An Alternative Kernel Adaptive Filtering Algorithm for Quaternion-Valued Data

Tokunbo Ogunfunmi* and Thomas Paul†
* Santa Clara University, Santa Clara CA, USA
E-mail: togunfunmi@scu.edu
† Santa Clara University, Santa Clara CA, USA
E-mail: tpaul@scu.edu

***Abstract*— Nonlinear adaptive filters are getting more common and are useful especially where performance of linear adaptive filters may be unacceptable. Such areas include communications, image processing and biological systems. Quaternion valued data has also been drawing recent interest in various areas of statistical signal processing, including adaptive filtering, image pattern recognition, and modeling and tracking of motion. The benefit for quaternion valued processing includes performing data transformations in a 3 or 4-dimensional space in a more convenient fashion than using vector algebra.**

In this paper we describe an alternative kernel adaptive filter for quaternion valued data we refer to as the involution Quaternion Kernel Least Mean Square (iQuat-KLMS) algorithm. The approach is based on the Quaternion KLMS (Quat-KLMS) algorithm obtained previously, as well as the recently-developed involution gradient (i-gradient). A modified HR Calculus for Hilbert spaces is used for finding cost function gradients defined on a quaternion RKHS. Simulation tests with a synthetic quaternion channel are used to verify the benefit of iQuat-KLMS in convergence compared to Quat-KLMS.

I. INTRODUCTION

Interest in nonlinear adaptive filtering has become more prevalent, providing benefit in modeling real-world physical processes which often contain nonlinearities. The references [1-5] describe the design of these filters.

Quaternions for statistical signal processing have also been drawing increasing interest in applications such as image pattern recognition, and modeling/tracking of motion [6-10]. For example, linear and widely linear adaptive filters using quaternions can be found in references [6], [7], and gradient operations for obtaining them, referred to as $\mathbb{HR}$ calculus, are described in [8], [9].

Processing using quaternions allows data transformations in 3 or 4-dimension space to be performed conveniently vs. vector algebra. The multiplication operation for quaternions differs from real/complex multiplication in that quaternion multiplication is non-commutative, making the gradient operation challenging.

Here we describe an alternative kernel adaptive filter for quaternions. The method is based on the Quaternion Kernel Least Mean Square (Quat-KLMS) algorithm derived in [11], which is summarized here. The new approach makes use of the involution gradient (or i-gradient) of [9], and is referred to as the involution Quaternion Kernel LMS (iQuat-KLMS).

The paper is organized as follows. In Section II background on quaternion-valued data is provided. Section III describes the approach for kernel adaptive filtering with quaternions. Section IV describes the derivation of Quat-KLMS and the iQuat-KLMS using the i-gradient. Section V provides simulation results showing the improvement in convergence of iQuat-KLMS for learning quaternion channels with nonlinearities. Tests include synthetic quaternion channels with nonlinearity, as well as practical EEG data. Finally, Section VI concludes the paper.

II. BACKGROUND ON QUATERNION DATA

Here we provide a brief description on quaternion data properties. More information may be found from [6-8].

A quaternion input vector may be expressed

$$\mathbf{x}(n) = \mathbf{x}_a(n) + i\mathbf{x}_b(n) + j\mathbf{x}_c(n) + k\mathbf{x}_d(n) \tag{1}$$

where $\mathbf{x}_a(n)$, $\mathbf{x}_b(n)$, $\mathbf{x}_c(n)$, and $\mathbf{x}_d(n)$ are real-valued vectors, and i, j, and k are orthogonal unit direction vectors comprising the quaternion input (where $i^2=j^2=k^2=-1$). Their multiplication is also non-commutative, i.e. $ij=-ji=k$, $jk=-kj=i$, and $ki=-ik=j$.

It was shown previously that the following three perpendicular quaternion involutions may be formed [2]:

$$\begin{aligned}\mathbf{x}^i(n) &= -i\mathbf{x}(n)i = \mathbf{x}_a(n) + i\mathbf{x}_b(n) - j\mathbf{x}_c(n) - k\mathbf{x}_d(n)\\ \mathbf{x}^j(n) &= -j\mathbf{x}(n)j = \mathbf{x}_a(n) - i\mathbf{x}_b(n) + j\mathbf{x}_c(n) - k\mathbf{x}_d(n)\\ \mathbf{x}^k(n) &= -k\mathbf{x}(n)k = \mathbf{x}_a(n) - i\mathbf{x}_b(n) - j\mathbf{x}_c(n) + k\mathbf{x}_d(n)\end{aligned} \tag{2}$$

which can be used to determine components $\mathbf{x}_a(n)$, $\mathbf{x}_b(n)$, $\mathbf{x}_c(n)$, and $\mathbf{x}_d(n)$, as well as form a basis (when combined with $\mathbf{x}(n)$) for the gradient operations to be described. They can also be used to express the conjugate for quaternion data, which is:

$$\mathbf{x}^*(n) = \frac{1}{2}\left(\mathbf{x}^i(n) + \mathbf{x}^j(n) + \mathbf{x}^k(n) - \mathbf{x}(n)\right) \tag{3}$$

The perpendicular involutions, conjugation operation will be used for the analysis.

III. COMBINING QUATERNIONS WITH KERNEL METHODS

To develop kernel adaptive filtering algorithms for quaternions, we proceed as follows. For the kernel filter, the goal is to estimate the channel output, where the desired response is of form: $d(n) = f(\mathbf{x}(n)) + v(n)$. The channel function $f(.)$ here is some unknown nonlinear quaternion-valued function and $v(n)$ is additive noise. The estimator form is:

$$\hat{d}_{\mathbb{Q}}(n) = <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}} \quad (4)$$

where $\mathbf{x}(n)$ is the quaternion-valued input vector, and $\mathbf{w}_{\mathbb{Q}}$ the quaternion-valued weight vector. The notation $\hat{\mathbf{\Phi}}(.)$ is a kernel transform mapping of the input to a quaternion RKHS, and $<.,.>_{\mathbb{Q}}$ the inner product for this RKHS which is denoted by $\mathbb{Q}$ here.

To proceed we require a description of a quaternion RKHS, which is summarized here. A detailed description of the approach used for obtaining kernel functions suitable for quaternion-valued data can be found in [11].

A function κ can be seen to be a non-negative definite kernel (or simply a kernel) if it satisfies the Mercer condition [5], [10]. In other words, κ may be seen to be a valid kernel if

$$\kappa(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{v})^H \varphi(\mathbf{u}) \quad (5)$$

where $\mathbf{u}$, $\mathbf{v}$ are the quaternion inputs, and $\varphi(.)$ an arbitrary mapping to a RKHS for the quaternion input. Equation (10) forms the definition of the inner product of $\mathbf{u}$, $\mathbf{v}$ in the RKHS.

A suitable kernel for quaternions, shown in Appendix A, is

$$\kappa_{\sigma,\mathbb{Q}^d}(\mathbf{x}, \mathbf{w}) = 4\exp(-\sigma^2 \sum_{j=1}^{d} \left| x_j - w_j \right|^2) \quad (6)$$

We note this kernel function, though suitable for quaternion data, is designed using a real-valued RKHS. This fact was found to aid the derivation of the Quat-KLMS.

IV. Quaternion and iQuaternion Kernel LMS Algorithms

For the Quat-KLMS, the algorithm's goal is finding weights, based on (4), minimizing the mean square error, $E[|e(n)|^2]$, where $e(n) = d(n) - \hat{d}_{\mathbb{Q}}(n)$, with $d(n)$ being the desired response, and $\hat{d}_{\mathbb{Q}}(n)$ the estimator output. Using an LMS-based approach, the instantaneous square error $|e(n)|^2$ is used for the cost function instead. The weight update has the form:

$$\mathbf{w}_{\mathbb{Q}}(n) = \mathbf{w}_{\mathbb{Q}}(n-1) - \eta \nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{|e(n)|^2\} \quad (7)$$

where $\nabla_{\mathbf{w}_{\mathbb{Q}}^*}$ is the gradient with respect to $\mathbf{w}_{\mathbb{Q}}^*$.

Expanding the gradient expression yields

$$\begin{aligned} \nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{|e(n)|^2\} &= \nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{|d(n) - <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}}|^2\} \\ &= \nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{(d(n) - <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}})(d(n) - <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}})^*\} \\ &= \nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{(d(n) - <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}})(d^*(n) - <\mathbf{w}_{\mathbb{Q}}, \hat{\mathbf{\Phi}}(\mathbf{x}(n))>_{\mathbb{Q}})\} \end{aligned} \quad (8)$$

At this point, we consider that for the gradient operation here, the function to be differentiated is defined on a quaternion RKHS, and has a real-valued output. For this case, the modified $\mathbb{HR}$ Calculus obtained from [11] applies. A description of this modified calculus, as well as the properties used, may be found in Appendix B.

The resulting gradient can be seen to be

$$\nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{|e(n)|^2\} = -\hat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n) + \frac{1}{2}e(n)\hat{\mathbf{\Phi}}^*(\mathbf{x}(n))$$

The Quat-KLMS weight update can be written as

$$\mathbf{w}_{\mathbb{Q}}(n) = \mathbf{w}_{\mathbb{Q}}(n-1) + \eta\left(\hat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n) - \frac{1}{2}e(n)\hat{\mathbf{\Phi}}^*(\mathbf{x}(n))\right) \quad (9)$$

It was observed in [11] that use of weight update (9) to form Quat-KLMS resulted in difficulty casting the algorithm in inner products for use of the 'kernel trick' of [5], [12].

However, based on the RKHS mapping of Appendix A, the Quat-KLMS estimates were found to be

$$\hat{d}_{\mathbb{Q}}(n) = \eta \sum_{m=1}^{n}\left[\left(e(m)|s_{\Phi}|^2 - \frac{1}{2}s_{\Phi}e^*(m)s_{\Phi}\right)\kappa_{\sigma,\mathbb{R}^d}(\mathbf{x}_{comp}(n), \mathbf{x}_{comp}(m))\right] \quad (10)$$

where $\kappa_{\sigma,\mathbb{R}^d}(.,.)$ is the real Gaussian kernel, and s_{Φ}, $\mathbf{x}_{comp}$ are defined in Appendix A.

An alternative gradient may be obtained based on the involution gradient (or i-gradient). The i-gradient uses an alternative basis, which is shown to yield convergence benefit in [9].

Use of the i-gradient yields (see Appendix C)

$$\nabla_{\mathbf{w}_{\mathbb{Q}}^*}\{|e(n)|^2\} = -\frac{3}{2}\hat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n) \quad (11)$$

The weight update becomes

$$\mathbf{w}_{\mathbb{Q}}(n) = \mathbf{w}_{\mathbb{Q}}(n-1) + \eta\left(\frac{3}{2}\hat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n)\right)$$

and for iteration n (assuming $\mathbf{w}(0)$=0) we get

$$\mathbf{w}_{\mathbb{Q}}(n) = \eta \sum_{m=1}^{n}\left(\hat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n)\right)$$

where the scaling 3/2 is included in stepsize η.

The resulting estimates become

$$\begin{aligned} \hat{d}_{\mathbb{Q}}(n) &= <\hat{\mathbf{\Phi}}(\mathbf{x}(n)), \mathbf{w}_{\mathbb{Q}}>_{\mathbb{Q}} \\ &= \left[\eta \sum_{m=1}^{n}\left(\hat{\mathbf{\Phi}}(\mathbf{x}(m))e^*(m)\right)\right]^H \hat{\mathbf{\Phi}}(\mathbf{x}(n)) \\ &= \eta \sum_{m=1}^{n}\left[e(m)\hat{\mathbf{\Phi}}^H(\mathbf{x}(m))\hat{\mathbf{\Phi}}(\mathbf{x}(n))\right] \\ &= \eta \sum_{m=1}^{n}\left[e(m)\kappa_{\sigma,\mathbb{Q}^d}(\mathbf{x}(n), \mathbf{x}(m))\right] \end{aligned} \quad (12)$$

which can be seen to have the same form as the Complex Kernel LMS algorithm from reference [12], aside from the quaternionic kernel (A.4) used. We refer to this as the iQuaternion Kernel LMS (iQuat-KLMS) algorithm.

Simulation results comparing the performance of iQuat-KLMS, Quat-KLMS are shown in Section V.

V. Simulations

We use a synthetic quaternion nonlinear channel of the following form for simulations comparing iQuat-KLMS and Quat-KLMS algorithms:.

The channel consisted of initial stage

$$\begin{aligned} z = {} & g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n) \\ & + h_1^* u(n\text{-}1) + h_2^* u^i(n\text{-}1) + h_3^* u^j(n\text{-}1) + h_4^* u^k(n\text{-}1) \end{aligned}$$

followed by the nonlinearity

$$y = z + az^2 + bz^3$$

where coefficients $g_1 \ldots g_4$, $h_1 \ldots h_4$, a, b are random-generated quaternion values.

For quaternion input sequence $u(n)$, the data was either of the form:

$$u(n) = 0.5u_a(n) + i0.5u_b(n) + j0.5u_c(n) + k0.5u_d(n)$$

or:

$$u(n) = (\sqrt{28/32})u_a(n) + i(\sqrt{2/32})u_b(n) + j(\sqrt{1/32})u_c(n) + k(\sqrt{1/32})u_d(n)$$

where $u_a(n)$ to $u_d(n)$ are white Gaussian noise with unit variance. The former form is referred to as $\mathbb{Q}$-proper, while the latter is referred to as $\mathbb{Q}$-improper. The notion of proper refers to the dependence of the input probability distribution to rotations [6], [7]. If the probability distribution is rotation invariant, it is referred to as $\mathbb{Q}$-proper, otherwise it is $\mathbb{Q}$-improper. $\mathbb{Q}$-proper noise was also added at -17dBm.

Graphs of both iQuat-KLMS and Quat-KLMS for the simulated channel are shown in Figures 1-2. For both algorithms, the parameters used were $\eta = 0.35$ and $\sigma = \sqrt{1/8}$, which were found to yield best performance.

For both iQuat-KLMS and Quat-KLMS, the MSE reached -14dBm for both $\mathbb{Q}$-proper and $\mathbb{Q}$-improper data. However, an improvement in convergence can be seen based on use of iQuat-KLMS vs. Quat-KLMS (converges 500-1000 iterations sooner), while the steady-state MSE remains the same. This is consistent with the discussion of Appendix C (as well as reference [9]) on the benefit of i-gradient for improving the convergence steepness with real cost functions.

VI. Conclusions

Here we described an alternative algorithm (the involution Quaternion KLMS, or iQuat-KLMS) for kernel adaptive filtering using quaternion data. The iQuat-KLMS was derived using an alternative formulation of the gradient vs. the Quat-KLMS, which may be seen to provide improved convergence for real-valued cost functions. Simulation tests verified the benefit of the iQuat-KLMS approach.

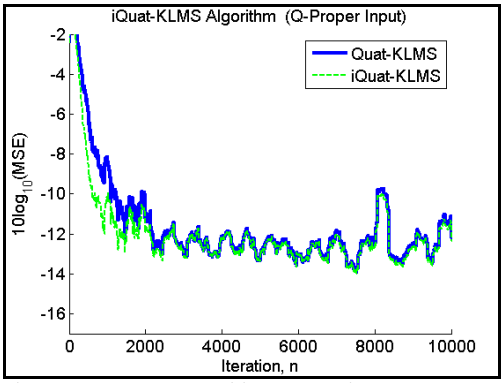

Fig. 1 Quat-KLMS Test with Q-Proper data

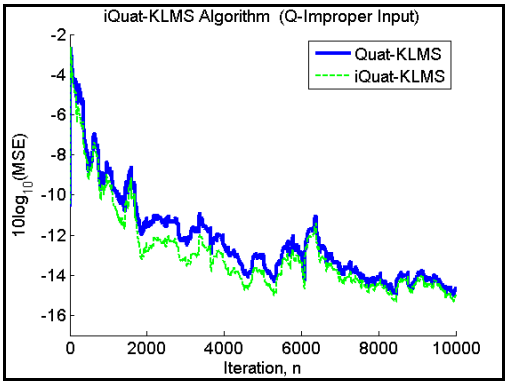

Fig. 2 Quat-KLMS Test with Q-Improper data

Appendix A

For a Gaussian kernel for quaternions, we use the approach of [11] for extending a real RKHS for quaternion data.

For the RKHS mapping, we use the rule:

$$\begin{aligned}\hat{\Phi}(x) &= \hat{\Phi}(x_a + ix_b + jx_c + kx_d) \\ &= \hat{\Phi}(x_a, x_b, x_c, x_d) \\ &= \Phi(x_a, x_b, x_c, x_d) + i\cdot\Phi(x_a, x_b, x_c, x_d) \\ &\quad + j\cdot\Phi(x_a, x_b, x_c, x_d) + k\cdot\Phi(x_a, x_b, x_c, x_d)\end{aligned} \tag{A.1}$$

where $\hat{\Phi}$ is the mapping for quaternion $x = x_a + ix_b + jx_c + kx_d$, and Φ is the feature map of a real kernel κ, defined $\Phi(x_a, x_b, x_c, x_d) = \kappa(\cdot, (x_a, x_b, x_c, x_d))$. The mapping for each real component of the quaternion output (based on the components of the quaternion input) is chosen the same, i.e. feature map Φ of real kernel κ. The resulting RKHS, $\mathcal{Q}$, we refer to as a quaternion-extended RKHS.

The mapping can be simplified to

$$\begin{aligned}\hat{\Phi}(x) &= \Phi(x_a, x_b, x_c, x_d) + i\cdot\Phi(x_a, x_b, x_c, x_d) \\ &\quad + j\cdot\Phi(x_a, x_b, x_c, x_d) + k\cdot\Phi(x_a, x_b, x_c, x_d) \\ &= (1+i+j+k)\cdot\Phi(x_a, x_b, x_c, x_d)\end{aligned} \tag{A.2}$$

due to the fact the kernel transform mapping $\Phi(x_a, x_b, x_c, x_d)$ is the same for each orthogonal direction 1, i, j, k. Note the mapping for a real kernel function is to a real-valued RKHS.

The inner product in RKHS $\mathcal{Q}$ can be expressed as

$$\begin{aligned}<\hat{\Phi}(\mathbf{x}), \hat{\Phi}(\mathbf{w})>_{\mathbb{Q}} &= \hat{\Phi}^H(\mathbf{w})\hat{\Phi}(\mathbf{x}) \\ &= \Phi^T(\mathbf{w}_a, \mathbf{w}_b, \mathbf{w}_c, \mathbf{w}_d)\Phi(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d)\left|1-i-j-k\right|^2 \\ &= 4\kappa\left((\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d), (\mathbf{w}_a, \mathbf{w}_b, \mathbf{w}_c, \mathbf{w}_d)\right)\end{aligned}$$

where $\mathbf{w} = \mathbf{w}_a + i\mathbf{w}_b + j\mathbf{w}_c + k\mathbf{w}_d$ and $\mathbf{x} = \mathbf{x}_a + i\mathbf{x}_b + j\mathbf{x}_c + k\mathbf{x}_d$ are quaternion data vectors.

The result shows any linear method formed by quaternion inner products (i.e. $\mathbf{w}^H\mathbf{x}$) can be converted to a nonlinear method by replacing each of the inner products using a real kernel, whose arguments are the components of inputs $\mathbf{x}, \mathbf{w}$, i.e. $\kappa((\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d), (\mathbf{w}_a, \mathbf{w}_b, \mathbf{w}_c, \mathbf{w}_d))$. This forms an alternative RKHS for nonlinear filtering with quaternions.

With real Gaussian kernel

$$\kappa_{\sigma,\mathbb{R}^d}(\mathbf{u}, \mathbf{u}') = \exp(-\sigma^2\textstyle\sum_{j=1}^{d}(u_j - u_j')^2) \tag{A.3}$$

The quaternion-extended real Gaussian kernel (or Quat-Ex Gaussian kernel for short) is:

$$\begin{aligned}\kappa_{\sigma,\mathbb{Q}^d}(\mathbf{x}, \mathbf{w}) &= 4\exp\Big(-\sigma^2\textstyle\sum_{j=1}^{d}\big[(x_{a,j} - w_{a,j})^2 + (x_{b,j} - w_{b,j})^2 \\ &\qquad + (x_{c,j} - w_{c,j})^2 + (x_{d,j} - w_{d,j})^2\big]\Big) \\ &= 4\exp(-\sigma^2\textstyle\sum_{j=1}^{d}\left|x_j - w_j\right|^2)\end{aligned} \tag{A.4}$$

The function here satisfies the Mercer condition, and thus may be considered a valid kernel form.

If an algorithm may not be easily cast in inner products (depending on the gradient) due to the non-commutative nature of quaternion multiplication, it may be convenient to express transform map $\widehat{\boldsymbol{\Phi}}(\mathbf{x})$ of quaternion input $\mathbf{x}$ as

$$\widehat{\boldsymbol{\Phi}}(\mathbf{x}) = s_{\Phi} \cdot \Phi(\mathbf{x}_{comp}) \tag{A.5}$$

where $s_{\Phi} = (1+i+j+k)$ is a quaternion scalar, and $\mathbf{x}_{comp} = [\mathbf{x}_a^T \mathbf{x}_b^T \mathbf{x}_c^T \mathbf{x}_d^T]^T$ is a vector of the components of $\mathbf{x}$.

APPENDIX B

The modified $\mathbb{HR}$ Calculus for Hilbert spaces of [11] is summarized here. The modified calculus describes a set of rules for gradient evaluations of functions defined on quaternion RKHSs (i.e. quaternion Hilbert spaces). No analyticity restrictions are present from $\mathbb{HR}$ Calculus of [8], which extends to the modified calculus also.

Considering $\mathbb{H}$, $\mathbb{R}^4$ may be viewed as isomorphic, a quaternion transformation $\boldsymbol{T}(.)$ may be equivalently expressed

$$\boldsymbol{T}(\boldsymbol{f}) = \boldsymbol{T}(f_a + i\boldsymbol{f}_b + j\boldsymbol{f}_c + k\boldsymbol{f}_d) = \boldsymbol{T}(f_a, \boldsymbol{f}_b, \boldsymbol{f}_c, \boldsymbol{f}_d) = \boldsymbol{T}(\boldsymbol{f}, \boldsymbol{f}^i, \boldsymbol{f}^j, \boldsymbol{f}^k)$$

where $\boldsymbol{f} = f_a + i\boldsymbol{f}_b + j\boldsymbol{f}_c + k\boldsymbol{f}_d$, and $\boldsymbol{f}^i$, $\boldsymbol{f}^j$, $\boldsymbol{f}^k$ are the involutions.

Rules for differentiability of $\boldsymbol{T}(\boldsymbol{f})$ w.r.t. $\boldsymbol{f}$ and its involutions based on the isomorphism described are outlined in [8] (i.e. the $\mathbb{HR}$ Calculus) and are extended, based on the Fréchet derivative, for Hilbert spaces in [11].

Fréchet -based $\mathbb{HR}$ gradients were found to be:

$$\begin{bmatrix} \nabla_{\boldsymbol{f}} \boldsymbol{T}(\boldsymbol{f}, \boldsymbol{f}^i, \boldsymbol{f}^j, \boldsymbol{f}^k)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^i} \boldsymbol{T}(\boldsymbol{f}, \boldsymbol{f}^i \boldsymbol{f}^j \boldsymbol{f}^k)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^j} \boldsymbol{T}(\boldsymbol{f}, \boldsymbol{f}^i, \boldsymbol{f}^j, \boldsymbol{f}^k)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^k} \boldsymbol{T}(\boldsymbol{f}, \boldsymbol{f}^i, \boldsymbol{f}^j, \boldsymbol{f}^k)(\boldsymbol{c}) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & -i & -j & -k \\ 1 & -i & j & k \\ 1 & i & -j & k \\ 1 & i & j & -k \end{bmatrix} \begin{bmatrix} \nabla_{f_a} \boldsymbol{T}(f_a, \boldsymbol{f}_b, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_b} \boldsymbol{T}(f_a, \boldsymbol{f}_b, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_c} \boldsymbol{T}(f_a, \boldsymbol{f}_b, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_d} \boldsymbol{T}(f_a, \boldsymbol{f}_b, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \end{bmatrix} \tag{B.1}$$

Similarly, Fréchet $\mathbb{HR}^*$ gradients were:

$$\begin{bmatrix} \nabla_{\boldsymbol{f}^*} \boldsymbol{T}(\boldsymbol{f}^*, \boldsymbol{f}^{i*}, \boldsymbol{f}^{j*}, \boldsymbol{f}^{k*})(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^{i*}} \boldsymbol{T}(\boldsymbol{f}^*, \boldsymbol{f}^{i*}, \boldsymbol{f}^{j*}, \boldsymbol{f}^{k*})(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^{j*}} \boldsymbol{T}(\boldsymbol{f}^*, \boldsymbol{f}^{i*}, \boldsymbol{f}^{j*}, \boldsymbol{f}^{k*})(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}^{k*}} \boldsymbol{T}(\boldsymbol{f}^*, \boldsymbol{f}^{i*}, \boldsymbol{f}^{j*}, \boldsymbol{f}^{k*})(\boldsymbol{c}) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & i & j & k \\ 1 & i & -j & -k \\ 1 & -i & j & -k \\ 1 & -i & -j & k \end{bmatrix} \begin{bmatrix} \nabla_{f_a} \boldsymbol{T}(f_a, \boldsymbol{f}_b\, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_b} \boldsymbol{T}(f_a, \boldsymbol{f}_b\, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_c} \boldsymbol{T}(f_a, \boldsymbol{f}_b\, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \\ \nabla_{\boldsymbol{f}_d} \boldsymbol{T}(f_a, \boldsymbol{f}_b\, \boldsymbol{f}_c, \boldsymbol{f}_d)(\boldsymbol{c}) \end{bmatrix} \tag{B.2}$$

where the gradient is considered at point $\boldsymbol{c}$ in all cases.

Some properties based on the modified $\mathbb{HR}$ calculus are:

1. If $\boldsymbol{T}(\boldsymbol{f}) = \langle \boldsymbol{h}, \boldsymbol{f}^* \rangle_{\mathcal{Q}}$, then $\nabla_{\boldsymbol{f}} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}$ and $\nabla_{\boldsymbol{f}^{\eta}} \boldsymbol{T}(\boldsymbol{c}) = 0$ for $\eta \in \{i, j, k\}$ at every $\boldsymbol{c}$. In addition, we also have: $\nabla_{\boldsymbol{f}^*} \boldsymbol{T}(\boldsymbol{c}) = -\boldsymbol{h}/2$ and $\nabla_{\boldsymbol{f}^{\eta *}} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}/2$ for $\eta \in \{i, j, k\}$.
2. If $\boldsymbol{T}(\boldsymbol{f}) = \langle \boldsymbol{h}, \boldsymbol{f} \rangle_{\mathcal{Q}}$, then $\nabla_{\boldsymbol{f}^*} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}$ and $\nabla_{\boldsymbol{f}^{\eta *}} \boldsymbol{T}(\boldsymbol{c}) = 0$ for $\eta \in \{i, j, k\}$ at every $\boldsymbol{c}$. In addition, we also have: $\nabla_{\boldsymbol{f}} \boldsymbol{T}(\boldsymbol{c}) = -\boldsymbol{h}/2$ and $\nabla_{\boldsymbol{f}^{\eta}} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}/2$ for $\eta \in \{i, j, k\}$.
3. If $\boldsymbol{T}(\boldsymbol{f}) = \langle \boldsymbol{h}, \boldsymbol{f}^i \rangle_{\mathcal{Q}}$, then $\nabla_{\boldsymbol{f}^{i*}} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}$ and $\nabla_{\boldsymbol{f}^*} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^{j*}} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^{k*}} \boldsymbol{T}(\boldsymbol{c}) = 0$ at every $\boldsymbol{c}$. In addition, we also have: $\nabla_{\boldsymbol{f}^i} \boldsymbol{T}(\boldsymbol{c}) = -\boldsymbol{h}/2$ and $\nabla_{\boldsymbol{f}} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^j} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^k} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}/2$
4. If $\boldsymbol{T}(\boldsymbol{f}) = \langle \boldsymbol{h}, \boldsymbol{f}^{i*} \rangle_{\mathcal{Q}}$, then $\nabla_{\boldsymbol{f}^i} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}$ and $\nabla_{\boldsymbol{f}} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^j} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^k} \boldsymbol{T}(\boldsymbol{c}) = 0$ at every $\boldsymbol{c}$. In addition, we also have: $\nabla_{\boldsymbol{f}^{i*}} \boldsymbol{T}(\boldsymbol{c}) = -\boldsymbol{h}/2$ and $\nabla_{\boldsymbol{f}^*} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^{j*}} \boldsymbol{T}(\boldsymbol{c}) = \nabla_{\boldsymbol{f}^{k*}} \boldsymbol{T}(\boldsymbol{c}) = \boldsymbol{h}/2$.
5. Product Rule: When $\boldsymbol{T}, \boldsymbol{S} : \mathcal{Q} \to \mathbb{H}$ are Fréchet differentiable in the real sense at $\boldsymbol{c} \in \mathcal{Q}$, then:
$\nabla_{\chi}(\boldsymbol{T} \cdot \boldsymbol{S})(\boldsymbol{c}) = [\nabla_{\chi} \boldsymbol{T}(\boldsymbol{c})] \boldsymbol{S}(\boldsymbol{c}) + \boldsymbol{T}(\boldsymbol{c}) [\nabla_{\chi} \boldsymbol{S}(\boldsymbol{c})]$
where $\chi \in \{\boldsymbol{f}, \boldsymbol{f}^*, \boldsymbol{f}^{\eta}, \boldsymbol{f}^{\eta *}\}$, $\forall \eta \in \{i, j, k\}$ (note quaternion multiplication is non-commutative).

These gradient properties were used to obtain the result (9).

APPENDIX C

In [9], the involution gradient (or i-gradient) is based on an alternative definition for the gradient when compared to [8]. The i-gradient was formed based on viewing gradient $\partial f / \partial q^*$ in terms of involution-wise partial derivatives $\partial f / \partial q^i$, $\partial f / \partial q^j$, and $\partial f / \partial q^k$.

Based on the gradients

$$\frac{\partial f(q, q^i, q^j, q^k)}{\partial q} = \frac{1}{4} \left[\frac{\partial f}{\partial q_a} - i \frac{\partial f}{\partial q_b} - j \frac{\partial f}{\partial q_c} - k \frac{\partial f}{\partial q_d} \right] \tag{C.1}$$

$$\frac{\partial f(q, q^i, q^j, q^k)}{\partial q^*} = \frac{1}{4} \left[\frac{\partial f}{\partial q_a} + i \frac{\partial f}{\partial q_b} + j \frac{\partial f}{\partial q_c} + k \frac{\partial f}{\partial q_d} \right]$$

it was shown that $\partial f / \partial q^*$ may be expressed, in terms of the involution-wise partial derivatives, as

$$\mathrm{Re}\left[\frac{\partial f}{\partial q^*} \right] = \frac{1}{3} \mathrm{Re}\left[\frac{\partial f}{\partial q^i} + \frac{\partial f}{\partial q^j} + \frac{\partial f}{\partial q^k} \right] \tag{C.2}$$

$$\mathrm{Im}\left[\frac{\partial f}{\partial q^*} \right] = \mathrm{Im}\left[\frac{\partial f}{\partial q^i} + \frac{\partial f}{\partial q^j} + \frac{\partial f}{\partial q^k} \right]$$

Thus, defining the gradient alternatively as

$$\nabla_{\omega^n} f = \frac{\partial f}{\partial q^i} + \frac{\partial f}{\partial q^j} + \frac{\partial f}{\partial q^k} \tag{C.3}$$

only differs from $\partial f / \partial q^*$ of (C.1) by the magnitude of the real component (i.e. scaled by factor of 3).

The i-gradient approach was also shown from [9] to differ from the original gradient by added term $0.5\, \partial f / \partial q_a$, i.e.

$$\nabla_{\omega^n} f = \frac{\partial f}{\partial q^*} + \frac{1}{2} \frac{\partial f}{\partial q_a}$$

Thus, using the i-gradient can be seen to accelerate the descent for minimizing a real-valued cost function w.r.t. the real component of quaternion q (i.e. q_a). This should yield faster convergence.

For finding the i-gradient of equation (8) in Section IV, if we use

$$\mathbf{w}_Q = \tfrac{1}{2} (\mathbf{w}_Q^{i*} + \mathbf{w}_Q^{j*} + \mathbf{w}_Q^{k*} - \mathbf{w}_Q^*)$$

with both $e(n)$ and $e^*(n)$, it can be seen that

$$\frac{\partial e(n)}{\partial \mathbf{w}_Q^{\beta}} = -\frac{1}{2} \widehat{\boldsymbol{\Phi}}(\mathbf{x}(n)) \qquad \frac{\partial e^*(n)}{\partial \mathbf{w}_Q^{\beta}} = 0 \qquad \forall \beta \in \{i, j, k\}$$

Thus using (C.3), along with Property 5 in Appendix B, we obtain

$$\nabla_{\mathbf{w}_o^*}\{|e(n)|^2\} = -\frac{3}{2}\widehat{\mathbf{\Phi}}(\mathbf{x}(n))e^*(n)$$

which is the i-gradient result shown in Section IV.

REFERENCES

[1] Tokunbo Ogunfunmi, *Adaptive Nonlinear System Identification: The Volterra and Wiener Model Approaches*, published by Springer Publishers, 2007.

[2] Tokunbo Ogunfunmi and Thomas Paul, "On the Complex Kernel-based Adaptive Filter", *Proceedings of the IEEE Int. Symp. on Circuits and Systems*, pp. 1263-1266, May 2011.

[3] Thomas Paul and Tokunbo Ogunfunmi, "A Study of the Convergence Behavior of the Complex Kernel LMS Algorithm", submitted to *IEEE Transactions on Neural Networks and Learning Systems*, June 2012.

[4] Tokunbo Ogunfunmi and Thomas Paul, "Kernel-Based APA Adaptive Filters for Complex Data", Proceedings of the *Asia Pacific Signal and Information Processing. Association. (APSIPA) Conference*, Oct. 2011.

[5] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering : A Comprehensive Introduction*, published by Wiley Publishers, 2010.

[6] S. Javidi, C. Cheong Took, C. Jahanchahi, N. Le Bihan, and D. P. Mandic, "Blind Extraction of Improper Quaternion Sources", *IEEE Proc. of the International Conference on Acoustic Speech and Signal Processing.*, May 2011. Available at: http://www.commsp.ee.ic.ac.uk/~mandic/research

[7] C. Cheong Took, and D. P. Mandic, "A Quaternion Widely Linear Adaptive Filter", *IEEE Transactions on Signal Processing*, vol. 58, no. 8, August 2010.

[8] D. P. Mandic, C. Jahanchahi, and C. Cheong Took, "A Quaternion Gradient Operator and its Applications", *IEEE Signal Processing Letters*, vol. 18, no. 1, January 2011.

[9] C. Jahanchahi, C. Cheong Took, and D. P. Mandic, "On Gradient Calculations in Quaternion Adaptive Filtering", *Proc. of Inter. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2012, pp. 3773-3776.

[10] A. Shilton, D. T. H. Lai, "Quaternionic and Complex-Valued Support Vector Regression for Equalization and Function Approximation", *Proc. of Inter. Joint Conf. on Neural Networks*, Aug. 2007.

[11] Thomas Paul and Tokunbo Ogunfunmi, "A Kernel Adaptive Filtering Algorithm for Quaternion-Valued Inputs", submitted to *IEEE Transactions on Signal Processing*, July 2012.

[12] P. Bouboulis, and S. Theodoridis, "Extension of Wirtinger's Calculus to Reproducing Kernel Hilbert Spaces and the Complex Kernel LMS", *IEEE Transactions on Signal Processing*, vol. 59, no. 3, March 2011.

[13] I. Steinwart, D. Hush, C. Scovel, "An Explicit Description of the Reproducing Kernel Hilbert Spaces of Gaussian RBF kernels", *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4635–4643, 2006