mathbf{v}_n &= \mathbf{v}_{n-1} + \mu e\,[n]\,\mathbf{x}_n^{\mathbf{k}*}. \end{aligned} \tag{5.17}$$

Since the execution of four quaternion adaptation processes may require a huge computational effort, we propose in Par. 5.4 a version of WL-QLMS operating periodically in blocks. This algorithm is particularly suitable for 3D (audio) applications, for example.

5.4 Widely Linear Block QLMS

The Widely Linear Block Quaternion Least Mean Square algorithm is a block algorithm, i.e. it has a periodic weight update equation. It differs from Block QLMS since it embeds a widely linear model of the system and it exploits the signal augmented statistics. The algorithm was founded on the revised version of QLMS proposed in [8] and a step-by-step description of it is given just below.

5.4.1 Overview of the WL-BQLMS algorithm

Before running the algorithm, the initial values of all variables and vectors involved need to be chosen. The WL-BQLMS algorithm exploits quaternion involutions, so four sets of filter weights have to be initialized and updated in the process: $\mathbf{w}_{init}$, $\mathbf{h}_{init}$, $\mathbf{u}_{init}$, $\mathbf{v}_{init} \in \mathbb{H}^{M\times 1}$, where M is the filter length. All filter weight vectors $(\mathbf{w}_k, \mathbf{h}_k, \mathbf{u}_k, \mathbf{v}_k)$ are defined in a way similar to $\mathbf{w}_k = [w_0[k]\ w_1[k]\ \ldots\ w_{M-1}[k]]^{\mathrm{T}} \in \mathbb{H}^{M\times 1}$. After completing the initialization, the following operations are executed at each new input block k.

Given the input block defined as $\mathbf{x}_k = [x[kL]\ x[kL-1]\ \ldots\ x[kL-L+1]]^{\mathrm{T}}$, its involutions ($\mathbf{x}^{\mathbf{i}}$, $\mathbf{x}^{\mathbf{j}}$, $\mathbf{x}^{\mathbf{k}}$) have to be derived. We denote the filter length and the block length with M and L, respectively. We can now compute the filter output by convolving each weight vector with its corresponding input involution:

$$\begin{aligned} y_w\,[k] &= \mathbf{w}_k^T \mathbf{x}_k = \sum_{l=0}^{M-1} w_k\,[l]\,x\,[kL+i-l] \\ y_h\,[k] &= \mathbf{h}_k^T \mathbf{x}^{\mathbf{i}}{}_k = \sum_{l=0}^{M-1} h_k\,[l]\,x^{\mathbf{i}}\,[kL+i-l] \\ y_u\,[k] &= \mathbf{u}_k^T \mathbf{x}^{\mathbf{j}}{}_k = \sum_{l=0}^{M-1} u_k\,[l]\,x^{\mathbf{j}}\,[kL+i-l] \\ y_v\,[k] &= \mathbf{v}_k^T \mathbf{x}^{\mathbf{k}}{}_k = \sum_{l=0}^{M-1} v_k\,[l]\,x^{\mathbf{k}}\,[kL+i-l] \end{aligned} \tag{5.18}$$

and summing all four contributions:

$$y\,[k] = y_w\,[k] + y_h\,[k] + y_u\,[k] + y_v\,[k]\,. \tag{5.19}$$

The filter error at block k is defined as the difference between the desired signal $d[k]$ and the filter output $y[k]$:

$$e\,[k] = d\,[k] - y\,[k]\,. \tag{5.20}$$

We can finally update the filter weights:

$$\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k + \mu \sum_{i=0}^{L-1} e\,[kL+i]\,\mathbf{x}^*_{kL+i} \\
\mathbf{h}_{k+1} &= \mathbf{h}_k + \mu \sum_{i=0}^{L-1} e\,[kL+i]\,\mathbf{x}^{\mathbf{i}^*}_{kL+i} \\
\mathbf{u}_{k+1} &= \mathbf{u}_k + \mu \sum_{i=0}^{L-1} e\,[kL+i]\,\mathbf{x}^{\mathbf{j}^*}_{kL+i} \\
\mathbf{v}_{k+1} &= \mathbf{v}_k + \mu \sum_{i=0}^{L-1} e\,[kL+i]\,\mathbf{x}^{\mathbf{k}^*}_{kL+i}
\end{aligned} \tag{5.21}$$

where the step size μ includes the average term of the block algorithm ($\mu = \mu_B/L$).

5.4.2 Computational cost

Our goal in implementing the WL-BQLMS was to achieve efficiency from the computational point of view in comparison with WL-QLMS. In WL-QLMS, given a filter length M and a number of samples N, the computation of both the filter output and the cross-correlation in the update equations requires $2 \times 4 \times (4 \times 4)M \times N = 128M \times N$ multiplications. Remember that each quaternion multiplication requires 4×4 real-valued multiplications. This operation has to be repeated 4 times (one for each widely linear channel) and such a computational effort is required twice for each sample (filter output and weight update equation). The critical processing paths are the computation of the filter output and the cross-correlation in the update equation. In WL-BQLMS the number of algorithm iterations is simply divided by the block length L with respect to WL-QLMS. Definitively, we have the computational cost ratio between WL-BQLMS and WL-QLMS equal to $1/L$:

$$\frac{C_{WL-BQLMS}}{C_{WL-QLMS}} = \frac{128M \times (N/L)}{128M \times N} = \frac{1}{L}. \tag{5.22}$$

5.5 3D improper sound fields

As introduced in Par. 5.1.1, the widely linear model suits those systems having some disparities in the dynamics of its channels and correlated signal components. The 3D audio recording/processing technique called Ambisonics and described in Chapter 4 belongs to such a category of systems. We still consider the 1st order *B-Format* here (Fig. 4.2). This technique employs an array of microphones, with coincident capsules, orthogonal to one another. Because of such a layout (geometric placement with respect to the sound source and different types of polar pattern for each capsule), disparities of the signal levels in the channels are likely to occur.

We have seen in Chapter 4 how to transform and represent B-Format signals in a quaternion format:

$$B^Q[n] = B_W[n] + B_X[n]\mathbf{i} + B_Y[n]\mathbf{j} + B_Z[n]\mathbf{k}. \tag{5.23}$$

5.6 Direct system modeling with Ambisonic signals

The simulations below are aimed at comparing the performances of WL-BQLMS and BQLMS when improper processes are involved. We propose a very simple example: a direct system modeling problem with an Ambisonic B-Format input signal.

5.6.1 Direct system modeling performance

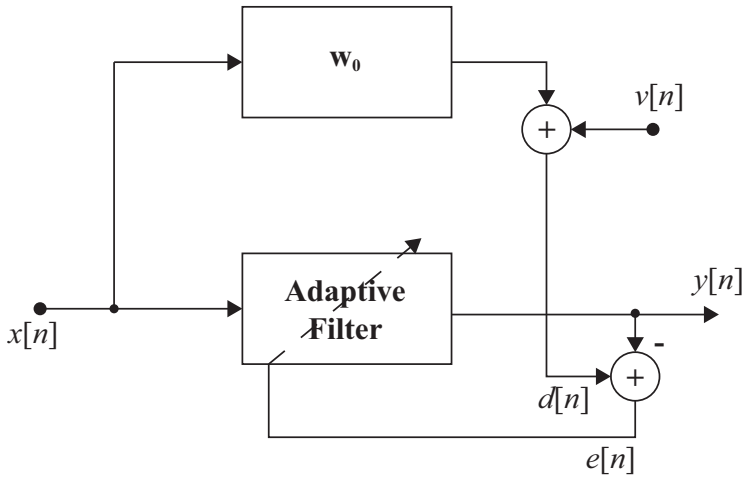


Figure 5.1. System modeling block diagram.

Given a system $\mathbf{w}_0$ to be identified, we compare the behavior of the WL-BQLMS and the BQLMS algorithms when the adaptive filter has an improper input signal. The improper quaternion input signal was recorded by means of a 1st order B-Format array as defined in Par. 4.2.1. The signal samples have the form as in (5.23) and its source is a monodimensional unit-variance white Gaussian noise. Signal $B_W[n]$ is the omnidirectional component and signals $B_X[n], B_Y[n], B_Z[n]$ are the three figure-of-eight components. The source was placed at a distance of 20 cm from the array, 45° off-axis with the X microphone. The recording environment is an anechoic simulated room. Additive unit-variance white Gaussian noise $\nu[n] \in \mathbb{H}$, with $n = 0, 2..., P-1$, is summed to the output signal of the system to be identified ($d[n] = \mathbf{w}_0^T \mathbf{B}^Q + \nu[n]$). Signal *improperness* can be easily proved by checking the properties 1 and 2 listed in Par. 5.1.2 for Q-proper signals:

1. $E\{q_m^2\} = \sigma^2, \forall m = a, b, c, d$
2. $E\{q_m q_n\} = 0, \forall m, n = a, b, c, d$ and $m \neq n$

As a matter of simplicity, the filter length M in WL-BQLMS is chosen equal to the block length L ($M = L$). Choosing the filter parameters as $M = 4, \mu = 0.6$ ($\mu_B = \mu L$), we obtain the results reported in Fig. 5.2.

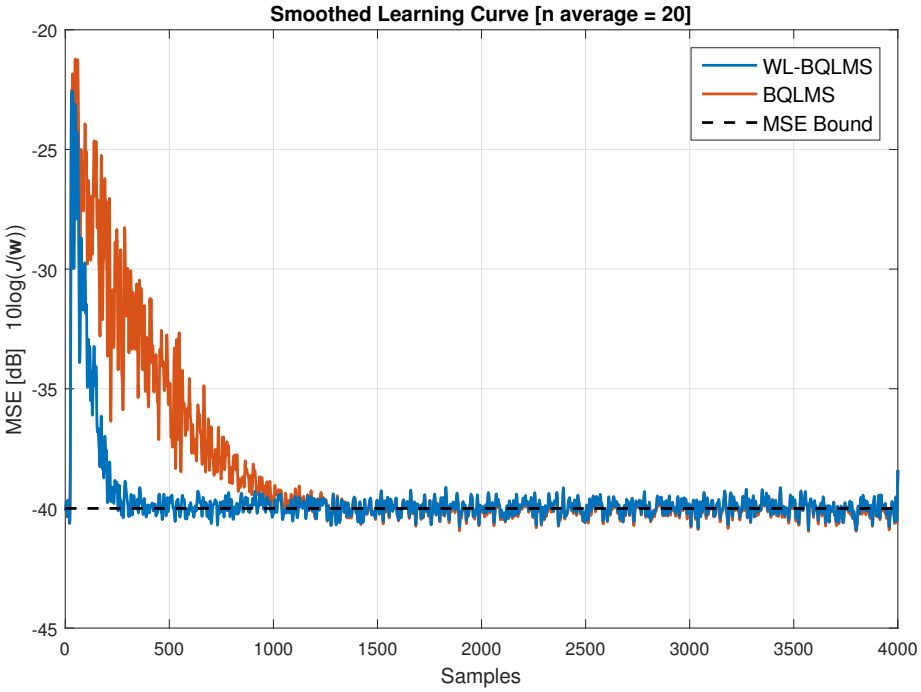


Figure 5.2. WL-BQLMS vs. BQLMS. Direct system modeling with quaternion-valued ambisonic input signal (improper). Mean Square Error (MSE)

It is worth noting that, when the input signal is improper, non-widely linear algorithms, such as (B)QLMS, typically converge slower than widely linear algorithms [99, 100]. When the input signal is proper the situation is turned around. In fact, when the input signal is proper, widely linear and non-widely linear algorithms should not perform differently (aside from minor numerical artefacts). As a proof, we propose another simple experiment of the same kind. This time the quaternion input signal is a proper colored signal defined as $x[n] = bx[n-1] + \frac{\sqrt{1-b^2}}{\sqrt{4}}\eta[n]$, where $\eta[n] \in \mathbb{H}$, with $n = 0, 1, ..., P-1$, is a unit-variance white Gaussian noise sequence and b is a filtering coefficient (chosen as $b = 0.7$ in this simulation). The filter parameters are chosen as $M = 4, \mu = 0.01$. The MSE curve is given in Fig. 5.3.

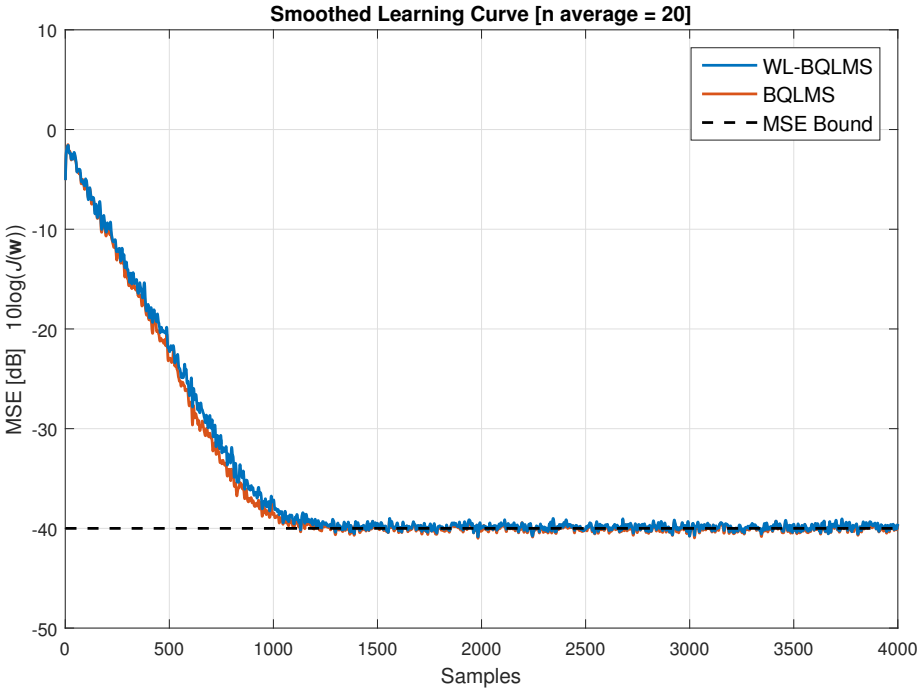


Figure 5.3. WL-BQLMS vs. BQLMS. Direct system modeling with quaternion-valued proper input signal. Mean Square Error (MSE).

5.7 Widely linear algorithms in the frequency domain

In Par. 3.2, we introduced the Overlap-Save Quaternion Frequency Domain Filter (OS-QFDAF). It is possible to extend this algorithm to a widely linear architecture. In [73], the authors introduced a preliminary version of the Widely Linear Overlap-Save Quaternion Frequency Domain Filter (WL-OS-QFDAF). This algorithm applies the convolution theorem in the *conventional* form $Y(\omega) = H(\omega)X(\omega)$ (*spectrum product*). However, we said in Chapter 2 that, in quaternion algebra, spectrum product corresponds to convolution in the time-domain only in the case one of the two functions has even simmetry. Generally, spectrum product corresponds to

$$y(t) = \int_{-\infty}^{\infty} h_a(t-\tau)x(\tau)d\tau + \int_{-\infty}^{\infty} h_b(t+\tau)\boldsymbol{\nu_2}x(\tau)d\tau \tag{5.24}$$

if the quantities in the frequency domain derive from the left transform, or

$$y(t) = \int_{-\infty}^{\infty} h(t-\tau)x_a(\tau)d\tau + \int_{-\infty}^{\infty} h(\tau-t)\boldsymbol{\nu_2}x_b(\tau)d\tau \tag{5.25}$$

if the right transform is employed. Here, we present the full algorithm, with the convolution and crosscorrelation theorems introduced in Par. 2.3 and Par. 3.2.

5.7.1 Widely Linear Overlap-Save Quaternion Frequency Domain Filter - Algorithm Overview

Algorithm initialization:

$$\begin{aligned}&\mathbf{W}_{init} = \mathbf{H}_{init} = \mathbf{U}_{init} = \mathbf{V}_{init} = \mathbf{0} \ \text{(M-by-1 null weight vectors)}\\&\mu = \mu^{\mathbf{i}} = \mu^{\mathbf{j}} = \mu^{\mathbf{k}} = \mu_0,\\&P_0(m) = P_0^{\mathbf{i}}(m) = P_0^{\mathbf{j}}(m) = P_0^{\mathbf{k}}(m) = \delta, m = 0, 1, ..., N-1\end{aligned} \tag{5.26}$$

$P_k(m)$: 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:

$$\mathbf{X}_k = \text{diag}\left[\text{QFFT}\left[\mathbf{x}_{old}^M \quad \mathbf{x}_k^L\right]^T\right] \tag{5.27}$$

where the input block consists of $\mathbf{x}_{old}^M$ and $\mathbf{x}_k^L$, defined as

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

Compute the input matrices $\mathbf{X}_k^{\mathbf{i}}.\mathbf{X}_k^{\mathbf{j}}, \mathbf{X}_k^{\mathbf{k}}$ for the involutions of $\mathbf{x}_k$.
Note: diagonalization in (5.27) allows a formalism similar to Block LMS when the filter output is computed in (5.28) [104].

Compute the filter output in the frequency domain and anti-transform:

$$\begin{aligned}\mathbf{Y}_k^w &= \mathbf{W}_k^a\mathbf{X}_k + \mathbf{W}_k^b\boldsymbol{\nu}_{\mathbf{2}}\mathbf{X}_{-k} \quad \mathbf{Y}_k^u = \mathbf{U}_k^a\mathbf{X}_k + \mathbf{U}_k^b\boldsymbol{\nu}_{\mathbf{2}}\mathbf{X}_{-k}\\\mathbf{Y}_k^h &= \mathbf{H}_k^a\mathbf{X}_k + \mathbf{H}_k^b\boldsymbol{\nu}_{\mathbf{2}}\mathbf{X}_{-k} \quad \mathbf{Y}_k^v = \mathbf{V}_k^a\mathbf{X}_k + \mathbf{V}_k^b\boldsymbol{\nu}_{\mathbf{2}}\mathbf{X}_{-k}\end{aligned} \tag{5.28}$$

$$\mathbf{Y}_k = \mathbf{Y}_k^w + \mathbf{Y}_k^h + \mathbf{Y}_k^u + \mathbf{Y}_k^v \tag{5.29}$$

$$\hat{\mathbf{y}}_k = [\text{IQFFT}(\mathbf{Y}_k)]^{\lfloor L \rfloor} \tag{5.30}$$

$\mathbf{Y}_k$: filter output in the frequency domain.
$\hat{\mathbf{y}}_k$: filter output in the time domain (take only the last L samples).

$\mathbf{X}_{-k} \equiv \mathbf{X}_k(-\omega)$: frequency reversed signal of $\mathbf{X}_k(\omega)$
$\mathbf{W}_k^a$, $\mathbf{W}_k^b$: simplex and perplex parts ($\mathbf{W}_k = \mathbf{W}_k^a + \mathbf{W}_k^b\boldsymbol{\nu}_{\mathbf{2}}$).

Error calculation:

$$\hat{\mathbf{e}}_k = \hat{\mathbf{d}}_k - \hat{\mathbf{y}}_k \tag{5.31}$$

$$\mathbf{E}_k = \text{QFFT}\left(\left[0_M \quad \hat{\mathbf{e}}_k\right]^T\right) \tag{5.32}$$

$\mathbf{d}_k$: desired output vector in the time domain at block k,
$\hat{\mathbf{d}}_k = \left[d(kL) \quad d(kL+1) \quad \cdots \quad d(kL+L-1)\right]^T$.

$\mathbf{E}_k$: error vector in the frequency domain at block k.

Update the learning rates (*Power Normalization*):

$$\boldsymbol{\mu}_k = \mu \cdot \operatorname{diag}\left[P_k^{-1}(0), \ldots, P_k^{-1}(N-1)\right] \tag{5.33}$$

$$P_k(m) = \lambda P_{k-1}(m) + (1-\lambda)\left|X_k(m)\right|^2 \tag{5.34}$$

$\lambda \in [0,1]$: forgetting factor.
Compute $(\boldsymbol{\mu}_k^{\mathbf{i}}, P_k^{\mathbf{i}})$, $(\boldsymbol{\mu}_k^{\mathbf{j}}, P_k^{\mathbf{j}})$, $(\boldsymbol{\mu}_k^{\mathbf{k}}, P_k^{\mathbf{k}})$ for all involutions.

Note: Power Normalization reduces the disparity (*Eigenspread*) between the eigenvalues of the input autocorrelation matrix by assigning each filter weight a step size of its own. Decreasing the eigenspread makes the algorithm modes converge at the same rate.

Update the filter weights:

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k + \boldsymbol{\mu}_k \mathbf{C}_k \quad & \mathbf{U}_{k+1} &= \mathbf{U}_k + \boldsymbol{\mu}_k \mathbf{C}_k^{\mathbf{j}} \\ \mathbf{H}_{k+1} &= \mathbf{H}_k + \boldsymbol{\mu}_k \mathbf{C}_k^{\mathbf{i}} \quad & \mathbf{V}_{k+1} &= \mathbf{V}_k + \boldsymbol{\mu}_k \mathbf{C}_k^{\mathbf{k}} \end{aligned} \tag{5.35}$$

where

$$\begin{aligned} \mathbf{C}_k &= E_k^a(\omega)\left[X_k^{aH}(\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^b(-\omega)\right] + E_k^b(\omega)\boldsymbol{\nu}_{\mathbf{2}}\left[X_k^{aH}(-\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^b(\omega)\right] \\ \mathbf{C}_k^{\mathbf{i}} &= E_k^a(\omega)\left[X_k^{\mathbf{i}aH}(\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{i}b}(-\omega)\right] + E_k^b(\omega)\boldsymbol{\nu}_{\mathbf{2}}\left[X_k^{\mathbf{i}aH}(-\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{i}b}(\omega)\right] \\ \mathbf{C}_k^{\mathbf{j}} &= E_k^a(\omega)\left[X_k^{\mathbf{j}aH}(\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{j}b}(-\omega)\right] + E_k^b(\omega)\boldsymbol{\nu}_{\mathbf{2}}\left[X_k^{\mathbf{j}aH}(-\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{j}b}(\omega)\right] \\ \mathbf{C}_k^{\mathbf{i}} &= E_k^a(\omega)\left[X_k^{\mathbf{k}aH}(\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{k}b}(-\omega)\right] + E_k^b(\omega)\boldsymbol{\nu}_{\mathbf{2}}\left[X_k^{\mathbf{k}aH}(-\omega) - \boldsymbol{\nu}_{\mathbf{2}} X_k^{\mathbf{k}b}(\omega)\right] \end{aligned} \tag{5.36}$$

where $\mathbf{X}_{-k} \equiv \mathbf{X}_k(-\omega)$, $\mathbf{E}_k = \mathbf{E}_k^a + \mathbf{E}_k^b \boldsymbol{\nu}_{\mathbf{2}}$, $\mathbf{X}_k = \mathbf{X}_k^a + \mathbf{X}_k^b \boldsymbol{\nu}_{\mathbf{2}}$.
As in the OS-QFDAF algorithm we can constrain the gradient as follows:

$$\mathbf{W}_{k+1} = \mathrm{QFFT}\begin{pmatrix} [\mathrm{IQFFT}(\mathbf{W}_{k+1})]^{\lceil M \rceil} \\ \mathbf{0}_L \end{pmatrix}. \tag{5.37}$$

The same procedure will be repeated for all other weights $\mathbf{H}$, $\mathbf{U}$, $\mathbf{V}$.

5.7.2 Computational cost analysis

Our goal in implementing the WL-OS-QFDAF was to achieve efficiency from the computational point of view in comparison with WL-QLMS.

The computation of one complex FFT involves $N\log_2 N$ multiplications and each QFFT requires the execution of 2 complex FFTs. The WL-OS-QFDAF algorithm includes 14 (I)QFFTs, $8 \times 16N$ multiplications to compute the filter output and $8 \times 16N$ multiplications to update the filter weights. Given the QFFT length $N = 2L = 2M$ samples, the computational cost for WL-OS-QFDAF is approximately

$$\begin{aligned} C_{WL-OS-QFDAF} &\simeq 2 \cdot 14N\log_2 N + 2 \cdot 8 \cdot 16N \\ &= 2 \cdot 28M\log_2 M + 2 \cdot 256M. \end{aligned} \tag{5.38}$$

In WL-QLMS, the computation of both the filter output and the cross-correlation in the update equations requires $4 \cdot 4 \cdot 4M = 64M$ multiplications for each sample. Overall, for M samples, the computational cost for WL-QLMS is approximately $C_{WL-QLMS} \simeq 128M \cdot M = 128M^2$. The complexity ratio between WL-OS-QFDAF and WL-QLMS is

$$\frac{C_{WL-OS-QFDAF}}{C_{WL-QLMS}} = \frac{7\log_2 2M + 64}{16M} \quad . \tag{5.39}$$

For example, for $M = 16$, the WL-OS-QFDAF algorithm is about 2.5 times more efficient than WL-QLMS.

5.7.3 Simulations

We will repropose here the simulation in the scheme of Fig. 5.1 for testing the performance of WL-OS-QFDAF in comparison with WL-QLMS which operates in the time domain. The system to be identified is characterized by a set of random weights $\mathbf{w}_0$ (in the time domain), uniformly distributed in the range $[-1, 1]$. The quaternion filter input signal $x[n]$ is a unit variance colored noise with length of 20,000 samples, obtained by filtering the white Gaussian noise $\eta[n]$ as

$$x\,[n] = bx\,[n-1] + \frac{\sqrt{1-b^2}}{\sqrt{4}}\eta\,[n] \tag{5.40}$$

where b determines the bandwidth of the signal. The parameter b is equal to 0 (white noise) during the first 10,000 samples, and 0.9 (narrow band) during the last 10,000 samples. The additive signal $v[n]$ is unit-variance white Gaussian noise with an SNR of 50 dB. The choice of the overlap length and the block length ($M = L$) determines the FFT length ($N = M + L$). In this test, $M = 8$ and $L = 10$.

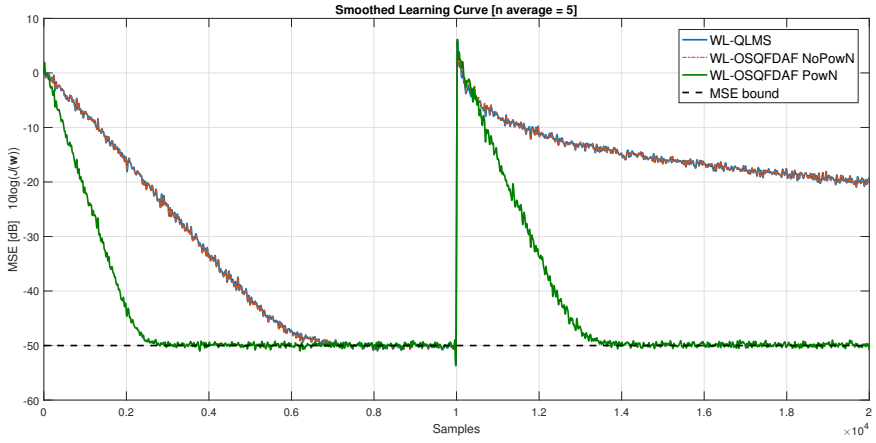


Figure 5.4. WL-QLMS vs. WL-OS-QFDAF with and without power normalization. Direct system modeling with quaternion-valued input signal. Mean Square Error (MSE)

In Fig. 5.4 we can see that the WL-QLMS and the WL-OS-QFDAF without power normalization behave the same way. In particular, the algorithm convergence

is slower in the case the input signal has a narrow band ($b = 0.9$). The power-normalized WL-OS-QFDAF outperforms the other two algorithms in both cases $b = 0$ and $b = 0.9$.

Chapter 6

A comparison study with other hypercomplex algebras

Contents

As previously introduced, multidimensionality is somehow intrinsic to the nature of the data: it arises from the need for processing correlated data (not necessarily homogeneous, as in [98]). The investigation topic in this chapter considers two hypercomplex algebras having the same dimensions: quaternions and tessarines (the latter also known as *bicomplex numbers*). Both of them are a 4-dimensional algebra. So, which are the reasons why we should choose one algebra or another? Here we present a couple of examples where different results are obtained from different mathematical representations of the systems. A line of reasoning is also suggested. With regard to adaptive signal processing, the first adaptive algorithm implemented in a hypercomplex algebra has been the Quaternion Least Mean Square (QLMS) algorithm [98]. Because of its straight comprehensibility and ease of implementation, this algorithm offered an instrument on hand for studying quaternion algebra combined with adaptive filtering. We adopted this algorithm to make a comparison

of the two 4-dimensional algebras above mentioned. On this occasion, we derived and implemented a tessarine version of the LMS algorithm, namely TLMS. In a second step, we searched for a modification of both the QLMS and TLMS algorithms into a *widely linear* form (including full second order statistics) [54, 99, 100] and their behaviour was tested with *proper* and *improper* input signals [69, 70]. The highlight in this chapter is the evidence that the choice of a specific algebra may condition a filter behaviour. We analysed this fact by introducing Ambisonic 3D audio signals (see Chapter 4) into a 4-dimensional system. The main goal of the current work is to examine whether the good results obtained with quaternion algebra persist in other 4-dimensional algebras.

In this chapter, after itemizing the main differences between quaternions and tessarines and describing 4D algorithms (Par. 6.1 and Par. 6.2), we present two simulations contexts (Par. 6.4). The first test is aimed at studying the LMS filter performance in tessarine and quaternion algebra in case of correlated and uncorrelated input signals. The second test works with only correlated signals, but the array geometry is changed. The microphone layouts are described in Par. 6.3.

6.1 Differences between quaternion and tessarine algebras

The geometric algebras belonging to Clifford algebras permit generating units as either $\mathbf{i}_{n+1}^2 = -1$ or $\mathbf{i}_{n+1}^2 = +1$. The number of generators p and q in both types determine completely the particular Clifford algebra over the field $\mathbb{K}$ with $\mathcal{C}\ell(p,q)(\mathbb{K})$. Typically, all Clifford algebras as associative, but not necessarily division algebras. Tessarine algebra is an example of this kind. Here we briefly recall the main properties of quaternions ($\mathbb{H}$) and we make a comparison with the 4-dimensional algebra of tessarines ($\mathbb{T}$). Other 4-D algebras exist, e.g. the Sklyanin algebras described in [88]; however, we chose these two algebras because they are characterized by the fact that their *outfit* is exactly the same: $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ with $q \in \mathbb{H}$ and $t = t_0 + t_1\mathbf{i} + t_2\mathbf{j} + t_3\mathbf{k}$ with $t \in \mathbb{T}$. Despite that, the algebraic rules governing quaternion and tessarine algebras are radically different. Both quaternions and tessarines can be split into a real (scalar) part (i.e. q_0 and t_0), and a full-imaginary (vector) part built on the imaginary axes $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$. The properties below are those responsible of the essential differences between quaternion and tessarine numerical systems:

Quaternion algebra:

$$\mathbf{ij} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{i}, \quad \mathbf{ki} = \mathbf{j}, \tag{6.1}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1. \tag{6.2}$$

Tessarine algebra:

$$\mathbf{ij} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{i}, \quad \mathbf{ki} = -\mathbf{j}, \tag{6.3}$$

$$\mathbf{i}^2 = \mathbf{k}^2 = -1, \quad \mathbf{j}^2 = +1. \tag{6.4}$$

Axioms (6.1)–(6.4) generate the discrepancies between the two algebras, e.g. they dictate the rules for product and other mathematical operations (convolution, correlation, etc.).
Given quaternions q_a, $q_b \in \mathbb{H}$, we compute their product as

$$\begin{aligned} q_a q_b &= (a_0 + a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k})\,(b_0 + b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}) \\ &= (a_0 b_0 - a_1 b_1 - a_2 b_2 - a_3 b_3) \\ &+ (a_0 b_1 + a_1 b_0 + a_2 b_3 - a_3 b_2)\,\mathbf{i} \\ &+ (a_0 b_2 - a_1 b_3 + a_2 b_0 + a_3 b_1)\,\mathbf{j} \\ &+ (a_0 b_3 + a_1 b_2 - a_2 b_1 + a_3 b_0)\,\mathbf{k}. \end{aligned} \tag{6.5}$$

Given tessarines t_a, $t_b \in \mathbb{T}$, we compute their product as

$$\begin{aligned} t_a t_b &= (a_0 + a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k})\,(b_0 + b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}) \\ &= (a_0 b_0 - a_1 b_1 + a_2 b_2 - a_3 b_3) \\ &+ (a_0 b_1 + a_1 b_0 + a_2 b_3 + a_3 b_2)\,\mathbf{i} \\ &+ (a_0 b_2 - a_1 b_3 + a_2 b_0 - a_3 b_1)\,\mathbf{j} \\ &+ (a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0)\,\mathbf{k}. \end{aligned} \tag{6.6}$$

Product rules can be also expressed with tables (see Table 6.1 and Table 6.2):

	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	-1	**k**	**-j**
j	**j**	**-k**	-1	**i**
k	**k**	**j**	**-i**	-1

Table 6.1. Quaternion Multiplication

	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	-1	**k**	**-j**
j	**j**	**k**	1	**i**
k	**k**	**-j**	**i**	-1

Table 6.2. Tessarine Multiplication

It can be seen that commutativity features only in tessarine algebra. As a consequence of this, for example, we can detect an effect on the convolution theorem. In fact, the classic convolution theorem (valid in $\mathbb{R}, \mathbb{C}$) does not hold in quaternion algebra [67,82]. On the contrary, the author of this thesis experimented that the theorem is still valid in tessarine algebra. This result is reported in Par. 6.5. On the other hand, the sum is computed the same way with either quaternions or tessarines (component-by-component):

$$\begin{aligned} q \pm p &= (q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}) \pm (p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}) \\ &= (q_0 \pm p_0) + (q_1 \pm p_1)\,\mathbf{i} + (q_2 \pm p_2)\,\mathbf{j} + (q_3 \pm p_3)\,\mathbf{k}. \end{aligned} \tag{6.7}$$

Another consequence of the divergent algebraic background is the distinct definition of the conjugate of a quaternion and a tessarine:

$$\begin{aligned} q^* &= q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \in \mathbb{H} \\ t^* &= t_0 - t_1\mathbf{i} + t_2\mathbf{j} - t_3\mathbf{k} \in \mathbb{T}. \end{aligned} \tag{6.8}$$

However, the *poly-conjugation* can generalize both:

$$p^* = p_0 + \sum_{\nu=1}^{3} p_\nu \mathbf{e}_\nu^3 \tag{6.9}$$

where $\mathbf{e}_\nu$ with $\nu = 1, 2, 3$ form an orthonormal basis.

If we consider commutative, associative, non-division algebras with $n > 2$, these always exhibit zero divisors [1]. For example, for $n = 2$:

$$(1 - j)(1 + j) = 1 - j^2 = 1 - 1 = 0 \tag{6.10}$$

with $(1 - j) \neq 0$ and $(1 + j) \neq 0$.
One of the potentialities of zero divisors is the *orthogonal decomposition*, which allows representing signals in orthogonal components, thus reducing the computational cost in those operations involving multiplication:

- Regular 4D hypercomplex multiplication: $4^2 = 16$ real multiplications.
- Orthogonal decomposition: 4 real multiplications (component-wise).

Orthogonal decomposition is based on the fact that each commutative and associative algebra over $\mathbb{R}$ is isomorphic to a direct sum in $\mathbb{R}$.
Drawbacks with zero divisors: Euclidean norm is not multiplicative (i.e. $|a|\,|b| \neq |ab|$), so, for instance, the definition of *energy* is under discussion for non-Euclidean algebras.

Unlike non-commutative algebras, the properties of commutative algebras still hold with increasing n.

6.2 A comparison of 4D adaptive filters

In quaternion algebra, the development of adaptive filters has been progressing for about a decade. Since the first Quaternion Least Mean Square algorithm was presented in [98], improvements and other quaternion algorithm architectures were proposed. On the contrary, tessarine algebra did not arise the same interest as quaternions, so the literature lacks of filters in this algebra. On account of this, the author of this thesis *et al.* implemented the basic Tessarine LMS algorithm in order to make a comparison with its quaternion counterpart [69, 70]. The main information about the basic LMS algorithm is provided in the following.

We recall here the main facts about the LMS algorithm, extensively debated in Par. 3.1. In the LMS adaptation process, we define a cost function to be minimized,

[1] If A is a commutative ring, then a non-zero element $a \in A$ is called *zero divisor* if there exists a non-zero element $b \in A$, such that $ab = 0$.

e.g. the Mean Square Error (MSE) $J(\mathbf{w}_{n-1}) = E\{e[n]e^*[n]\}$. The error $e[n]$ is measured as the difference between a desired signal and the adaptive filter output ($e[n] = d[n] - y[n]$). Considering the properties of quaternion and tessarine algebras, the adaptive filter output is slightly different in the two algebras:

$$y[n] = \mathbf{w}_{n-1}^T \mathbf{x}_n = \begin{bmatrix} \mathbf{w}_a^T\mathbf{x}_a - \mathbf{w}_b^T\mathbf{x}_b - \mathbf{w}_c^T\mathbf{x}_c - \mathbf{w}_d^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_b + \mathbf{w}_b^T\mathbf{x}_a + \mathbf{w}_c^T\mathbf{x}_d - \mathbf{w}_d^T\mathbf{x}_c \\ \mathbf{w}_a^T\mathbf{x}_c + \mathbf{w}_c^T\mathbf{x}_a + \mathbf{w}_d^T\mathbf{x}_b - \mathbf{w}_b^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_d + \mathbf{w}_d^T\mathbf{x}_a + \mathbf{w}_b^T\mathbf{x}_c - \mathbf{w}_c^T\mathbf{x}_b \end{bmatrix} \in \mathbb{H} \tag{6.11}$$

for quaternions, and

$$y[n] = \mathbf{w}_{n-1}^T \mathbf{x}_n = \begin{bmatrix} \mathbf{w}_a^T\mathbf{x}_a - \mathbf{w}_b^T\mathbf{x}_b + \mathbf{w}_c^T\mathbf{x}_c - \mathbf{w}_d^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_b + \mathbf{w}_b^T\mathbf{x}_a + \mathbf{w}_c^T\mathbf{x}_d + \mathbf{w}_d^T\mathbf{x}_c \\ \mathbf{w}_a^T\mathbf{x}_c + \mathbf{w}_c^T\mathbf{x}_a - \mathbf{w}_d^T\mathbf{x}_b - \mathbf{w}_b^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_d + \mathbf{w}_d^T\mathbf{x}_a + \mathbf{w}_b^T\mathbf{x}_c + \mathbf{w}_c^T\mathbf{x}_b \end{bmatrix} \in \mathbb{T} \tag{6.12}$$

for tessarines, where $\mathbf{x}_n$ is the filter input vector at iteration n and $\mathbf{w}_{n-1}$ are the filter weights at iteration $n-1$: $\mathbf{x}_n = \begin{bmatrix} x[n] & x[n-1] & \cdots & x[n-M] \end{bmatrix}^T$, $\mathbf{w}_n = \begin{bmatrix} w_0[n] & w_1[n] & \cdots & w_M[n] \end{bmatrix}^T$, with M the filter length. The minimum of the cost function $J(\mathbf{w}_{n-1})$ can be found by computing and setting to zero its gradient. The final adaptation equation for both QLMS and TLMS algorithms results in

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\mathbf{x}_n^*. \tag{6.13}$$

where μ is the step size along the direction of the gradient and the conjugate $\mathbf{x}_n^*$ is derived as in (6.9). The QLMS and the TLMS algorithms will be employed in the simulations described in Par. 6.4.

6.2.1 Widely linear modification

Recent works about both complex and hypercomplex filtering showed a particular interest in widely linear algorithms [54, 60, 100, 105]. It has been observed that most real world signals are *improper* (or *noncircular*) in nature [76] and, in this case, a filter performance can be improved significantly if the full second order statistics of the signals is taken into account and included into the algorithm. We said in Par. 5.1.2 that, if a random variable has a rotation-invariant probability distribution (with respect to all six pairs of rotation axes $(\mathbf{1}, \mathbf{i})$,$(\mathbf{1}, \mathbf{j})$,$(\mathbf{1}, \mathbf{k})$, $(\mathbf{i}, \mathbf{j})$, $(\mathbf{k}, \mathbf{j})$, $(\mathbf{k}, \mathbf{i})$), it must be considered *proper*, or second-order circular. We recall from Par. 5.1.2 that signal properness in $\mathbb{H}$ can be verified by checking if the following properties hold for a quaternion random variable $q = q_a + q_b\mathbf{i} + q_c\mathbf{j} + q_d\mathbf{k}$ [100]:

1. $E\{q_m^2\} = \sigma^2,\ \forall m = a, b, c, d$
2. $E\{q_m q_n\} = 0,\ \forall m, n = a, b, c, d$ and $m \neq n$
3. $E\{qq\} = -2E\{q_m^2\} = -2\sigma^2,\ \forall m = a, b, c, d$
4. $E\left\{|q|^2\right\} = 4E\{q_m^2\} = 4\sigma^2,\ \forall m = a, b, c, d$

We said in Par. 5.1.2 that since for a quaternion proper signal the complementary covariance matrices, defined as $\mathcal{C}_{\mathbf{q}}^{\mathbf{i}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{i}H}\right\}, \mathcal{C}_{\mathbf{q}}^{\mathbf{j}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{j}H}\right\}, \mathcal{C}_{\mathbf{q}}^{\mathbf{k}} = E\left\{\mathbf{q}\mathbf{q}^{\mathbf{k}H}\right\}$, vanish, widely linear algorithms incorporate and exploit this second order information on purpose. It is useful to report about the quaternion *involutions* here and make a comparison with tessarine involution later in this chapter:

$$\begin{aligned} q^{\mathbf{i}} &= -\mathbf{i}q\mathbf{i} = q_a + \mathbf{i}q_b - \mathbf{j}q_c - \mathbf{k}q_d \\ q^{\mathbf{j}} &= -\mathbf{j}q\mathbf{j} = q_a - \mathbf{i}q_b + \mathbf{j}q_c - \mathbf{k}q_d \\ q^{\mathbf{k}} &= -\mathbf{k}q\mathbf{k} = q_a - \mathbf{i}q_b - \mathbf{j}q_c + \mathbf{k}q_d. \end{aligned} \tag{6.14}$$

Involutions are functions $f(.)$ chosen in a way that, given $q, p \in \mathbb{H}$, we have the conditions:

1. $f(f(q)) = q$
2. $f(q+p) = f(q) + f(p)$ and $f(\lambda q) = \lambda f(q)$
3. $f(qp) = f(q)f(p)$

In Par. 5.3, we saw that the WL-QLMS algorithm updates four sets of filter weights: $\mathbf{w}$, $\mathbf{h}$, $\mathbf{u}$, $\mathbf{v} \in \mathbb{H}^{M\times 1}$, where M is the filter length. Accordingly, the filter output is computed by convoluting each weight vector with its corresponding input involution:

$$y_w[n] = \mathbf{w}_{n-1}^T \mathbf{x}_n, \quad y_h[n] = \mathbf{h}_{n-1}^T \mathbf{x}^{\mathbf{i}}{}_n, \quad y_u[n] = \mathbf{u}_{n-1}^T \mathbf{x}^{\mathbf{j}}{}_n, \quad y_v[n] = \mathbf{v}_{n-1}^T \mathbf{x}^{\mathbf{k}}{}_n \tag{6.15}$$

and summing all four contributions:

$$y[n] = y_w[n] + y_h[n] + y_u[n] + y_v[n]. \tag{6.16}$$

In conclusion, we have four adaptation equations:

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}_{n-1} + \mu e[n]\,\mathbf{x}_n^*, \quad \mathbf{h}_n = \mathbf{h}_{n-1} + \mu e[n]\,\mathbf{x}_n^{\mathbf{i}*} \\ \mathbf{u}_n &= \mathbf{u}_{n-1} + \mu e[n]\,\mathbf{x}_n^{\mathbf{j}*}, \quad \mathbf{v}_n = \mathbf{v}_{n-1} + \mu e[n]\,\mathbf{x}_n^{\mathbf{k}*}. \end{aligned} \tag{6.17}$$

Is it possible to obtain a similar algorithm with tessarines? Well, if we apply the three conditions above in order to find tessarine involutions we obtain the following results ($t \in \mathbb{T}$):

$$\begin{aligned} t^{\mathbf{i}} &= -\mathbf{i}t\mathbf{i} = t_a + \mathbf{i}t_b + \mathbf{j}t_c + \mathbf{k}t_d = t \\ t^{\mathbf{j}} &= +\mathbf{j}t\mathbf{j} = t_a + \mathbf{i}t_b + \mathbf{j}t_c + \mathbf{k}t_d = t \\ t^{\mathbf{k}} &= -\mathbf{k}t\mathbf{k} = t_a + \mathbf{i}t_b + \mathbf{j}t_c + \mathbf{k}t_d = t. \end{aligned} \tag{6.18}$$

From (6.18), we see that tessarines are auto-involutive, so a widely linear model is possible to the extent that it is defined the same way as for complex numbers [78].

6.3 Microphone array geometries and mathematical representation of space

The experiments we are going to present in this chapter make use of two different microphone configurations to pick up a sound source. Both layouts mount four microphones and each microphone signal is assigned to a quaternion/tessarine component. The first microphone configuration is the coincident Ambisonics B-Format array (Fig. 6.1), presented in Chapter 4. The second is a simple 4-element uniform linear array (Fig. 6.2).

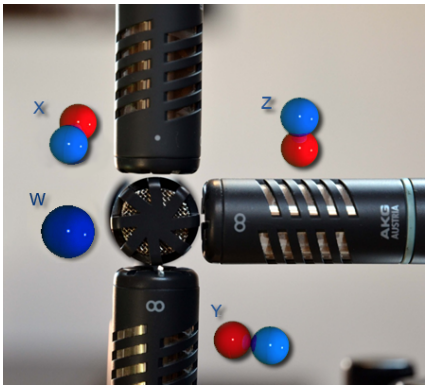


Figure 6.1. Typical Ambisonic B-Format layout.

Figure 6.2. 4-Microphone Uniform Linear Array.

Besides the evident different shape of the arrays, a transformation of the sound space is applied before placing each signal into a mathematical dimension. Details about this transformation are given just below.

6.3.1 Ambisonic coincident array

We saw in Chapter 4 that Ambisonics is a renowned 3D audio technique [28–30] that can be used for either recording or reconstructing a 3-dimensional sound field. Ambisonics typically uses coincident microphone arrays with a configuration depending on the specific *format*. Ambisonic formats were tailored to a special audio equipment or purpose (studio recording, audio broadcasting, public address, etc.) and it is possible to transcode from one format to another by means of uncomplicated matrix transformations [81].

We still focus here on the Ambisonic *B-Format*. The choice and the arrangement of the microphones in this format are in line with the physical decomposition of the sound field according to the Ambisonic theory. Ambisonics describes the sound field, $p(\vec{r})$, as a linear combination of *spherical harmonics* (Y_{mn}^{σ}), multiplied by coefficients representing the recorded audio signals (B_{mn}^{σ}):

$$p(\vec{r}) = \sum_{m=0}^{\infty} (2m+1)\, j^m J_m(kr) \sum_{\substack{0 \le n \le m, \\ \sigma = \pm 1}} B_{mn}^{\sigma} Y_{mn}^{\sigma}(\theta, \varphi) \tag{6.19}$$

where m, n, σ, k are the degree, the order, the spin and the wave number ($2\pi f/c$), respectively. The polar coordinates (θ, φ) denote the azimuth and the elevation. The decomposition in (6.19) refers to a plane wave and a sound field with external sources only. The other functions in the formula, $J_m(kr)$, are radial functions called *spherical Bessel functions of the first kind* [19]. The *1st-order B-Format* considers harmonics up to first order. Graphically, the zeroth and first order harmonics are represented by the colored bubbles in Fig. 6.1. The microphones used in this technique (coincident and orthogonal to one another) must have polar patterns fitting the shape of these harmonics. For this reason, one omnidirectional microphone (W) and three figure-of-eight microphones (X,Y,Z) are adopted here.

A transformation of the sound field from the traditional Euler representation into a quaternion-valued form has been already discussed and experimented by the author in Chapter 4 and in [66, 69, 71]. It is worth recalling and pointing out that the group of 3D Euclidean rotations SO(3) has a representation on the $(2m+1)$-dimensional Hilbert space with spherical harmonics, span $\{Y_{mn}^{\sigma}(\theta, \phi), 0 \le n \le m, \sigma = \pm 1\}$. In addition, the SO(3) group is isomorphic to the subspace of full-imaginary quaternions. That said, the transformation is possible and straightforward. In conclusion, 1st-order B-format can be encapsulated into a quaternion form as

$$B^{Q}[n] = B_W[n] + B_X[n]\mathbf{i} + B_Y[n]\mathbf{j} + B_Z[n]\mathbf{k}. \tag{6.20}$$

Later in this chapter, we will see if a similar transformation is effective in tessarine mathematics, too.

6.3.2 Uniform Linear Array

The second microphone configuration we present here consists of four uniformly spaced microphones, aligned on a single space dimension (Fig. 6.2). The microphone polar pattern may be of any kind. It is preferable that all microphones have the same directional characteristic. Such arrays are usually employed in highly directive beamforming applications, such as sound source localization. System directivity depends on the number of sensors, their spacing and the wavelength of the impinging wave. The spatial position of each microphone in our 4-element array is defined as

$$r_p = [(p-1)d \;\; 0 \;\; 0]^T, \quad \text{for } p = 1, ..., P \tag{6.21}$$

where d is the distance between two microphones and $P = 4$. In a uniform linear array (ULA) the delay time between sensors is $\tau = \frac{d \cos\theta}{c}$, where θ is the angle of incidence of the acoustic wave. The ULA array steering vector can be expressed as

$$\mathbf{a} = \begin{bmatrix} 1 & e^{jkd\cos\theta} & \dots & e^{j(1-P)kd\cos\theta} \end{bmatrix}^T \tag{6.22}$$

with $P = 4$ and $k = \omega/c$ (c is the speed of sound). Given the source signal $s(t)$, the sound signal at the p-th microphone can be represented as

$$x_p(t) = s(t - (p-1)\tau). \tag{6.23}$$

Since we have four microphone signals and they are correlated to one another, we may want to assign each signal to a quaternion or a tessarine component. No particular space transformation is applied here. We are just encapsulating signals into one single multidimensional object, because they belong to the same context:

$$x^{Q,T}[n] = x_1[n] + x_2[n]\mathbf{i} + x_3[n]\mathbf{j} + x_4[n]\mathbf{k}. \tag{6.24}$$

6.4 Simulations

In this paragraph, we propose two examples in order to make a performance comparison between quaternion and tessarine filtering according to the input signals. The simulation layout is represented in Fig. 6.3. In both simulations we have a system $\mathbf{w}_0$ to be identified, which is defined in the time domain by a set of random weights, uniformly distributed in the range $[-1, 1]$.

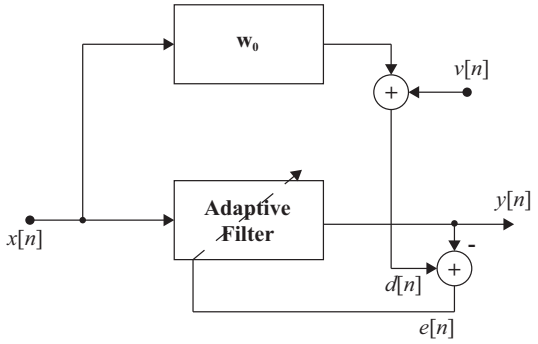


Figure 6.3. Simulation layout.

6.4.1 Generic circular input signals

In this first example, we apply QLMS and TLMS in a context where the input signal $x[n]$ is considered as either a quaternion-valued or tessarine-valued colored noise with unit variance and it was obtained by filtering the white Gaussian noise $\eta[n]$ as $x[n] = bx[n-1] + \frac{\sqrt{1-b^2}}{\sqrt{4}}\eta[n]$, where b is a filtering parameter (here it was chosen as $b = 0.7$). The additive signal $v[n]$ is defined the same way as $x[n]$, but the parameter b is set to zero.

Signal $x[n]$ is circular, all its components are uncorrelated to each other, so, at first glance, it seems to be equivalent to consider it as a quaternion or a tessarine. In effect, our results are concordant with the expectations (Fig. 6.4): given the same filter parameters ($M = 12$, $\mu = 0.008$), the QLMS and TLMS exhibit the same MSE. In this simulation, after 5000 samples, the weights $\mathbf{w}_0$ change abruptly. The two filters run after the variation with the same rate.

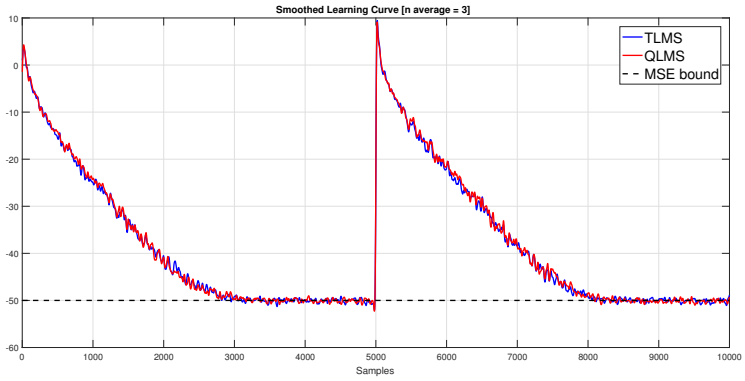


Figure 6.4. MSE: QLMS vs TLMS with proper input signal.

6.4.2 Ambisonic improper audio input signals

The second example we propose here makes use of a well-structured 3D audio input signal. This signal has 4 components which were recorded by 4 microphones in accordance with the 3D audio technique called Ambisonics (B-Format). The first order Ambisonic B-Format technique mounts 4 coincident microphones, orthogonal to one another: one omnidirectional microphone (W) and three figure-of-eight microphones (X, Y, Z). Each microphone signal can be assigned to a 4-dimensional algebra component as

$$x[n] = x_W[n] + x_X[n]\mathbf{i} + x_Y[n]\mathbf{j} + x_Z[n]\mathbf{k}. \tag{6.25}$$

However, we want to prove that this assignment is not merely a matter of convenience. In fact, Fig. 6.5 shows how the choice of a different algebra, defining the mathematical space, determines the filter performance. Giving an interpretation of Fig. 6.5, we understand that a B-Format signal is rather inclined to be represented by quaternion algebra than by tessarines (the QLMS converges faster than TLMS on equal terms). In truth, in previous works [68, 71], the authors found a relation between the sound field as decomposed by Ambisonics and a quaternion-valued representation. It is known that the subspace of pure quaternions (those quaternions with null real component) is isomorphic to rotations SO(3) which can be represented by spherical harmonics. That said, the quaternion representation of Ambisonics does not simply consist in a compact formalism, but it has a physical and geometrical meaning.

In our simulation, the source is a monodimensional unit-variance white Gaussian noise in a computer-generated anechoic room. The source was placed at a distance of 20 cm from the B-Format array, 45° off-axis with the X microphone. Additive unit-variance white Gaussian noise $\nu[n] \in \mathbb{H}$, with $n = 0, 2..., P-1$, was summed to the output signal of the system to be identified ($d[n] = \mathbf{w}_0^T\mathbf{x} + \nu[n]$). The filter parameters were chosen as $M = 12$, $\mu = 0.3$.

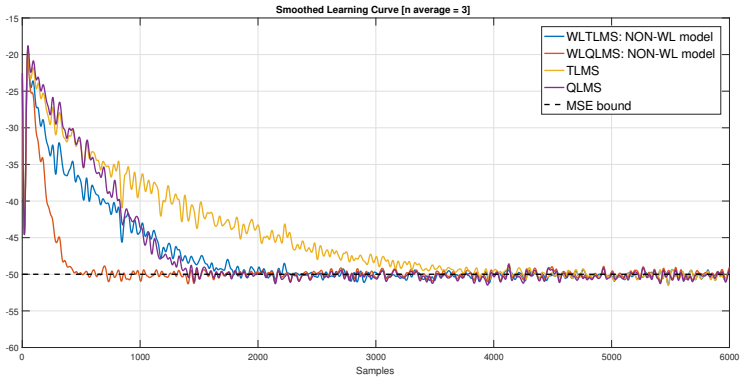


Figure 6.5. MSE: (WL-)QLMS vs (WL-)TLMS with Ambisonic improper input signal. *NON-WL model* in the legend box refers to a system to be identified which only has $\mathbf{w}_0$ weights.

In addition, in Par. 6.2.1, we emphasized the possibility to build a widely linear algorithm (WL-QLMS, WL-TLMS). Since the Ambisonic B-Format is improper, we expect the WL-QLMS and WL-TLMS to outperform QLMS and TLMS, respectively. The results from our simulation meet the expectations (Fig. 6.5).

6.4.3 Microphone array geometry

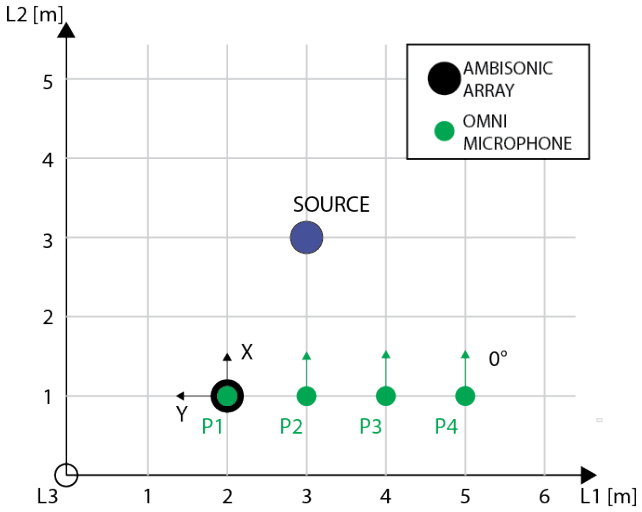


Figure 6.6. Simulation room scenario.

In this experiment, we chose two different microphone configurations having a precise geometry. The microphones are set up as described in Par. 6.3.1 and Par. 6.3.2. The ULA array is composed of omnidirectional microphones. The four signal components are correlated in both cases. The positions of the source and the microphones in the room are described in Fig. 6.6. The resulting impulse responses, characterizing the path between the source and each sensor, in the case of Ambisonics and ULA are reported in Fig. 6.7 and Fig. 6.8, respectively.

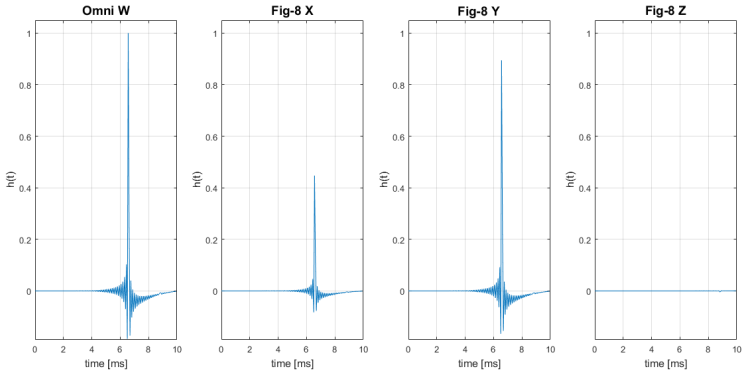


Figure 6.7. B-Format impulse responses.

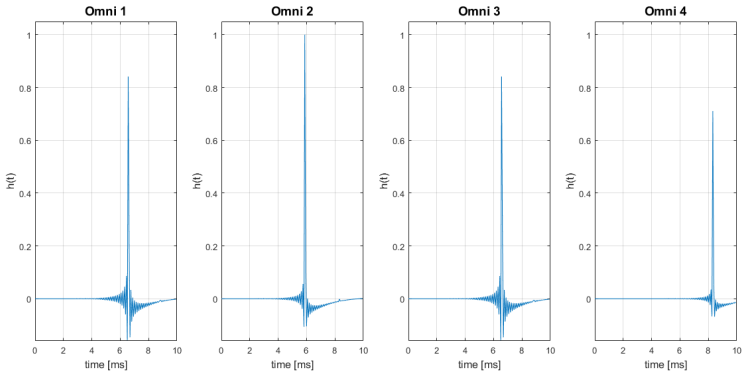


Figure 6.8. Uniform Linear Array impulse responses (omnidirectional mics).

The experiment consists in the identification of a 4-dimensional system, previously defined in the time domain by the random weights $\mathbf{w}_0$, uniformly distributed in the range $[-1, 1]$. The adaptive core accomplishing this task is one of the algorithms presented in Par. 6.2. The filter input is recorded in the two techniques. The source generates unit-variance white Gaussian noise. Given the same filter parameters for both QLMS and TLMS ($M = 12, \mu = 0.8$), we obtained the results plotted in Fig. 6.9 and Fig. 6.10 with B-Format and ULA, respectively. We considered as a measure of comparison the MSE at the steady-state and the time (in terms of samples) the algorithm takes to reach this value.

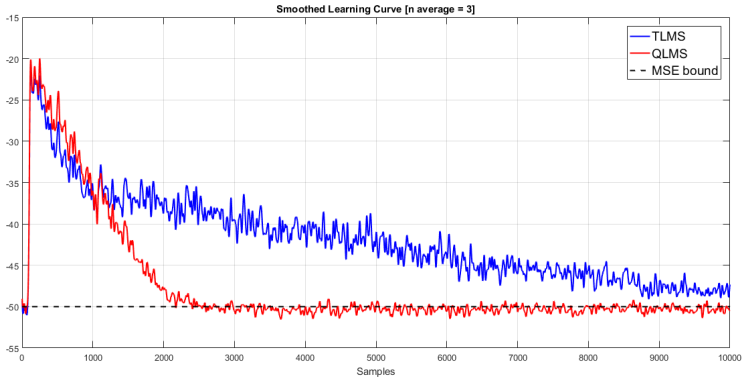


Figure 6.9. System identification with B-Format input signal. QLMS and TLMS filter performance.

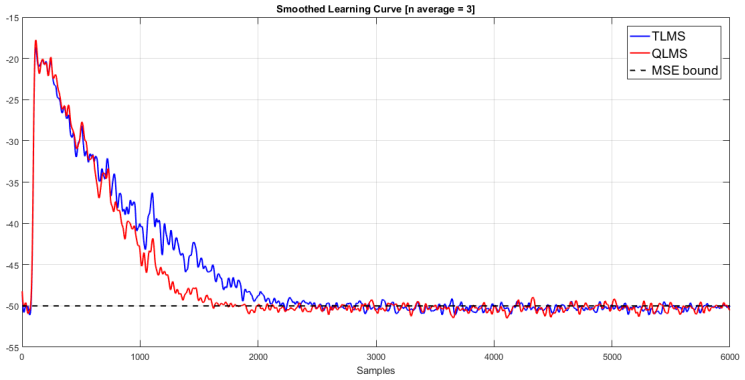


Figure 6.10. System identification with ULA (omnidirectional mics) input signal. QLMS and TLMS filter performance.

In both cases, the QLMS has an edge over TLMS. However, when the input signal is consistent with the quaternion transformation we discussed in Par. 6.3.1, this advantage is distinct. Apparently, a similar transformation is not feasible with tessarines, since, unlike rotations and quaternions, tessarine algebra is commutative. In the example above, geometry wins over mere correlation. Nevertheless, we are interested in observing whether it is possible to turn the situation around. We present here a minor modification of the scenario. Keeping the position of microphone P1 unchanged, the omnidirectional ULA microphones are moved closer to one another ($d = 0.5$ [m]). The impulse responses are shown in Fig. 6.11 and the result from the simulation is plotted in Fig. 6.12. Here, the TLMS converges faster than the QLMS. In conclusion, correlation matters, but geometry is preponderant. In fact, geometry implies a particular configuration of correlation.

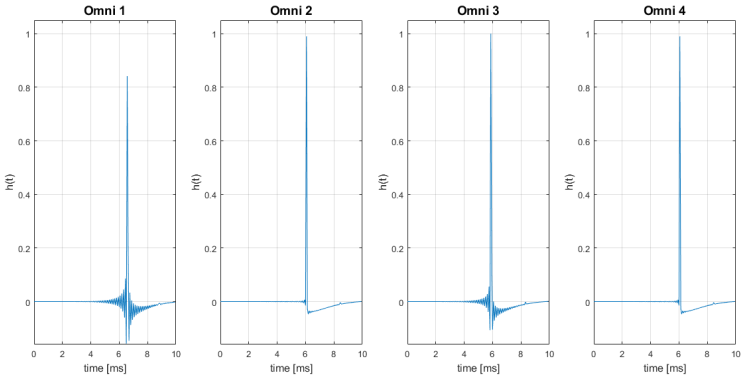


Figure 6.11. Uniform Linear Array impulse responses (closer omni mics).

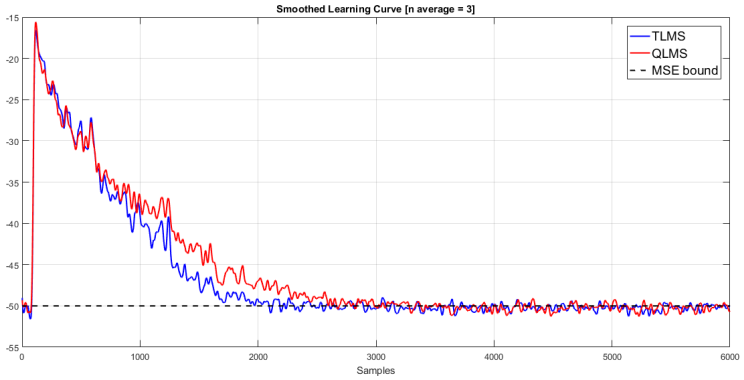


Figure 6.12. System identification with ULA (closer omni mics) input signal. QLMS and TLMS filter performance.

6.5 Tessarine algorithms in the frequency domain

One of the most important differences between quaternion and tessarine algebras concerns product commutativity. A consequence of this is the possibility to define a tessarine convolution theorem which *formally* equals the traditional real- and complex-valued therem and is commutative:

$$y[n] = w[n] * x[n] \Leftrightarrow Y(\omega) = W(\omega)X(\omega) = X(\omega)W(\omega). \tag{6.26}$$

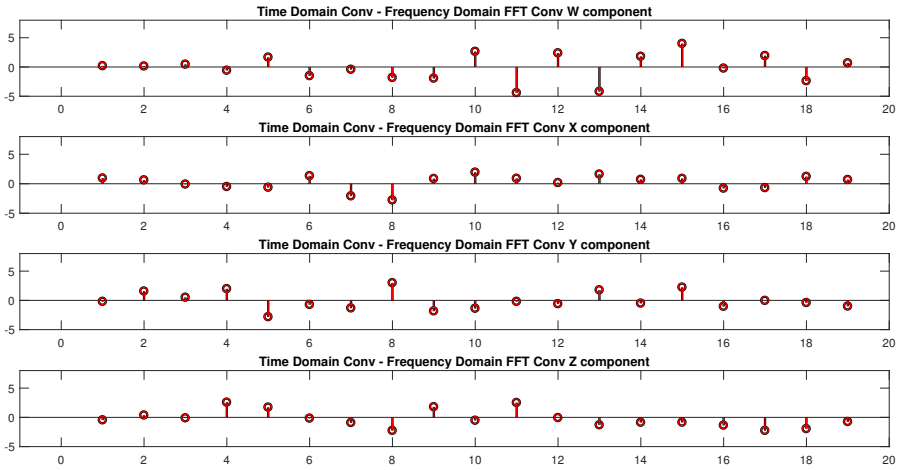


Figure 6.13. Tessarine convolution theorem: time domain output (red circles) and anti-transformed frequency domain output (black circles) coincide.

In Par. 3.2, we saw that the convolution theorem in quaternion algebra is more complicated:

$$\mathbf{Y}_k = \mathbf{W}_k^a \mathbf{X}_k + \mathbf{W}_k^b \boldsymbol{\nu_2} \mathbf{X}_{-k}, \tag{6.27}$$

where $\mathbf{X}_{-k} \equiv \mathbf{X}_k(-\omega)$ is the frequency reversed signal of $\mathbf{X}_k(\omega)$, while $\mathbf{W}_k^a$ and $\mathbf{W}_k^b$ are the simplex and perplex parts such that $\mathbf{W}_k = \mathbf{W}_k^a + \mathbf{W}_k^b \boldsymbol{\nu_2}$.

6.5.1 Tessarine Fourier Transform

In tessarine algebra, it is still possible to compute the Fourier transform by means of Cayley-Dickson decomposition in a way similar to quaternions (see Par. 2.1.1):

$$\begin{aligned} F(u) = &\sum_{n=0}^{N-1} \mathrm{e}^{\left(-2\pi \boldsymbol{\nu} \frac{nu}{N}\right)} \left(w_1(n) + x_1(n) \boldsymbol{\nu_1}\right) \\ &+ \sum_{n=0}^{N-1} e^{\left(-2\pi \boldsymbol{\nu} \frac{nu}{N}\right)} \left(y_1(n) + z_1(n) \boldsymbol{\nu_1}\right) \boldsymbol{\nu_2}. \end{aligned} \tag{6.28}$$

The versors $(\boldsymbol{\nu_2}, \boldsymbol{\nu_3})$ must be chosen in a way that $\boldsymbol{\nu_1} \perp \boldsymbol{\nu_2} \perp \boldsymbol{\nu_3}$, $\boldsymbol{\nu_1}\boldsymbol{\nu_2} = \boldsymbol{\nu_3}$ and $\boldsymbol{\nu_1}\boldsymbol{\nu_2}\boldsymbol{\nu_3} = -1$. Unlike quaternion transforms, left and right transforms coincide.

6.5.2 Overlap-Save Tessarine Frequency Domain Adaptive Filter

Thanks to commutative product, the Overlap-Save Tessarine Frequency Domain Adaptive Filter (OS-TFDAF) has the same architecture of real- and complex-valued OS-FDAF [104]. The algorithm block diagram is shown in Fig. 6.14.

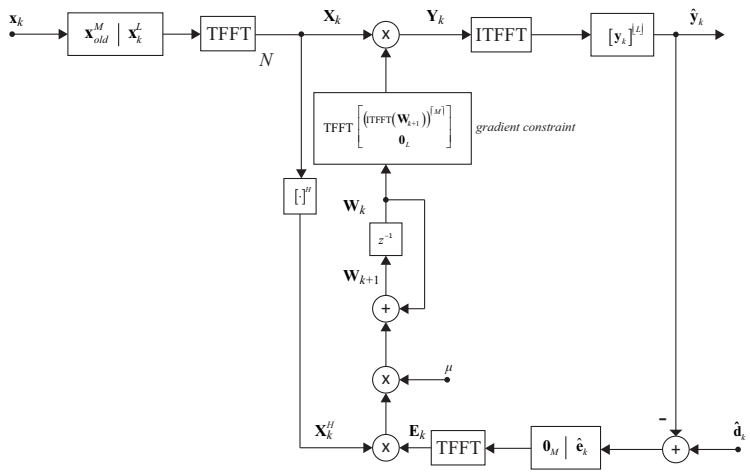


Figure 6.14. Overlap-Save Tessarine Frequency Domain Adaptive Filter. Block diagram.

A brief overview of the algorithm is given just below.
Algorithm initialization:

$$\begin{aligned} &\mathbf{W}_{init} = \mathbf{0} \text{ (2M-by-1 null vector)} \\ &\mu = \mu_0 \end{aligned} \tag{6.29}$$

For each new input block k do:

$$\mathbf{X}_k = \text{diag}\left\{ \text{TFFT}\left[\mathbf{x}_{old}^{M} \quad \mathbf{x}_{k}^{L} \right]^{T} \right\} \tag{6.30}$$

where the input block consists of $\mathbf{x}_{old}^{M}$ and $\mathbf{x}_{k}^{L}$, defined as

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

Compute the filter output in the frequency domain and anti-transform:

$$\mathbf{Y}_k = \mathbf{W}_k \mathbf{X}_k. \tag{6.31}$$

$$\hat{\mathbf{y}}_k = [\text{ITFFT}\,(\mathbf{Y}_k)]^{\lfloor L \rfloor} \tag{6.32}$$

Given the desired output vector $\hat{\mathbf{d}}_k$ in the time domain, compute and transform the error vector:

$$\hat{\mathbf{d}}_k = \left[d\,(kL) \quad d\,(kL+1) \quad \cdots \quad d\,(kL+L-1) \right]^{T} \tag{6.33}$$

$$\hat{\mathbf{e}}_k = \hat{\mathbf{d}}_k - \hat{\mathbf{y}}_k \tag{6.34}$$

$$\mathbf{E}_k = \text{TFFT}\left(\left[0_M \quad \hat{\mathbf{e}}_k \right]^{T} \right). \tag{6.35}$$

Finally, update the filter weights and apply the contraint on the gradient

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu \mathbf{X}_k^H \mathbf{E}_k \tag{6.36}$$

$$\mathbf{W}_{k+1} = \text{TFFT}\left(\begin{array}{c} [\text{ITFFT}(\mathbf{W}_{k+1})]^{\lceil M \rceil} \\ \mathbf{0}_L \end{array} \right). \tag{6.37}$$

6.5.3 Simulations

It is possible to verify that the OS-TQFDAF can reach the same performance capabilites of TLMS. In the simulation scenario depicted in Fig. 6.3, we choose the filter parameters as $M = 12$ (in both TLMS and OS-TFDAF), $L = 16$, $\mu = 8 \times 10^{-3}$. The input signal $x[n]$ is unit-variance colored Gaussian noise defined as $x[n] = 0.5x[n-1] + \frac{\sqrt{1-0.5^2}}{\sqrt{4}}\eta[n]$, where $\eta[n] \in \mathbb{H}$, with $n = 0, 1, ..., P-1$, is a unit-variance white Gaussian noise. The resulting learning curves are reported in Fig.6.15.

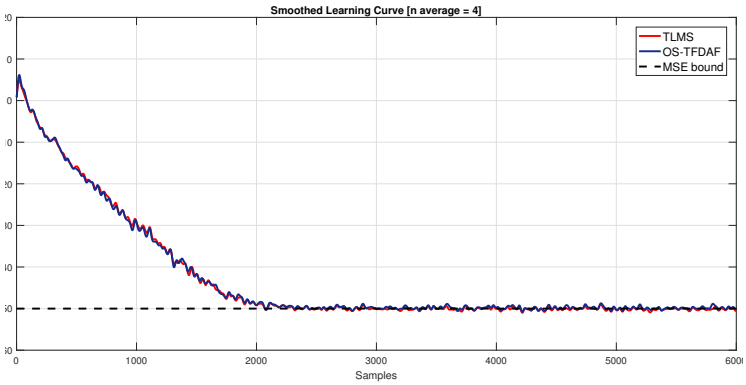


Figure 6.15. MSE: TLMS vs OS-TFDAF ($M = 12$, $L = 16$, $\mu = 8 \times 10^{-3}$).

Chapter 7

Hypercircuits

Contents

7.1 The problem of discrete-time hypercircuits

In the previous chapters, we said that a higher-dimensional numerical space allows the description of hypercomplex objects and problems, highlighting typical aspects visible only within a hypercomplex field. As known from the experience with complex numbers, the complex plane made it possible to visualize phenomena like the electric and magnetic storage and release of charges in capacitors and inductors, respectively, or to treat signals by considering their phase relations, thus helping the development of all those applications concerning signal processing, radio technique, electronics and so on. The question now is: what results are achievable by means of hypercomplex algebras? These algebras, quaternions in particular, raised the interest of digital signal processing engineers. In digital signal processing, the most popular approach considers filters as circuits. Even though digital filters are non-physical structures, it is still possible to define constitutive relations for component elements and operate computations within the circuit topology in the digital domain.

During the last few years, the efforts of our research were oriented towards the study of the properties of quaternion algebra, with the aim of developing digital adaptive filters in the quaternion domain. However, adaptive filters have self-adjusting structures and the engineer intervention is restricted to the architecture design and the initial parameter setup. At every step, the adaptive filter finds a configuration which gets closer to the optimal solution. This is possible by means of reference signals. What about designing a static filter according to some initial specifications? Let us assume, for example, that we are asked to implement a shelving filter for audio applications in real-valued algebra. We can write the shelving filter

analogue trasfer funtion in the Laplace domain

$$H(s) = \frac{s + K\omega_a}{s + \omega_a} \tag{7.1}$$

where ω_a is the *analogue* cut-off frequency and K can be either $K > 1$ (bass-boost) or $K < 1$ (bass-cut). The filter in (7.1) can be Z-transformed in order to get a digital filter, making use of the *bilinear* transform

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{7.2}$$

where T is the sampling period, and the frequency *pre-warping*

$$\omega_d = \frac{2}{T}\arctan\left(\omega_a \frac{T}{2}\right) \tag{7.3}$$

with ω_d the *digital* frequency. This procedure is quite common in digital signal processing and it allows to decide easily for the filter slope, cut off frequency, gain/attenuation. Is it possible (and does it make sense) to follow a similar procedure in hypercomplex algebra?

The concept of discrete-time *hypercircuit* has always been known to digital signal processing engineers working with high-dimensional (hypercomplex) systems. Nevertheless, a definition of discrete-time *hypercircuit* was never given explicitly. Which are the conditions for which we can affirm that it makes sense to talk about hypercircuits? In order to give a proper answer to this question, it is better to look back upon the circuit theory and recall the basic notions.

7.2 Fundamentals of Circuit Theory

Digital circuits derive from the lumped element model of electrical circuits [16]. The idea behind the concept of circuit is to build a connection between elements, whose behaviour is characterized by a constitutive relation among electrical quantities. That said, the circuital approach becomes an easy way to describe many and different natural phenomena, since it is possible to use elementary *equivalent* electrical elements and electrical connections to represent a physical structure under analysis.

Given the fundamental notions of *connection* and *constitutive relations*, circuit modeling is based on the representation of phenomena by means of two kinds of quantities: *across* and *through*. The topological constraints in the circuit graph determine the physical properties of the system. Examples of equivalent electrical quantities for contexts different from electricity are reported in Table 7.1.

	Through	**Across**
Electricity	Current	Voltage
Fluidodynamics	Current Flux	Pressure
Mechanics	Torque	Angular Velocity

Table 7.1. Examples of equivalent electrical quantities for physical problems different from electricity.

Besides *through* and *across* quantities, it is possible to define equivalent electrical elements to describe non-electrical objects. A nice example is the application of the circuit theory to acoustics (see Table 7.2).

Quantity	Expression	Analogy
Mechanical force	F	Voltage
Velocity	u	Current
Mechanical impedence	$Z_m = \frac{F}{u}$	Impedence
Mass	M	Inductance
Mechanical flexibility	$C_m = \frac{1}{K}$	Capacitance

Table 7.2. Maxwell's mechanical analogy for acoustics.

The circuit models considered here are lumped element models, i.e. ideal circuits, made up with ideal elements whose geometrical dimensions are not taken into account, since they are considerably smaller compared to the wavelength of the physical phenomenon occurring in the circuit. In other words, the propagation time of the phenomenon is not crucial in the analysis and design of the circuit.

The physical quantities in the circuit must comply with two physical laws: the *Kirchhoff* laws. The Kirchhoff Current Law (KCL) states that "*the sum of the currents flowing into any circuit node is equal to the sum of currents flowing out of that node*" ($\sum_{n=1}^{N} I_n = 0$). The Kirchhoff Voltage Law (KVL) states that "*in a closed mesh, the sum of the voltage drops is zero*" ($\sum_{n=1}^{N} V_n = 0$). These laws derive from Maxwell laws, under the assumption that the circuit is in the steady state (i.e. $\frac{\partial \Phi_B}{\partial t} = 0$, the change of the magnetic flux in time outside the conductor is zero, $\frac{\partial q}{\partial t} = 0$, the change of the charge in time inside conducting elements is zero). These two laws allow to operate computation within the circuit topology.

7.2.1 Digital circuits

From available literature [64], we know that it is possible to extend the circuit theory to the digital world. Digital circuits are non-physical structures, but the circuital approach can still be adopted. Unlike generic electrical circuits, the concept of *cause-effect* is explicit in digital circuits. For this reason, we often talk about *uni-directional* structures. An important theorem for digital circuits states that a uni-directional time discrete circuit must not include closed loops lacking of delay elements.

In digital circuits it is still possible to define constitutive relations for elements:

$$\begin{aligned} y[n] &= Ax[n] \quad \text{element without memory} \\ y[n] &= x[n-D] \quad \text{element with memory} \end{aligned} \tag{7.4}$$

where $A \in \mathbb{H}$ is a constant (multiplier, similar to a linear resistor), $D \in \mathbb{R}$ is a delay element and $x[n], y[n] \in \mathbb{R}, \mathbb{C}, \mathbb{H}$, etc. are the *cause* and the *effect*, respectively. The nodes in the circuit are sum elements.

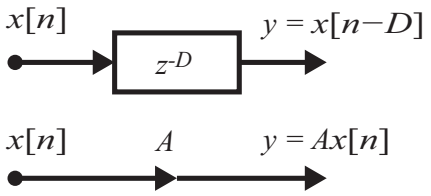


Figure 7.1. Discrete-time elements: *with* and *without memory*.

7.3 Quaternion Z-Transform

Z-Transform is a powerful mathematical tool for the analysis and the design of digital circuits. It applies to Linear Time Invariant (LTI) systems and it allows a simple expression of system transfer functions. In general, the Z-Transform is a useful instrument for studying the behavior of LTI systems.

First and foremost, because of the peculiar characteristics of quaternion algebra, it is worth recalling some basic notion about quaternion transforms, before defining the quaternion-valued Z-Transform (QZT). As shown in Par. 1.2.2, quaternion product is not commutative. This gives rise to the existence of the two-handed (left, right) and sandwich transforms we saw in Chapter 2.

We learned in Chapter 2 that it is possible to exploit the Cayley-Dickson decomposition in order to simplify the calculation of the discrete Fourier transform (QDFT) in the quaternion domain. Cayley-Dickson decomposition splits a quaternion function $f[n]$ into the combination of two complex functions (*simplex* and *perplex* parts) as

$$f[n] = (w[n] + x[n]\boldsymbol{\nu_1}) + (y[n] + z[n]\boldsymbol{\nu_1})\,\boldsymbol{\nu_2}\ . \tag{7.5}$$

The basis of versors must be chosen in a way that $\boldsymbol{\nu_1} \perp \boldsymbol{\nu_2} \perp \boldsymbol{\nu_3}$, $\boldsymbol{\nu_1}\boldsymbol{\nu_2} = \boldsymbol{\nu_3}$ and $\boldsymbol{\nu_1}\boldsymbol{\nu_2}\boldsymbol{\nu_3} = -1$. So, a quaternion transform can be split, as well:

$$F\left(e^{\boldsymbol{\nu}\omega}\right) = \sum_{n=0}^{N-1} e^{-\boldsymbol{\nu}\omega n} f\left[n\right] = F_s\left(e^{\boldsymbol{\nu}\omega}\right) + F_p\left(e^{\boldsymbol{\nu}\omega}\right)\boldsymbol{\nu_2} \tag{7.6a}$$

$$f\left[n\right] = \frac{1}{N}\sum_{u=0}^{N-1} e^{\boldsymbol{\nu}\omega n} F\left(e^{\boldsymbol{\nu}\omega}\right) = f_s\left[n\right] + f_p\left[n\right]\boldsymbol{\nu_2} \tag{7.6b}$$

Equations (7.6a) and (7.6b) represent the *left* QDFT and *left* inverse QDFT of $f[n]$. Of course, this kind of decomposition is valid thanks to the linearity property of the Fourier transform. Versor $\boldsymbol{\nu}$ is an arbitrarily chosen pure unitary quaternion versor. Besides the *left-handed* Fourier transform, there exists a *right-handed* transform. The two forms are transpose with one another and definitions are summed up in Table 7.3. The table reveals in advance that we have *left* and *right* Z-Transforms, as well.

As we can find in [64], Z-Transform can be defined from the discrete Fourier transform of a sequence $f[n]$. Actually, the DFT is a special case of the Z-Transform. From (7.6a), we can define the quaternion *left* Z-Transform (coincident with the *left* DFT) by substitution of the quaternion-valued variable $z = e^{\boldsymbol{\nu}\omega}$:

$$F\left(z\right) = \sum_{n=0}^{N-1} z^{-n} f[n] = F_s\left(z\right) + F_p\left(z\right)\boldsymbol{\nu_2}. \tag{7.7}$$

Table 7.3. Kernel Definitions for Monodimensional Quaternion Transforms

Transform	Left	Right
QDFT	$e^{-\nu\omega x} f[\cdot]$	$f[\cdot]\, e^{-\nu\omega x}$
QZT	$z^{-n} f[\cdot]$	$f[\cdot]\, z^{-n}$

More generally, we can choose $z = re^{\nu\omega}$, where $r \in \mathbb{R}$ is the radius of a *hypersphere*. Those points belonging to the unit hypersphere ($r = 1$, $|z| = 1$) are related to the Fourier transform. An n-dimensional *hypersphere* is the locus of points $x_1, \cdots, x_n$ such that

$$x_1^2 + x_2^2 + \cdots + x_n^2 \leq r^2 \tag{7.8}$$

where r is the radius of the hypersphere.

A linear system can be represented on the basis of an autoregressive moving-average equation [64]:

$$\sum_{p=0}^{N} y[n-p]a_p = \sum_{q=0}^{M} x[n-q]b_q \ . \tag{7.9}$$

Taking the Z-Transform of all terms, we have:

$$Y(z)\sum_{p=0}^{N} z^{-p}a_p = X(z)\sum_{q=0}^{M} z^{-q}b_q \ . \tag{7.10}$$

Rearranging (7.10), we get the system *transfer function*:

$$\begin{aligned} H(z) = \frac{Y(z)}{X(z)} &= \frac{\sum_{q=0}^{M} z^{-q}b_q}{\sum_{p=0}^{N} z^{-p}a_p} \\ &= \frac{b_0 + z^{-1}b_1 + z^{-2}b_2 + \cdots + z^{-M}b_M}{a_0 + z^{-1}a_1 + z^{-2}a_2 + \cdots + z^{-N}a_N}. \end{aligned} \tag{7.11}$$

The validation of (7.10) is possible by retracing the proof of the *Time Shifting* property of Z-Transform. We intend demonstrating in $\mathbb{H}$ that

$$\mathcal{Z}\{y[n-p]\} = z^{-p}Y(z) \tag{7.12}$$

where $n, p \in \mathbb{R}$, $y[n] \in \mathbb{H}$. The left Z-Transform of $y[n-p]$ is computed as

$$\mathcal{Z}\{y[n-p]\} = \sum_{n=-\infty}^{\infty} z^{-n}y[n-p]. \tag{7.13}$$

We operate the substitution of the variable $m = n - p$, so that $n = p + m$ and

$$\mathcal{Z}\{y[n-p]\} = \sum_{m=-\infty}^{\infty} z^{-p}z^{-m}y[m] = z^{-p}Y(z). \tag{7.14}$$

Equation (7.14) justifies (7.10).

7.3.1 Examples: Design of a quaternion digital filter

In this subsection, we will discuss about the implementation and analysis of a digital quaternion filter. We propose a very simple numerical exercise aimed at analyzing a quaternion IIR transfer function:

$$H(z) = \frac{b_0}{1 - a_1 z^{-1}}. \tag{7.15}$$

Alternatively, we can express the filter in (7.15) as

$$y[n] = a_1 y[n-1] + b_0 x[n]. \tag{7.16}$$

Equations (7.15) and (7.16) refer to the circuit in Fig. 7.2.

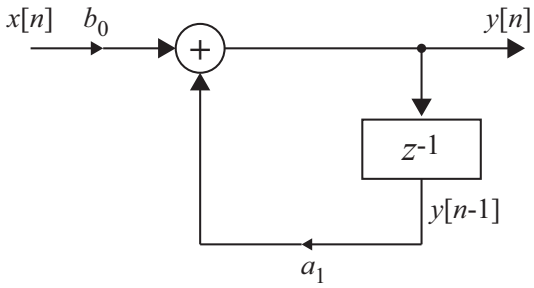


Figure 7.2. Simple IIR circuit diagram.

We assume the filter has coefficients $b_0 = 1 + 1\mathbf{i} + 1\mathbf{j} + 1\mathbf{k}$ and $a_1 = -0.1 - 0.1\mathbf{i} + 0.5\mathbf{j} - 0.5\mathbf{k}$. In this case, the filter is stable. Please note that the pole in a_1 has module $|z_1| = 0.7211 < 1$ and lies inside the unit hypersphere. The filter response components (in the frequency domain) are plotted in Fig. 7.3.

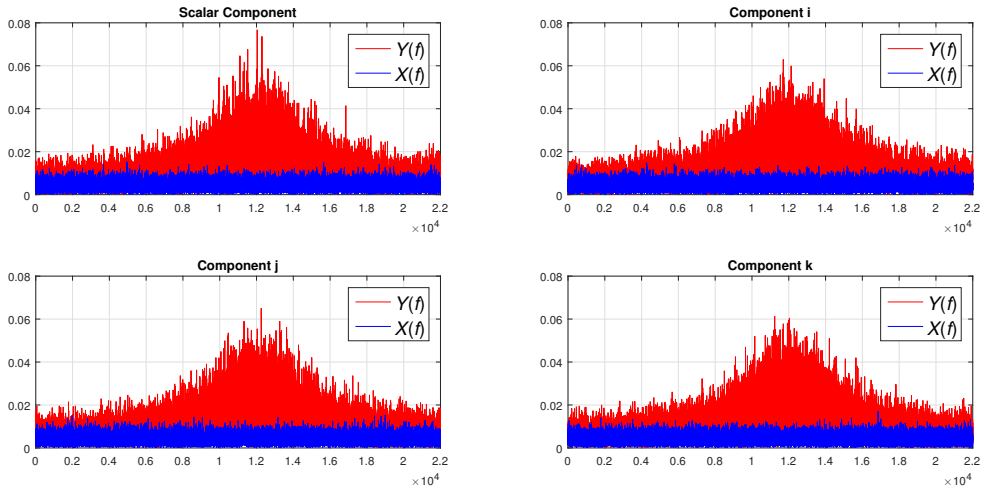


Figure 7.3. Stable quaternion IIR filter response (7.15) with $b_0 = 1 + 1\mathbf{i} + 1\mathbf{j} + 1\mathbf{k}$ and $a_1 = -0.1 - 0.1\mathbf{i} + 0.5\mathbf{j} - 0.5\mathbf{k}$. $X(f)$ is the filter input (white Gaussian noise). $Y(f)$ is the filter output.

If now we assume the filter has coefficients $b_0 = 1 + 1\mathbf{i} + 1\mathbf{j} + 1\mathbf{k}$ and $a_1 = -0.1 - 0.1\mathbf{i} + 1\mathbf{j} + 0.5\mathbf{k}$, the pole in a_1 has module $|z_1| = 1.1269 > 1$. Consequently, the pole lies outside the unit hypersphere and we expect the filter to be unstable. The filter behaviour can be understood well in the time domain in Fig. 7.4.

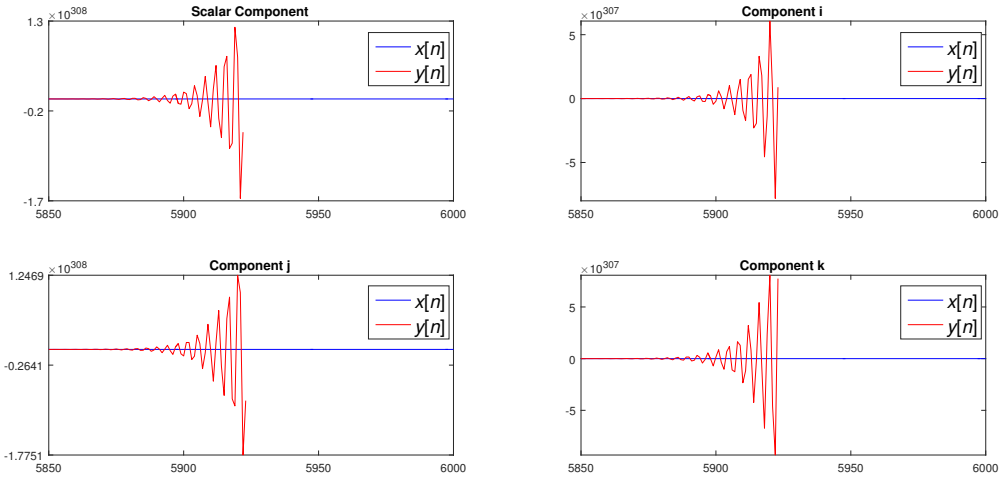


Figure 7.4. Unstable quaternion IIR filter response (7.15) with $b_0 = 1 + 1\mathbf{i} + 1\mathbf{j} + 1\mathbf{k}$ and $a_1 = -0.1 - 0.1\mathbf{i} + 1\mathbf{j} + 0.5\mathbf{k}$. $x[n]$ is the filter input (white Gaussian noise). $y[n]$ is the filter output.

The results obtained from this simple exercise still have to formalized and a proper and complete theory still has to be enunciated.

Chapter 8

Hypercomplex Adaptive Filtering Applications

Contents

8.1 Quaternion-valued Adaptive Filtering for 3D Audio Equalization

In today's audio systems we can take advantage of adaptive filtering in order to correct diverse acoustic flaws in a room electronically, i.e. repair holes in the room frequency response, level off undesired peaks and control vibrations and noise, as well. The same method can be exploited when the flaws concern loudspeakers and microphones.

Equalization can be accomplished in many ways. The simplest layout in an adaptive equalizer exploits a single sensor and this system is typically known as single-point equalizer [25]. Later, multiple-point equalization was proposed with the idea of controlling the room response at points away from a single-point equalization microphone. Within this approach, a certain number of microphones are deployed over the area to be controlled [25, 51]. The signals picked up by the microphones are processed by means of an adaptive algorithm. There exist many algorithms and solutions for accomplishing this task. However, the algorithms traditionally used in adaptive single- and multiple-point equalization are those derived from the Least Mean Square (LMS) algorithm, i.e. Filtered Reference-LMS (FX-LMS), Filtered Error-LMS (FE-LMS) or a modification of these algorithms [51, 85, 91].

The equalization system we propose in this work still uses multiple sensors to pick up the acoustic sources, but all sensors are placed at a single point. In fact, the proposed method exploits the 3D audio technique called *Ambisonics* and presented in Chapter 4, which uses arrays of coincident microphones (starting with arrays of four elements). Unlike traditional multiple-point techniques, Ambisonics does not suffer from cross-talk artefacts, produced by the sound coming from different directions and reaching all microphones with a difference in phase and intensity, thus causing unwanted filtering effects. Unlike single-point equalization, Ambisonics allows a full-3D sound pickup. How can Ambisonics succeed in sampling the sound field properly? Ambisonics decomposes the soundfield into a linear combination of spherical harmonics. Microphones should be chosen in a way that they fit the spherical harmonics, so that each recorded signal represents a coefficient multiplying its specific harmonic.

Underneath the physical description of the sound field with Ambisonics, we can decide for a proper mathematical representation of it. Typically, Ambisonic signals are processed separately as real-valued objects. In this work, we propose the quaternion-valued representation of Ambisonics presented in Chapter 4. The signal dimensions determine the format of the adaptive algorithm. Quaternion-valued signals require quaternion-valued filters, so our 3D equalizer embeds a quaternion version of an LMS-based algorithm. Quaternion-valued adaptive filters were already experimented in 3D audio and non-audio applications and the peculiar statistical properties of quaternion algebra make the use of quaternions a promising approach in multidimensional signal processing [68, 71, 100].

The debate about the usage of quaternion filtering in signal processing blames quaternions for being merely a sophisticated way to represent things. There are other mathematical formalisms equivalent to quaternions and they *apparently* employ only real-valued arithmetics. In reality, the quaternion algebraic rules are embedded into the real-valued format. We will show an example by building up and testing a real-valued MIMO$(4, 4)$ system to be compared with quaternions. Despite the intrinsic system equivalence, we will see how quaternion algorithms and quaternion-like MIMO algorithms do not perform the same way.

8.1.1 3D equalization

In multi-sensor equalization (either single- or multiple-point), each sensor output is subtracted from a desired signal. The method we propose here exploits quaternion algebra to represent multidimensional data as a single element and reduces multi-dimensional (multi-sensor) equalization to a quaternion-valued single-point equalization problem.

Filtered-Error QLMS

Adaptive equalization requires some adaptive filter as the core engine of the system. The algorithms commonly used in this context are the Filtered Reference LMS (FX-LMS), the Filtered Error LMS (FE-LMS) and modifications of these two algorithms. Since we are working in the quaternion domain, we apply an algorithm based on the quaternion-valued LMS (QLMS) [98]. The algorithm we adopted is the FE-QLMS

algorithm shown in Fig. 8.1.

In classic FE-QLMS, the filter inputs are a delayed version of the reference signal $x[n]$ and an error signal filtered by an inverse estimate of the delayed secondary path, namely $\hat{S}_{\Delta}^{-1}(z)$. In adaptive equalization, the secondary path $S(z)$ represents the room transfer function to be equalized. In case the secondary path transfer function $S(z)$ is minimum-phase, the delay Δ is not required ($\Delta = 0$). The ideal inverse of a function is obtained by exchanging the poles and the zeros of the original function. That said, if $S(z)$ is not minimum-phase, only the delayed inverse function exists. A rule of thumb for finding Δ is to set it empirically and check for the algorithm convergence (e.g. if the plant is minimum-phase with more poles than zeros, $\Delta = 1$ is probably a good solution; a typical choice is $\Delta = M/2$, where M is the filter length). The secondary path inverse model can be estimated by means of the QLMS algorithm as shown in Fig. 8.2. Function $P(z)$ is the desired transfer function. FE-QLMS is an error-correction based algorithm, so our target is to minimize a cost function defined as $J_n = E\{e^2[n]\}$, where $E\{.\}$ denotes the estimation. We consider the error $e[n]$ as $e[n] = d[n] - y[n]$, where

$$y[n] = \sum_{p=0}^{P-1} s[p]y'[n-p] \tag{8.1}$$

while

$$y'[n] = \sum_{m=0}^{M-1} w[m]x[n-m]. \tag{8.2}$$

We substitute (8.2) into (8.1)

$$y[n] = \sum_{p=0}^{P-1} s[p] \sum_{m=0}^{M-1} w[m]x[n-m-p]. \tag{8.3}$$

Defining $r[n] = \sum_{p=0}^{P-1} s[p]x[n-p]$, we finally get

$$y[n] = \sum_{m=0}^{M-1} w[m]r[n-m] = \mathbf{w}_n^T \mathbf{r}_n. \tag{8.4}$$

Recalling the adaptive filter weight update equation in QLMS [8, 98], we have

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e'[n]\mathbf{x}_n^* \tag{8.5}$$

where $e'[n] = \mathbf{s}_{\Delta n}^{-1T}\mathbf{e}_n$, given $\mathbf{s}_{\Delta n}^{-1}$ the delayed inverse of $\mathbf{s}_n$.

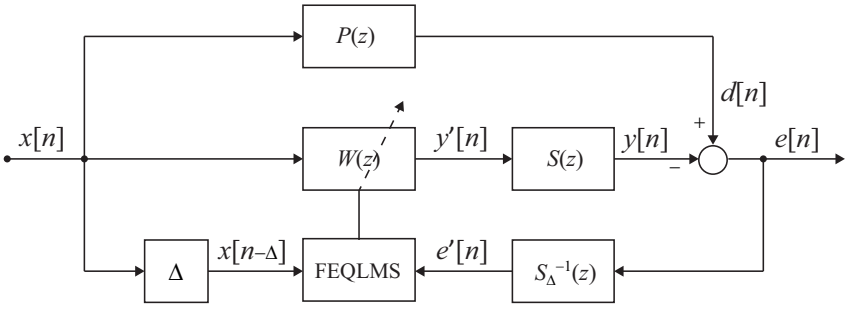


Figure 8.1. FE-QLMS algorithm block diagram.

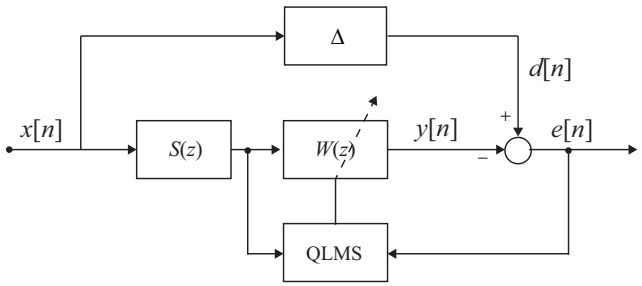


Figure 8.2. Secondary path delayed inverse model estimation.

Multichannel Filtered-Error LMS

A more typical approach in multichannel signal processing adopts MIMO systems and algorithms [43, 44]. A generic MIMO system has P inputs and Q outputs and can be represented by a matrix. A MIMO$(4, 4)$ system can have the form

$$\mathbf{H}[n] = \begin{bmatrix} \mathbf{h}_{11}[n] & \mathbf{h}_{12}[n] & \mathbf{h}_{13}[n] & \mathbf{h}_{14}[n] \\ \mathbf{h}_{21}[n] & \mathbf{h}_{22}[n] & \mathbf{h}_{23}[n] & \mathbf{h}_{24}[n] \\ \mathbf{h}_{31}[n] & \mathbf{h}_{32}[n] & \mathbf{h}_{33}[n] & \mathbf{h}_{34}[n] \\ \mathbf{h}_{41}[n] & \mathbf{h}_{42}[n] & \mathbf{h}_{43}[n] & \mathbf{h}_{44}[n] \end{bmatrix} \tag{8.6}$$

and, given an input vector $\mathbf{x}[n] = [\mathbf{x}_1[n] \ \dots \ \mathbf{x}_4[n]]^T$, we can compute the system output as

$$\mathbf{y}[n] = \mathbf{H}^T[n]\mathbf{x}[n]. \tag{8.7}$$

With such a notation and system structure, the real-valued MIMO LMS algorithm has $P \times Q$ weight adaptation equations [43, 104]. These equations are of the kind

$$\mathbf{h}_{pq}[n+1] = \mathbf{h}_{pq}[n] + \mu e_q[n]\mathbf{x}_p[n]. \tag{8.8}$$

We are now interested in making a comparison with quaternion maths. Since quaternion signal processing was introduced, scientists investigated the benefits and drawbacks of employing this algebra in algorithms. It was demonstrated in comparison with real systems that quaternion (hypercomplex) algebra considers phase information and permits to exploit the full second order statistics of signals in order to improve a filter performance [66, 97, 100, 111]. However, it is still possible to define a MIMO system which embodies the algebraic properties of quaternion algebra and keep on working with real-valued arithmetics. We can read in [2] that every associative algebra can be transformed into an isomorphic $\mathbb{K}$-valued $n \times n$ matrix algebra. In the case of quaternions we can define a 4×4 matrix of component impulse responses as

$$\mathbf{W}[n] = \begin{bmatrix} \mathbf{w}_a[n] & -\mathbf{w}_b[n] & -\mathbf{w}_c[n] & -\mathbf{w}_d[n] \\ \mathbf{w}_b[n] & \mathbf{w}_a[n] & -\mathbf{w}_d[n] & \mathbf{w}_c[n] \\ \mathbf{w}_c[n] & \mathbf{w}_d[n] & \mathbf{w}_a[n] & -\mathbf{w}_b[n] \\ \mathbf{w}_d[n] & -\mathbf{w}_c[n] & \mathbf{w}_b[n] & \mathbf{w}_a[n] \end{bmatrix} \tag{8.9}$$

which is equivalent to a quaternion impulse response $\mathbf{w}^Q[n] = \mathbf{w}_a[n] + \mathbf{w}_b[n]\mathbf{i} + \mathbf{w}_c[n]\mathbf{j} + \mathbf{w}_d[n]\mathbf{k}$. The formalism shown in (8.9) embeds the rules of quaternion

product and can be used to define a real-valued MIMO$(4,4)$ system equivalent to a quaternion system.

The question now is: is it convenient to work with the equivalent MIMO system in (8.9) in comparison with quaternion arithmetics? At first glance, we understand that we have to keep in memory 16 impulse responses instead of the 4 quaternion components. However, today's machines probably have enough computational resources to work it out. We will see in Par. 8.1.2 that, given the same filter parameters, the FE-QLMS and MIMO FE-LMS do not perform the same way. The main difference concerns the dynamics of the weight updates. We repropose here a demonstration presented in [95], re-calculated over the correct version of the QLMS algorithm (see [8]) on which our FE-QLMS algorithm is based. The weight adaptation equation in LMS has an *innovation* term which is defined as

$$\begin{aligned} \Delta\mathbf{w}^Q[n] &= \mu e[n]\mathbf{x}^*[n] \\ \Delta\mathbf{h}_{pq}[n] &= \mu e_q[n]\mathbf{x}_p[n], \quad p,q = a,b,c,d \end{aligned} \tag{8.10}$$

in QLMS and in a generic MIMO real-valued LMS, respectively. We can expand the calculation in (8.10) and obtain

$$\begin{aligned} \Delta\mathbf{w}_a^Q[n] &= \mu(e_a[n]x_a[n] + e_b[n]x_b[n] + e_c[n]x_c[n] + e_d[n]x_d[n]) \\ \Delta\mathbf{w}_b^Q[n] &= \mu(e_b[n]x_a[n] - e_a[n]x_b[n] - e_c[n]x_d[n] + e_d[n]x_c[n]) \\ \Delta\mathbf{w}_c^Q[n] &= \mu(e_c[n]x_a[n] - e_a[n]x_c[n] + e_b[n]x_d[n] - e_d[n]x_b[n]) \\ \Delta\mathbf{w}_d^Q[n] &= \mu(e_c[n]x_b[n] - e_b[n]x_c[n] - e_a[n]x_d[n] + e_d[n]x_a[n]) \end{aligned}. \tag{8.11}$$

We can associate each quaternion component with a MIMO channel, so that $e_{a,b,c,d} \equiv e_{1,2,3,4}$ and $\mathbf{x}_{a,b,c,d} \equiv \mathbf{x}_{1,2,3,4}$ and find the corrispondence below, considering the generic MIMO matrix in (8.6):

$$\begin{aligned} \Delta\mathbf{w}_a^Q[n] &= \Delta\mathbf{h}_{11}[n] + \Delta\mathbf{h}_{22}[n] + \Delta\mathbf{h}_{33}[n] + \Delta\mathbf{h}_{44}[n] \\ \Delta\mathbf{w}_b^Q[n] &= \Delta\mathbf{h}_{21}[n] - \Delta\mathbf{h}_{12}[n] - \Delta\mathbf{h}_{34}[n] + \Delta\mathbf{h}_{43}[n] \\ \Delta\mathbf{w}_c^Q[n] &= \Delta\mathbf{h}_{31}[n] - \Delta\mathbf{h}_{13}[n] + \Delta\mathbf{h}_{24}[n] - \Delta\mathbf{h}_{42}[n] \\ \Delta\mathbf{w}_d^Q[n] &= \Delta\mathbf{h}_{32}[n] - \Delta\mathbf{h}_{23}[n] - \Delta\mathbf{h}_{14}[n] + \Delta\mathbf{h}_{41}[n] \end{aligned}. \tag{8.12}$$

From (8.9), we finally find

$$\begin{aligned} \Delta\mathbf{w}_a^Q[n] &= 4\Delta\mathbf{w}_a[n] \\ \Delta\mathbf{w}_b^Q[n] &= 4\Delta\mathbf{w}_b[n] \\ \Delta\mathbf{w}_c^Q[n] &= 4\Delta\mathbf{w}_c[n] \\ \Delta\mathbf{w}_d^Q[n] &= 4\Delta\mathbf{w}_d[n] \end{aligned}. \tag{8.13}$$

Given the results in (8.13), we see that the innovation term components in QLMS are 4 times greater than each homonymous component in MIMO. A practical consequence of this will be evident in the simulations proposed in Par. 8.1.2.

8.1.2 Simulations

The simulation scenario proposed here is a little bit more complicated than the scheme of FE-QLMS of Fig. 8.1, since unit-variance white Gaussian noise $\eta[n]$ is added to the desired path (Fig. 8.3).

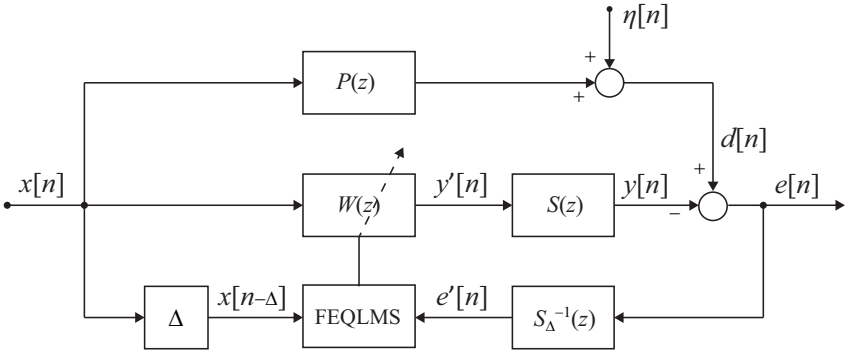


Figure 8.3. Adaptive 3D equalization with FE-QLMS

Both the unknown path $S(z)$ and the desired response $P(z)$ have an Ambisonic format and their impulse responses $s[n]$ and $p[n]$ are quaternion-valued. The input signal is unit-variance white Gaussian noise. The delayed inverse of $S(z)$ was estimated as in Fig. 8.2. The delay Δ in the estimation process is the same delay applied to $x[n]$ before entering the adaptive algorithm. In this case, the delay has been chosen as $\Delta = \frac{M}{2}$, where M is the adaptive filter length. The primary and secondary path impulse responses exhibit different reverberation tails and they are shown in Fig. 8.4.

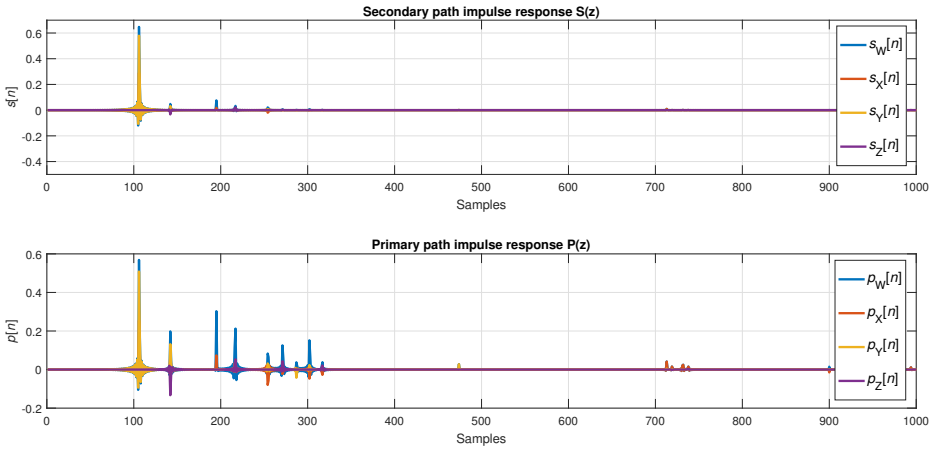


Figure 8.4. Primary and secondary impulse responses ($p[n], s[n]$).

After the adaptation, the controller weights result at steady state as shown in Fig. 8.5. The filter performance can be analysed by inspection of the algorithm learning curve. Given the same step size for FE-QLMS and MIMO FE-LMS ($\mu_Q = \mu_{MIMO} = 10^{-4}$), the Mean Square Error (MSE) curve confirms the theoretic

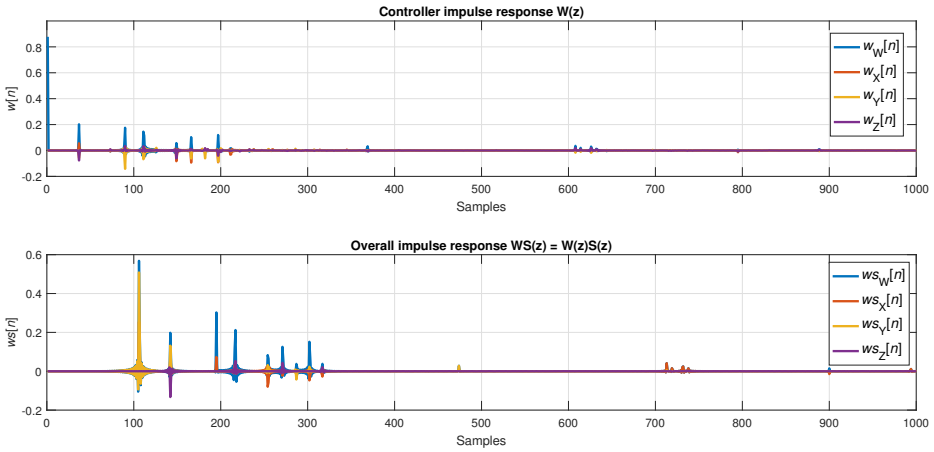


Figure 8.5. Controller weights $w[n]$ and overall controller/secondary path response.

calculations in Par. 8.1.1. In other words, the MIMO algorithm requires a larger number of iterations to get to optimum (see Fig.8.6).

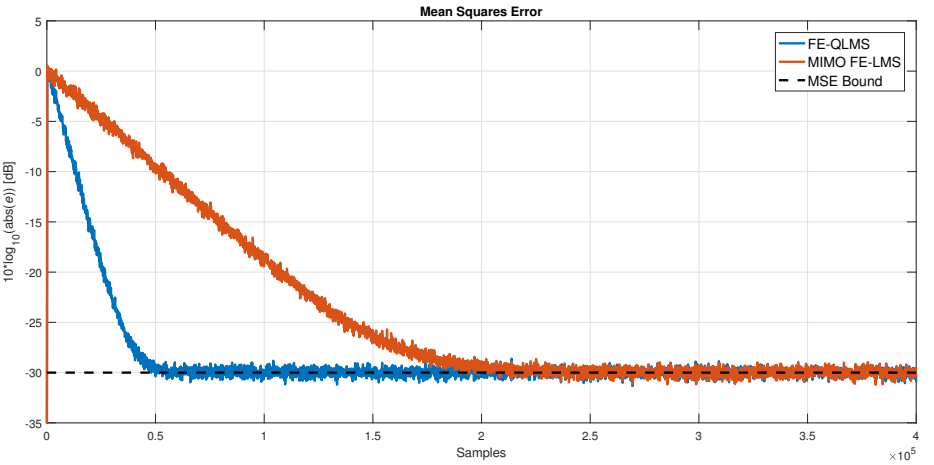


Figure 8.6. FE-QLMS vs MIMO FE-LMS ($\mu_Q = \mu_{MIMO} = 10^{-4}$).

Tests revealed that the FE-QLMS and MIMO FE-LMS converge at the same rate if $\mu_{MIMO} = 4\mu_Q$. However, if we try to speed up the execution, the step size may result excessively large thus causing an excess MSE at steady state or filter instability (e.g. $\mu_{MIMO} = 4\mu_Q = 4 \times 10^{-4}$, Fig. 8.7).

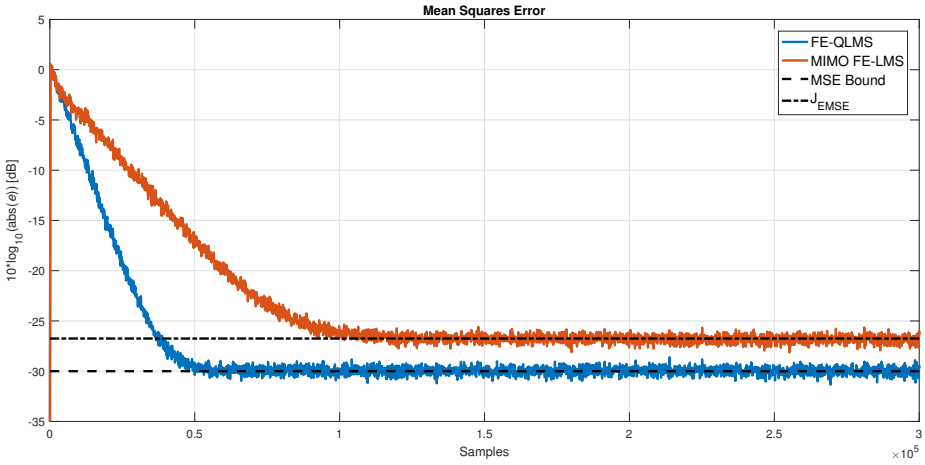


Figure 8.7. FE-QLMS vs MIMO FE-LMS ($\mu_{MIMO} = 4\mu_Q = 4 \times 10^{-4}$).

8.2 Quaternion-valued inverse system modeling

In the 3D audio equalizer presented in Par. 8.1, we made use of inverse modeling to estimate the secondary path impulse response 8.2. In the experiment proposed, inverse modeling was accomplished by choosing a long enough test signal $x[n]$ and a loose step size in order to guarantee convergence to optimum for both QLMS and MIMO LMS. However, in Par. 8.1.2, we only highlighted the slow convergence rate in MIMO FE-LMS in comparison with FE-QLMS. Which are the consequences of slow rate in inverse system modeling? If perfect inversion is reached, the convolution of the *unknown plant* $s[n]$ with the estimated response $w[n] = \hat{s}[n]$ should return a real-valued unit response at the instant of the chosen delay Δ and zero response at all other times. When errors in the estimation occur, the plot is not clean, but it exhibits small side lobes about the main spike. Let us consider the unknown plant in Fig. 8.8. We are not interested here about the detailed characterists of the unknown plant. It was generated by an acoustic simulator and it represents a generic room impulse response recorded by an Ambisonic B-Format array. The length of the impulse response is $M = 1280$ samples. We estimate its inverse model by means of QLMS and MIMO LMS applying the delay $\Delta = \frac{M}{2}$. MIMO is defined as in (8.6). The test signal $x[n]$ is white Gaussian noise and its length is 100000 samples (it was chosen in a way that it is not long enough for MIMO to reach the optimum solution). The step size is chosen as $\mu = 7 \times 10^{-4}$. The results of convolution of the unknown plant with the estimated inverse in the two cases is reported in Fig. 8.9 and Fig. 8.10.

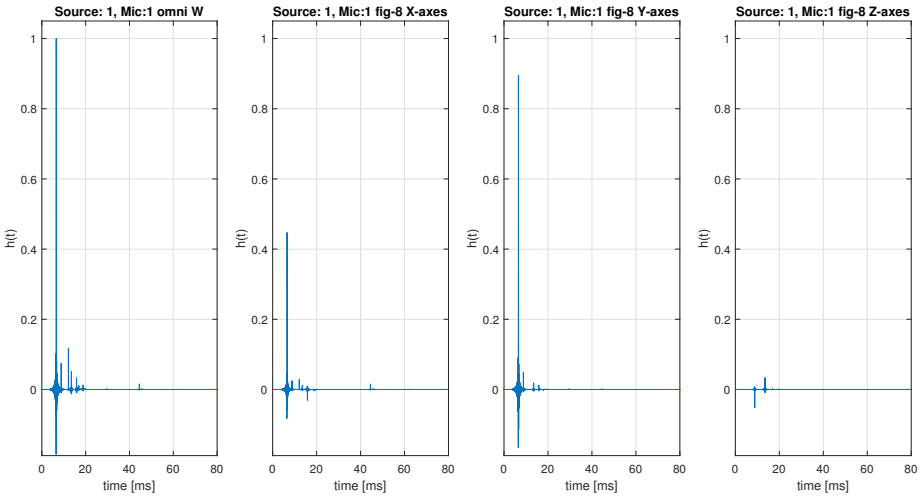


Figure 8.8. Generic Ambisonic unknown impulse response $s[n]$.

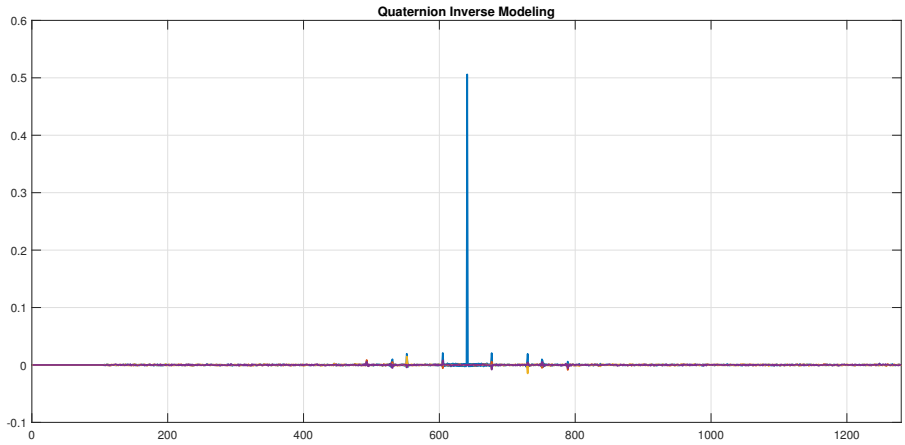


Figure 8.9. Quality check of Quaternion inverse modeling ($\mathbf{s}^T\hat{\mathbf{s}}$).

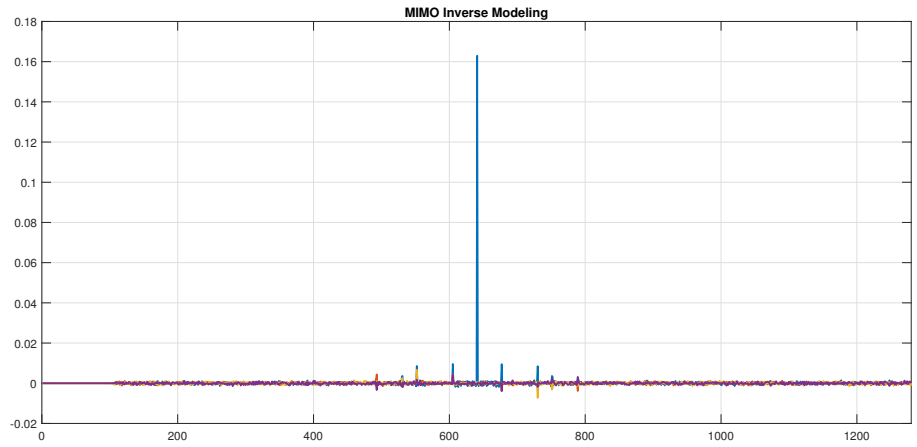


Figure 8.10. Quality check of MIMO inverse modeling ($\mathbf{s}^T\hat{\mathbf{s}}$).

In both cases, the main spike is exactly in $\Delta = \frac{M}{2}$ and its amplitude is greater in the quaternion inverse modeling. However, in order to give a correct interpretation to the graphics, it is better to check for the total power of the imaginary quaternion components (second, third and fourth components in MIMO) related to the real component (first component in MIMO). Given the quaternion convolution $c[n] = \mathbf{s}^T\hat{\mathbf{s}} = c_a[n] + c_b[n]\mathbf{i} + c_c[n]\mathbf{j} + c_d[n]\mathbf{k} \in \mathbb{H}$, we compute

$$p_b = \frac{\sum_{n=1}^{N} c_b^2[n]}{\sum_{n=1}^{N} c_a^2[n]}, \quad p_c = \frac{\sum_{n=1}^{N} c_c^2[n]}{\sum_{n=1}^{N} c_a^2[n]}, \quad p_d = \frac{\sum_{n=1}^{N} c_d^2[n]}{\sum_{m=1}^{N} c_a^2[n]} \tag{8.14}$$

where N is the length of $c[n]$. The calculation for MIMO is similar:

$$p_2 = \frac{\sum_{n=1}^{N} c_2^2[n]}{\sum_{n=1}^{N} c_1^2[n]}, \quad p_3 = \frac{\sum_{n=1}^{N} c_3^2[n]}{\sum_{n=1}^{N} c_1^2[n]}, \quad p_4 = \frac{\sum_{n=1}^{N} c_4^2[n]}{\sum_{m=1}^{N} c_1^2[n]} \tag{8.15}$$

The power components are shown in Fig. 8.11.

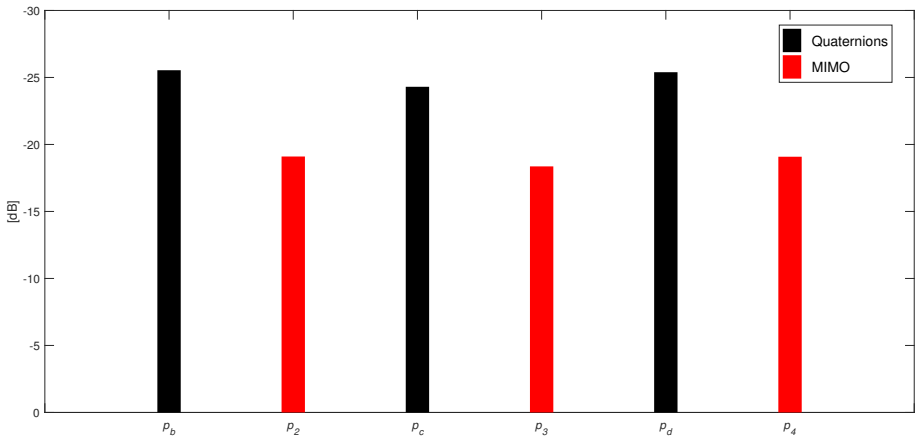


Figure 8.11. $\mathbf{s}^T\hat{\mathbf{s}}$: "imaginary" components total power related to real component in quaternion and equivalent MIMO inverse modeling.

In MIMO, components $c_2[n], c_3[n], c_4[n]$ have a greater power related to $c_1[n]$ in comparison with $c_b[n], c_c[n], c_d[n]$ related to $c_a[n]$. In other words, the side lobes produced by the "imaginary" components are more *disturbing* in MIMO than in quaternion processing. Further investigation and study are required in order to provide a robust interpretation of the results.

Chapter 9

Conclusion

The reasons why hypercomplex numbers attract the attention of the scientific community reside in the mathematical properties of these algebras, capable of bringing to light aspects of physics and engineering which could not be highlighted by real and complex numbers. In fact, within the hypercomplex number subfields, it is possible to consider different entities, possibly related to one another, as a whole and take advantage of a single filter to process multidimensional data, combining all hypercomplex dimensions and exploiting the mutual information between all components. The goals in this research project have been the investigation of hypercomplex (quaternion) adaptive filters and their integration into 3-dimensional context, e.g. acoustics and 3D audio. Moreover, one of our main targets has been to raise the awareness for the foundation of the concepts of discrete-time *Hypercircuit* and discrete-time hypercircuit theory.

9.1 3D Audio and quaternion signal processing

Hypercomplex filtering offers advanced geometric capabilities for multidimensional data processing. That said, 3D audio is not merely multisensor data, but it is possible to consider a 3D sound space with a mathematical structure over the field of quaternions. We have explored some recent applications of quaternion algebra to 3D audio problems. In particular, we have seen how the Ambisonic sound field can be expressed in a quaternionic formalism, thus obtaining a compact form, without the loss of information. We have pinpointed the fundamental problems of 3D audio adaptive signal processing: the choice of a proper algebraic representation, the computational effort, the possibility of exploiting the statistical properties of the audio signals. We have presented a new class of algorithms working in the frequency domain with the aim of reducing the computational cost. In addition, thanks to the definition of widely linear systems, it is possible to fully exploit the second order stastistics of the quaternion signal and obtain improved performance in terms of filter convergence with improper input signals. In the context of 3D audio, we have seen that the Ambisonic B-Format signals are in fact improper and we helped validate the theory of widely linear filters with simulations. In order to test whether quaternions are the right algebraic choice with Ambisonics, we compared the performance of quaternion adaptive filters with tessarine adaptive filters of the same kind.

9.2 Quaternion Adaptive Filters in the Frequency Domain

In this work, we have introduced the OS-QFDAF algorithm, along with the Unconstrained QFDAF and Sliding Window QFFT-QLMS versions, to process quaternion-valued signals in the frequency domain. After deriving the OS-QFDAF algorithm, we analyzed its convergence properties and computational cost. We obtained the stability range of the step size and found a mathematical relation between the Excess Mean Square Error (EMSE) and the algorithm parameters. The OS-QFDAF was achieved by transforming the Block QLMS algorithm into the frequency domain and following the rules of the overlap-save method to obtain a fast convolution. One of the difficulties in working with quaternion signals in the frequency domain is that the quaternion convolution theorem is not straightforward as the conventional theorem in real and complex algebras. In fact, spectrum product does not correspond to time-domain convolution, generally. In Chapter 2, we have reported the quaternion convolution theorem in all its forms. Different formulations of the theorem depend on the adoption of the left or the right Fourier transform. We have seen that, because of the non-commutative quaternion product, there exist left, right and sandwich transforms in the quaternion domain. However, we found by tests that the rule of thumb is to keep the transform rotation direction unchanged during the execution of an algorithm for all the signals to be transformed.

9.3 Widely Linear algorithms

Quaternion-valued adaptive filters are known for their property of exploiting the cross-correlation among all components of a multidimensional (quaternion-valued) signal. However, in order to improve the filter performance, it is necessary to design filters capable of handling signals of any nature. We saw that a complete insight of quaternion second order statistics is needed in order to fully exploit the information coupling within all quaternion channels. Widely linear quaternion algorithms were introduced and showed improved performance with the processing of improper signals. Unfortunately, the computational cost is an issue due to the adaptation of four types of filter weights. We proposed and tested a version of the WL-QLMS algorithm operating weight adaptation periodically (WL-BQLMS). From WL-BQLMS we derived the Widely Linear version of the OS-QFDAF. Periodic adaptation allows the computational cost to be reduced dramatically. Moreover, periodic adaptation has an intrisic smoothing capability which contributes to the improvement of a filter performance.

9.4 Other algebras

The question today about hypercomplex filtering is whether we really need such hypercomplex models to represent our systems. Besides that, what determines the rejection of one algebra in favor of another? In this work, we proposed a simple comparison between two 4-dimensional hypercomplex algebras: quaternions and tessarines. We learned from simulations that some systems can be considered either

quaternion-valued or tessarine-valued. In other cases, the choice of the algebraic representation determines the performance of the whole system. For instance, we introduced the Ambisonic B-Format signals into a 4-dimensional system and we saw that a quaternion adaptive algorithm converges much faster than its tessarine counterpart. Moreover, we were able to exploit widely linear modifications of QLMS and TLMS (tessarine LMS) and take the most out of them with improper signals like Ambisonic 3D audio.

We have investigated the choice of a microphone array layout in combination with the mathematical description of the surrounding environment on the performance of adaptive filters. We made use of a coincident B-Format array and a uniform linear array of microphones, both of them made up of four elements. We have transformed the sound space into a 4-dimensional format and we have applied the resulting multidimensional signal to 4-dimensional adaptive filters (QLMS and TLMS). We have seen that, in the special case the space is decomposed into spherical harmonics and the Ambisonic B-Format technique is used to record the sound field, the choice of quaternion algebra in signal processing works like the right key in the lock. In this instance, the QLMS algorithm converges faster than the TLMS algorithm. The outcome of this test suggests that, in this special context, quaternion algebra is a better choice. In the case signals are simply correlated, but the sound space does not undergo any special transformation, there may be still an advantage of quaternion signal processing over tessarine algebra, although not so evident. In fact, we have seen how changing the distance between the microphones overturned the result. Further investigation is absolutely required and the results need to be formalized.

9.4.1 MIMO systems

Typically, multichannel data can be processed by means of MIMO (Multiple-Input, Multiple-Output) systems. It is possible to define a MIMO$(4, 4)$ system wich is isomorphic to quaternions and keep on operating in real-valued algebra. We proposed the implementation of a 3D audio equalizer. In contrast to conventional multichannel/multiple-point signal processing, we proposed a multichannel/single-point audio layout, exploiting the 3D recording technique called Ambisonics. We encapsulated Ambisonic signals into the quaternion format as the input of a traditional algorithm for adaptive equalization (FE-QLMS). Because of the differences in the filter dynamics in weight adaptation, the quaternion filter converges faster than the MIMO option, even though the two systems are equivalent to each other.

9.5 3D rotations: Quaternion algebra vs 3D vector algebra

In Chapter 4 we compared the representation of 3D rotations with matrices and quaternion operators. We saw that a matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$ maps a point $x \in \mathbb{R}^3$ onto a point $y \in \mathbb{R}^3$. On the other hand, a quaternion $q_A \in \mathbb{H}$ transforms a pure quaternion $q_x \in \mathbb{H}$ into another pure quaternion $q_y \in \mathbb{H}$. The two transformations

are summarized below:

$$\begin{aligned} y &= \mathbf{A}x \in \mathbb{R}^3 \qquad \text{MATRICES} \\ q_y &= q_A q_x q_A^* \in \mathbb{H} \quad \text{QUATERNIONS} \end{aligned} \tag{9.1}$$

We recall some remarks from [47]:

1. Matrix mapping requires 9 coefficients to relate two vectors in $\mathbb{R}^3$ (3×3 matrix). However, only 4 parameters are necessary to define a rotation (2 for the axis of rotation, 1 for the angle of rotation, 1 for the scaling factor). The 4 elements of a quaternion can express these parameters without the addition of any redundant information.

2. In matrix rotations, if any two axes are aligned, a degree of freedom is lost. We called this phenomenon *Gimbal Lock*. Nothing like this happens with quaternions.

3. Numerical conditioning is better in quaternion algebra with respect to matrix rotations. The only requirement for q_A is to be a pure quaternion, whereas $\mathbf{A}$ must comply with $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and $\det(\mathbf{A}) = 1$.

9.6 What's next?

9.6.1 Energetic issues

During the development of our algorithms presented in [67], we focused on the study of the convengence properties for each algorithm. Since convergence is related to energy, we soon encountered a problem. Quaternion algebra generally produces left and right eigenvalues because of the non-commutativity of the quaternion product. The eigenvalue disparity in a filter, also known as *eigenspread*, determines the convergence rate of the algorithm, so the physical meaning of the algorithm eigenvalues, responsible for the algorithm natural modes, should not be ignored since it is related to energy. This fact of quaternion algebra was not completely explored and the literature does not provide full information about the quaternion eigenvalue definition. Further investigation would supply the missing information and possibly reveal aspects of energetic problems which have been ingnored up to now.

9.6.2 Hypercircuit theory

One of the design and analysis approaches in digital signal processing considers filters as circuits. It is possible to define constitutive relations for component elements and operate computations within the circuit topology in the digital domain [64]. Even though discrete-time hypercircuits have been widely studied in signal processing (image processing, adaptive filters, neural networks, etc.), a definition of discrete-time hypercircuit was never given explicitly. It is important to confirm the classic theorems of the circuit theory or find an extended definition of them. The aim for the future is to provide a reliable definition of circuit, since circuit theory is a powerful design tool in digital signal processing. Our research group started this *mission* in 2016 [72], proposing a version of quaternion *Z-Transform* (to be established,

yet) and recalling a resolute definition of quaternion convolution and correlation in both time and frequency domain. We started testing a filter stability examining the position of quaternion-valued poles. It is worth continuing this study, since it would provide an aid for digital filter design. Unlike adaptive systems, fixed filters require higher mastery over transfer function implementation. It is necessary to provide the designer with a stable circuit theory, including mathematical tools such as transforms, convolutions and correlations, system stability analysis and so on.

9.6.3 Neural networks and nonlinear processing

At present time, the proponent has very little experience with neural networks and nonlinear processing. The idea for this project is to start exploring the branch of intelligent signal processing and embedding our hypercomplex algorithms in the conventional learning schemes for neural networks. Some studies concerning the use of quaternion neural networks can be found in literature. Amongst many examples, they include the definition and application to engineering problems of a quaternionic *Multilayer Perceptron* (QMLP) model [5, 52, 63] and the development of quaternionic *Hopfield-type networks* [45, 113]. Since we need to build the basic instruments to start working, the idea for this project is to implement some elementary scheme in a hypercomplex format and continue with more advanced architectures. The main reason for undertaking this experimentation is that the widespread availability of intelligent signal processing is growing in size very quickly. It would be a pity not to be up-to-date.

Bibliography

[1] Number, geometry and nature. *Hypercomplex Numbers in Geometry and Physics*, **1** (2004), 3 .

[2] Alfsmann, D., Göckler, H., Sangwine, S. J., and Ell, T. A. Hypercomplex algebras in digital signal processing: benefits and drawbacks. In *Proc. of the 15th European Signal Processing Conference (EUSIPCO)*, pp. 1322 – 1326 (2007).

[3] Amblard, P. O. and Bihan, N. L. On properness of quaternion valued random variables. In *Proc. of Int. Conf. on Mathematics (IMA) in Signal Process.* (2004).

[4] Anttila, L., Valkama, M., and Renfors, M. Frequency-selective i/q mismatch calibration of wideband direct-conversion transmitters. *IEEE Trans. Circuits Syst. II, Exp. Briefs*, **55** (2008), 359 .

[5] Arena, P., Fortuna, L., Muscato, G., and Xibilia, M. Multilayer perceptrons to approximate quaternion valued functions. *Neural Netw.*, **10** (1997), 335.

[6] Baez, J. C. The octonions. *Bull. Amer. Math. Soc.*, **39** (2001), 145.

[7] Baker, A. Right eigenvalues for quaternionic matrices: a topological approach. *Linear Algebra and its Applications*, (1999), 303.

[8] Barthélemy, Q., Larue, A., and Mars, J. I. About qlms derivations. *IEEE Trans. Signal Process. Letters*, **21** (2014), 240.

[9] Berner, P. Technical Concepts - Orientation, Rotation, Velocity and Acceleration, and the SRM (2008). Available from: `http://www.sedris.org/wg8home/Documents/WG80485.pdf`.

[10] Boré, G. and Peus, S. *Microphones for Studio and Home-Recording Applications - Operation Principles and Type Examples.* Georg Neumann GmbH (1999).

[11] Brenner, J. L. Matrices of Quaternions. *Pacific J. Math.*, **1** (1951), 329.

[12] Cao, J., Khong, A. W. H., and Gannot, S. On the performance of widely linear quaternion based MVDR beamformer for an acoustic vector sensor. In *Proc. 14th Int. Wksp on Acoustic Signal Enhancement (IWAENC)*, p. 303 – 307 (2014).

[13] Cayley, A. and Forsyth, A. R. *The collected mathematical papers of Arthur Cayley.* Cambridge University Press (1889).

[14] Chen, J. and Benesty, J. Binaural noise reduction in the time domain with a stereo setup. *IEEE Trans. Signal Process.*, **19** (2011), 2260 .

[15] Chrisitianto, V. and Smarandachey, F. A derivation of Maxwell equations in quaternion space. *Progress in Physics*, **2** (2010), 23 .

[16] Chua, L., C.A.Desoer, and E.S.Kuh. *Linear and Nonlinear Circuits.* Mcgraw-Hill College (1987).

[17] Dam, E. B. and Koch, M. M. Quaternions, interpolation and animation. `"http://web.mit.edu/2.998/www/QuaternionReport1.pdf"` (1998). [Online].

[18] Daniel, J. *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia.* Ph.D. thesis, Université de Paris 6 (2001).

[19] Daniel, J., Nicol, R., and Moreau, S. Further investigations of high order ambisonics and wavefield synthesis for holophonic sound imaging. In *AES 114th Conv.* (2003).

[20] D.Eberly. Quaternion algebra and calculus. `http://www.geometrictools.com/Documentation/Quaternions.pdf` (2010). [Online].

[21] Ell, T. A. *Hypercomplex Spectral Transformations.* Ph.D. thesis, University of Minnesota (1992).

[22] Ell, T. A., Bihan, N. L., and Sangwine, S. J. *Quaternion Fourier Transforms for Signal and Image Processing.* Wiley (2014).

[23] Ell, T. A. and Sangwine, S. J. Decomposition of 2D Hypercomplex Fourier Transforms into Pairs of Complex Fourier Transforms. *Proc. 10th Eur. Signal Process. Conf. (EUSIPCO)*, (2000).

[24] Elliott, S. and Nelson, P. Active noise control. *IEEE Signal Processing Magazine*, **10** (1993), 12 .

[25] Elliott, S. J. and Nelson, P. A. Multiple-point equalization in a room using adaptive digital filters. *J. Audio Eng. Soc.*, **37** (1989), 899 .

[26] Eneroth, P., Benesty, J., and Gay, S. State of the art of stereophonic acoustic echo cancellation. In *in Proc. RVK99* (1999).

[27] Farhang-Boroujeny, B. and Chan, K. S. Analysis of the Frequency-Domain Block LMS Algorithm. *IEEE Trans. Signal Proc.*, **48** (2000), 2332 .

[28] Fellgett, P. Ambisonics. part one: General system description. *Studio Sound*, **17** (1975), 20.

[29] Gerzon, M. A. Ambisonics. Part two: Studio techniques. *Studio Sound*, **17** (1975), 24.

[30] Gerzon, M. A. Ambisonics in multichannel broadcasting and video. *J. Audio Eng. Soc.*, **33** (1985), 859–871.

[31] Gou, X., Liu, Z., and Xu, Y. Biquaternion cumulant-music for doa estimation of noncircular signals. *Signal Process.*, **93** (2013), 874 .

[32] Gray, B. *Homotopy Theory – An introduction to algebraic topology*. Academic Press (1975).

[33] Greiter, M. and Schuricht, D. Imaginary in all directions: an elegant formulation of special relativity and classical electrodynamics. *European Journal of Physics*, **24** (2003), 397 .

[34] Hamilton, W. R. *Elements of Quaternions*. Longmans, Green & Co. (1866).

[35] Hanson, A. *Quaternions applied to physics in non-euclidean space*. Elsevier (2006).

[36] Hanson, A. J. *Visualizing Quaternions*. Elsevier (2006).

[37] Hitzer, E. Introduction to Clifford's geometric algebra. *SICE J. Control, Measurement, and System Integration*, **4** (2011), 1 .

[38] Hitzer, E. and Sangwine, S. J. *Quaternion and Clifford Fourier Transforms and Wavelets*. Birkenhäuser, Springer (2010).

[39] Holm, D. Geometric mechanics, part ii: Rotating, translating and rolling (2011). Available from: `http://wwwf.imperial.ac.uk/~dholm/classnotes/GeomMech2-2nd.pdf`.

[40] Hu, J., Zhang, H., Feng, J., Huang, H., Ma, H., and G.Wang. A scale adaptive kalman filter method based on quaternion correlation in object tracking. In *3rd Int. Conf. on Networking and Distributed Computing*, pp. 170 – 174 (ICNDC 2012).

[41] Hu, Q. and Zou, L. Several theorems for the trace of self-conjugate quaternion matrix. *Modern Applied Science*, **2** (2008), 21.

[42] Huang, L. and So, W. On left eigenvalues of a quaternionic matrix. *Linear Algebra Appl.*, **323** (2001), 105.

[43] Huang, Y. and Benesty, J. *Audio Signal Processing for Next-Generation Multimedia Communication Systems*. Kluwer Academic Publishers (2004).

[44] Huang, Y., Benesty, J., and Jingdong, C. *Acoustic MIMO Signal Processing*. Springer (2006).

[45] Isokawa, T., Nishimura, H., and Matsui, N. Quaternionic multilayer perceptron with local analyticity. *Information*, **3** (2012), 756.

[46] Jahanchahi, C., Took, C. C., and Mandic, D. P. A class of quaternion-valued affine projection algorithms. *Signal Process.*, (2013), 1712 .

[47] Jahanehahi, C. and Mandic, D. P. A class of quaternion kalman filters. *IEEE Trans. Neural Netw. Learn. Syst.*, **25** (2014), 533.

[48] Katunin, A. Three-dimensional octonion wavelet transform. *J. Applied Mathematics and Computational Mechanics*, **13** (2014), 33 .

[49] Khalil, M. I. Applying Quaternion Fourier Transforms for Enhancing Color Images. *Int. J. Image, Graphics and Signal Process.*, **4** (2012), 9 .

[50] Kraft, E. A quaternion-based unscented Kalman filter for orientation tracking. vol. 1, pp. 47 – 54 (2003).

[51] Kuo, S. M. and Morgan, D. R. Active noise control: A tutorial review. *Proc. of the IEEE*, **87** (1999), 943 .

[52] Kusamichi, H., Isokawa, T., Matsui, N., Y. Ogawa, Y., and Maeda, K. A new scheme for color night vision by quaternion neural network. In *Proc. 2nd Int. Conf. Autonomous Robots and Agents*, pp. 101–106 (2004).

[53] Lee, H. Eigenvalues and canonical forms of matrices with quatemion coefficients. *Proc. of the Royal Irish Academy, Section A*, **52** (1949), 137.

[54] Li, X. and Adali, T. Complex-valued linear and widely linear filtering using mse and gaussian entropy. *IEEE Trans. on Sig. Proc.*, **60** (2012), 5672 .

[55] Lie, S. Über complexe, insbesondere linien- und kugel-complexe, mit anwendung auf die theorie partieller differentialgleichungen. *Math. Ann.*, **5** (1872), 145.

[56] Mandic, D. P. and Goh, V. S. L. *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models.* Wiley (2009).

[57] McWhorter, T. and Schreier, P. Widely-linear beamforming. In *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1 (2004).

[58] Molineros, J., Behringer, R., and Tam, C. Vision-based augmented reality for pilot guidance in airport runways and taxiways. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004)* (2004).

[59] Navarro-Moreno, J., Fernández-Alcalá, R. M., Took, C., and Mandic, D. P. Prediction of wide-sense stationary quaternion random signals. *Signal Process.*, **93** (2013), 2573 .

[60] Neeser, F. D. and Massey, J. L. Proper complex random processes with applications to information theory. *IEEE Trans. Inf. Theory*, **39** (1993), 1293 .

[61] Nicol, R. and Emerit, M. 3D sound reproduction over an extensive listening area: a hybrid method derived from holophony and ambisonics. In *Proc. AES 16th Int. Conf.*, pp. 436 – 453 (1999).

[62] Nitta, T. A theoretical foundation for the widely linear processing of quaternion-valued data. *Applied Mathematics*, (2013), 1616 .

[63] Nitta, T. An extension of the back-propagation algorithm to quaternion. In *3rd Int. Conf. Neural Information Process.*, p. 247 – 250 (ICONIP 1996).

[64] Oppenheim, A. V. and Schafer, R. W. *Discrete-Time Signal Processing*. Prentice Hall (1989).

[65] Ortolani, F. Introduction to Ambisonics - A tutorial for beginners in 3D audio. `"http://www.ironbridge-elt.com/downloads/FrancescaOrtolani-IntroductionToAmbisonics.pdf"` (2014). [Online].

[66] Ortolani, F., Comminiello, D., Scarpiniti, M., and Uncini, A. Advances in hypercomplex adaptive filtering for 3D audio processing. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (2017).

[67] Ortolani, F., Comminiello, D., Scarpiniti, M., and Uncini, A. Frequency domain quaternion adaptive filters: Algorithms and convergence performance. *Signal Processing*, **136** (2017), 69 .

[68] Ortolani, F., Comminiello, D., and Uncini, A. The widely linear block quaternion least mean square algorithm for fast computation in 3d audio systems. In *Proc. 26th International Workshop on Machine Learning for Signal Processing* (MLSP, 2016).

[69] Ortolani, F., Scarpiniti, M., Comminiello, D., and Uncini, A. On 4-dimensional hypercomplex algebras in adaptive signal processing. In *27th Italian Workshop on Neural Networks (WIRN)* (2017).

[70] Ortolani, F., Scarpiniti, M., Comminiello, D., and Uncini, A. On the influence of microphone array geometry on the behavior of hypercomplex adaptive filters. In *IEEE Microwaves, Radar and Remote Sensing Symposium MRRS-2017* (2017).

[71] Ortolani, F. and Uncini, A. A new approach to acoustic beamforming from virtual microphones based on Ambisonics for adaptive noise cancelling. In *IEEE 36th Int. Conf. on Electronics and Nanotechnology (ELNANO)* (2016).

[72] Ortolani, F. and Uncini, A. Quaternion digital signal processing: A hypercomplex approach to information processing. In *2016 Int. Siberian Conf. on Control and Communications (SIBCON)* (2016).

[73] Ortolani, F. and Uncini, A. Widely linear quaternion adaptive filtering in the frequency domain. In *2016 IEEE International Conference on Mathematical Methods in Electromagnetic Theory (MMET)* (2016).

[74] Pei, S. C., Ding, J. J., and Chang, J. H. Efficient Implementation of Quaternion Fourier Transform, Convolution, and Correlation by 2-D Complex FFT. *IEEE Trans. Signal Process.*, **11** (2001), 2783 .

[75] Peirce, B. Linear associative algebra. *American J. of Mathematics*, **1** (1881), 221.

[76] Picinbono, B. On circularity. *IEEE Trans. on Sig. Proc.*, **42** (1994), 3473 .

[77] Picinbono, B. and Chevalier, P. Widely linear estimation with complex data. *IEEE Trans. on Signal Processing*, **43** (1995), 2030 .

[78] Picinbono, B. and Chevalier, P. Widely linear estimation with complex data. *IEEE Trans. on Sig. Proc.*, **43** (1995), 2030 .

[79] Rediess, H. A. An augmented reality pilot display for airport operations under low and zero visibility conditions. (1997).

[80] Roth, B. *Advances in Robot Kinematics and Computational Geometry.* Springer (1994).

[81] Rumsey, F. *Spatial Audio.* Focal Press (2001).

[82] Said, S., Bihan, N. L., and Sangwine, S. J. Fast complexified quaternion Fourier transform. *IEEE Trans. Signal Process.*, **56** (2008), 1522 .

[83] Sangwine, S. and T-Ell. Colour image filters based on hypercomplex convolution. *IEEE Proc. Vision, Image and Signal Process.*, **147** (2000), 89 .

[84] Schreier, P. J. and Scharf, L. L. *Statistical Signal Processing of Complex-Valued Data: The Theory of Improper and Noncircular Signals.* Cambridge University Press (2010).

[85] Shaffer, S. and Williams, C. S. The filtered error lms algorithm. In *Proc. of the 8th IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 41 – 44 (1983).

[86] Shoemake, K. *Animating rotation with quaternion calculus.* ACM SIGGRAPH Course Notes.

[87] Siu, L. S. *A Study of Polynomials, Determinants, Eigenvalues and Numerical Ranges over Real Quaternions.* Ph.D. thesis, University of Hong Kong (1997).

[88] Smith, S. The four-dimensional Sklyanin algebras. *K-Theory*, **8** (1994), 65.

[89] Sondhi, M. M. An adaptive echo canceller. *The Bell System Technical Journal*, **66** (1967), 497 .

[90] Starner, T., Mann, S., Rhodes, B., Levine, J., Healey, J., Kirsch, D., Picard, R., and Pentland, A. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments archive*, **6** (1997), 386 .

[91] Sujbert, L. A filtered reference – filtered error lms algorithm. In *Proc. of the IEEE Int. Workshop on Intelligent Signal Processing*, pp. 204 – 209 (1999).

[92] Sundareswaran, S., Vassiliou, M., McGee, J., Behringer, R., and Tam, C. Two wearable testbeds for augmented reality: itwarns and wimmis. In *2012 16th International Symposium on Wearable Computers* (2000).

[93] Sweetser, D. B. Doing physics with quaternions (2005). Available from: `http://www.theworld.com/~sweetser/quaternions/ps/book.pdf`.

[94] Taber, H. On hypercomplex number systems. *Trans. of the American Mathematical Soc.*, **5** (1904), 509.

[95] Took, C., Mandic, D., and Benesty, J. Study of the quaternion LMS and four-channel LMS algorithms. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (2009).

[96] Took, C. C., Aihara, K., and Mandic, D. P. Quaternion-valued short term forecasting of wind profile. In *Int. Joint Conf. on Neural Networks*, pp. 1–6 (IJCNN 2010).

[97] Took, C. C. and Mandic, D. P. Fusion of heterogeneous data sources: A quaternionic approach. In *IEEE Workshop on Machine Learning for Signal Process. (MLSP)*, pp. 456–461 (2008).

[98] Took, C. C. and Mandic, D. P. The quaternion LMS algorithm for adaptive filtering of hypercomplex processes. *IEEE Trans. Signal Process.*, **57** (2009), 1316.

[99] Took, C. C. and Mandic, D. P. A quaternion widely linear adaptive filter. *IEEE Trans. Signal Process.*, **58** (2010), 4427 .

[100] Took, C. C. and Mandic, D. P. Augmented second-order statistics of quaternion random signals. *Signal Process.*, (2011), 214 .

[101] Took, C. C., Strbac, G., Aihara, K., and Mandic, D. P. Quaternion-valued short-term joint forecasting of three-dimensional wind and atmospheric parameters. *Renewable Energy*, (2011), 1754.

[102] Ujang, B. C., Jahanchahi, C., Took, C. C., and Mandic, D. P. Adaptive convex combination approach for the identification of improper quaternion processes. *IEEE Trans. Neural Netw.*, **25** (2014), 172 .

[103] Ujang, B. C., Took, C. C., and Mandic, D. Quaternion-valued nonlinear adaptive filtering. *IEEE Trans. Neural Netw.*, **22** (2011), 1193 .

[104] Uncini, A. *Fundamentals of Adaptive Signal Processing*. Springer (2015).

[105] Vakhania, N. N. Random vectors with values in quaternion Hilbert spaces. *Theory of Probability and its Applications*, **43** (1998), 18 .

[106] Wang, G., Liua, Y., and Zhao, T. A quaternion-based switching filter for colour image denoising. *Signal Process.*, **102** (2014), 216.

[107] Waser, A. Quaternions in electrodynamics (2001). Available from: http://hotstreamer.deanostoybox.com/temp/QuaternionsInElectrodynamicsEN02.pdf.

[108] Widrow, B., Glover, J., McCool, J., Kaunitz, J., Williams, C., Hearn, R., Zeidler, J., Dong, J. E., and Goodlin, R. Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, **63** (1975), 1692 .

[109] Wiegmann, N. A. The Structure of Unitary and Orthogonal Quaternion Matrices. *Illinois J. Math.*, **2** (1958), 402.

[110] Witten, B. and Shragge, J. Quaternion-based signal processing. In *SEG/Annual Meeting* (2006).

[111] Witten, B. and Shragge, J. Quaternion-based Signal Processing. In *Proc. Soc. of Exploration Geophysicists Annual Meeting (SEG)* (2006).

[112] Wood, R. Quaternionic eigenvalues. *Bulletin of the London Mathematical Society*, **15** (1985), 137.

[113] Yoshida, M., Kuroe, Y., and Mori, T. Models of hopfield-type quaternion neural networks and their energy function. *Int. J. Neural Syst.*, **15** (2005), 129.

[114] Zhang, F. Quaternions and matrices of quaternions. *Linear Algebra Appl.*, **251** (1997), 21.

[115] Zhang, X., Liu, W., Xu, Y., and Liu, Z. Quaternion-valued robust adaptive beamformer for electromagnetic vector-sensor arrays with worst-case constraint. *Signal Process.*, **104** (2014), 274 .