2019

Quaternion Information Theoretic Learning Adaptive Algorithms for Nonlinear Adaptive

Carlo Safarian

Quaternion Information Theoretic Learning Adaptive Algorithms for Nonlinear Adaptive Systems

By

Carlo Safarian

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering
in the School of Engineering at
Santa Clara University, 2019

Santa Clara, California

Acknowledgments

"16 Always be rejoicing. 17 Pray constantly. 18 Give thanks for everything. This is God's will for you in Christ Jesus." ***1 Thessalonians 5:16-18***

I would like to express my sincere gratitude to my advisor Prof. Dr. Tokunbo Ogunfunmi for his support, patience, motivation, knowledge and encouragement. His guidance and technical knowledge has taught me numerous lessons and insights in the field of my academic research and Ph.D study.

Besides my advisor, I would like to thank my thesis committee: Prof. Dr. Shoba Krishnan, Prof. Dr. Maryam Khanbaghi, Prof. Dr. Sarah Wilson and Prof. Dr. Xiaoshu Qian for their insightful comments and encouragement and also for their all deep questions which gave me widen vision from various perspectives in my research.

I would like to dedicate this dissertation to my beloved parents, Mr. Sepanous Safarian and Mrs. Sedik Safarian. They have always supported me and sacrificed their life to raise me to be a better person. I hope, I have made both of them proud. I also would like to express my special appreciation and thanks to my lovely siblings, Carolin Safarian and Catherine Safarian for their endless love, support and prayers.

Finally, I would like to thank everyone who has been a part of my life directly or indirectly and supported me during this entire journey.

Quaternion Information Theoretic Learning Adaptive Algorithms for Nonlinear Adaptive Systems

Carlo Safarian

Department of Electrical Engineering
Santa Clara University
Santa Clara, California
2019

ABSTRACT

Information Theoretic Learning (ITL) is gaining popularity for designing adaptive filters for a non-stationary or non-Gaussian environment [1] [2] . ITL cost functions such as the Minimum Error Entropy (MEE) have been applied to both linear and nonlinear adaptive filtering with better overall performance compared with the typical mean squared error (MSE) and least-squares type adaptive filtering, especially for nonlinear systems in higher-order statistic noise environments [3].

Quaternion valued data processing is beneficial in applications such as robotics and image processing, particularly for performing transformations in 3-dimensional space. Particularly 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 [4, 5, 6, 7, 8]. Adaptive filtering in quaterion domain operates intrinsically based on the augmented statistics which the quaternion input vector covariance is taken into account naturally and as a result it incorporates component-wise real valued cross-correlation or the coupling within the dimensions of the quaternion input [9].

The generalized Hamilton-real calculus (GHR) for the quaternion data simplified product and chain rules and allows us to calculate the gradient and Hessian of quaternion

based cost function of the learning algorithms efficiently [10][11] . The quaternion reproducing kernel Hilbert spaces and its uniqueness provide a mathematical foundation to develop the quaternion value kernel learning algorithms [12]. The reproducing property of the feature space replace the inner product of feature samples with kernel evaluation.

In this dissertation, we first propose a kernel adaptive filter for quaternion data based on minimum error entropy cost function. The new algorithm is based on error entropy function and is referred to as the quaternion kernel minimum error entropy (QKMEE) algorithm [13]. We apply generalized Hamilton-real (GHR) calculus that is applicable to quaternion Hilbert space for evaluating the cost function gradient to develop the QKMEE algorithm. The minimum error entropy (MEE) algorithm [3, 14, 15] minimizes Renyis quadratic entropy of the error between the filter output and desired response or indirectly maximizing the error information potential. ITL methodology improves the performance of adaptive algorithm in biased or non-Gaussian signals and noise enviorments compared to the mean squared error (MSE) criterion algorithms such as the kernel least mean square algorithm.

Second, we develop a kernel adaptive filter for quaternion data based on normalized minimum error entropy cost function [14]. We apply generalized Hamilton-real (GHR) calculus that is applicable to Hilbert space for evaluating the cost function gradient to develop the quaternion kernel normalized minimum error entropy (QKNMEE) algorithm [16]. The new proposed algorithm enhanced QKMEE algorithm where the filter update stepsize selection will be independent of the input power and the kernel size.

Third, we develop a kernel adaptive filter for quaternion domain data, based on information theoretic learning cost function which could be useful for quaternion based kernel applications of nonlinear filtering. The new algorithm is based on error entropy function with fiducial point and is referred to as the quaternion kernel minimum error entropy with fiducial point (QKMEEF) algorithm [17]. In our previous work we de-

veloped quaternion kernel adaptive filter based on minimum error entropy referred to as the QKMEE algorithm [13]. Since entropy does not change with the mean of the distribution, the algorithm may converge to a set of optimal weights without having zero mean error. Traditionally, to make the zero mean output error, the output during testing session was biased with the mean of errors of training session. However, for non-symmetric or heavy tails error PDF the estimation of error mean is problematic [18]. The minimum error entropy criterion, minimizes Renyi's quadratic entropy of the error between the filter output and desired response or indirectly maximizing the error information potential [19]. Here, the approach is applied to quaternions. Adaptive filtering in quaterion domain intrinsically incorporates component-wise real valued cross-correlation or the coupling within the dimensions of the quaternion input. We apply generalized Hamilton-real (GHR) calculus that is applicable to Hilbert space for evaluating the cost function gradient to develop the Quaternion Minimum Error Entropy Algorithm with Fiducial point. Simulation results are used to show the behavior of the new algorithm (QKMEEF) when signal is non-Gaussian in presence of unimodal noise versus bi-modal noise distributions. Simulation results also show that the new algorithm QKMEEF can track and predict the 4-Dimensional non-stationary process signals where there are correlations between components better than quadruple real-valued KMEEF and Quat-KLMS algorithms.

Fourth, we develop a kernel adaptive filter for quaternion data, using stochastic information gradient (SIG) cost function based on the information theoretic learning (ITL) approach. The new algorithm (QKSIG) is useful for quaternion-based kernel applications of nonlinear filtering [20]. Adaptive filtering in quaterion domain intrinsically incorporates component-wise real valued cross-correlation or the coupling within the dimensions of the quaternion input. We apply generalized Hamilton-real (GHR) calculus that is applicable to quaternion Hilbert space for evaluating the cost function gradi-

ent. The QKSIG algorithm minimizes Shannon's entropy of the error between the filter output and desired response and minimizes the divergence between the joint densities of input-desired and input-output pairs. The SIG technique reduces the computational complexity of the error entropy estimation. Here, ITL with SIG approach is applied to quaternion adaptive filtering for three different reasons. First, it reduces the algorithm computational complexity compared to our previous work quaternion kernel minimum error entropy algorithm (QKMEE). Second, it improves the filtering performance by considering the coupling within the dimensions of the quaternion input. Third, it performs better in biased or non-Gaussian signal and noise environments due to ITL approach. We present convergence analysis and steady-state performance analysis results of the new algorithm (QKSIG). Simulation results are used to show the behavior of the new algorithm QKSIG in quaternion non-Gaussian signal and noise environments compared to the existing ones such as quadruple real-valued kernel stochastic information gradient (KSIG) and quaternion kernel LMS (QKLMS) algorithms.

Fifth, we develop a kernel adaptive filter for quaternion data, based on stochastic information gradient (SIG) cost function with self adjusting step-size. The new algorithm (QKSIG-SAS) is based on the information theoretic learning (ITL) approach. The new algorithm (QKSIG-SAS) has faster speed of convergence as compared to our previous work QKSIG algorithm.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information Theoretic Learning (ITL) [1][2] such as minimum error entropy is gaining popularity for designing adaptive filters for a non-stationary or non-Gaussian environment. The adaptive filtering cost function based on mean squared error (MSE) uses only second order statistics and does not capture the probability of error distribution in the system. An information theoretic learning (ITL) alternative such as minimum error entropy (MEE) uses minimum error entropy as a cost function. Minimizing error entropy minimizes the distance between the joint probability distribution of the input-desired and input-output of adaptive system and is expected to perform better with biased or non-Gaussian signals compared to mean-squared-error (MSE) criteria adaptive filters [3].

Quaternion domain statistical signal processing has wide range of applications in areas such as as adaptive filtering for earthquake prediction, wind forecasting, stock market prediction and EEG. These applications have multi dimensional nature with correlations or couplings between dimensions. In real world applications such as wind forecasting, there are correlations between different dimensions such as east-north, east-vertical, and north-vertical, respectively. Therefore, considering the correlations between different dimensions leads to better prediction. Adaptive filtering in quaternion domain operates intrinsically based on the augmented statistics which the quaternion input vector $\mathbf{q}$ covariance $E[\mathbf{q}\mathbf{q}^H]$ is taken into account naturally [22]. As a result it incorporates component-wise real valued cross-correlation or the coupling within the dimensions of

the quaternion input [9]. Adaptive filtering of 4-Dimensional signals in real domain requires multiple uni-variate filters without incorporating the mutual information or coupling between all four components [22].

1.1 Purpose of this Work

The quaternion kernel estimation is an emerging field and some algorithms are developed in quaternion domain [23],[24],[25] and [26]. The purpose of this thesis is to develop kernel adaptive filter algorithms for quaternion domain data, based on information theoretic learning cost functions which could be useful for quaternion based kernel applications of nonlinear filtering. In order to develop quaternion kernel nonlinear filters based on information theoretic learning, first the information theory concepts such as entropy is extended to quaternion domain; second, the generalized Hamilton-real calculus (GHR) for the quaternion data [10] are used to enable the gradient calculation of the optimization problems. The GHR calculus simplified product and chain rules allows us to calculate the quaternion based gradient and Hessian of cost function efficiently and use them for the learning algorithms [11],[27]. The quaternion reproducing kernel Hilbert spaces and its uniqueness [12], provides a mathematical foundation to develop the quaternion value kernel learning algorithms.

1.2 Organization of the Thesis

This thesis is organized as follows. The Chapters 2 and 3 describe background material which formed the basis of the new research. The areas covered in these chapters are:

1. Introduction to Information Theoretic Learning (Chapter 2).

2. Quaternions and Properties (Chapter 3).

The chapters 4 to 8 describe the new algorithms and related analysis. These are:

1. Quaternion Kernel Minimum Error Entropy Adaptive Filter (Chapter 4).
2. Quaternion Kernel Normalized Minimum Error Entropy Algorithm (Chapter 5).
3. Quaternion Kernel Minimum Error Entropy Algorithm with Fiducial point for Nonlinear Adaptive Systems (Chapter 6).
4. Quaternion Kernel Stochastic Information Gradient Algorithm for Nonlinear Adaptive Systems (Chapter 7).
5. Quaternion Kernel Stochastic Information Gradient Algorithm with Self Adjusting Step-size for Nonlinear Adaptive Systems (Chapter 8).

Finally, Chapter 9 summarizes the research and describes future work.

Part I

Background

Chapter 2

Introduction to Information Theoretic Learning

This chapter describes the fundamental methodology which uses information theory to develop adaptive information filters named information theoretic learning (ITL). In information theoretic learning (ITL) methodology, the cost function criterion of traditional adaptive filters, replaced by new criterion based on information theory principles such as entropy [28]. The ITL methodology plays important growing roles in adaptive signal processing and machine learning areas [1][29].

In all supervised adaptive signal processing problems such as system identification, noise canceling and channel equalization the goal is to minimize the difference between the desired and the system outputs. In traditional adaptive filtering the mean-squared-error (MSE) is used as the optimal criterion to minimize the error between the desired and the system outputs. The MSE criterion, minimizes the error energy and its implementation is easy. The main drawback of the MSE criterion lays on its statistical characteristic nature that only takes into account the second order statistics and is only optimal in the case of Gaussian signals and linear filters [30] [31].

In adaptive filtering especially for nonlinear signal processing, instead of constraining directly energy of error, constraining the information content of signal achieves better performance [3]. Entropy, such as Shannon entropy [32] and Renyi's entropy [33], is a scalar quantity that provides a measure for the average information contained in a given PDF. When error entropy is minimized, all moments of the error PDF are constrained

[31][29]. The entropy criterion has been utilized as an alternative for MSE in supervised adaptive learning and minimizing error entropy (MEE) shows more robust criterion than MSE in adaptive learning process especially for nonlinear systems.

Information theory measures the statistical uncertainty in random processes, statistical similarity and dependencies between multiple random processes. The two main statistical measures in information theory are entropy and divergence.

2.1 Entropy

Entropy measures the uncertainty of the random vector X with probability distribution function (PDF) p(x) which is a generalization of variance to processes with non-Gaussian distributions, and is defined by Shannon as [32][34]

$$H_S(X) = -\int_{\infty}^{\infty} p(x) \log p(x) dx = E[-\log p(x)] \tag{2.1}$$

where $p(x)$ is probability distribution function of random variable X.

2.2 Divergence

Divergence, a measure of statistical similarity, such as Kullback-Leibler divergence (KLD), measures the distance between two distributions p(x) and q(x) and is defined as [34]

$$D_{KL}(p||q) = \int_{\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx. \tag{2.2}$$

This measure becomes zero if and only if p and q are identical distributions and is

positive otherwise.

2.3 Mutual information

Mutual information, which is a measure of statistical dependency, is a generalized form of correlation to arbitrary nonlinear transforms of multiple processes having arbitrary distributions [1]. The Mutual information as a special case of KLD, measures the distance between the joint probability distribution and the product of the marginal distributions and is defined as

$$I_S(p;q) = \int_{\infty}^{\infty} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy. \tag{2.3}$$

These are some of the key measures in information theory and can be leveraged to information theoretic learning (ITL) to derive algorithms based on information costs instead of second-order (quadratic) costs. However adaptive filtering deals with continuous random variables, described by their PDF. This means that the analytic approach taken in information theory must be modified with continuous and differentiable non-parametric estimators of entropy and divergence. Parzen estimation estimates the probability distributions of signals in the system using non-parametric sample estimators and has advantage of linking information theory, adaptation, and kernel methods.

2.4 Parzen window

The *Parzen window* [35] for a set of N statistically independent random samples $\{x_i\}_{i=1}^{N}$ of random variable x, computes the estimate of the probability distribution function

$p(x)$ as

$$\hat{p(x)} = \frac{1}{N}\sum_{l=1}^{N}\kappa_\sigma(x - x_l) \tag{2.4}$$

where κ_σ is Gaussian kernel defined as

$$\begin{aligned}\kappa_\sigma(x-y) &= \frac{4}{\sqrt{2\pi}\sigma}\exp\left[\frac{-1}{2\sigma^2}(x-y)^2\right] \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left[\frac{-1}{2\sigma^2}|x-y|^2\right].\end{aligned}$$

2.5 Adaptive Information Filtering

In the conventional adaptive filtering method such as least mean squares, the goal is to minimize the mean-squared-error (MSE) between the desired response z and the system output $y = f_{\mathbf{w}}(x)$ with respect to free parameter $\mathbf{w}$ as

$$\min_{\forall \mathbf{w}} \quad J(w) = E[(z - f_{\mathbf{w}}(x))^2]. \tag{2.5}$$

This corresponds to estimating the orthogonal projection of the system desired response z in the space which is spanned by the system input signal x.

Alternatively, the problem of estimating the system parameters $\mathbf{w}$ can be done by minimizing the divergence between PDFs of system input-output and input-desired [1]. The desired response z can be estimated by an unknown mapper of the input vector x defined as $z = f_{\mathbf{w}}(x) + e$, where e is the error and $f_{\mathbf{w}}(x)$ is linear or non-linear mapping function. Therefore, the joint PDF $p(x, z)$ of input-desired fully characterizes the mapping relationship between input and desired. The mapper estimates the system output y, which is a parametric function of the input, with parameter vector $\mathbf{w}$. Intuitively the

system output can be an estimator $\hat{p}_{\mathbf{w}}(x,z)$ of $p(x,z)$. Therefore, in statistical view, the optimization problem is to minimize the KLD between these two distributions as:

$$\min_{\forall \mathbf{w}} \quad J(w) = \int_{\infty}^{\infty} \int_{\infty}^{\infty} p(x,z) \log \frac{p(x,z)}{\hat{p}_{\mathbf{w}}(x,z)} dxdz. \tag{2.6}$$

It can be shown that minimizing the Kullback-Leibler divergence between the joint probability distribution of the input-desired and input-output is equivalent to minimizing the entropy of the error signal.

$$\begin{aligned} &\min_{\forall \mathbf{w}} \quad -\int_{\infty} p_e(\varepsilon) \log p_e(\varepsilon) d\varepsilon \\ &\equiv \min_{\forall \mathbf{w} \in \mathcal{H}} \int_{\infty} \int_{\infty} \int_{\infty} p_{\mathbf{y},\mathbf{x}|\mathbf{w}}(y,x) \log \frac{p_{\mathbf{y},\mathbf{x}|\mathbf{w}}(y,x)}{p_{\mathbf{x},\mathbf{z}}(x,\zeta)} dydxd\zeta \\ &= \min_{\forall \mathbf{w} \in \mathcal{H}} D_{KL}(p_{\mathbf{y},\mathbf{x}|\mathbf{w}} || p_{\mathbf{x},\mathbf{z}}) \end{aligned} \tag{2.7}$$

where $p_{\mathbf{y},\mathbf{x}}$ is the input-output joint PDF and $p_{\mathbf{x},\mathbf{z}}$ is the input-desired joint PDF.

ITL is defined as a set of algorithms to implement adaptive information filtering. In many signal processing and machine learning problems, the probability density function of the data is unknown, therefore fundamental issue in ITL is how to estimate entropy and divergence directly from samples. ITL cost functions such as the Minimum Error Entropy (MEE) have been applied to both linear and nonlinear adaptive filtering with better overall performance compared with the typical mean squared error (MSE) and least-squares type adaptive filtering, especially for nonlinear systems in higher-order statistic noise environments.

2.6 Non-Linear Kernel Adaptive Filter Theory

In this section we describe the basics of the non-linear adaptive filter theory. Details about the materials could be found in "Kernel Adaptive Filtering: A Comprehensive Introduction" [21].

2.6.1 Reproducing Kernel Hilbert Space

A Hilbert space is a vector space $\mathbb{H}$ with an inner product $< f, g >$ such that the norm defined by $\|f\|_{\mathbb{H}}^2 =< f, f >$, turns $\mathbb{H}$ into a complete metric space. An inner product space $\mathbb{H}$ is complete if every Cauchy sequence of vectors taken from the space $\mathbb{H}$ converges to a limit in $\mathbb{H}$. A complete inner product space is called a Hilbert space.

Suppose an inner product space $\mathbb{H}$ has an orthonormal basis $\{x_k\}_{k=1}^{\infty}$ then the vectors sequence $\{\mathbf{y}_n\}_{k=1}^{\infty}$ spanned by the basis $\{x_k\}_{k=1}^{\infty}$ defined as

$$\mathbf{y}_n = \sum_{k=1}^{n} a_k \mathbf{x}_k, a_k \in \mathbb{R} \tag{2.8}$$

is Cauchy sequence if the squared Euclidean distance between the vector $\mathbf{y}_n$ and $\mathbf{y}_m$ for $(m > n)$, can meet the following condition:

$$\lim_{(m,n)\to\infty} \|\mathbf{y}_n - \mathbf{y}_m\|_{\mathbb{H}}^2 = 0. \tag{2.9}$$

A *Mercer kernel* is a continuous, symmetric, positive definite function $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ where $\mathbb{U}$ is the input domain and subset of $\mathbb{R}^L$. The Gaussian kernel and the polynomial kernel are two kernels which are used commonly and defined respectively as follows:

$$\kappa(\mathbf{u}, \mathbf{u}') = \exp(\left\|\mathbf{u} - \mathbf{u}'\right\|_{\mathbb{U}}^2) \tag{2.10}$$

$$\kappa(\mathbf{u}, \mathbf{u}') = (\mathbf{u}^T\mathbf{u}' + 1)^p \tag{2.11}$$

where p is positive integer number.

Suppose $\mathbb{H}$ is a vector space of all real-valued functions of $\mathbf{u}$ that are generated by the kernel $\kappa(\mathbf{u}, .)$. The bilinear form of two functions $h(.)$ and $g(.)$ in space $\mathbb{H}$ can be defined as

$$< h, g >= \sum_{i=1}^{l}\sum_{j=1}^{m} a_i\kappa(\mathbf{c}_i, \hat{\mathbf{c}}j)b_j \tag{2.12}$$

where functions $h(.)$ and $g(.)$ are in space $\mathbb{H}$ defined respectively as below

$$h = \sum_{i=1}^{l} a_i\kappa(\mathbf{c}_i, .) \tag{2.13}$$

$$g = \sum_{j=1}^{m} b_j\kappa(\hat{\mathbf{c}}j, .) \tag{2.14}$$

where the a_i and the b_j are expansion coefficients and $\mathbf{c}_i$ and $\hat{\mathbf{c}}j \in \mathbb{U}$ for all i and j .

The bilinear form of two functions $h(.)$ and $g(.)$ satisfies the following properties:

1. Symmetry, $< h, g >=< g, h >$.
2. Scaling and distributive property, $< cf + dg, h >= c < f, g > +d < g, h >$.
3. Squared norm, $\|f\|^2 =< f, f >\geqslant 0$.

The bilinear term $< h, g >$ is indeed an inner product.

One of the properties of bilinear term $< h, g >$ is the reproducing property which is defined as

$$< h, \kappa(\mathbf{u}, .) >= h(\mathbf{u}). \tag{2.15}$$

By setting $g(.) = \kappa(\mathbf{u}, .)$ we can simplify the bilinear term $< h, g >$ as

$$< h, \kappa(\mathbf{u}, .) >= \sum_{i=1}^{l} a_i \kappa(\mathbf{c}_i, \mathbf{u}) = h(\mathbf{u}). \tag{2.16}$$

Definition 2.6.1. (Reproducing kernel Hilbert space (RKHS)): The kernel $\kappa(\mathbf{u}, \mathbf{u}')$, which represents a function of the two vectors $\mathbf{u}$, $\mathbf{u}' \in \mathbb{U}$, is called a reproducing kernel of the vector space $\mathbb{H}$ if it satisfies the following three conditions:

1. For every $\mathbf{u} \in \mathbb{U}$, $\kappa(\mathbf{u}, \mathbf{u}')$ as a function of the vector $\mathbf{u}'$ belongs to $\mathbb{H}$
2. It satisfies the reproducing property

If the inner product space $\mathbb{H}$, in which the reproducing kernel space is defined, is also complete, then it is called a reproducing kernel Hilbert space (RKHS).

Theorem 1. (Mercer theorem): Any reproducing kernel $\kappa(\mathbf{u}, \mathbf{u}')$ can be expanded as follows:

$$\kappa(\mathbf{u}, \mathbf{u}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{u}) \phi_i(\mathbf{u}') \tag{2.17}$$

where λ_i and ϕ_i are the eigenvalues and the eigenfunctions respectively, and eigenvalues are non-negative.

Therefore, a mapping φ can be constructed as

$$\varphi : \mathbb{U} \longrightarrow \mathbb{F}$$
$$\varphi(\mathbf{u}) = [\sqrt{\lambda_1}\phi_1(\mathbf{u}), \sqrt{\lambda_2}\phi_2(\mathbf{u}), ...]$$

where φ is called the feature mapping and $\varphi(\mathbf{u})$ is the transformed feature vector lying in the feature space $\mathbb{F}$. The dimension of $\mathbb{F}$ is determined by the number of strictly positive eigenvalues, which are infinite in the Gaussian kernel case.

Therefore, we can obtain the following equation which shows the inner product of two vectors in $\mathbb{F}$ as

$$\varphi(\mathbf{u})^T\varphi(\mathbf{u}') = \kappa(\mathbf{u}, \mathbf{u}'). \tag{2.18}$$

By defining $\varphi(\mathbf{u}) = \kappa(\mathbf{u}, .)$ where $\varphi(\mathbf{u})$ and $\kappa(\mathbf{u}, .)$ are the basis of the feature spaces $\mathbb{F}$ and $\mathbb{H}$ respectively, the two spaces $\mathbb{F}$ and $\mathbb{H}$ will be same space.

Example 2.6.1. Let $\kappa(., .)$ be a polynomial kernel defined as

$$\kappa(\mathbf{u}, \mathbf{c}) = (1 + \mathbf{u}^T\mathbf{c})^2 \tag{2.19}$$

where $\mathbf{u} = [u_1, u_2]^T$ and $\mathbf{c} = [c_1, c_2]^T$. By expressing the polynomial kernel in terms of monomials of various orders, we have

$$\kappa(\mathbf{u}, \mathbf{c}) = 1 + u_1^2c_1^2 + 2u_1u_2c_1c_2 + u_2^2c_2^2 + 2u_1c_1 + 2u_2c_2 \tag{2.20}$$

Therefore, the image of the input vector $\mathbf{u}$ in the feature space may be written as

$$\varphi(\mathbf{u}) = [1, u_1^2, \sqrt{2}u_1u_2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2] \tag{2.21}$$

and similarly we have

$$\varphi(\mathbf{c}) = [1, c_1^2, \sqrt{2}c_1c_2, c_2^2, \sqrt{2}c_1, \sqrt{2}c_2]. \tag{2.22}$$

Based on $\varphi(\mathbf{u})$ and $\varphi(\mathbf{c})$ definitions, it is easy to verify that kernel function is indeed the inner product of two vectors in feature space as

$$\varphi(\mathbf{u})^T\varphi(\mathbf{c}) = \kappa(\mathbf{u}, \mathbf{c}). \tag{2.23}$$

2.6.2 Kernel Adaptive Filters

The kernel method is a powerful non-parametric modeling tool for nonlinear systems. The Kernel function or equivalent feature map transform the input data into a high-dimensional feature space via a reproducing kernel such that the inner product operation in the feature space can be computed efficiently through the kernel evaluation. This transformation enables us to transform nonlinear system to higher dimensional space which can apply appropriate linear methods on the transformed data. The algorithm can be formulated in terms of inner products (or equivalent kernel evaluation) without performing inner product in the high-dimensional feature space. This method is called the 'kernel trick'. The underlying reproducing kernel Hilbert space plays a central role in providing linearity, convexity, and universal approximation capability. Some examples of the kernel methodology are support vector machines, kernel principal component analysis, and Fisher discriminant analysis [21].

Now, we will show that projecting the input into a feature space could help in learning.

As an example, consider the function of a two-dimensional input $\mathbf{u} = [u_1, u_2]^T$ [21]

$$f(u_1, u_2) = a_1 u_1 + a_2 u_2 + a_3 u_1^2 + a_4 u_2^2 \tag{2.24}$$

it is clear that the function has second order non linearity. Therefore we cannot find any linear combination of u_1 and u_2 to approximate f. However, by using the kernel and its feature mapping φ, we can transform two dimensional input space to a higher order feature space and find a new representation of the input as [21]

$$\varphi : \mathbb{U} \longrightarrow \mathbb{H}$$

$$\varphi(u_1, u_2) = [x_1, x_2, x_3, x_4, x_5, x_6] = [1, u_1^2, \sqrt{2}u_1 u_2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2]$$

Now, f can be represented by a linear combinations of $(x_1, x_2, x_3, x_4, x_5, x_6)$ as

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = 0x_1 + a_3 x_2 + 0x_3 + a_4 x_4 + \frac{a_4}{\sqrt{2}} x_5 + \frac{a_2}{\sqrt{2}} x_6. \tag{2.25}$$

The fact that mapping the input into a feature space can simplify the learning task has been well known for a long time in machine learning as exemplified by polynomial regression and Volterra series.

Fig.2.1 shows the block model of a kernel adaptive filter for non-linear system identification.

The goal of the kernel adaptive filter can be stated as finding filter $w(n)$ to estimate desired response $d(n)$, where $d(n) = f(\mathbf{u}(n)) + \mathbf{v}(n)$, with $f(n)$ being a nonlinear function, where $\mathbf{u}(n)$ is the system input and $\mathbf{v}(n)$ is additive noise. The application here may be seen to be similar to the system identification task for linear filters.

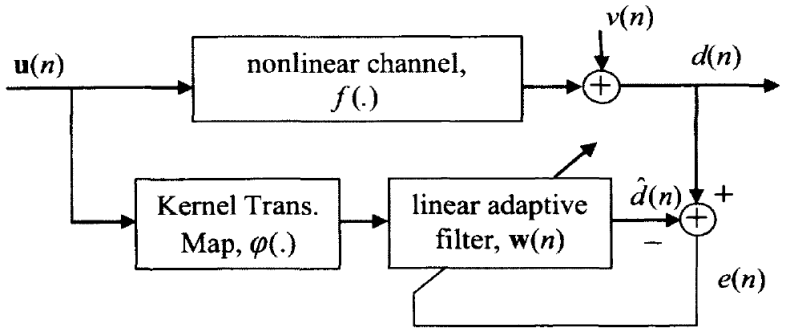


Fig. 2.1: Kernel Adaptive Filter for Non-Linear system Identification

The mapping of the input to an RKHS allows for the learning of a nonlinear channel using the linear filter $\mathbf{w}(n)$. By considering $\mathbf{w}(n)$ as a vector in a reproducing kernel Hilbert space (RKHS), the gradient methods may be used for optimization problem to find the optimal weights. The estimate $\hat{d}(n)$ may be expressed as

$$\hat{d}(n) =< \varphi(\mathbf{u}), \mathbf{w}(n) >= \varphi(\mathbf{u})^T \mathbf{w}(n). \tag{2.26}$$

Many approaches may be used for updating the weights $\mathbf{w}(n)$ for the kernel adaptive filter. An example approach is the Kernel Least Mean Square (KLMS) algorithm.

2.6.3 Kernel Least Mean Square (KLMS) algorithm

For the Kernel Least Mean Square (KLMS) algorithm, the goal is to minimize the cost function $J(n) = (e(n))^2$ respect to free parameter $\mathbf{w}_n$ as

$$\begin{aligned} \min_{\forall \mathbf{w}_n \in \mathcal{H}} \quad & J(n) = (e(n))^2 \\ \text{s.t.} \quad & e(n) = d(n) - y_n \\ & y_n =< \varphi(\mathbf{u}_n), \mathbf{w}_n >= \mathbf{w}_n^T \varphi_n \end{aligned} \tag{2.27}$$

where d is desired signal, $\mathbf{u}_n$ input signal, φ_n is the kernel map to a RKHS.

Minimizing the cost function $J(n)$ can be done with unconstrained optimization algorithm such as gradient descent algorithm as below

$$\begin{aligned} \mathbf{w}_{n+1} =& \mathbf{w}_n - 0.5\eta \nabla_{\mathbf{w}_n} J(n) \\ =& \mathbf{w}_n - \eta \left(\frac{\partial (J(n))}{\partial \mathbf{w}_n} \right)^T = \mathbf{w}_n + \eta e(n) \varphi(\mathbf{u}_n). \end{aligned} \tag{2.28}$$

The weight-update equation through iterations by assuming the initial weights $\mathbf{w}_0 = 0$ yields

$$
\begin{aligned}
\mathbf{w}_{n+1} =& \mathbf{w}_n + \eta e(n)\varphi(\mathbf{u}_n) \\
=& \underbrace{\mathbf{w}_{n-1} + \eta e(n-1)\varphi(\mathbf{u}_{n-1})}_{\mathbf{w}_n} + \eta e(n)\varphi(\mathbf{u}_n) \\
=& \mathbf{w}_{n-1} + \eta \Big[e(n-1)\varphi(\mathbf{u}_{n-1}) + e(n)\varphi(\mathbf{u}_n) \Big] \Big] \\
=& \mathbf{w}_{n-2} + \eta \sum_{i=n-2}^{n} e(i)\varphi(\mathbf{u}_i) \\
=& \mathbf{w}_0 + \eta \sum_{i=0}^{n} e(i)\varphi(\mathbf{u}_i) \\
=& \eta \sum_{i=0}^{n} e(i)\varphi(\mathbf{u}_i).
\end{aligned}
\tag{2.29}
$$

By substituting the weight update in the $y_n = \mathbf{w}_n^T \varphi_n$ and using properties of Reproducing Kernel Hilbert Space (RKHS) and the 'kernel trick,' to replace the inner product of two vectors with kernel κ_σ, we can simplify the equation in kernel form as

$$
\begin{aligned}
y_n &= \left(\eta \sum_{i=0}^{n-1} e(i)\varphi(\mathbf{u}_i) \right)^T \varphi(\mathbf{u}_n) \\
&= \eta \sum_{i=0}^{n-1} e(i)\varphi(\mathbf{u}_i)^T \varphi(\mathbf{u}_n) \\
&= \eta \sum_{i=0}^{n-1} e(i)\kappa_\sigma(\mathbf{u}_i, \mathbf{u}_n).
\end{aligned}
\tag{2.30}
$$

Example 2.6.2. Mackey-Glass Time Series Prediction: In this example we want to show the short term prediction of the Mackey-Glass (MG) chaotic time series using KLMS adaptive filter. The Mackey-Glass (MG) time series is generated from the following time delay differential equation as:

$$\frac{dx(t)}{d(t)} = -bx(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} \tag{2.31}$$

with $b = 0.1$, $a = 0.2$, and $\tau = 30$. The time series is sampled with period of 6 seconds [21].

The input and desired input are corrupted by additive Gaussian noise with zero mean and 0.04 standard deviation. The purpose of the experiment is to compare the performance of a linear combiner trained with LMS and KLMS. The step size parameter for LMS is 0.2. For KLMS, the Gaussian kernel with $a = 1$ is chosen and the step size parameter is also 0.2. Figure 2.2 [21] shows the learning curves of LMS and KLMS algorithms. As expected, KLMS converges to a smaller value of MSE because of its nonlinear nature [21].

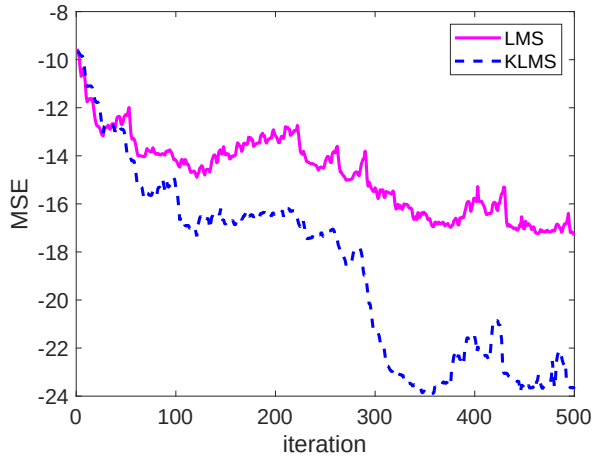


Fig. 2.2: Learning curves of LMS and KLMS in Mackey-Glass time series prediction (Source: [21])

Chapter 3

Quaternions and Properties

Quaternions are a 4-D associative, noncommutative, normed division algebra over the real numbers, defined as in [36]

$$\mathbb{H} = \{q_r + iq_i + jq_j + kq_k \ \ | q_r, q_i, q_j, q_k \in \mathbb{R}\} \tag{3.1}$$

where $\{1, i, j, k\}$ is a basis for $\mathbb{H}$ and has the the following rules

$$ij = k \quad jk = i \quad ki = j \text{ where } \quad i^2 = j^2 = k^2 = -1.$$

For any quaternion $q \in \mathbb{H}$

$$q = S_q + V_q \tag{3.2}$$

the S_q is the real part of q and denoted by $q_r = S_q = \Re(q)$ and V_q is the imaginary part of q and denoted by $iq_i + jq_j + kq_k = V_q = \Im(q)$. The real part of the q is a scalar in real number domain $\mathbb{R}$ and the imaginary part is a 3-D vector. The conjugate of a quaternion q is $q^* = q_r - iq_i - jq_j - kq_k$.

For any two quaternion numbers q and $p \in \mathbb{H}$, the following properties exist

$$(pq)^* = (q^*p^*) \ , \quad |q| = \sqrt{qq^*} \ , \quad |pq| = |p||q|.$$

The inverse of a quaternion $q \neq 0$ is $q^{-1} = q^*/|q|^2$ and for any two quaternion numbers q and $p \in \mathbb{H}$ the inverse of (pq) is equal to $(pq)^{-1} = (q^{-1}p^{-1})$. If $|q| = 1$ then it is called unit quaternion. If $\Re(q) = 0$ then $q^* = -q$ and $q^2 = -|q|^2$. For any quaternion q the quaternion rotation is defined as the transformation

$$q^{\mu} \triangleq \mu q \mu^{-1} \tag{3.3}$$

and geometrically describes a 3-D rotation of the vector part of q by an angle 2θ about the vector part of μ [11]. The quaternion rotation (3.3) becomes quaternion involution such as q^i , q^j , q^k defined by

$$\begin{aligned} q^i &= -iqi = q_r + iq_i - jq_j - kq_k, \\ q^j &= -jqj = q_r - iq_i + jq_j - kq_k, \\ q^k &= -kqk = q_r - iq_i - jq_j + kq_k. \end{aligned} \tag{3.4}$$

whose conjugates q^{i*} , q^{j*} , q^{k*} are defined as

$$\begin{aligned} q^{i*} &= q_r - iq_i + jq_j + kq_k, \\ q^{j*} &= q_r + iq_i - jq_j + kq_k, \\ q^{k*} &= q_r + iq_i + jq_j - kq_k. \end{aligned} \tag{3.5}$$

For any two quaternion numbers p and q the quaternion rotation has the following properties

$$q^{\mu} = q^{\mu/|\mu|}, \;\; pq^{\mu} = p^{\mu}q^{\mu}, \;\; pq = q^{p}p = qp^{q^*}. \tag{3.6}$$

The real representation in (3.1) can be easily generalized to a general orthogonal system $\{1, i^{\mu}, j^{\mu}, k^{\mu}\}$, where the following properties hold

$$i^{\mu}i^{\mu} = j^{\mu}j^{\mu} = k^{\mu}k^{\mu} = -1. \tag{3.7}$$

3.1 Quaternion Involutions and the Augmented Basis Vector

In complex domain, the real and imaginary part of a complex number $z = z_a + iz_b$ can be calculated as $z_a = \frac{1}{2}(z + z^*)$ and $z_b = \frac{1}{2i}(z - z^*)$. The corresponding bivariate signal $(z_a, z_b) \in \mathbb{R}^2$ is used as a basis for the augmented complex statistics, where the augmented basis vector is $z^a = [z, z^*]^T$. In quaternion domain, the relation between the elements of a quadrivariate vector in $\mathbb{R}^4$ and the elements of a quaternion valued variable in $\mathbb{H}$ is not straight forward. To deal with this issue, the four components of the quaternion $q = q_a + iq_b + jq_c + kq_d$ can be used which are expressed based on three

perpendicular quaternion involution as follows [9]:

$$\begin{aligned} q_a &= \frac{1}{2}(q + q^*), \\ q_b &= \frac{1}{2i}(q - q^{i*}), \\ q_c &= \frac{1}{2j}(q - q^{j*}), \\ q_d &= \frac{1}{2k}(q - q^{k*}). \end{aligned} \tag{3.8}$$

The combination of $\{q, q^*, q^{i*}, q^{j*}, q^{k*}\}$ is used to define the augmented quaternion vector $\mathbf{q}^a = [\mathbf{q}^T \mathbf{q}^{iT} \mathbf{q}^{jT} \mathbf{q}^{kT}]^T \in \mathbb{H}^{4N\times 1}$, and the relation with its real vector counterpart $\mathbf{q}^r = [\mathbf{q}^T \mathbf{q}_a^T \mathbf{q}_b^T \mathbf{q}_c^T \mathbf{q}_d^T]^T \in \mathbb{R}^{4N}$ is defined as [9]:

$$\underbrace{\begin{pmatrix} \mathbf{q} \\ \mathbf{q}^i \\ \mathbf{q}^j \\ \mathbf{q}^k \end{pmatrix}}_{\mathbf{q}^a} = \underbrace{\begin{pmatrix} \mathbf{I}_N & i\mathbf{I}_N & j\mathbf{I}_N & k\mathbf{I}_N \\ \mathbf{I}_N & i\mathbf{I}_N & -j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & -j\mathbf{I}_N & k\mathbf{I}_N \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix}}_{\mathbf{q}^r} \tag{3.9}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix, $\mathbf{q} = [q_1, q_2, ...q_N]^T \in \mathbb{H}^{N\times 1}$ and similarly same description for $\mathbf{q}^i, \mathbf{q}^j, \mathbf{q}^k \in \mathbb{H}^{N\times 1}$, and $\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c, \mathbf{q}_d \in \mathbb{R}^{N\times 1}$. The $4N \times 4N$ matrix $\mathbf{A}$ provides an invertible mapping between the augmented quaternion value signal $\mathbf{q}^a$ and the quadrivariate composite real valued vector $\mathbf{q}^r$ as:

$$\mathbf{q}^a = \mathbf{A}\mathbf{q}^r$$
$$\mathbf{q}^r = \mathbf{A}^{-1}\mathbf{q}^a = \frac{1}{4}\mathbf{A}^H\mathbf{q}^a. \tag{3.10}$$

3.2 GHR Calculus

In this section we briefly highlight the HR and GHR calculus properties. More details can be found in [10] [27]. Traditional quaternion pseudoderivatives use component-wise real derivatives of quaternion components. However, traditional method is not suitable to apply on optimization algorithms due to the complexity of the calculation. To overcome this issue, a recent and more elegant approach such as HR calculus and its generalized form GHR calculus are used to derive the gradient and Hessian of cost functions for quaternion optimization algorithms. The HR derivatives are given by

$$\begin{pmatrix} \frac{\partial f}{\partial q_r} \\ \frac{\partial f}{\partial q^i} \\ \frac{\partial f}{\partial q^j} \\ \frac{\partial f}{\partial q^k} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & -i & -j & -k \\ 1 & -i & j & k \\ 1 & i & -j & -k \\ 1 & i & j & -k \end{pmatrix} \begin{pmatrix} \frac{\partial f}{\partial q_r} \\ \frac{\partial f}{\partial q_i} \\ \frac{\partial f}{\partial q_j} \\ \frac{\partial f}{\partial q_k} \end{pmatrix} \tag{3.11}$$

and the conjugate HR derivatives (HR* derivatives)

$$\begin{pmatrix} \frac{\partial f}{\partial q^*} \\ \frac{\partial f}{\partial q^{i*}} \\ \frac{\partial f}{\partial q^{j*}} \\ \frac{\partial f}{\partial q^{k*}} \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & i & j & k \\ 1 & i & -j & -k \\ 1 & -i & j & -k \\ 1 & -i & -j & k \end{pmatrix} \begin{pmatrix} \frac{\partial f}{\partial q_r} \\ \frac{\partial f}{\partial q_i} \\ \frac{\partial f}{\partial q_j} \\ \frac{\partial f}{\partial q_k} \end{pmatrix}. \tag{3.12}$$

The traditional product and chain rules are not valid for the HR calculus and the GHR calculus has solved the issue.

Definition 3.2.1. (GHR Derivatives): Let $f : \mathbb{H} \to \mathbb{H}$. Then, the left GHR derivatives of $f(q)$ with respect to q^μ and $q^{\mu*}$ ($\mu \neq 0$ and $\mu \in \mathbb{H}$) are defined as [11]

$$\frac{\partial f}{\partial q^\mu} = \frac{1}{4}\left(\frac{\partial f}{\partial q_r} - \frac{\partial f}{\partial q_i} i^\mu - \frac{\partial f}{\partial q_j} j^\mu - \frac{\partial f}{\partial q_k} k^\mu\right) \in \mathbb{H} \tag{3.13}$$

$$\frac{\partial f}{\partial q^{\mu*}} = \frac{1}{4}\left(\frac{\partial f}{\partial q_r} + \frac{\partial f}{\partial q_i} i^\mu + \frac{\partial f}{\partial q_j} j^\mu + \frac{\partial f}{\partial q_k} k^\mu\right) \in \mathbb{H} \tag{3.14}$$

while the right GHR derivatives are defined as [11]

$$\frac{\partial_r f}{\partial q^\mu} = \frac{1}{4}\left(\frac{\partial f}{\partial q_r} - i^\mu \frac{\partial f}{\partial q_i} - j^\mu \frac{\partial f}{\partial q_j} - k^\mu \frac{\partial f}{\partial q_k}\right) \in \mathbb{H} \tag{3.15}$$

$$\frac{\partial_r f}{\partial q^{\mu*}} = \frac{1}{4}\left(\frac{\partial f}{\partial q_r} + i^\mu \frac{\partial f}{\partial q_i} + j^\mu \frac{\partial f}{\partial q_j} + k^\mu \frac{\partial f}{\partial q_k}\right) \in \mathbb{H} \tag{3.16}$$

where $\frac{\partial f}{\partial q_r}$, $\frac{\partial f}{\partial q_i}$, $\frac{\partial f}{\partial q_j}$ and $\frac{\partial f}{\partial q_k} \in \mathbb{H}$ are the partial derivatives of f with respect to q_r , q_i, q_j, and q_k , respectively, where $\{1, i^\mu, j^\mu, k^\mu\}$ is orthogonal basis of $\mathbb{H}$.

Some properties of the left GHR derivatives are [11]

$$\text{First Product Rule: } \frac{\partial(fg)}{\partial q^{\mu}} = f\frac{\partial g}{\partial q^{\mu}} + \frac{\partial f}{\partial q^{g\mu}}g, \tag{3.17}$$

$$\text{Second Product Rule: } \frac{\partial(fg)}{\partial q^{\mu *}} = f\frac{\partial g}{\partial q^{\mu *}} + \frac{\partial f}{\partial q^{g\mu *}}g, \tag{3.18}$$

$$\text{First Chain Rule: } \frac{\partial(f(g(q)))}{\partial q^{\mu}} = \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^{v}}\frac{\partial g^{v}}{\partial q^{\mu}}, \tag{3.19}$$

$$\text{Second Chain Rule: } \frac{\partial(f(g(q)))}{\partial q^{\mu *}} = \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^{v*}}\frac{\partial g^{v*}}{\partial q^{\mu *}}, \tag{3.20}$$

$$\text{Rotation rule: } \left(\frac{\partial f}{\partial q^{\mu}}\right)^{v} = \frac{\partial f^{v}}{\partial q^{v\mu}}, \left(\frac{\partial f}{\partial q^{\mu *}}\right)^{v} = \frac{\partial f^{v}}{\partial q^{v\mu *}}, \tag{3.21}$$

$$\text{Conjugate rule: } \left(\frac{\partial f}{\partial q^{\mu}}\right)^{*} = \frac{\partial_{r} f^{*}}{\partial q^{\mu *}}, \left(\frac{\partial f}{\partial q^{\mu *}}\right)^{*} = \frac{\partial_{r} f^{*}}{\partial q^{\mu}}, \tag{3.22}$$

$$\text{if } f \text{ is real then: } \left(\frac{\partial f}{\partial q^{\mu}}\right)^{*} = \frac{\partial f}{\partial q^{\mu *}}, \left(\frac{\partial f}{\partial q^{\mu *}}\right)^{*} = \frac{\partial f}{\partial q^{\mu}}. \tag{3.23}$$

Example 3.2.1. Find the GHR derivatives of the functions

$$f(q) = wqv + \lambda, \qquad g(q) = wq^{*}v$$

Solution: Using product rule and setting $\mu = 1$, we have

$$\begin{aligned}
\frac{\partial f(q)}{\partial q} =& \frac{\partial wqv}{\partial q} = wq\frac{\partial v}{\partial q} + \frac{\partial wq}{\partial q^{v}}v = w\frac{\partial q}{\partial q^{v}}v \\
&= w(\frac{1}{4})\left(\frac{\partial q}{\partial q_r} - \frac{\partial q}{\partial q_i}i^{v} - \frac{\partial q}{\partial q_j}j^{v} - \frac{\partial q}{\partial q_k}k^{v}\right)v \\
&= w(\frac{1}{4})\left(1 - ii^{v} - jj^{v} - kk^{v}\right)v \\
&= w(\frac{1}{4})\left(1 - iviv^{-1} - jvjv^{-1} - kvkv^{-1}\right)v \\
&= w(\frac{1}{4})\left(v - ivi - jvj - kvk\right) \\
&= w(\frac{1}{4})\left(v + v^{i} + v^{j} + v^{k}\right) \\
&= (\frac{1}{4})w\mathfrak{Re}(v).
\end{aligned} \tag{3.24}$$

In a similar way it can be solved that:

$$
\begin{aligned}
\frac{\partial g(q)}{\partial q^*} =& \frac{\partial wqv}{\partial q^*} = wq\frac{\partial v}{\partial q^*} + \frac{\partial wq}{\partial q^{v*}}v = w\frac{\partial q}{\partial q^{v*}}v \\
&= w(\frac{1}{4})\Big(1 + ii^v + jj^v + kk^v\Big)v \\
&= w(\frac{1}{4})\Big(1 + iviv^{-1} + jvjv^{-1} + kvkv^{-1}\Big)v \\
&= w(\frac{1}{4})\Big(v + ivi + jvj + kvk\Big) \\
&= w(\frac{1}{4})\Big(v - v^i - v^j - v^k\Big) \\
&= w(\frac{-1}{4})\Big(- v + v^i + v^j + v^k\Big) \\
&= w(\frac{-1}{4})2v^* \\
&= \frac{-1}{2}wv^*.
\end{aligned}
\tag{3.25}
$$

Definition 3.2.2. (Quaternion Gradient): The two quaternion gradients of a function $f\colon \mathbb{H}^{N\times 1} \to \mathbb{H}$ are defined as [11]

$$
\nabla_{\mathbf{q}} f \triangleq \left(\frac{\partial f}{\partial \mathbf{q}}\right)^T = \left(\frac{\partial f}{\partial q_1}, ..., \frac{\partial f}{\partial q_N}\right)^T \in \mathbb{H}^{N\times 1} \tag{3.26}
$$

$$
\nabla_{\mathbf{q}^*} f \triangleq \left(\frac{\partial f}{\partial \mathbf{q}^*}\right)^T = \left(\frac{\partial f}{\partial q_1^*}, ..., \frac{\partial f}{\partial q_N^*}\right)^T \in \mathbb{H}^{N\times 1} \tag{3.27}
$$

Definition 3.2.3. (Quaternion Jacobian Matrix): The quaternion Jacobian matrices of $\mathbf{f}\colon \mathbb{H}^{N\times 1} \to \mathbb{H}^{M\times 1}$ are defined as [11]

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}} = \begin{pmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_N} \\ & \ddots & \\ \frac{\partial f_M}{\partial q_1} & \cdots & \frac{\partial f_M}{\partial q_N} \end{pmatrix} \tag{3.28}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{q}^*} = \begin{pmatrix} \frac{\partial f_1}{\partial q_1^*} & \cdots & \frac{\partial f_1}{\partial q_N^*} \\ & \ddots & \\ \frac{\partial f_M}{\partial q_1^*} & \cdots & \frac{\partial f_M}{\partial q_N^*} \end{pmatrix} \tag{3.29}$$

3.3 Optimization in Quaternion Field

To show the intuitive link between the real and quaternion vectors, consider a quaternion vector $\mathbf{q} = \mathbf{q}_a + i\mathbf{q}_b + j\mathbf{q}_c + k\mathbf{q}_d \in \mathbb{H}^{N\times 1}$, expressed by its real coordinate vectors $\mathbf{q}_a$, $\mathbf{q}_a$, $\mathbf{q}_a$, and $\mathbf{q}_a \in \mathbb{R}^{N\times 1}$. The augmented quaternion vector $\mathbf{h} \in \mathbb{H}^{4N\times 1}$ and its relationship with dual-quadrivariate real vector $\mathbf{r} \in \mathbb{R}^{4N\times 1}$ is defined as [11]

$$\underbrace{\begin{pmatrix} \mathbf{q} \\ \mathbf{q}^i \\ \mathbf{q}^j \\ \mathbf{q}^k \end{pmatrix}}_{\mathbf{h}} = \underbrace{\begin{pmatrix} \mathbf{I}_N & i\mathbf{I}_N & j\mathbf{I}_N & k\mathbf{I}_N \\ \mathbf{I}_N & i\mathbf{I}_N & -j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & j\mathbf{I}_N & -k\mathbf{I}_N \\ \mathbf{I}_N & -i\mathbf{I}_N & -j\mathbf{I}_N & k\mathbf{I}_N \end{pmatrix}}_{\mathbf{J}} \underbrace{\begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix}}_{\mathbf{r}} \tag{3.30}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix, and $\mathbf{J}$ is the $4N \times 4N$ matrix. Multiplying both sides of (3.30) by $(1/4)\mathbf{J}^H$ and noting that $(1/4)\mathbf{J}^H\mathbf{J} = \mathbf{I}_{4N}$, we have

$$\mathbf{r} = (1/4)\mathbf{J}^H\mathbf{h}. \tag{3.31}$$

Since $\mathbf{r}$ is a real vector, it follows that:

$$\frac{\partial f}{\partial \mathbf{h}} = \frac{\partial f}{\partial \mathbf{r}}\frac{\partial r}{\partial \mathbf{h}} = \frac{1}{4}\frac{\partial f}{\partial \mathbf{h}}\mathbf{J}^H \Leftrightarrow \frac{\partial f}{\partial \mathbf{r}} = \frac{\partial f}{\partial \mathbf{h}}\mathbf{J}, \quad \in \mathbb{R}^{1\times 4N} \tag{3.32}$$

where $\frac{\partial f}{\partial \mathbf{h}} \in \mathbb{H}^{1\times 4N}$ and $\frac{\partial f}{\partial \mathbf{r}} \in \mathbb{R}^{1\times 4N}$. Since f and r are real valued, we have

$$\begin{aligned}
\nabla_{\mathbf{r}} f &\triangleq \left(\frac{\partial f}{\partial \mathbf{r}}\right)^T = \left(\frac{\partial f}{\partial \mathbf{r}}\right)^H \\
&= \left(\frac{\partial f}{\partial \mathbf{h}}\mathbf{J}\right)^H \\
&= \mathbf{J}^H \left(\frac{\partial f}{\partial \mathbf{h}}\right)^H \\
&= \mathbf{J}^H \left(\frac{\partial f}{\partial \mathbf{h}^*}\right)^T \\
&= \mathbf{J}^H \nabla_{\mathbf{h}^*} f.
\end{aligned} \tag{3.33}$$

This shows that the real gradient $\nabla_{\mathbf{r}} f \in \mathbb{R}^{4N\times 1}$ and the augmented quaternion gradient $\nabla_{\mathbf{h}^*} f \in \mathbb{H}^{4N\times 1}$ are related by a simple invertible linear transformation $\mathbf{J}^H$.

Optimization algorithms such Gradient descent or steepest descent algorithms find a local minimum of a function by taking iterative steps proportional to the negative of the gradient of the function. From (3.30), a real scalar function $f(\mathbf{q})\colon \mathbb{H}^{N\times 1} \to \mathbb{R}$ can also be viewed as $f(\mathbf{r})\colon \mathbb{R}^{4N\times 1} \to \mathbb{R}$ for which the quadrivariate real gradient descent update rule is given by [11]

$$\Delta r = -\alpha \nabla_{\mathbf{r}} f, \mathbf{r} \quad \in \mathbb{R}^{4N \times 1} \tag{3.34}$$

where Δr denotes a small increment in r and $\alpha \in \mathbb{R}^{+}$ is the step size. Therefore

$$\Delta h = \mathbf{J}\Delta r = -\alpha \mathbf{J} \nabla_{\mathbf{r}} f = -\alpha \mathbf{J}\mathbf{J}^{H} \nabla_{\mathbf{h}^*} f = -4\alpha \nabla_{\mathbf{h}^*} f. \tag{3.35}$$

Thus, this gives the quaternion gradient descent update rule in the form

$$\Delta q = -4\alpha \nabla_{\mathbf{q}^*} f = -4\alpha \left(\frac{\partial f}{\partial \mathbf{q}^*} \right)^{T} = -4\alpha \left(\frac{\partial f}{\partial \mathbf{q}} \right)^{H}, \quad f \in \mathbb{R}. \tag{3.36}$$

3.4 Augmented Quaternion Statistics

The standard covariance matrix $C_{\mathbf{qq}}$ of a quaternion random vector $\mathbf{q} = [q_1, ..., q_N]^T$ can be calculated by $C_{\mathbf{qq}} = E[\mathbf{q}\mathbf{q}^H]$ and the real and imaginary parts of it is shown in table 3.1 [9]. As shown in Table 3.1 the real and imaginary parts of $C_{\mathbf{qq}}$ are linear functions of the real-valued covariance and cross-covariance matrices of the component vectors $\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c$ and $\mathbf{q}_d \in \mathbb{R}^{N \times 1}$. The complementary covariance matrices, the i-covariance $C_{\mathbf{q}^i}$, the j-covariance $C_{\mathbf{q}^j}$ and the k-covariance $C_{\mathbf{q}^k}$ will be used to augment the information within the covariance and are defined as [9]:

$$
\begin{aligned}
C_{\mathbf{q}^i} &= E[\mathbf{q}\mathbf{q}^{iH}] \\
C_{\mathbf{q}^j} &= E[\mathbf{q}\mathbf{q}^{jH}] \\
C_{\mathbf{q}^k} &= E[\mathbf{q}\mathbf{q}^{kH}]
\end{aligned}
\tag{3.37}
$$

where $\mathbf{q}^{iH} = [q_1^{i*}, ..., q_N^{i*}]^T$, $\mathbf{q}^{jH} = [q_1^{j*}, ..., q_N^{j*}]^T$ and $\mathbf{q}^{kH} = [q_1^{k*}, ..., q_N^{k*}]^T$. The details about the real and imaginary parts of the complementary covariance matrices are shown in table 3.1 and table 3.2. The complementary covariance matrices, the i-covariance $C_{\mathbf{q}^i}$, the j-covariance $C_{\mathbf{q}^j}$ and the k-covariance $C_{\mathbf{q}^k}$ are i-Hermitian, j-Hermitian and k-Hermitian respectively which is:

$$
\begin{aligned}
C_{\mathbf{q}^i} &= C_{\mathbf{q}^i}^{iH} \\
C_{\mathbf{q}^j} &= C_{\mathbf{q}^j}^{jH} \\
C_{\mathbf{q}^k} &= C_{\mathbf{q}^k}^{kH}
\end{aligned}
\tag{3.38}
$$

Table 3.1: Covariance matrix (part 1)

Covariance matrix	$C_{\mathbf{qq}} = E[\mathbf{q}\mathbf{q}^H]$	$C_{\mathbf{q}^i} = E[\mathbf{q}\mathbf{q}^{iH}]$
$\Re(.)$	$C_{\mathbf{q}_a} + C_{\mathbf{q}_b} + C_{\mathbf{q}_c} + C_{\mathbf{q}_d}$	$C_{\mathbf{q}_a} + C_{\mathbf{q}_b} - C_{\mathbf{q}_c} - C_{\mathbf{q}_d}$
$\Im_i(.)$	$C_{\mathbf{q}_b\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_b} + C_{\mathbf{q}_d\mathbf{q}_c} - C_{\mathbf{q}_c\mathbf{q}_d}$	$C_{\mathbf{q}_b\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_b} + C_{\mathbf{q}_c\mathbf{q}_d} - C_{\mathbf{q}_d\mathbf{q}_c}$
$\Im_j(.)$	$C_{\mathbf{q}_c\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_c} + C_{\mathbf{q}_b\mathbf{q}_d} - C_{\mathbf{q}_d\mathbf{q}_b}$	$C_{\mathbf{q}_a\mathbf{q}_c} + C_{\mathbf{q}_c\mathbf{q}_a} - C_{\mathbf{q}_d\mathbf{q}_b} - C_{\mathbf{q}_b\mathbf{q}_d}$
$\Im_k(.)$	$C_{\mathbf{q}_d\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_d} + C_{\mathbf{q}_c\mathbf{q}_b} - C_{\mathbf{q}_b\mathbf{q}_c}$	$C_{\mathbf{q}_d\mathbf{q}_a} + C_{\mathbf{q}_a\mathbf{q}_d} + C_{\mathbf{q}_b\mathbf{q}_d} + C_{\mathbf{q}_d\mathbf{q}_b}$

Following on these results, the quadrivariate real-valued correlation matrices of each single component $\mathbf{q}_a$, $\mathbf{q}_b$, $\mathbf{q}_c$ and $\mathbf{q}_d$ of the quaternion random vector $\mathbf{q}$ can be expressed in terms of the quaternion-valued covariance and the complementary covariance matrices as follows [9]:

Table 3.2: Covariance matrix (part 2)

Covariance matrix	$C_{\mathbf{qq}} = E[\mathbf{q}\mathbf{q}^{jH}]$	$C_{\mathbf{q}^i} = E[\mathbf{q}\mathbf{q}^{kH}]$
$\Re(.)$	$C_{\mathbf{q}_a} - C_{\mathbf{q}_b} + C_{\mathbf{q}_c} - C_{\mathbf{q}_d}$	$C_{\mathbf{q}_a} - C_{\mathbf{q}_b} - C_{\mathbf{q}_c} + C_{\mathbf{q}_d}$
$\Im_i(.)$	$C_{\mathbf{q}_b\mathbf{q}_a} + C_{\mathbf{q}_a\mathbf{q}_b} + C_{\mathbf{q}_d\mathbf{q}_c} + C_{\mathbf{q}_c\mathbf{q}_d}$	$C_{\mathbf{q}_b\mathbf{q}_a} + C_{\mathbf{q}_a\mathbf{q}_b} - C_{\mathbf{q}_c\mathbf{q}_d} - C_{\mathbf{q}_d\mathbf{q}_c}$
$\Im_j(.)$	$C_{\mathbf{q}_c\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_c} + C_{\mathbf{q}_d\mathbf{q}_b} - C_{\mathbf{q}_b\mathbf{q}_d}$	$C_{\mathbf{q}_a\mathbf{q}_c} + C_{\mathbf{q}_c\mathbf{q}_a} + C_{\mathbf{q}_d\mathbf{q}_b} + C_{\mathbf{q}_b\mathbf{q}_d}$
$\Im_k(.)$	$C_{\mathbf{q}_d\mathbf{q}_a} + C_{\mathbf{q}_a\mathbf{q}_d} - C_{\mathbf{q}_b\mathbf{q}_c} - C_{\mathbf{q}_c\mathbf{q}_b}$	$C_{\mathbf{q}_d\mathbf{q}_a} - C_{\mathbf{q}_a\mathbf{q}_d} + C_{\mathbf{q}_b\mathbf{q}_c} - C_{\mathbf{q}_c\mathbf{q}_b}$

$$
\begin{aligned}
C_{\mathbf{q}_a} &= \frac{1}{4}\Re\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} + C_{\mathbf{q}^j} + C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_b} &= \frac{1}{4}\Re\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} - C_{\mathbf{q}^j} - C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_c} &= \frac{1}{4}\Re\{C_{\mathbf{qq}} - C_{\mathbf{q}^i} + C_{\mathbf{q}^j} - C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_d} &= \frac{1}{4}\Re\{C_{\mathbf{qq}} - C_{\mathbf{q}^i} - C_{\mathbf{q}^j} + C_{\mathbf{q}^k}\}
\end{aligned}
$$

$$
\begin{aligned}
C_{\mathbf{q}_b\mathbf{q}_a} &= \frac{1}{4}\Im_i\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} + C_{\mathbf{q}^j} + C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_c\mathbf{q}_a} &= \frac{1}{4}\Im_j\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} + C_{\mathbf{q}^j} + C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_d\mathbf{q}_a} &= \frac{1}{4}\Im_k\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} + C_{\mathbf{q}^j} + C_{\mathbf{q}^k}\} \\
C_{\mathbf{q}_c\mathbf{q}_a} &= \frac{1}{4}\Im_k\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} - C_{\mathbf{q}^j} - C_{\mathbf{q}^k}\}
\end{aligned}
\tag{3.39}
$$

$$
C_{\mathbf{q}_d\mathbf{q}_b} = \frac{-1}{4}\Im_k\{C_{\mathbf{qq}} + C_{\mathbf{q}^i} - C_{\mathbf{q}^j} - C_{\mathbf{q}^k}\}
$$

Therefore, the augmented quaternion-valued covariance matrix of an augmented random

vector $\mathbf{q}^a = [\mathbf{q}^T \mathbf{q}^{iT} \mathbf{q}^{jT} \mathbf{q}^{kT}]^T \in \mathbb{H}^{4N\times 1}$ can be calculated as follows:

$$C_{\mathbf{q}^a} = E\{\mathbf{q}^a \mathbf{q}^{aH}\} = \begin{pmatrix} C_{\mathbf{qq}} & C_{\mathbf{q}^i} & C_{\mathbf{q}^j} & C_{\mathbf{q}^k} \\ C^H_{\mathbf{q}^i} & C_{\mathbf{q}^i\mathbf{q}^i} & C_{\mathbf{q}^i\mathbf{q}^j} & C_{\mathbf{q}^i\mathbf{q}^k} \\ C^H_{\mathbf{q}^j} & C_{\mathbf{q}^j\mathbf{q}^i} & C_{\mathbf{q}^j\mathbf{q}^j} & C_{\mathbf{q}^j\mathbf{q}^k} \\ C^H_{\mathbf{q}^k} & C_{\mathbf{q}^k\mathbf{q}^i} & C_{\mathbf{q}^k\mathbf{q}^j} & C_{\mathbf{q}^k\mathbf{q}^k} \end{pmatrix} \tag{3.40}$$

The corresponding real valued quadrivariate covariance matrix C_R can be defined as [9]

$$C_R = E\{\mathbf{q}^r \mathbf{q}^{rT}\} = \begin{pmatrix} C_{\mathbf{q}_a} & C_{\mathbf{q}_a\mathbf{q}_b} & C_{\mathbf{q}_a\mathbf{q}_c} & C_{\mathbf{q}_a\mathbf{q}_d} \\ C_{\mathbf{q}_b\mathbf{q}_a} & C_{\mathbf{q}_b} & C_{\mathbf{q}_b\mathbf{q}_c} & C_{\mathbf{q}_b\mathbf{q}_d} \\ C_{\mathbf{q}_c\mathbf{q}_a} & C_{\mathbf{q}_c\mathbf{q}_b} & C_{\mathbf{q}_c} & C_{\mathbf{q}_c\mathbf{q}_d} \\ C_{\mathbf{q}_d\mathbf{q}_a} & C_{\mathbf{q}_d\mathbf{q}_b} & C_{\mathbf{q}_d\mathbf{q}_c} & C_{\mathbf{q}_d} \end{pmatrix} \tag{3.41}$$

Based on the relationship between the augmented quaternion-valued vector $\mathbf{q}^a$ and the corresponding real valued composite vector $\mathbf{q}^r$, $\mathbf{q}^r = \mathbf{A}^{-1}\mathbf{q}^a = \frac{1}{4}\mathbf{A}^H\mathbf{q}^a$, the real valued quadrivariate covariance matrix can be expressed in terms of the augmented quaternion valued covariance matrix as:

$$\begin{aligned} C_R &= E\{\mathbf{q}^r \mathbf{q}^{rT}\} \\ &= E\{\frac{1}{4}\mathbf{A}^H \mathbf{q}^a \frac{1}{4}\mathbf{q}^{aH}\mathbf{A}\} \\ &= \frac{1}{16}\mathbf{A}^H E\{\mathbf{q}^a \mathbf{q}^{aH}\}\mathbf{A} \\ &= \frac{1}{16}\mathbf{A}^H C_{\mathbf{q}^a}\mathbf{A} \end{aligned} \tag{3.42}$$

3.5 Quaternion Vector Spaces

3.5.1 The Quaternion Division Ring

The quaternion set $\mathbb{H}$ is a four-dimensional vector space over the real field $\mathbb{R}$ spanned by the linearly independent basis $\{1, i, j, k\}$ [12]. Therefore, any element $q \in \mathbb{H}$ can be written as a linear combination of basis as $q = a + ib + jc + kd$, where $a, b, c, d \in \mathbb{R}$. For any two quaternions q_1 and q_2, the sum and the scalar multiplication are defined in $\mathbb{R}^4$ as:

$$\begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} + \begin{pmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ a_1b_2 - b_1a_2 - c_1d_2 - d_1c_2 \\ a_1c_2 - b_1d_2 - c_1a_2 - d_1b_2 \\ a_1d_2 - b_1c_2 - c_1b_2 - d_1a_2. \end{pmatrix} \tag{3.43}$$

and for any scalar $\alpha \in \mathbb{R}$, the scalar multiplication are defined as

$$\alpha q = \alpha \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} \alpha a \\ \alpha b \\ \alpha c \\ \alpha d \end{pmatrix} \tag{3.44}$$

Remark 1. The pair $(\mathbb{H}, +)$ is an Abelian group [12], for which the addition operation is defined in (3.43) and the additive identity is $\mathbf{0} = (0, 0, 0, 0) \in \mathbb{H}$.

The quaternion multiplication or Hamilton product is a bilinear mapping

$\mathbb{H} \times \mathbb{H} \to \mathbb{H}, (p, q) \to pq$ defined by

$$pq = \begin{pmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} \begin{pmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} a_1 + a_2 \\ b_1 + b_2 \\ c_1 + c_2 \\ d_1 + d_2 \end{pmatrix} \tag{3.45}$$

Remark 2. The quaternion product defined in (3.45) distributes over the sum.

$$\begin{aligned} p(q + r) &= pq + pr \\ (p + q)r &= pr + qr \end{aligned} \tag{3.46}$$

Remark 3. The pair $(\mathbb{H}, .)$ equipped with the identity element is a monoid under multiplication, while the inclusion of the multiplicative inverse makes $(\mathbb{H}\backslash\{0\}, .)$ a group [12].

Remark 4. Since$(\mathbb{H}, +)$ is an Abelian group (Remark 1), $(\mathbb{H}, .)$ is a group (Remark 3), and the quaternion product distributes over the sum (Remark 2), the triplet $(\mathbb{H}, +, .)$ is a non-commutative division ring [12].

3.5.2 Quaternion-Valued Hilbert Spaces

In order to construct general vector space over $\mathbb{H}$, it requires division field. As shown in previous section, the $(\mathbb{H}, +, .)$ lacks the commutativity property and it is a division ring only. However, it is possible to construct a left-module. The left-module $\mathcal{H}$ over $\mathbb{H}$ as vector space [12] in which the non-commutative scalar multiplication $\mathbb{H} \times \mathcal{H} \to \mathcal{H}$ is defined on the left-hand side by $(q, \mathbf{x}) \to q\mathbf{x}$.

Definition 3.5.1. (Quaternion Left Hilbert Space): A nonempty set $\mathcal{H}$ is called a

quaternion left Hilbert space if it is a quaternion left module, and there exists a quaternion-valued function $<.,.>: \mathbb{H} \times \mathcal{H} \rightarrow \mathcal{H}$ with the following properties:

1. Conjugate symmetry: $< \mathbf{x}, \mathbf{y} >=< \mathbf{y}, \mathbf{x} >^*$
2. Linearity: $< p\mathbf{x} + q\mathbf{y}, \mathbf{z} >= p < \mathbf{x}, \mathbf{z} > +q < \mathbf{y}, \mathbf{z} >$
3. Conjugate linearity: $< \mathbf{x}, p\mathbf{y} + q\mathbf{z} >=< \mathbf{x}, \mathbf{y} > p^* + < \mathbf{y}, \mathbf{z} > q^*$
4. if $< \mathbf{x}, \mathbf{x} >\geqslant 0$ and $< \mathbf{x}, \mathbf{x} >= 0$ then $\mathbf{x} = \mathbf{0}$
5. Completeness: If $\{\mathbf{x}_n\}_{n=1}^{\infty} \in \mathcal{H}$ is a Cauchy sequence, then $\mathbf{x} = \lim_{n\rightarrow\infty} \mathbf{x}_n \in \mathcal{H}$

3.6 Quaternion Reproducing Kernel Hilbert Spaces

Definition 3.6.1. (Quaternion Reproducing Kernel Hilbert Space): [12] Let X be an arbitrary set and $\mathcal{H}$ a left quaternion Hilbert space of functions from X to $\mathcal{H}$. We say that $\mathcal{H}$ is a quaternion reproducing kernel Hilbert space (QRKHS) if the linear evaluation map

$$L_{\mathbf{x}} : \mathcal{H} \longrightarrow \mathbb{H}$$
$$f \longmapsto f(\mathbf{x})$$

is bounded for $\forall \mathbf{x} \in X$.

3.6.1 Riesz Representation Theorem

Theorem 2. (Quaternion Riesz Representation Theorem): [12] For every bounded linear function L defined over a quaternion left Hilbert space $\mathbb{H}$, there exists a unique element $g \in \mathcal{H}$ such that $L(f) =< f, g >$, $\forall f \in \mathcal{H}$.

Corollary 2.1. (Reproducing Property): [12] For $\forall f \in \mathcal{H}$, there exists a unique element $K_{\mathbf{x}} \in \mathcal{H}$ such that the evaluation map $L_{\mathbf{x}} = f(\mathbf{x})$ can be expressed as $L_{\mathbf{x}} =< f, K_{\mathbf{x}} >$.

Since $K_{\mathbf{x}}(.) \in \mathcal{H}$, it can be evaluated for any $\mathbf{y} \in X$. This allows us to define

$$K : X \times X \longrightarrow \mathbb{H}$$
$$(\mathbf{x}, \mathbf{y}) \longmapsto K(\mathbf{x}, \mathbf{y}) = K_{\mathbf{x}}(\mathbf{y})$$

whereby the function K is referred to as the reproducing kernel of the QRKHS $\mathcal{H}$. Its existence and uniqueness properties are a direct consequence of the quaternion Riesz representation theorem.

The following relationships are readily obtained by applying the reproducing property on the functions $K_{\mathbf{x}} = K(\mathbf{x}, .) \in \mathcal{H}$ and $K_{\mathbf{y}} = K(\mathbf{y}, .) \in \mathcal{H}$:

1. $K(\mathbf{x}, \mathbf{y}) =< K_{\mathbf{x}}, K_{\mathbf{y}} >=< K_{\mathbf{y}}, K_{\mathbf{x}} >^* = K^*(\mathbf{y}, \mathbf{x})$
2. $K(\mathbf{x}, \mathbf{x}) = \|K_{\mathbf{x}}\| \geqslant 0$
3. $K_{\mathbf{x}} = 0 \Longleftrightarrow f(\mathbf{x}) =< f, K_{\mathbf{x}} >= 0, \forall f \in \mathcal{H}$

3.6.2 Moore-Aronszajn Theorem

Definition 3.6.2. (Positive Definiteness - Integral Form): [12] A Hermitian kernel $K(\mathbf{x}, \mathbf{y}) = K^*(\mathbf{y}, \mathbf{x})$ is positive definite on the set X iff for any integrable function $\theta : X \times X \longrightarrow \mathbb{H}, \theta \neq 0$ it obeys

$$\int_X \int_X \theta^*(\mathbf{x}) K(\mathbf{x}, \mathbf{y}) \theta(\mathbf{y}) d\mathbf{x} d\mathbf{y} > 0.$$

Definition 3.6.3. (Positive Definiteness - Matrix Form): [12] A Hermitian kernel $K(\mathbf{x}, \mathbf{y}) = K^*(\mathbf{y}, \mathbf{x})$ is positive definite on the set X iff the kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is positive definite for any choice of the set $S_{\mathbf{x}} = \{\mathbf{x}_1, ..., \mathbf{x}_m\} \subset X, m \in \mathbb{N}$.

Theorem 3. (Quaternion Moore-Aronszajn Theorem):[12] For any positive definite quaternion-valued kernel K defined over a set X, there exists a unique (up to an isomorphism) left quaternion Hilbert space of functions $\mathcal{H}$ for which $\mathbf{K}$ is a reproducing kernel.

3.6.3 Quaternion-Valued Gaussian Kernel

The Gaussian kernel can be extended to quarenion domain by using the quaternion norm in its argument. The real-valued Gaussian kernel K in quaternion domain can be defined as [12]

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(- A((\mathbf{x} - \mathbf{y})^H (\mathbf{x} - \mathbf{y})\right)$$

where $A > 0$ is the kernel parameter. Also, we can show that, based on Definition 3.6.3, the K is positive definite in the quaternion domain.

First, for any arbitrary non-zero vector $\mathbf{x} \in \mathbb{H}^n$, the quadratic form $\mathbf{x}^H \mathbf{K} \mathbf{x}$ is real.

$$2\Im\{\mathbf{x}^H \mathbf{K} \mathbf{x}\} = \{\mathbf{x}^H \mathbf{K} \mathbf{x}\} - \{\mathbf{x}^H \mathbf{K} \mathbf{x}\}^H = 0.$$

By expanding the vector $\mathbf{x} = \mathbf{x}_r + i\mathbf{x}_i + j\mathbf{x}_j + k\mathbf{x}_k$ within $\Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\}$ using its real and imaginary parts, we can write

$$\Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} = \{\mathbf{x}_r^T\mathbf{K}\mathbf{x}_r\} + \{\mathbf{x}_i^T\mathbf{K}\mathbf{x}_i\} + \{\mathbf{x}_j^T\mathbf{K}\mathbf{x}_j\} + \{\mathbf{x}_k^T\mathbf{K}\mathbf{x}_k\}.$$

Since $\mathbf{K}$ is positive definite in the real domain, the arbitrary components $\mathbf{x}_r, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$ are real-valued, and the quadratic form $\{\mathbf{x}^H\mathbf{K}\mathbf{x}\}$ is positive. Therefore, we have $\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} = \Re\{\mathbf{x}^H\mathbf{K}\mathbf{x}\} > 0$, which shows that the real Gaussian kernel $K \in \mathbb{H}$ is positive definite.

Part II

New Algorithms

Chapter 4

Quaternion Kernel Minimum Error Entropy Adaptive Filter

In this chapter, we describe and develop quaternion kernel adaptive filter based on information theoretic learning (ITL) cost function referred to quaternion kernel minimum error entropy (QKMEE) algorithm. We use the generalized Hamilton-real calculus (GHR) to derive the gradient of the optimization problem in quaternion domain [10]. The GHR calculus simplified product and chain rules and allows us to calculate the quaternion based gradient and Hessian of cost function efficiently, and use them for the learning algorithms [11].

The quaternion reproducing kernel Hilbert spaces and its uniqueness is established in [12]. These provide a mathematical foundation to develop the quaternion value kernel learning algorithms. The reproducing property of the feature space replace the inner product of feature samples with kernel evaluation. The existence and uniqueness of quaternion reproducing kernel Hilbert space (QRKHS) provide a theoretical basis for kernel algorithms operating in quaternion feature spaces.

The new algorithm minimizes the Renyi's entropy of the errors of the adaptive filter in quaternion domain.

Definition 4.0.1. (Renyi's Entropy for Quaternion data): Renyi's entropy [33] such as

the *order–α Renyi's entropy* for quaternion data can be defined as

$$H_\alpha(e) = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} p_e^\alpha(\varepsilon) d\varepsilon \tag{4.1}$$

where $\alpha \in \mathbb{R}^+ \backslash \{1\}$ and p_e is probability distribution function of quaternion random variable e.

We can define *order–α information potential* V_α as

$$V_\alpha(e) = \int_{-\infty}^{\infty} p_e^\alpha(\varepsilon) d\varepsilon. \tag{4.2}$$

In practice the entropy function is not accessible since it is a function of the PDF of relative random variable e. The entropy can be estimated by using some specific method such as the *Parzen window* which is a good estimation of the *order–2 Renyi's entropy* function.

Definition 4.0.2. (Parzen Window for Quaternion data): For a set of N statistically independent random samples $\{e_i\}_{i=1}^N$ of quaternion random variable e, the *Parzen window* computes the estimate of the probability distribution function p_e as

$$\hat{p_e}(\varepsilon) = \frac{1}{N} \sum_{l=1}^{N} \kappa_{\sigma/\sqrt{2}}(\varepsilon - e_l) \tag{4.3}$$

where κ_σ is Gaussian-based kernel for quaternion data defined as

$$\kappa_\sigma(X-Y) = \frac{4}{\sqrt{2\pi}\sigma}\exp\{\frac{-1}{2\sigma^2}(X_r-Y_r)^2 + (X_i-Y_i)^2 + (X_j-Y_j)^2 + (X_k-Y_k)^2\} = \frac{4}{\sqrt{2\pi}\sigma}\exp\{\frac{-1}{2\sigma^2}|X-Y|^2\}$$

and X and Y are quaternion numbers $\in \mathbb{H}$ in forms of
$X = X_r + iX_i - jX_j - kX_k$ and $Y = Y_r + iY_i - jY_j - kY_k$.
More details of the quaternion kernel is provided in [25].

The information potential $V(e)$ can be estimated using Parzen window as

$$\hat{V}(e) = \frac{1}{N^2}\sum_{l_1=1}^{N}\sum_{l_2=1}^{N}\kappa_\sigma(e_{l_1} - e_{l_2}). \tag{4.4}$$

The global solution of maximization of the $V(e)$ is the same as global solution of $\hat{V}(e)$ with the Parzen window estimation and the global solution is achieved when all related errors are constant and the maximum value of $V(e)$ is shown by $V(0)$ or equally $\hat{V}(0) = \hat{V}(0) = \frac{4}{\sqrt{2\pi}\sigma}$.

Minimizing the error entropy can be done by maximizing the error information potential cost function $J_n(e)$ in quaternion domain $\mathbb{H}$ which can be defined as

$$J_n(e) = \frac{1}{N^2}\sum_{i,j=1}^{N}\kappa_\sigma(e(n-i) - e(n-j)). \tag{4.5}$$

Based on (4.5) we develop *The Quaternion Kernel Minimum Error Entropy Algorithm*

in next section.

4.1 Quaternion Kernel Minimum Error Entropy Algorithm Derivation

For the quaternion kernel adaptive filter based on minimum entropy (QKMEE) with quaternion data, the goal is to maximize the cost function $J_n(e)$ (4.5) with respect to free parameter $\mathbf{w}_n$ as

$$\begin{aligned} \max_{\forall \mathbf{w}_n \in \mathcal{H}} \quad & J_n(e) \\ \text{s.t.} \quad & e(n) = d - y_n \\ & y_n =< \Phi(\mathbf{u}_n), \mathbf{w}_n >= \mathbf{w}_n^H \varphi_n \end{aligned} \tag{4.6}$$

where d is desired signal, $\mathbf{u}_n$ input vector and φ_n= $\Phi(\mathbf{u}_n)$ which $\Phi(.)$ is the kernel map to a quaternion RKHS [25] defined as

$$\begin{aligned} \Phi(\mathbf{u}) =& \Phi(\mathbf{u}_r + i\mathbf{u}_i + j\mathbf{u}_j + k\mathbf{u}_k) \\ &= \phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{q}_k^T]^T) + i.\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) \\ &+ j.\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) + k.\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) \end{aligned}$$

where ϕ is the feature map of real kernel κ defined as

$$\phi(u_r, u_i, u_j, u_k) = \kappa(., (u_r, u_i, u_j, u_k)).$$

Maximizing the information potential cost function $J_n(e)$ (4.5) can be done with uncon-

strained optimization algorithm such as gradient ascent algorithm as

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \eta \nabla_{\mathbf{w}_n^*} J_n(e) \tag{4.7}$$

where η is adaptation step size.

To derive the gradient of cost function, we define functions $f\colon \mathbb{H} \to \mathbb{H}$ and $g_{l,t}\colon \mathbb{H} \to \mathbb{R}$ as

$$f(x) = \exp(x) \tag{4.8}$$

$$g_{l,t}(\mathbf{w}_n) = -\frac{|e(n-l) - e(n-t)|^2}{2\sigma^2} \tag{4.9}$$

where $e(n-l) = d(n-l) - \mathbf{w}_n^H \varphi_{n-l}$ are *a posteriori errors* for all $l : 1 \leqslant l \leqslant N$.
To simplify the notation for function $g_{l,t}$ in our derivative for a given l and t, $1 \leqslant l \leqslant N$ and $1 \leqslant t \leqslant N$, we define $g(\mathbf{w}_n) = g_{l,t}(\mathbf{w}_n) = -\frac{\left|e(n-l)-e(n-t)\right|^2}{2\sigma^2}$.
With the above notation the equation (9) can be written as

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \left(\sum_{l=1}^{N} \sum_{t=1}^{N} \frac{\partial \Big[f(g_{l,t}(\mathbf{w}_n)) \Big]}{\partial \mathbf{w}_n} \right)^H \tag{4.10}$$

where $\mu = \eta \frac{1}{N^2} \frac{4}{\sqrt{2\pi}\sigma}$.
For a given l and t, the partial derivative can be calculated with GHR chain rule as

$$\begin{aligned} \frac{\partial \Big[f(g_{l,t}(\mathbf{w}_n)) \Big]}{\partial \mathbf{w}_n} &= \frac{\partial \Big[f(g(\mathbf{w}_n)) \Big]}{\partial \mathbf{w}_n} \\ &= \sum_{v \in \{1,i,j,k\}} \frac{\partial f}{\partial g^v} \frac{\partial g^v}{\partial \mathbf{w}_n}. \end{aligned} \tag{4.11}$$

Using HR derivative property and quaternion rotation, for $\forall v \in \{i, j, k\}$ we can show that $\frac{\partial f}{\partial g^v} = 0$. Suppose $v = i$ then

$$\begin{aligned}
\frac{\partial f}{\partial g^i} &= \frac{1}{4}\left(\frac{\partial f}{\partial g_r} - i\frac{\partial f}{\partial g_i} + j\frac{\partial f}{\partial g_j} + k\frac{\partial f}{\partial g_k}\right) \\
&= \frac{1}{4}\left(\frac{\partial \exp(g)}{\partial g_r} - i\frac{\partial \exp(g)}{\partial g_i} + j\frac{\partial \exp(g)}{\partial g_j} + k\frac{\partial \exp(g)}{\partial g_k}\right) \\
&= \frac{1}{4}\Big(\exp(g) - ii\exp(g) + jj\exp(g) + kk\exp(g)\Big) \\
&= \frac{1}{4}\Big(\exp(g) + \exp(g) - \exp(g) - \exp(g)\Big) = 0
\end{aligned} \tag{4.12}$$

and if $v = 1$ then

$$\begin{aligned}
\frac{\partial f}{\partial g} &= \frac{1}{4}\left(\frac{\partial f}{\partial g_r} - i\frac{\partial f}{\partial g_i} - j\frac{\partial f}{\partial g_j} - k\frac{\partial f}{\partial g_k}\right) \\
&= \exp(g).
\end{aligned} \tag{4.13}$$

By substituting (4.12) and (4.13) in (4.11) we can simplify (4.11) as follow

$$\sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n} = \exp(g)\frac{\partial g}{\partial \mathbf{w}_n}. \tag{4.14}$$

To calculate the derivative of function g , we can expand it as follows

$$\begin{aligned} g(\mathbf{w}_n) &= -\frac{|e(n-l)-e(n-t)|^2}{2\sigma^2} \\ &= \frac{-1}{2\sigma^2}|e(n-l)-e(n-t)|^2 \\ &= \frac{-1}{2\sigma^2}\Big[(e(n-l)-e(n-t))^*(e(n-l)-e(n-t))\Big] \\ &= \frac{-1}{2\sigma^2}\Big[(e^*(n-l)-e^*(n-t))(e(n-l)-e(n-t))\Big] \\ &= \frac{-1}{2\sigma^2}\Big[|e(n-l)|^2-|e(n-t)|^2-e(n-l)e^*(n-t) \\ &- e^*(n-l)e(n-t)\Big]. \end{aligned} \tag{4.15}$$

Therefore by substituting (4.15) in (4.14) we can find partial derivative of g using GHR calculus as below

$$\begin{aligned} \frac{\partial g}{\partial \mathbf{w}_n} &= \left(\frac{-1}{2\sigma^2}\right)\Big[\frac{\partial|e(n-l)|^2}{\partial \mathbf{w}_n}+\frac{\partial|e(n-t)|^2}{\partial \mathbf{w}_n} \\ &- \frac{\partial e(n-t)e^*(n-l)}{\partial \mathbf{w}_n}-\frac{\partial e(n-l)e^*(n-t)}{\partial \mathbf{w}_n}\Big]. \end{aligned} \tag{4.16}$$

By substituting $e(n-l)=d(n-l)-\mathbf{w}_n^H\varphi_{n-l}$ in (4.16) and using GHR calculus, we can compute each partial derivative of (4.16) as

$$\begin{aligned} \frac{\partial|e(n-l)|^2}{\partial \mathbf{w}_n} &= \frac{\partial e(n-l)e^*(n-l)}{\partial \mathbf{w}_n} \\ &= e(n-l)\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n}+\frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}}e^*(n-l) \end{aligned} \tag{4.17}$$

where

$$\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H\frac{\partial \mathbf{w}_n}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H \tag{4.18}$$

and the second term of (4.17) can be calculated as

$$\begin{aligned}
&\frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) = -\frac{\partial \mathbf{w}_n^H \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) \\
&= -\mathbf{w}_n^H \frac{\partial \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) - \frac{\partial \mathbf{w}_n^H}{\partial \mathbf{w}_n^{\varphi_{n-l} e^*(n-l)}} \varphi_{n-l} e^*(n-l) \\
&= \frac{1}{2}\Big(\varphi_{n-l} e^*(n-l)\Big)^H \\
&= \frac{1}{2} e(n-l)\varphi_{n-l}^H.
\end{aligned} \tag{4.19}$$

By substituting (4.18) and (4.19) in (4.17) we can obtain

$$\frac{\partial |e(n-l)|^2}{\partial \mathbf{w}_n} = -\frac{1}{2} e(n-l)\varphi_{n-l}^H. \tag{4.20}$$

Using the same method, the other terms of (4.16) can be calculated. By substituting all partial derivatives , we can simplify (4.16) as below

$$\begin{aligned}
\frac{\partial g}{\partial \mathbf{w}_n} &= \left(\frac{-1}{2\sigma^2}\right)\Bigg(-\frac{1}{2} e(n-l)\varphi_{n-l}^H + -\frac{1}{2} e(n-t)\varphi_{n-t}^H \\
&\quad + \frac{1}{2} e(n-l)\varphi_{n-t}^H + \frac{1}{2} e(n-t)\varphi_{n-l}^H \Bigg) \\
&= \left(\frac{1}{4\sigma^2}\right)\Big[e(n-l) - e(n-t)\Big]\Big[\varphi_{n-l}^H - \varphi_{n-t}^H\Big].
\end{aligned} \tag{4.21}$$

Therefore by substituting (4.21) in (4.14) we can obtain

$$\begin{aligned}
\frac{\partial J(n)}{\partial \mathbf{w}_n} &= \sum_{l=1}^{N}\sum_{t=1}^{N} \frac{\partial\Big[f(g_{l,t}(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n} \\
&= \sum_{l=1}^{N}\sum_{t=1}^{N} \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n} \\
&= \left(\frac{1}{4\sigma^2}\right) \times \sum_{l=1}^{N}\sum_{t=1}^{N} \exp(g_{l,t}(\mathbf{w}_n)) \\
&\times \Big[e(n-l) - e(n-t)\Big]\Big[\varphi_{n-l}^{H} - \varphi_{n-t}^{H}\Big]\Bigg).
\end{aligned} \tag{4.22}$$

Thus, the gradient of the cost function $J_n(e)$ can be calculated based on the following equation

$$\begin{aligned}
\nabla_{\mathbf{w}_n^*} J_n(e) &= \left(\frac{\partial J(n)}{\partial \mathbf{w}_n}\right)^{H} \\
&= \left(\frac{1}{4\sigma^2}\right) \times \Bigg[\sum_{l=1}^{N}\sum_{t=1}^{N} \exp(g_{l,t}(\mathbf{w}_n)) \\
&\times \Big[e(n-l) - e(n-t)\Big]\Big[\varphi_{n-l}^{H} - \varphi_{n-t}^{H}\Big]\Bigg]^{H}.
\end{aligned} \tag{4.23}$$

By setting $\mathbf{w}_0 = 0$ and replacing $\exp(g)$ with its kernel equivalent κ_σ we can obtain filter output weight as

$$\mathbf{w}_n = \zeta \sum_{p=0}^{n} \sum_{l=1}^{N} \sum_{t=1}^{N} \Bigg(\Bigg[\kappa_\sigma \Big(e(p-l) - e(p-t) \Big) \Bigg] \times \Big[e(p-l) - e(p-t) \Big] \Big[\varphi_{p-l}^H - \varphi_{p-t}^H \Big] \Bigg)^H \tag{4.24}$$

where $\zeta = \mu\sqrt{2\pi}/16\sigma$.

By substituting the weight update in the $y_n = \mathbf{w}_n^H \varphi_n$ and using properties of Quaternion Reproducing Kernel Hilbert Space (QRKHS) and the 'kernel trick' to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, we can simplify the equation in kernel form as

$$\begin{aligned} y_n = \zeta \sum_{p=0}^{n} \sum_{l=1}^{N} \sum_{t=1}^{N} \Big[\kappa_\sigma (e(p-l) - e(p-t)) \Big] \\ \times \Big[e(p-l) - e(p-t) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-t}}, \mathbf{u_n}) \Big]. \end{aligned} \tag{4.25}$$

Based on equation (4.25), the pseudo code for QKMEE could be summarized in Algorithm 1 table.

Algorithm 1 QKMEE Algorithm

Input: signal and desired $\{(\mathbf{u_i}, d(i))\}_{i=1}^{\infty} \subset \mathbb{H}$, $y_0 = 0$

Output: Estimate desired output $\hat{d(n)} = y_n$ at time n, Residual $e(n)$

1: **Initialization** The kernel parameters $\bar{\sigma}$, σ using Silverman's rule $1.06 \times min\{\sigma_Y, R/1.34\} \times N^{1/5L}$

2: **while** $(\mathbf{u_n}, d(n))$, available **do**

3: {calculate filter output at iteration n}

$$\begin{aligned} y_n = \zeta \sum_{p=0}^{n} \sum_{l=1}^{N} \sum_{t=1}^{N} \Big[\kappa_\sigma(e(p-l) - e(p-t)) \Big] \\ \times \Big[e(p-l) - e(p-t) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-t}}, \mathbf{u_n}) \Big] \end{aligned} \tag{4.26}$$

4: $e(n) = d(n) - y_n$ {calculate error at iteration n}

5: $n = n + 1$

6: **end while**

4.2 Simulation Results

4.2.1 Quaternion Nonlinear Channel Estimation

4.2.1.1 Symmetric Unimodal Density Noise

The Quat-KMEE (QKMEE) algorithm was simulated for a nonlinear channel with symmetric unimodal Gaussian noise. The channel consisted of the quaternion filter, i.e.,

$$\begin{aligned} z(n) = g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n) \\ + h_1^* u(n-1) + h_2^* u^i(n-1) + h_3^* u^j(n-1) + h_4^* u^k(n-1) \end{aligned} \tag{4.27}$$

and nonlineraity, i.e.,

$$y(n) = z(n) + az^2(n) + bz^3(n) + v(n) \tag{4.28}$$

where $v(n)$ is Symmetric unimodal density Gaussian noise described later. Coefficients $g1, ..., g4, h1, ..., h4, a, b,$ and noise $v(n)$ are all quaternion valued.The coefficients used were [23]

$a = 0.075 + i0.35 + j0.1 - k0.05,$

$b = -0.025 - i0.25 - j0.05 + k0.03,$

$g1 = -0.40 + i0.30 + j0.15 - k0.45,$

$h1 = 0.175 - i0.025 + j0.1 + k0.15,$

$g2 = -0.35 - i0.15 - j0.05 + k0.20,$

$h2 = 0.15 - i0.225 + j0.125 - k0.075,$

$g3 = -0.10 - i0.40 + j0.20 - k0.05,$

$h3 = +0.025 + i0.075 - j0.05 - k0.05,$

$g4 = +0.35 + i0.10 - j0.10 - k0.15,$

$h4 = -0.05 - i0.075 - j0.075 + k0.175.$

For the tests, the input $u(n)$ was formed using impulsive Gaussian mixture models to form non-Gaussian signals as follows [23]:

$$\begin{aligned} p_u(i) = &(0.85N(1.0, 0.01) + 0.15N(3.0, 0.01)) \\ &+i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01)) \\ &+j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01)) \\ &+k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01)) \end{aligned} \tag{4.29}$$

And noise $v(n)$ was formed using symmetric unimodal Gaussian distributions as:

$$\begin{aligned} p_v(i) = &(N(0.0, 0.01)) \\ &+i(N(3.0, 0.01)) \\ &+j(N(1.0, 0.01)) \\ &+k(N(0.5, 0.01)) \end{aligned} \tag{4.30}$$

where $N(m_N, \sigma_N)$ denotes the normal (Gaussian) PDF with mean m_N and variance σ_N. The kernel parameters $\bar{\sigma} = 3.14$; $\sigma = 0.17$ for the Parzen window with size $N = 10$, are estimated using Silverman's rule $1.06 \times min\{\sigma_Y, R/1.34\} \times N^{1/5L}$ [37] where σ_Y is the data standard deviation, L is data dimension, R is the interquartile and N is the number of samples. The simulation results of Quat-KMEE algorithm for the nonlinear channel are shown in Fig.4.1 and Fig.4.2.

Fig.4.1 shows the mean-squared-error (MSE) of the Quat-KMEE and Quat-KLMS algorithms with step sizes $\zeta = 2.5$ and $\zeta_{KLMS} = 1$ respectively. Fig.4.2 shows the probability density of error signal real and imaginary components of the Quat-KMEE and Quat-KLMS algorithms. As shown in Fig.4.2 the error signal's real and imaginary components have symmetric unimodal Gaussian distributions. The results show improvement of Quat-KMEE for modeling nonlinear channel when the input signal is non-Gaussian compared with Quat-KLMS.

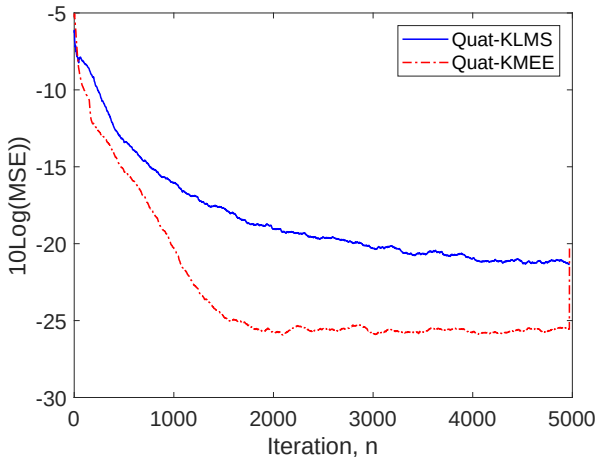
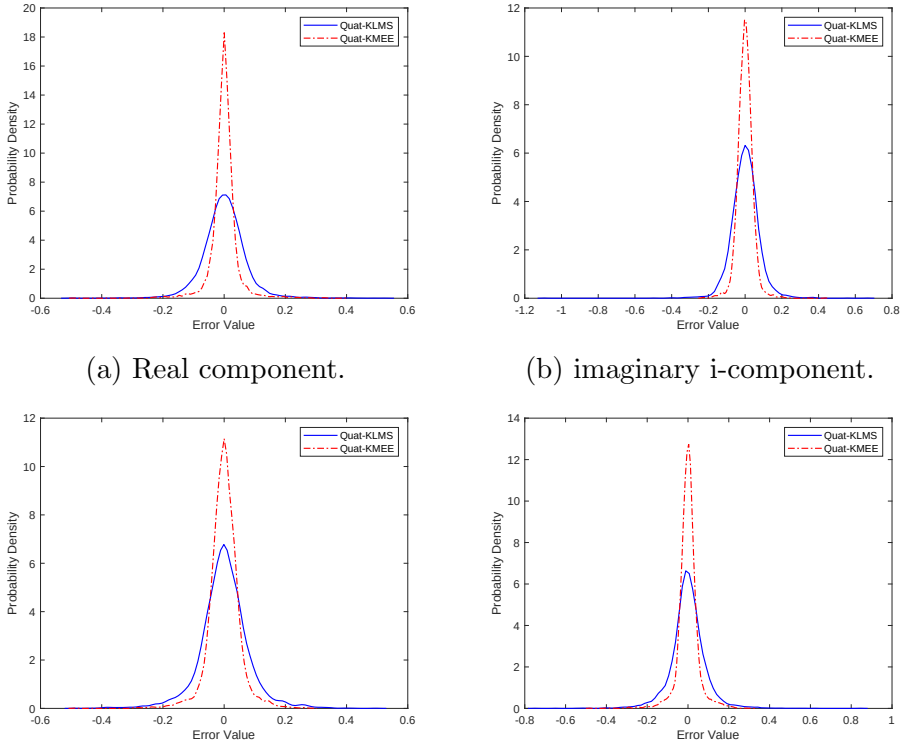


Fig. 4.1: Mean Squared Error of Quat-KLMS and Quat-KMEE for non-Gaussian signal



(a) Real component.

(b) imaginary i-component.

(c) imaginary j-component.

(d) imaginary k-component.

Fig. 4.2: Probability Density of Error Signal using unimodal density Noise.

4.2.2 Filtering of an Autoregressive Process

In this experiment, the model was described by a fourth-order quaternion autoregressive process AR(4) for one step ahead prediction as in equation (4.30)

$$\begin{aligned} z(n) =& 1.4z(n-1) - 0.7z(n-2) + 0.04z(n-3) \\ & - 0.05z(n-4) + u(n) \end{aligned} \tag{4.31}$$

where input noise $u(n)$ was formed using impulsive Gaussian mixture models to form non-Gaussian signal defined as (4.31). The state $z(n)$ was observed through the non-linearity described in (4.27), where $v(n)$ is non-Gaussian noise as (4.32). The system signal-to-noise ration was set to $SNR = 0.1$dB.

$$\begin{aligned} p_u(i) = & (0.85N(1.0, 0.01) + 0.15N(3.0, 0.01)) \\ & + i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01)) \\ & + j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01)) \\ & + k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01)) \end{aligned} \tag{4.32}$$

$$\begin{aligned} p_v(i) = & (0.90N(0.0, 0.01) + 0.10N(1.0, 0.01)) \\ & + i(0.70N(3.0, 0.01) + 0.30N(0.5, 0.01)) \\ & + j(0.45N(1.0, 0.01) + 0.55N(4.5, 0.01)) \\ & + k(0.80N(0.5, 0.01) + 0.20N(1.5, 0.01)) \end{aligned} \tag{4.33}$$

The parameters for the Quat-KMEE were $\zeta = 10$, $\bar{\sigma} = 6.5$, and $\sigma = 0.35$, and for the Quat-KLMS $\zeta = 0.9$, $\bar{\sigma} = 6.5$ were used. In simulation, the performance of algorithms were measured based on the prediction gain which can be described as

$R_p = 10\log(\sigma_y^2/\sigma_e^2)$ where σ_y^2 and σ_e^2 are the power of input and output error respectively [24]. Fig.4.3 shows the prediction gain of Quat-KMEE and Quat-KLMS algorithms. As shown in Fig.4.3 the Quat-KMEE has better steady-state prediction gain (around 20 dB) compared to Quat-KLMS. Fig.4.4 shows the probability densities of the real and imaginary parts of the error signal of the Quat-KMEE and Quat-KLMS filters. It is clear from Fig.4.4 that entropy criterion generates more concentrated error probability distribution PDF, whereas the variance (MSE) generates wider error probability distribution PDF.

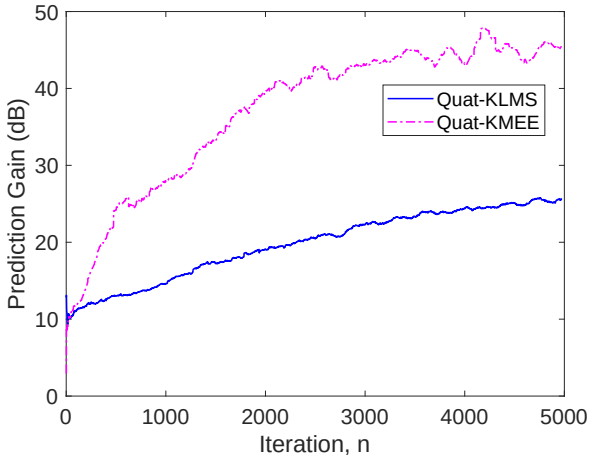
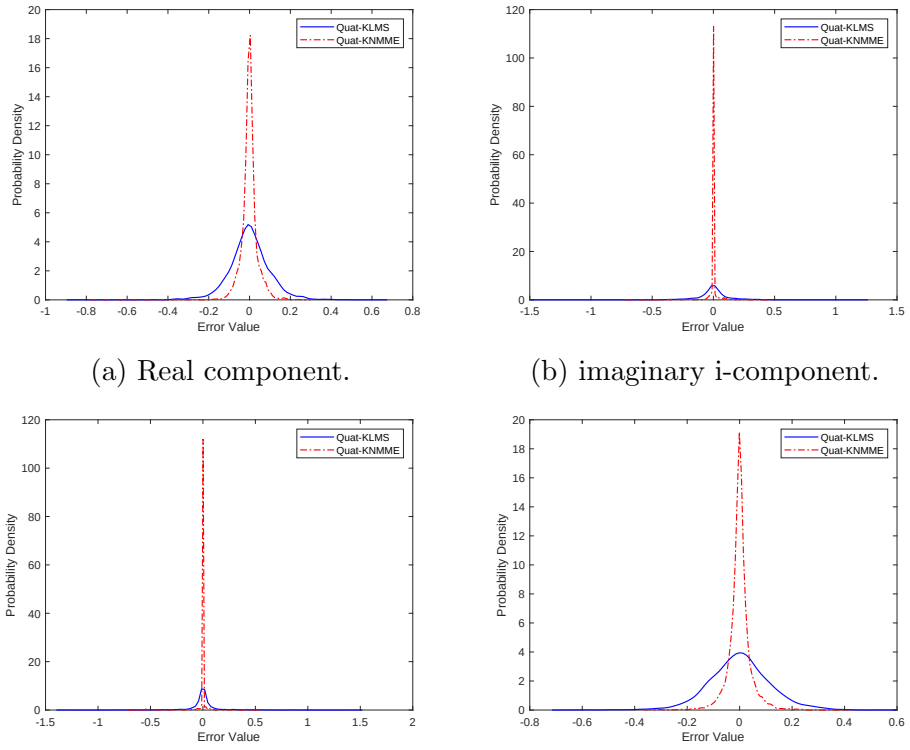


Fig. 4.3: Prediction Gain of Quat-KLMS and Quat-KMEE for non-Gaussian signal and noise of Filtering AR(4)



(a) Real component.

(b) imaginary i-component.

(c) imaginary j-component.

(d) imaginary k-component.

Fig. 4.4: Probability Density of Error signal for non-Gaussian signal and noise of Filtering AR(4)

4.3 Conclusion

We have shown the derivation and demonstration of convergence of a quaternion kernel adaptive algorithm based on minimum error entropy. The algorithm is based on information theoretic learning (ITL) cost function. The resulting algorithm is the Quat-KMEE algorithm. A gradient is derived based on GHR calculus applied on quaternion RKHS. Simulation results show the convergence of the mean-squared-error of the new algorithm (QKMEE) versus the existing algorithm (QKLMS). The QKMEE algorithm performed better with non-Gaussian signals and noise compared to QKLMS which is based on the MSE criteria adaptive filter and has better convergence misadjustment. Also the sim-

ulation results show that minimizing error entropy results in more concentrated error probability distribution PDF compared to MSE criterion.

Chapter 5

Quaternion Kernel Normalized Minimum Error Entropy Adaptive Algorithms

This chapter shows how to develop a kernel adaptive filter for quaternion data based on normalized minimum error entropy cost function. The generalized Hamilton-real (GHR) calculus which is applicable to Hilbert space for evaluating the cost function gradient is applied to develop the quaternion kernel normalized minimum error entropy (QKNMEE) algorithm. The new proposed algorithm enhanced QKMEE algorithm while the filter update step-size selection will be independent of the input power and the kernel size.

In chapter 4, the QKMEE algorithm was developed [13]. One of the main drawbacks of the minimum error entropy algorithm (MEE) is its strong dependency on the kernel size σ, and on the input signal power. In order to avoid these problems Han et al [38] [39] proposed the normalized minimum error entropy (NMEE) algorithm. Diniz et al [14] address some of the issues from the previous works and derived a new version for the linear-in-parameter NMEE algorithm which the solution is equivalent to the previous works .

The quaternion normalized minimum error entropy (QNMEE) algorithm, minimizes Renyis quadratic entropy of the error between the filter output and desired response. This approach improved performance for biased or non-Gaussian signals compared to the minimum mean square error criterion, while converges quickly with misadjustment.

This chapter is organized as follow: section 2 contains the algorithm derivation, section 3 convergence analysis, section 4 is simulation results and section 5 is conclusions.

5.1 The Quaternion Normalized Minimum Error Entropy Algorithm Derivation

In previous chapter (chapter 4), the QKMEE algorithm was developed. The goal was to maximize the information potential of the error signal. The adaptive filter could be expressed as $y_n = < \Phi(\mathbf{u}_n) \ ,\mathbf{w}_n >$, which also can be written as:

$$y_n = \mathbf{w}_n^H \varphi_n \tag{5.1}$$

where $\varphi_n = \Phi(\mathbf{u}_n)$ is the kernel map to a QRKHS [25].

The normalized minimum error entropy algorithm proposed in [14] was based on the real number domain $\mathbb{R}$. We use the same method as real domain case to develop quaternion kernel normalized minimum error entropy algorithm (QKNMEE). The proposed parameter to be estimated in quaternion domain may be described as follows:

$$\begin{aligned} \min_{\forall \mathbf{w}_{n+1} \in \mathcal{H}} \quad & \|\mathbf{w}_{n+1} - \mathbf{w}_n\|_2^2 \\ \text{s.t.} \quad & \epsilon(n) - \epsilon(l) = 0 \\ & \forall l \in \{n-N, ...n-1\} \end{aligned} \tag{5.2}$$

where $\mathcal{H}$ is a quaternion RKHS and $\epsilon(n-l) = d(n-l) - \mathbf{w}_{n+1}^H \varphi_{n-l}$ are *a posteriori errors* for $\forall l : 1 \leqslant l \leqslant N$.

The above constrained minimization problem (5.2) could be converted to the following

unconstrained minimization problem with cost function $J(n)$ using quaternion Lagrange multipliers $\lambda_{n-l} \in \mathbb{H}$ for $\forall l : 1 \leqslant l \leqslant N$:

$$J(n) = (\mathbf{w}_{n+1} - \mathbf{w}_n)^H (\mathbf{w}_{n+1} - \mathbf{w}_n) + \sum_{l=1}^{N} \lambda_{n-l}(\epsilon(n) - \epsilon(n-l)). \tag{5.3}$$

The minimum of $J(n)$ is reached when the gradient of $J(n)$ with respect to $\mathbf{w}_{n+1}$ is zero. The gradient of cost function $J(n)$ can be calculated in quaternion domain using GHR calculus as follow

$$\begin{aligned} \nabla_{\mathbf{w}^*_{n+1}} J(n) &= \left(\frac{\partial J(n)}{\partial \mathbf{w}_{n+1}} \right)^H \\ &= \left(\frac{\partial (\mathbf{w} - \mathbf{w}_n)^H (\mathbf{w} - \mathbf{w}_n)}{\partial \mathbf{w}_{n+1}} \right)^H \\ &+ \left(\frac{\partial \sum_{l=1}^{N} \lambda_{n-l}(\epsilon(n) - \epsilon(n-l))}{\partial \mathbf{w}_{n+1}} \right)^H \end{aligned} \tag{5.4}$$

where

$$\begin{aligned} &\frac{\partial J(n)}{\partial \mathbf{w}_{n+1}} = \\ &= \frac{\partial (\mathbf{w}^H_{n+1} \mathbf{w}_{n+1})}{\partial \mathbf{w}_{n+1}} - \frac{\partial (\mathbf{w}^H_{n+1} \mathbf{w}_n)}{\partial \mathbf{w}_{n+1}} \\ &- \frac{\partial (\mathbf{w}^H_n \mathbf{w}_{n+1})}{\partial \mathbf{w}_{n+1}} + \frac{\partial (\mathbf{w}^H_n \mathbf{w}_n)}{\partial \mathbf{w}_{n+1}} \\ &+ \sum_{l=1}^{N} \lambda_{n-l} \frac{\partial (\epsilon(n) - \epsilon(n-l))}{\partial \mathbf{w}_{n+1}}. \end{aligned} \tag{5.5}$$

Using product rule of GHR calculus, the gradient can be calculated as:

$$\begin{aligned}
&\frac{\partial J(n)}{\partial \mathbf{w}_{n+1}} = \\
&\mathbf{w}_{n+1}^{H}\frac{\partial \mathbf{w}_{n+1}}{\partial \mathbf{w}_{n+1}} + \frac{\partial \mathbf{w}_{n+1}^{H}}{\partial \mathbf{w}_{n+1}^{\mathbf{w}_{n+1}}}\mathbf{w}_{n+1} \\
&- \mathbf{w}_{n+1}^{H}\frac{\partial \mathbf{w}_{n}}{\partial \mathbf{w}_{n+1}} - \frac{\partial \mathbf{w}_{n+1}^{H}}{\partial \mathbf{w}_{n+1}^{\mathbf{w}_{n}}}\mathbf{w}_{n} \\
&- \mathbf{w}_{n}^{H}\frac{\partial \mathbf{w}_{n+1}}{\partial \mathbf{w}_{n+1}} - \frac{\partial \mathbf{w}_{n}^{H}}{\partial \mathbf{w}_{n+1}^{\mathbf{w}_{n+1}}}\mathbf{w}_{n+1} \\
&+ \mathbf{w}_{n}^{H}\frac{\partial \mathbf{w}_{n}}{\partial \mathbf{w}_{n+1}} + \frac{\partial \mathbf{w}_{n}^{H}}{\partial \mathbf{w}_{n}^{\mathbf{w}_{n}}}\mathbf{w}_{n} \\
&+ \sum_{l=1}^{N}\lambda_{n-l}(\frac{1}{2}\varphi_{n}^{H} - \frac{1}{2}\varphi_{n-l}^{H}).
\end{aligned} \tag{5.6}$$

Using GHR calculus and derivatives properties the gradient can be simplified as:

$$\begin{aligned}
&\frac{\partial J(n)}{\partial \mathbf{w}_{n+1}} \\
&= \frac{1}{2}\mathbf{w}_{n+1}^{H} - \frac{1}{2}\mathbf{w}_{n}^{H} \\
&+ \sum_{l=1}^{N}\lambda_{n-l}(\frac{1}{2}\varphi_{n}^{H} - \frac{1}{2}\varphi_{n-l}^{H}).
\end{aligned} \tag{5.7}$$

Therefore, by setting $\frac{\partial J(n)}{\partial \mathbf{w}_{n+1}} = 0$, the filter weight update can be calculated as :

$$\begin{aligned}
\mathbf{w}_{n+1}^{H} =& \mathbf{w}_{n}^{H} - \sum_{l=1}^{N}\lambda_{n-l}(\varphi_{n}^{H} - \varphi_{n-l}^{H}) \\
&= \mathbf{w}_{n}^{H} - \Lambda\Psi_{d}(n)^{H}
\end{aligned} \tag{5.8}$$

where $\Psi_d(n) = [\varphi_n - \varphi_{n-1}, ..., \varphi_n - \varphi_{n-N}] \in \mathcal{H}^{1\times N}$ which $\mathcal{H}$ is a quaternion RKHS and $\Lambda = [\lambda_{n-1}, ..., \lambda_{n-N}] \in \mathbb{H}^{1\times N}$ for N quaternion Lagrange multipliers.

The N Lagrange multipliers may be computed by the N constraint equations $\epsilon(n+1,n) = \epsilon(n+1,k)$ for $\forall k \in \{n-N, ...n-1\}$ using N previous posterior errors defined as $\epsilon(n,k) = d(k) - \mathbf{w}_n^H \varphi_k$. Therefore $\forall k \in \{n-N, ...n-1\}$ we have the following equation

$$d(n) - \mathbf{w}_{n+1}^H \varphi_n = d(k) - \mathbf{w}_{n+1}^H \varphi_k. \tag{5.9}$$

By substituting (5.8) in (5.9), we can obtain

$$\begin{aligned} &d(n) - \left(\mathbf{w}_n^H - \sum_{l=1}^{N} \lambda_{n-l} (\varphi_n^H - \varphi_{n-l}^H) \right) \varphi_n \\ &= d(k) - \left(\mathbf{w}_n^H - \sum_{l=1}^{N} \lambda_{n-l} (\varphi_n^H - \varphi_{n-l}^H) \right) \varphi_k. \end{aligned} \tag{5.10}$$

By using distributive property of quaternion RKHS, equation (5.10) can be expressed as:

$$\begin{aligned} &d(n) - \mathbf{w}_n^H \varphi_n - \sum_{l=1}^{N} \lambda_{n-l} (\varphi_n^H - \varphi_{n-l}^H) \varphi_n \\ &= d(k) - \mathbf{w}_n^H \varphi_k - \sum_{l=1}^{N} \lambda_{n-l} (\varphi_n^H - \varphi_{n-l}^H) \varphi_k. \end{aligned} \tag{5.11}$$

By substituting the posterior errors and changing the order in equation (5.11), it can be simplified to:

$$
\begin{aligned}
e(n) - \epsilon(n,k) &= -\sum_{l=1}^{N} \lambda_{n-l}(\varphi_n^H - \varphi_{n-l}^H)(\varphi_n - \varphi_k) \\
&= -\Lambda \Psi_d(n)^H (\varphi_n - \varphi_k).
\end{aligned} \tag{5.12}
$$

Now, we define $\epsilon_d = [e(n) - \epsilon(n, n-1), ..., e(n) - \epsilon(n, n-N)] \in \mathbb{H}^{1\times N}$, and rewrite N distinct delta error equations in matrix form as:

$$
\epsilon_d = -\Lambda \Psi_d(n)^H \Psi_d(n). \tag{5.13}
$$

Therefore the N quaternion Lagrange multipliers can be calculated as

$$
\Lambda = -\epsilon_d \Big(\Psi_d(n)^H \Psi_d(n) \Big)^{-1} \tag{5.14}
$$

where it is assumed that $\Psi_d(n)^H \Psi_d(n) \in \mathcal{H}^{N\times N}$ is non-singular and $\mathcal{H}$ is QRKHS. By substituting Λ in equation (5.8) we can simplify filter weight update recursion formula as:

$$
\mathbf{w}_{n+1}^H = \mathbf{w}_n^H + \epsilon_d \Big(\Psi_d(n)^H \Psi_d(n) \Big)^{-1} \Psi_d(n)^H. \tag{5.15}
$$

To simplify the weight update calculation and reduce the computational complexity due to matrix inversion, we use matrix inversion lemma and simplify the filter weight update equation (5.8) as:

$$\mathbf{w}_{n+1}^{H} = \mathbf{w}_{n}^{H} + \epsilon_d \Psi_d(n)^H \left(\Psi_d(n) \Psi_d(n)^H \right)^{-1} \tag{5.16}$$

or in element-wise form as:

$$\begin{aligned} \mathbf{w}_{n+1}^{H} = \mathbf{w}_{n}^{H} + & \left(\sum_{l=1}^{N} \Big[e(n) - \epsilon(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \right) \times \\ & \left(\Big[\varphi_n - \varphi_{n-l} \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] + ... + \right. \\ & \left. \Big[\varphi_n - \varphi_{n-N} \Big] \Big[\varphi_n^H - \varphi_{n-N}^H \Big] \right)^{-1} . \end{aligned} \tag{5.17}$$

Using quaternion left Hilbert space inner product properties we can simplify the equation (5.17) as:

$$\begin{aligned} \mathbf{w}_{n+1}^{H} = \mathbf{w}_{n}^{H} + & \left(\sum_{l=1}^{N} \Big[e(n) - \epsilon(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \right) \times \\ & \left(\Big[\varphi_n^H - \varphi_{n-l}^H \Big] \Big[\varphi_n - \varphi_{n-l} \Big] + ... + \right. \\ & \left. \Big[\varphi_n^H - \varphi_{n-N}^H \Big] \Big[\varphi_n - \varphi_{n-N} \Big] \right)^{-1} . \end{aligned} \tag{5.18}$$

Therefore by expanding the vectors multiplications we can overwrite equation (5.18) as follow:

$$\mathbf{w}_{n+1}^H = \mathbf{w}_n^H + \left(\sum_{l=1}^{N} \Big[e(n) - \epsilon(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \right) \times$$
$$\Bigg(\varphi_n^H \varphi_n - \varphi_n^H \varphi_{n-1} - \varphi_{n-1}^H \varphi_n + \varphi_{n-1}^H \varphi_{n-1} + ... \tag{5.19}$$
$$\varphi_n^H \varphi_n - \varphi_n^H \varphi_{n-N} - \varphi_{n-N}^H \varphi_n + \varphi_{n-N}^H \varphi_{n-N} \Bigg)^{-1} .$$

Using properties of QRKHS and the kernel trick to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, we can simplify the equation (5.19) in kernel form as:

$$\mathbf{w}_{n+1}^H = \mathbf{w}_n^H + \left(\sum_{l=1}^{N} \Big[e(n) - \epsilon(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \right) \times$$
$$\Bigg(\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_{n-1}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{n-1}}, \mathbf{u_{n-1}}) + ... \tag{5.20}$$
$$\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_{n-N}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{n-N}}, \mathbf{u_{n-N}})... \Bigg)^{-1} .$$

Using quaternion-extended real Gaussian kernel [25], $\bar{\kappa}_{\bar{\sigma}}(.,.)$ is real Gaussian kernel, the inverse term in equation (5.20) is changed to real number and can be moved to right or left side in quaternion multiplication. By setting $\mathbf{w}_0^H = 0$ and including a step size factor η, the weight update recursion can be calculated as:

$$\mathbf{w}_n^H = \eta \sum_{p=0}^{n-1} \left(\sum_{l=1}^{N} \Big[e(p) - \epsilon(p-l) \Big] \Big[\varphi_p^H - \varphi_{p-l}^H \Big] \right) \times \left(\sum_{l=1}^{N} \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_p}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_{p-l}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_{p-l}}) \right)^{-1} . \tag{5.21}$$

By substituting the weight update in the $y_n = \mathbf{w}_n^H \varphi_n$ and using properties of QRKHS and the 'kernel trick' to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, equation (5.21) can be simplified in kernel form as:

$$y_n = \eta \sum_{p=0}^{n-1} \left(\sum_{l=1}^{N} \Big[e(p) - \epsilon(p-l) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) \Big] \right) \times \left(\sum_{l=1}^{N} \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_p}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_{p-l}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_{p-l}}) \right)^{-1} . \tag{5.22}$$

5.2 Convergence Analysis

The goal of the convergence analysis is to find a range for learning step size η in equation (5.21) which QKNMEE converges to optimal set of weights. To studying convergence of QKMEE algorithm, we consider an approach using the energy conservation relation [40]. The weight error at iteration $n+1$ can be defined as:

$$\begin{aligned}\mathbf{v}_{n+1} &= \mathbf{w}^0 - \mathbf{w}_{n+1} \\ &= \mathbf{w}^0 - (\mathbf{w}_n + \Delta\mathbf{w}_n) \\ &= \mathbf{v}_n - \Delta\mathbf{w}_n.\end{aligned} \tag{5.23}$$

For checking energy conservation, we initially find *a priori* and *a posteriori* errors: $e_n^a = \mathbf{v}_n^H \varphi_n$ and $e_n^p = \mathbf{v}_{n+1}^H \varphi_n$ respectively, where

$$\begin{aligned}e_n^p &= \mathbf{v}_{n+1}^H \varphi_n \\ &= (\mathbf{v}_n^H - \Delta\mathbf{w}_n^H)\varphi_n \\ &= e_n^a - \Delta\mathbf{w}_n^H \varphi_n \\ &= e_n^a - \eta \\ &\times \left(\sum_{l=1}^{N} \Big[e(n) - e(n-l) \Big] \Big[\varphi_n^H \varphi_n - \varphi_{n-l}^H \varphi_n \Big] \right) \\ &\times \left(\sum_{l=1}^{N} \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_p}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_{p-l}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_{p-l}}) \right)^{-1}.\end{aligned} \tag{5.24}$$

To simplify calculation, function $\gamma(n)$ can be defined as:

$$\begin{aligned}\gamma(n) &\triangleq \left(\sum_{l=1}^{N} \Big[e(n) - e(n-l) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{n-l}}, \mathbf{u_n}) \Big] \right) \\ &\times \left(\sum_{l=1}^{N} \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_p}) - 2\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_{p-l}}) + \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_{p-l}}) \right)^{-1}\end{aligned} \tag{5.25}$$

thus, the energy can be expressed as:

$$\begin{aligned}
&\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2 = \left\|\mathbf{v}_{n}^{H}\varphi_n - \eta\gamma(n)\right\|^2 \\
&= \left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2 + \mathbf{v}_{n}^{H}\varphi_n\Big(- \eta\gamma^*(n)\Big) \\
&+ \Big(- \eta\gamma(n)\Big)\Big(\mathbf{v}_{n}^{H}\varphi_n\Big)^* \\
&+ \|-\eta\gamma(n)\|^2 \\
&= \left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2 - 2\eta\mathfrak{Re}\Big(\mathbf{v}_{n}^{H}\varphi_n\gamma^*(n)\Big) \\
&+ \eta^2\|\gamma(n)\|^2 .
\end{aligned} \tag{5.26}$$

Using Cauchy Schwarz inequality in Hilbert space and normalized kernel $\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) = 1$, we can expressed the following inequality as:

$$\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2 \leqslant \|\mathbf{v}_{n+1}\|^2\|\varphi_n\|^2 = \|\mathbf{v}_{n+1}\|^2\,\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) \tag{5.27}$$

therefore

$$\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2 \leqslant \|\mathbf{v}_{n+1}\|^2 \tag{5.28}$$

and

$$\left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2 \leqslant \|\mathbf{v}_{n}\|^2 . \tag{5.29}$$

By subtracting (5.29) from (5.28), inequality can be written as:

$$\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2 - \left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2 \leqslant \|\mathbf{v}_{n+1}\|^2 - \|\mathbf{v}_n\|^2 . \tag{5.30}$$

By taking expectation of both sides of (5.30) :

$$E\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2 - E\left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2 \leqslant E\|\mathbf{v}_{n+1}\|^2 - E\|\mathbf{v}_n\|^2 . \tag{5.31}$$

For convergence, the energy of the weight error vector should gradually reduce per iteration, thus

$$E\left[\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2\right] - E\left[\left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2\right] < 0. \tag{5.32}$$

Therefore, by taking expectation of both sides of equation (5.26) and using inequality (5.32)

$$\begin{aligned} & E\left[\left\|\mathbf{v}_{n+1}^{H}\varphi_n\right\|^2\right] - E\left[\left\|\mathbf{v}_{n}^{H}\varphi_n\right\|^2\right] \\ & = -2\eta E\left[\mathfrak{Re}\left(\mathbf{v}_{n}^{H}\varphi_n\gamma^{*}(n)\right)\right] \\ & + \eta^2 E\left[\|\gamma(n)\|^2\right] \\ & < 0. \end{aligned} \tag{5.33}$$

Thus, in order the algorithm converges, the convergence step size η should be:

$$\eta < 2\frac{E\left[\mathfrak{Re}\left(e_n^p \gamma^*(n)\right)\right]}{E\left[\|\gamma(n)\|^2\right]}. \tag{5.34}$$

5.3 Simulation Results

5.3.1 Channel Estimation Based on the Weiner Nonlinear Model

The Quat-KNMEE (QKNMEE) algorithm was simulated with Parzen Window lenght $N = 10$ for a nonlinear channel with non-Gaussian noise versus Quat-KLMS [23]. The channel consisted of the quaternion filter, i.e.,

$$z(n) = g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n)$$
$$+h_1^* u(n-1) + h_2^* u^i(n-1) + h_3^* u^j(n-1) + h_4^* u^k(n-1)$$

and nonlineraity, i.e.,

$$y(n) = z(n) + az^2(n) + bz^3(n) + v(n)$$

where $v(n)$ is added non-Gaussian noise described later. Coefficients $g1, ..., g4, h1, ..., h4, a, b$, and noise $v(n)$ are all quaternion valued.The coefficients used were

$$a = 0.075 + i0.35 + j0.1 - k0.05,$$
$$b = -0.025 - i0.25 - j0.05 + k0.03,$$

$$g1 = -0.40 + i0.30 + j0.15 - k0.45,$$
$$h1 = 0.175 - i0.025 + j0.1 + k0.15,$$
$$g2 = -0.35 - i0.15 - j0.05 + k0.20,$$
$$h2 = 0.15 - i0.225 + j0.125 - k0.075,$$
$$g3 = -0.10 - i0.40 + j0.20 - k0.05,$$
$$h3 = +0.025 + i0.075 - j0.05 - k0.05,$$
$$g4 = +0.35 + i0.10 - j0.10 - k0.15,$$
$$h4 = -0.05 - i0.075 - j0.075 + k0.175.$$

For the tests, both input $u(n)$ and noise $v(n)$ were formed using impulsive Gaussian mixture models to form non-Gaussian signals. A quaternion random variable with components from different real Gaussian distributions was formed [23]. The probability distributions used were

$$p_u(i) = (0.85N(1.0, 0.01) + 0.15N(3.0, 0.01))$$
$$+ i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01))$$
$$+ j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01))$$
$$+ k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01))$$

$$p_v(i) = (0.90N(0.0, 0.01) + 0.10N(1.0, 0.01))$$
$$+ i(0.70N(3.0, 0.01) + 0.30N(0.5, 0.01))$$
$$+ j(0.45N(1.0, 0.01) + 0.55N(4.5, 0.01))$$
$$+ k(0.80N(0.5, 0.01) + 0.20N(1.5, 0.01))$$

where $N(m_N, \sigma_N)$ denotes the normal (Gaussian) PDF with mean m_N and variance σ_N. The Quat-KNMEE and Quat-KLMS simulation results for the nonlinear channel

are shown in Fig.5.1 to Fig.5.2. The Fig.5.1 and Fig.5.2 are ensemble-averaged over 10 realizations.

To compare the performance of new proposed algorithm Quat-KNMEE with Quat-KMEE and Quat-KLMS, the parameters of all three algorithms were chosen that all three algorithms reached the same steady state mean square errors. For this reason the parameters for the Quat-KNMEE were $\eta = 3$, $\bar{\sigma} = 2.24$ and for the Quat-KMEE were $\eta = 0.4$, $\bar{\sigma} = 2.24$, and $\sigma = 0.736$ and for the Quat-KLMS $\eta = 0.5$, $\bar{\sigma} = 2.24$ were used.

Fig.5.1 shows the performance comparisons when the input power was set to 5.1 dB and measurement noise 12.5 dBm. As shown in Fig.5.1, The newly proposed algorithm Quat-KNMEE converged with 1000 iterations where the Quat-KMEE converged with 2000 iterations. It is clear from Fig.5.1, that the newly proposed algorithm Quat-KNMEE converges faster compared to the other two algorithms Quat-KMEE and Quat-KLMS.

To show that the Quat-KNMEE filter update step-size selection is independent of the input power, the input power of second simulation was increased to 1.75dB while all step-size of all three algorithms were kept the same as before. Fig.5.2 shows the input power impacts on three algorithms when input power was set to 1.75 dB. As shown in Fig.5.2, the convergence rate of the Quat-KNMEE didn't change and converged within 1000 iterations while the convergence rate and stability of the other two algorithms Quat-KMEE and Quat-KLMS changed and converged faster compared to the first simulation using smaller input power.

Fig.5.3 shows the learning curves of the Quat-KNMEE filter with different convergence step sizes. As shown in Fig.5.3 when the step size parameter η increases, the rate of convergence of Quat-KNMEE algorithm is correspondingly increased. When the step size is set to values greater than 5, the algorithm couldn't converge and become unstable.

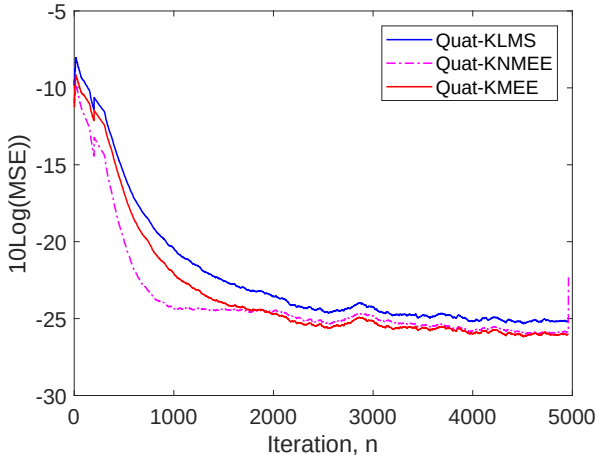


Fig. 5.1: Learning curves for Quat-KNMEE, Quat-KMEE and Quat-KLMS and for non-Gaussian signal with input power = -5.1 dB

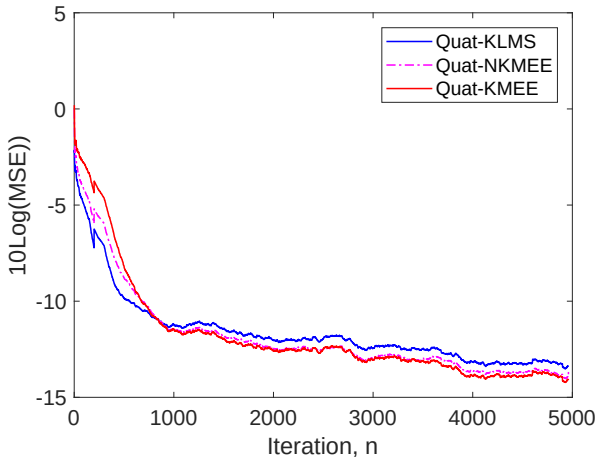


Fig. 5.2: Learning curves for Quat-KNMEE, Quat-KMEE and Quat-KLMS and for non-Gaussian signal with input power = 1.75 dB

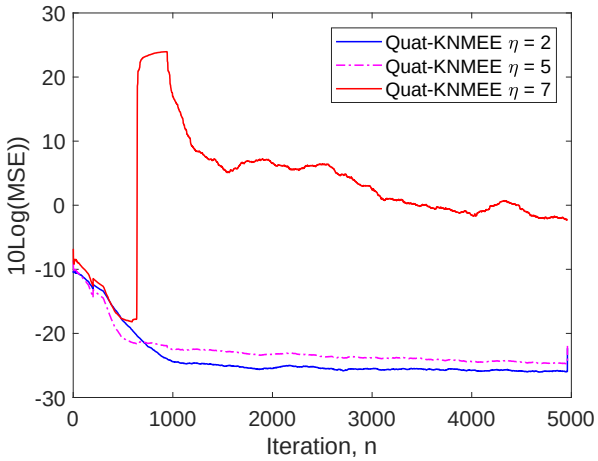


Fig. 5.3: Learning curves for Quat-KNMEE with different convergence step size for non-Gaussian signal with input power = -5.1 dB

5.4 Conclusion

We have shown the derivation and convergence analysis of a quaternion kernel adaptive algorithm based on normalized minimum error entropy. The algorithm is based on information theoretic learning (ITL) cost function. The resulting algorithm is the Quat-KNMEE (QKNMEE) algorithm. We used GHR calculus to derive the gradient of cost function in QRKHS. Simulation results show the convergence curve of the mean square error of the new algorithm (QKNMEE) versus the existing algorithms Quat-KMEE (QKMEE) and Quat-KLMS (QKLMS). The algorithm's convergence is very fast and outperforms the existing one QKMEE and QKLMS. The convergence analysis (5.34) shows that convergence step-size is independent of kernel size. The simulation results show that the convergence rate of the Quat-KNMEE is independent of the input power and the kernel size. QKNMEE algorithm gives better performance for low signal to noise ratio (SNR) environments.

Chapter 6

Quaternion Kernel Minimum Error Entropy Algorithm with Fiducial point for Nonlinear Adaptive Systems

In chapter 4, we developed quaternion kernel adaptive filter based on minimum error entropy referred to as the quaternion KMEE (QKMEE) algorithm [13]. Since entropy does not change with the mean of the distribution, the algorithm may converge to set of optimal weights without having zero mean error [3]. To make the zero mean output error, the output during testing session was biased with the mean of errors of training session. However, for non-symmetric or heavy tails error PDF the estimation of error mean is problematic.

In statistical signal processing and machine learning applications, one of the challenging problem is the estimation of unknown parameters or hidden functions from noisy observations [41]. In Bayesian estimation framework the solution is by calculating the posterior probability density functions PDF of the unknowns. Although the computation of statistical quantities related to these posterior distribution is analytically impossible, sampling techniques such as Monte Carlo (MC) solve the problem [41]. To reduce the computational complexity and variance of the corresponding estimators, some deterministic sampling techniques are applied to the algorithms. As a result, high-order information about the distribution can be captured with a fixed and small number of

points [42]. Techniques such as Herding [43] [44] can be used to sub-select a small collection of samples from a much larger set of MC samples to minimize the squared error between expected feature values evaluated at the true distribution and the empirical distribution obtained from herding. Similarly, in Quasi-Monte Carlo (QMC) methods [45], deterministic sequences of samples are selected based on the concept of low-discrepancy to avoid all kind of randomness. Other techniques such as quadrature, unscented transformations [41] are applied for deterministic approximations of the posterior distribution. These techniques generate a set of deterministically selected samples which are called sigma points, to do better estimation of moments of the posterior density function. These techniques are as extension of the standard Kalman filter and are used in filtering applications as alternative to the particle filtering techniques based on MC sampling and most of them are derived for the Gaussian distribution [41].

In our previous work we developed quaternion kernel adaptive filter based on minimum error entropy referred to as the quaternion KMEE (QKMEE) algorithm [13]. Since entropy does not change with the mean of the distribution, the algorithm may converge to set of optimal weights without having zero mean error [3]. To make the zero mean output error, the output during testing session was biased with the mean of errors of training session. However, for non-symmetric or heavy tails error PDF the estimation of error mean is problematic. Another way to locate the peak of the error probability distribution function at the origin is incorporate this operation in the cost function using fiducial points [18]. Fiducial points technique for estimating the error probability distribution function does not require extra steps in estimation algorithms. Instead, it integrates fixed points in cost function during the optimization process. However, QMC or other sigma points techniques require using deterministic selection of samples.

In this chapter, we describe a quaternion kernel adaptive filter based on minimum error entropy cost function with Fiducial point referred to as the quaternion minimum error

entropy with fiducial point (QKMEEF) algorithm.

As described in chapter 4, the Renyi's entropy [33] such as the *order–α Renyi's entropy* for quaternion data could be defined as:

$$H_\alpha(e) = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} p_e^\alpha(\varepsilon) d\varepsilon \tag{6.1}$$

where $\alpha \in \mathbb{R}^+ \backslash \{1\}$ and p_e is probability distribution function of quaternion random variable e.

We can define *order–α information potential* V_α as

$$V_\alpha(e) = \int_{-\infty}^{\infty} p_e^\alpha(\varepsilon) d\varepsilon. \tag{6.2}$$

In practice the entropy function is not accessible since it is a function of the PDF of relative random variable e. The entropy can be estimated by using some specific method such as the *Parzen window* which is a good estimation of the *order–2 Renyi's entropy* function.

For a set of N statistically independent random samples $\{e_i\}_{i=1}^N$ of quaternion random variable e,

the *Parzen window* computes the estimate of the probability distribution function p_e as

$$\hat{p_e}(\varepsilon) = \frac{1}{N} \sum_{l=1}^{N} \kappa_{\sigma/\sqrt{2}}(\varepsilon - e_l) \tag{6.3}$$

where κ_σ is Gaussian-based kernel for quaternion data defined as

$$\kappa_\sigma(X-Y) = \frac{4}{\sqrt{2\pi}\sigma}\exp\{\frac{-1}{2\sigma^2}(X_r-Y_r)^2 + (X_i-Y_i)^2 + (X_j-Y_j)^2 + (X_k-Y_k)^2\} = \frac{4}{\sqrt{2\pi}\sigma}\exp\{\frac{-1}{2\sigma^2}|X-Y|^2\}$$

where X and Y are quaternion numbers $\in \mathbb{H}$ in form of
$X = X_r + iX_i - jX_j - kX_k$ and $Y = Y_r + iY_i - jY_j - kY_k$.
More details of the quaternion kernel is provided in [25].

The information potential $V(e)$ can be estimated using Parzen window as

$$\hat{V}(e) = \frac{1}{N^2}\sum_{l_1=1}^{N}\sum_{l_2=1}^{N}\kappa_\sigma(e_{l_1} - e_{l_2}). \tag{6.4}$$

The global solution of maximization of the $\hat{V}(e)$ using *Parzen window* estimation is achieved when all related errors are constant and the maximum value reaches to $\hat{V}(0) = \frac{4}{\sqrt{2\pi}\sigma}$. We will show later that the maximization solution is the global solution.

In supervised learning the goal is to make most of the errors equal to zero, we can construct naturally an augmented criterion so that it minimizes the error entropy or (maximizing the error information potential) with locating the peak of the error PDF at the origin [27]. This can be done by maximizing the information potential cost function $\hat{V}(e)$ in quaternion domain $\mathbb{H}$ respect to a fiducial point $e_0 = 0$ where $\{e_i\}_{i=0}^{n}$ are the errors produced by adaptive filter. We can construct an augmented cost function $J_n(e)$ by adding a fiducial point to the information potential function (6.4) as

$$
\begin{aligned}
J_n(e) = & \frac{1}{(N+1)^2} \sum_{i,j=1}^{N} \kappa_\sigma(e(n-i) - e(n-j)) \\
& + \frac{1}{(N+1)^2} \left[2 \sum_{i=1}^{N} \kappa_\sigma(e(n-i)) + \kappa_\sigma(0)) \right].
\end{aligned}
\tag{6.5}
$$

Based on (6.5) we develop *The Quaternion Kernel Minimum Error Entropy with Fiducial Point Algorithm* in next section.

6.1 The Quaternion Kernel Minimum Error Entropy with Fiducial Point Algorithm Derivation

For the quaternion kernel adaptive filter based on minimum error entropy with fiducial point cost function (QKMEEF), the goal is to maximize the cost function $J_n(e)$ (6.5) respect to free parameter $\mathbf{w}_n$ as

$$
\begin{aligned}
\max_{\forall \mathbf{w}_n \in \mathcal{H}} \quad & J_n(e) \\
\text{s.t.} \quad & e(n) = d - y_n \\
& y_n =< \Phi(\mathbf{u}_n), \mathbf{w}_n >= \mathbf{w}_n^H \varphi_n
\end{aligned}
\tag{6.6}
$$

where d is desired signal, $\mathbf{u}_n$ input signal, φ_n= $\Phi(\mathbf{u}_n)$ and $\Phi(.)$ is the kernel map to a quaternion RKHS [25]. Maximizing the information potential cost function $J_n(e)$ (6.5) can be done with unconstrained optimization algorithm such as gradient ascent algorithm.

$$\begin{aligned}\mathbf{w}_{n+1} =& \mathbf{w}_n + \eta \nabla_{\mathbf{w}_n^*} J_n(e) \\ =& \mathbf{w}_n + \mu \left(\frac{\partial (J_1(n) + J_2(n))}{\partial \mathbf{w}_n} \right)^H \end{aligned} \tag{6.7}$$

where

$$J_1(n) = \sum_{l,t=1}^{N} \exp\{-\frac{|e(n-l) - e(n-t)|^2}{2\sigma^2}\} \tag{6.8}$$

and

$$J_2(n) = \sum_{l=1}^{N} \exp\{-\frac{|e(n-l)|^2}{2\sigma^2}\} \tag{6.9}$$

where η is adaptation step size and $\mu = \eta \frac{1}{(N+1)^2} \frac{4}{\sqrt{2\pi}\sigma}$,
and $e(n-l) = d(n-l) - \mathbf{w}_n^H \varphi_{n-l}$ are *a posteriori errors* for all $l : 1 \leqslant l \leqslant N$.

To derive the gradient of cost function we define functions $f \colon \mathbb{H} \to \mathbb{H}$ and $g_{l,t} \colon \mathbb{H} \to \mathbb{R}$ as

$$f(x) = \exp(x) \tag{6.10}$$

$$g_{l,t}(\mathbf{w}_n) = -\frac{|e(n-l) - e(n-t)|^2}{2\sigma^2}. \tag{6.11}$$

To simplify the notation for function $g_{l,t}$ in our derivative for a given l and t, $1 \leqslant l \leqslant N$ and $1 \leqslant t \leqslant N$ we define
$g(\mathbf{w}_n) = g_{l,t}(\mathbf{w}_n) = -\frac{\left|e(n-l)-e(n-t)\right|^2}{2\sigma^2}$.

With the above notation the derivative $\frac{\partial J_1(n)}{\partial \mathbf{w}_n}$ can be written as

$$\frac{\partial J_1(n)}{\partial \mathbf{w}_n} = \sum_{l=1}^{N}\sum_{t=1}^{N} \frac{\partial\Big[f(g_{l,t}(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n}. \tag{6.12}$$

For a given l and t, the partial derivative can be calculated with GHR chain rule as

$$\begin{aligned} \frac{\partial\Big[f(g_{l,t}(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n} &= \frac{\partial\Big[f(g(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n} \\ &= \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n}. \end{aligned} \tag{6.13}$$

Using HR derivative property and quaternion rotation, for $\forall v \in \{i,j,k\}$ we can show that $\frac{\partial f}{\partial g^v} = 0$. Suppose $v = i$ then

$$\begin{aligned} \frac{\partial f}{\partial g^i} &= \frac{1}{4}\left(\frac{\partial f}{\partial g_r} - i\frac{\partial f}{\partial g_i} + j\frac{\partial f}{\partial g_j} + k\frac{\partial f}{\partial g_k}\right) \\ &= \frac{1}{4}\left(\frac{\partial \exp(g)}{\partial g_r} - i\frac{\partial \exp(g)}{\partial g_i} + j\frac{\partial \exp(g)}{\partial g_j} + k\frac{\partial \exp(g)}{\partial g_k}\right) \\ &= \frac{1}{4}\Big(\exp(g) - ii\exp(g) + jj\exp(g) + kk\exp(g)\Big) \\ &= \frac{1}{4}\Big(\exp(g) + \exp(g) - \exp(g) - \exp(g)\Big) = 0 \end{aligned} \tag{6.14}$$

and if $v = 1$ then

$$\begin{aligned} \frac{\partial f}{\partial g} &= \frac{1}{4}\left(\frac{\partial f}{\partial g_r} - i\frac{\partial f}{\partial g_i} - j\frac{\partial f}{\partial g_j} - k\frac{\partial f}{\partial g_k}\right) \\ &= \exp(g). \end{aligned} \tag{6.15}$$

By substituting (6.14) and (6.15) in (6.13) we can simplify (6.13) as follows

$$\sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n} = \exp(g)\frac{\partial g}{\partial \mathbf{w}_n}. \tag{6.16}$$

To calculate the derivative of function g, we can expand it as follows

$$\begin{aligned} g(\mathbf{w}_n) &= -\frac{|e(n-l)-e(n-t)|^2}{2\sigma^2} \\ &= \frac{-1}{2\sigma^2}|e(n-l)-e(n-t)|^2 \\ &= \frac{-1}{2\sigma^2}\Big[(e(n-l)-e(n-t))(e(n-l)-e(n-t))^*\Big] \\ &= \frac{-1}{2\sigma^2}\Big[(e(n-l)-e(n-t))(e^*(n-l)-e^*(n-t))\Big] \\ &= \frac{-1}{2\sigma^2}\Big[|e(n-l)|^2+|e(n-t)|^2-e(n-l)e^*(n-t) \\ &- e(n-t)e^*(n-l)\Big] \end{aligned} \tag{6.17}$$

therefore by substituting (6.17) in (6.16) we can find partial derivative of g using GHR calculus as shown below

$$\begin{aligned} \frac{\partial g}{\partial \mathbf{w}_n} = \left(\frac{-1}{2\sigma^2}\right)\Bigg[&\frac{\partial |e(n-l)|^2}{\partial \mathbf{w}_n}+\frac{\partial |e(n-t)|^2}{\partial \mathbf{w}_n} \\ &-\frac{\partial e(n-l)e^*(n-t)}{\partial \mathbf{w}_n}-\frac{\partial e(n-t)e^*(n-l)}{\partial \mathbf{w}_n}\Bigg]. \end{aligned} \tag{6.18}$$

By substituting $e(n-l)=d(n-l)-\mathbf{w}_n^H\varphi_{n-l}$ in (6.18) and using GHR calculus, we can

compute each partial derivative of (6.18) as

$$\begin{aligned}\frac{\partial|e(n-l)|^2}{\partial \mathbf{w}_n} &= \frac{\partial e(n-l)e^*(n-l)}{\partial \mathbf{w}_n} \\ &= e(n-l)\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n} + \frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}}e^*(n-l)\end{aligned} \tag{6.19}$$

where

$$\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H \frac{\partial \mathbf{w}_n}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H. \tag{6.20}$$

The second term of (6.19) can be calculated as

$$\begin{aligned}&\frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}}e^*(n-l) = -\frac{\partial \mathbf{w}_n^H \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}}e^*(n-l) \\ &= -\mathbf{w}_n^H \frac{\partial \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}}e^*(n-l) - \frac{\partial \mathbf{w}_n^H}{\partial \mathbf{w}_n^{\varphi_{n-l}e^*(n-l)}}\varphi_{n-l}e^*(n-l) \\ &= \frac{1}{2}\left(\varphi_{n-l}e^*(n-l)\right)^H \\ &= \frac{1}{2}e(n-l)\varphi_{n-l}^H.\end{aligned} \tag{6.21}$$

By substituting (6.20) and (6.21) in (6.19) we can obtain

$$\frac{\partial|e(n-l)|^2}{\partial \mathbf{w}_n} = -\frac{1}{2}e(n-l)\varphi_{n-l}^H. \tag{6.22}$$

Using the same method, the other terms of (6.18) can be calculated. By substituting all partial derivatives, we can simplify (6.18) as below

$$\begin{aligned}\frac{\partial g}{\partial \mathbf{w}_n} &= \left(\frac{-1}{2\sigma^2}\right)\Bigg(-\frac{1}{2}e(n-l)\varphi_{n-l}^H - \frac{1}{2}e(n-t)\varphi_{n-t}^H \\ &\quad + \frac{1}{2}e(n-l)\varphi_{n-t}^H + \frac{1}{2}e(n-t)\varphi_{n-l}^H\Bigg) \\ &= \left(\frac{1}{4\sigma^2}\right)\Big[e(n-l)-e(n-t)\Big]\Big[\varphi_{n-l}^H - \varphi_{n-t}^H\Big].\end{aligned} \tag{6.23}$$

Therefore by substituting (6.23) in (6.16) we can obtain

$$\begin{aligned}\frac{\partial J_1(n)}{\partial \mathbf{w}_n} &= \sum_{l=1}^{N}\sum_{t=1}^{N}\frac{\partial\Big[f(g_{l,t}(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n} \\ &= \sum_{l=1}^{N}\sum_{t=1}^{N}\sum_{v\in\{1,i,j,k\}}\frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n} \\ &= \left(\frac{1}{4\sigma^2}\right)\times\sum_{l=1}^{N}\sum_{t=1}^{N}\exp(g_{l,t}(\mathbf{w}_n)) \\ &\quad\times\Big[e(n-l)-e(n-t)\Big]\Big[\varphi_{n-l}^H-\varphi_{n-t}^H\Big].\end{aligned} \tag{6.24}$$

We can use the same method to calculate the $\frac{\partial J_2(n)}{\partial \mathbf{w}_n}$ as below

$$\begin{aligned}\frac{\partial J_2(n)}{\partial \mathbf{w}_n} &= \left(\frac{1}{4\sigma^2}\right)\times 2\sum_{l=1}^{N}\exp(e(n-l)) \\ &\quad\times e(n-l)\varphi_{n-l}^H.\end{aligned} \tag{6.25}$$

Therefore the gradient of the cost function $J(n)$ can be calculated based on the following

equation

$$\begin{aligned}\nabla_{\mathbf{w}_n^*} J_n(e) &= \left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n}\right)^H \\ &= \frac{1}{4(N+1)^2\sigma^2} \times \Bigg[\sum_{l=1}^{N}\sum_{t=1}^{N} \kappa_\sigma\Big(e(n-l)-e(n-t)\Big) \\ &\times \Big[e(n-l)-e(n-t)\Big]\Big[\varphi_{n-l}^H - \varphi_{n-t}^H\Big] \\ &+ 2\sum_{l=1}^{N} \kappa_\sigma(e(n-l))e(n-l)\varphi_{n-l}^H\Bigg]^H .\end{aligned} \tag{6.26}$$

The gradient of the cost function $J_n(e)$ will be zero when

$$\nabla_{\mathbf{w}_n^*} J_n(e) = 0 \iff \{e(n-l) = 0 : \forall l : 1 \leqslant l \leqslant N\}. \tag{6.27}$$

The solution of (6.27) is the global solution of maximization problem (6.6). Otherwise there should exist $l : 1 \leqslant l \leqslant N$ which $e(n-l) \neq 0$. By plugging in the non zero solution in (19) the corresponding kernel term couldn't reach to the maximum value and results $J_n(e) < J_n(0) = \kappa_\sigma(0)$ which is smaller than the maximum value of the cost function.

By setting the initial weight $\mathbf{w}_0 = 0$, the filter update weight at iteration n could be calculated as

$$\begin{aligned}\mathbf{w}_n = \zeta \sum_{p=0}^{n-1} \Bigg(& \sum_{l=1}^{N}\sum_{t=1}^{N}\Big[\kappa_\sigma\Big(e(p-l)-e(p-t)\Big)\Big] \\ & \times \Big[e(p-l)-e(p-t)\Big]\Big[\varphi_{p-l}^H - \varphi_{p-t}^H\Big] \\ & + 2\sum_{l=1}^{N}\kappa_\sigma\Big(e(p-l)\Big)e(p-l)\varphi_{p-l}^H\Bigg)^H\end{aligned} \tag{6.28}$$

where $\zeta = \mu\sqrt{2\pi}/16\sigma = \eta\frac{1}{4(N+1)^2\sigma^2}$.

The weight update equation (6.28) includes the products of errors with Hermitian transpose of the input vectors φ_{p-l} in Reproducing Kernel Hilbert Space for $\forall l : 1 \leqslant l \leqslant N$

$$\begin{aligned}e(p-l)\varphi_{p-l}^H &= \Big[d(p-l) - \mathbf{w}_{p-l}^H\varphi_{p-l}\Big]\varphi_{p-l}^H \\ &= d(p-l)\varphi_{p-l}^H - \mathbf{w}_{p-l}^H\varphi_{p-l}\varphi_{p-l}^H \\ &= d(p-l)\Phi(\mathbf{u}_{p-l})^H - \mathbf{w}_{p-l}^H\Phi(\mathbf{u}_{p-l})\Phi(\mathbf{u}_{p-l})^H.\end{aligned} \tag{6.29}$$

The covariance term $\Phi(\mathbf{u}_{p-l})\Phi(\mathbf{u}_{p-l})^H$ in (6.29) indicates that the augmented statistics of quaternion input vector is inherent to the QKMEEF algorithm.

By substituting the weight update in the $y_n = \mathbf{w}_n^H\varphi_n$ and using properties of Quaternion Reproducing Kernel Hilbert Space (QRKHS) and the 'kernel trick' to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, we can simplify the equation in kernel form as

$$
\begin{aligned}
y_n = \zeta \sum_{p=0}^{n-1} \Bigg(& \sum_{l=1}^{N} \sum_{t=1}^{N} \Big[\kappa_\sigma(e(p-l) - e(p-t)) \Big] \\
& \times \Big[e(p-l) - e(p-t) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-t}}, \mathbf{u_n}) \Big] \\
& + 2 \sum_{l=1}^{N} \kappa_\sigma \Big(e(p-l) \Big) e(p-l) \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) \Bigg).
\end{aligned} \tag{6.30}
$$

6.2 Convergence Analysis

The goal of the convergence analysis is to find a range for learning step size η in equation (6.7) which the QKMEEF algorithm converges to optimal set of weights.

$$
\mathbf{w}_{n+1} = \mathbf{w}_n + \eta \nabla_{\mathbf{w}_n^*} J(n). \tag{6.31}
$$

To study convergence of the QKMEEF algorithm, we consider an approach using the energy conservation relation [40].

The weight error at iteration $n+1$ can be defined as

$$
\begin{aligned}
\mathbf{v}_{n+1} &= \mathbf{w}^0 - \mathbf{w}_{n+1} \\
&= \mathbf{w}^0 - (\mathbf{w}_n + \Delta \mathbf{w}_n) \\
&= \mathbf{v}_n - \Delta \mathbf{w}_n \\
&= \mathbf{v}_n - \eta \nabla_{\mathbf{w}_n^*} J(n) \\
&= \mathbf{v}_n - \frac{\eta}{4\sigma^2 (N+1)^2} \Big[\mathbf{\Lambda}(n) \mathbf{\Psi} + \mathbf{\Lambda_d}(n) \mathbf{\Psi_d}(n)^H \Big]^H
\end{aligned} \tag{6.32}
$$

where

$$\mathbf{\Psi_d}(n) = [\mathbf{\Phi_d}(n,1), ..., \mathbf{\Phi_d}(n,N)]$$

$$\mathbf{\Phi_d}(n,k) = [\varphi_{n-k} - \varphi_{n-1}, ..., \varphi_{n-k} - \varphi_{n-N}]$$

$$\mathbf{\Lambda_d}(n) = \Big[\mathbf{\Gamma_d}(n,1), ..., \mathbf{\Gamma_d}(n,N)\Big]$$

$$\mathbf{\Gamma_d}(n,k) = \Big[\kappa_\sigma(e(n-k) - e(n-1))(e(n-k) - e(n-1)),$$

$$, ..., \kappa_\sigma(e(n-k) - e(n-N))(e(n-k) - e(n-N))\Big]$$

and

$$\mathbf{\Psi}(n) = [\varphi_{n-1}, ..., \varphi_{n-N}]$$

$$\mathbf{\Lambda}(n) = \Big[\kappa_\sigma(e(n-1))(e(n-1)),$$

$$, ..., \kappa_\sigma(e(n-N))(e(n-N))\Big].$$

Based on energy conservation relation, both side of equation (6.32) should have the same energy, thus the energy can be expressed as:

$$\begin{aligned} &\|\mathbf{v}_{n+1}\|_{\mathbb{F}}^2 = \\ &\left\|\mathbf{v}_n - \frac{\eta}{4\sigma^2(N+1)^2}\Big[\mathbf{\Lambda}(n)\mathbf{\Psi} + \mathbf{\Lambda_d}(n)\mathbf{\Psi_d}(n)^H\Big]^H\right\|_{\mathbb{F}}^2 \end{aligned} \tag{6.33}$$

where $\|.\|_{\mathbb{F}}^2$ is the norm in Quaternion Reproducing Kernel Hilbert Space (QRKHS).

By Expanding the equation (6.33) we can obtain

$$
\begin{aligned}
&\mathbf{v}_{n+1}^H \mathbf{v}_{n+1} = \\
&\left(\mathbf{v}_n^H - \frac{\eta}{4\sigma^2(N+1)^2} \left[\mathbf{\Lambda}(n)\mathbf{\Psi}^H(n) + \mathbf{\Lambda_d}(n)\mathbf{\Psi_d}^H(n) \right] \right) \\
&\times \left(\mathbf{v}_n - \frac{\eta}{4\sigma^2(N+1)^2} \left[\mathbf{\Psi}(n)\mathbf{\Lambda}^H(n) + \mathbf{\Psi_d}(n)\mathbf{\Lambda_d}^H(n) \right] \right) \\
&= \mathbf{v}_n^H \mathbf{v}_n \\
&- 2\frac{\eta}{4\sigma^2(N+1)^2} \mathfrak{Re} \left(\mathbf{v}_n^H \mathbf{\Psi}(n)\mathbf{\Lambda}^H(n) + \mathbf{v}_n^H \mathbf{\Psi_d}(n)\mathbf{\Lambda_d}^H(n) \right) \\
&+ \frac{\eta^2}{16\sigma^4(N+1)^4} \Bigg(\mathbf{\Lambda}(n)\mathbf{\Psi}^H(n)\mathbf{\Psi}(n)\mathbf{\Lambda}^H(n) \\
&+ 2 \times \mathfrak{Re} \Big(\mathbf{\Lambda}(n)\mathbf{\Psi}^H(n)\mathbf{\Psi_d}(n)\mathbf{\Lambda_d}^H(n) \\
&+ \mathbf{\Lambda_d}(n)\mathbf{\Psi_d}^H(n)\mathbf{\Psi_d}(n)\mathbf{\Lambda_d}^H(n) \Big) \Bigg).
\end{aligned}
\tag{6.34}
$$

By taking expectation of both sides of equation (6.34) we obtain

$$
\begin{aligned}
E\left[\,\|\mathbf{v}_{n+1}\|_{\mathbb{F}}^{2}\,\right] &= E\left[\,\|\mathbf{v}_{n}\|_{\mathbb{F}}^{2}\,\right] \\
&- 2\frac{\eta}{4\sigma^{2}(N+1)^{2}}E\Bigg[\mathfrak{Re}\Bigg(\mathbf{v}_{n}^{H}\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^{H}(n)+ \\
&\mathbf{v}_{n}^{H}\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^{H}(n)\Bigg)\Bigg] \\
&+ \frac{\eta^{2}}{16\sigma^{4}(N+1)^{4}}E\Bigg[\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^{H}(n)\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^{H}(n) \\
&+ 2\times\mathfrak{Re}\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^{H}(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^{H}(n) \\
&+ \boldsymbol{\Lambda}_{\mathbf{d}}(n)\boldsymbol{\Psi_{\mathbf{d}}}^{H}(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^{H}(n)\Bigg)\Bigg].
\end{aligned}
\tag{6.35}
$$

For convergence, the energy of the weight error vector should gradually reduce per iteration, thus

$$
E\left[\,\|\mathbf{v}_{n+1}\|_{\mathbb{F}}^{2}\,\right] < E\left[\,\|\mathbf{v}_{n}\|_{\mathbb{F}}^{2}\,\right]. \tag{6.36}
$$

Therefore the algorithm converged if the following inequality could be satisfied:

$$
\begin{aligned}
&- 2\frac{\eta}{4\sigma^2(N+1)^2}E\Bigg[\mathfrak{Re}\Bigg(\mathbf{v}_n^H\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^H(n)+\\
&\mathbf{v}_n^H\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\Bigg)\Bigg]\\
&+\frac{\eta^2}{16\sigma^4(N+1)^4}E\Bigg[\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^H(n)\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^H(n)\\
&+2\times\mathfrak{Re}\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^H(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\\
&+\boldsymbol{\Lambda}_{\mathbf{d}}(n)\boldsymbol{\Psi_{\mathbf{d}}}^H(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\Bigg)\Bigg]<0
\end{aligned}
\tag{6.37}
$$

thus

$$
\begin{aligned}
&0<\eta<8\sigma^2(N+1)^2\\
&\times E\Bigg[\mathfrak{Re}\Bigg(\mathbf{v}_n^H\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^H(n)+\mathbf{v}_n^H\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\Bigg)\Bigg]/\\
&E\Bigg[\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^H(n)\boldsymbol{\Psi}(n)\boldsymbol{\Lambda}^H(n)\\
&+2\times\mathfrak{Re}\Bigg(\boldsymbol{\Lambda}(n)\boldsymbol{\Psi}^H(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\\
&+\boldsymbol{\Lambda}_{\mathbf{d}}(n)\boldsymbol{\Psi_{\mathbf{d}}}^H(n)\boldsymbol{\Psi}_{\mathbf{d}}(n)\boldsymbol{\Lambda_{\mathbf{d}}}^H(n)\Bigg)\Bigg].
\end{aligned}
\tag{6.38}
$$

After algorithm convergence, the $\boldsymbol{\Psi}_{\mathbf{d}}$ converged to $\mathbf{0}$ therefore (6.38) could be simplified as

$$0 < \eta < 8\sigma^2(N+1)^2 \frac{E\left[\mathfrak{Re}\left(\mathbf{v}_n^H \mathbf{\Psi}(n)\mathbf{\Lambda}^H(n)\right)\right]}{E\left[\left(\mathbf{\Lambda}(n)\mathbf{\Psi}^H(n)\mathbf{\Psi}(n)\mathbf{\Lambda}^H(n)\right)\right]}. \tag{6.39}$$

The upper bound for η satisfies the condition for the algorithm convergence.

Using normalized kernel $\varphi_n^H \varphi_n = \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n}) = 1$ the minimum of the upper bound of η could be approximated as

$$\begin{aligned}
&\min\left\{ 8\sigma^2(N+1)^2 \frac{E\left[\mathfrak{Re}\left(\mathbf{v}_n^H \mathbf{\Psi}(n)\mathbf{\Lambda}^H(n)\right)\right]}{E\left[\left(\mathbf{\Lambda}(n)\mathbf{\Psi}^H(n)\mathbf{\Psi}(n)\mathbf{\Lambda}^H(n)\right)\right]} \right\} \\
&\approx 8\sigma^2(N+1)^2 \frac{E\left[\mathfrak{Re}\left(N\kappa_\sigma(0)e(n)e^*(n)\right)\right]}{E\left[N^2\kappa_\sigma^2(0)e(n)e^*(n)\right]} \\
&= 8\sigma^2(N+1)^2 \frac{N\kappa_\sigma(0)\|e(n)\|_{\mathbb{F}}^2}{N^2\kappa_\sigma^2(0)\|e(n)\|_{\mathbb{F}}^2} = \frac{2\sqrt{2\pi}\sigma^3(N+1)^2}{N}.
\end{aligned} \tag{6.40}$$

6.3 Simulation Results

6.3.1 Quaternion Nonlinear Channel Estimation

6.3.1.1 Symmetric Unimodal Density Noise

The Quat-KMEEF (QKMEEF) algorithm was simulated for a nonlinear channel with symmetric unimodal Gaussian noise. The channel consisted of the quaternion filter, i.e.,

$$\begin{aligned} z(n) &= g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n) \\ &+ h_1^* u(n-1) + h_2^* u^i(n-1) + h_3^* u^j(n-1) + h_4^* u^k(n-1) \end{aligned} \tag{6.41}$$

and nonlineraity, i.e.,

$$y(n) = z(n) + az^2(n) + bz^3(n) + v(n) \tag{6.42}$$

where $v(n)$ is Symmetric unimodal density Gaussian noise described later. Coefficients $g1, ..., g4, h1, ..., h4, a, b$, and noise $v(n)$ are all quaternion valued.The coefficients used were [23]

$a = 0.075 + i0.35 + j0.1 - k0.05,$

$b = -0.025 - i0.25 - j0.05 + k0.03,$

$g1 = -0.40 + i0.30 + j0.15 - k0.45,$

$h1 = 0.175 - i0.025 + j0.1 + k0.15,$

$g2 = -0.35 - i0.15 - j0.05 + k0.20,$

$h2 = 0.15 - i0.225 + j0.125 - k0.075,$

$g3 = -0.10 - i0.40 + j0.20 - k0.05,$

$h3 = +0.025 + i0.075 - j0.05 - k0.05,$

$g4 = +0.35 + i0.10 - j0.10 - k0.15,$

$h4 = -0.05 - i0.075 - j0.075 + k0.175.$

For the tests, the input $u(n)$ was formed using impulsive Gaussian mixture models to form non-Gaussian signals as follows [23]:

$$\begin{aligned} p_u(i) = &(0.85N(1.0, 0.01) + 0.15N(3.0, 0.01)) \\ &+i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01)) \\ &+j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01)) \\ &+k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01)) \end{aligned} \tag{6.43}$$

And noise $v(n)$ was formed using symmetric unimodal Gaussian distributions as:

$$\begin{aligned} p_v(i) = &(N(0.0, 0.01)) \\ &+i(N(3.0, 0.01)) \\ &+j(N(1.0, 0.01)) \\ &+k(N(0.5, 0.01)) \end{aligned} \tag{6.44}$$

where $N(m_N, \sigma_N)$ denotes the normal (Gaussian) PDF with mean m_N and variance σ_N. The kernel parameters $\bar{\sigma} = 2.24$; $\sigma = 0.55$ for the Parzen window with size $N = 10$, are estimated using Silverman's rule $1.06 \times min\{\sigma_Y, R/1.34\} \times N^{1/5L}$ [40] where σ_Y is the data standard deviation, L is data dimension, R is the interquartile and N is the number of samples. The simulation results of Quat-KMEEF algorithm for the nonlinear channel are shown in Fig.1 to Fig.4. The theoretical upper bound of adaptation step size could be approximated by $\eta < 10$ using (6.40).

Fig.6.1 and Fig.6.2 show the convergence of the information potential (IP) and error variance (MSE) of the Quat-KMEEF algorithm with different convergence step η re-

spectively. Fig.6.3 shows the probability density of error signal real and imaginary components of the Quat-KMEEF algorithm with step size $\eta = 0.45$ and $\eta = 1$. As shown in Fig.6.3 the error signal real and imaginary components have symmetric unimodal Gaussian distributions. Based on [25], for appropriate subclasses of unimodal distributions, the variance of error signal (MSE) can be bounded in terms of the entropy power as $ce^{H(e)}$ where $H(e)$ is the error entropy and c is a positive constant. Therefore if error entropy $H(e) \rightarrow -\infty$ (or information potential $IP(n) \rightarrow +\infty$) the variance of error signal (MSE) can reach to zero. As shown in Fig.6.1 and Fig.6.2, when the algorithm information potentials (IP) converged the corresponding error variances (MSE) also converged. Fig.6.4 shows the impact of signal to noise ratio (SNR) on convergence misadjustment with step size $\eta = 1$. As shown in Fig.6.4 by increasing the SNR the convergence misadjustment degrades.

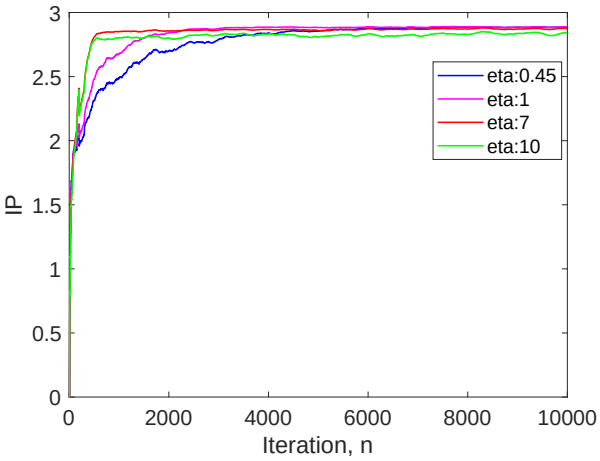


Fig. 6.1: IP using Symmetric Unimodal Density Noise.

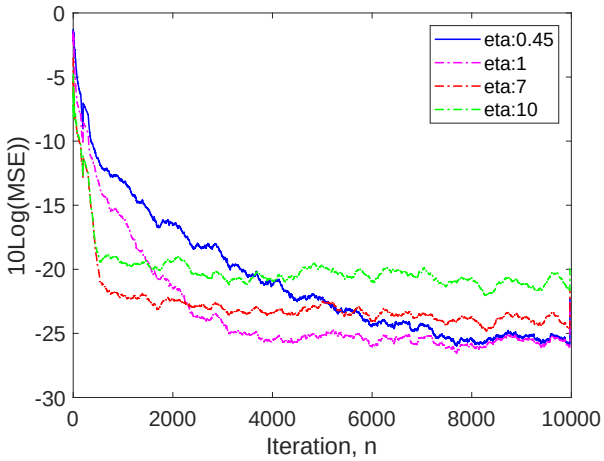
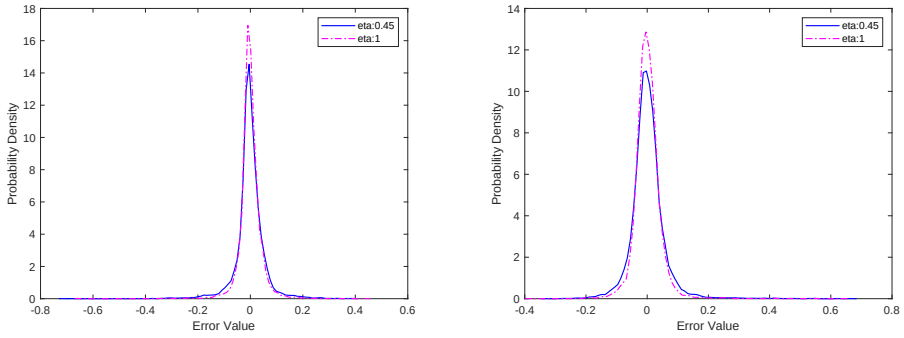
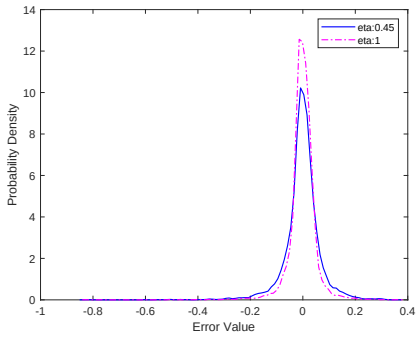


Fig. 6.2: Mean Square Errors vs. adaptation step size using Symmetric Unimodal Density Noise.

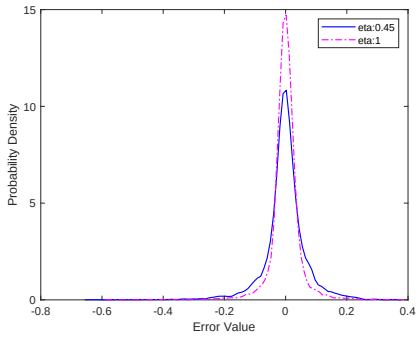


(a) Real component.

(b) imaginary i-component.



(c) imaginary j-component.



(d) imaginary k-component.

Fig. 6.3: Probability Density of Error Signal using Symmetric Unimodal Density Noise.

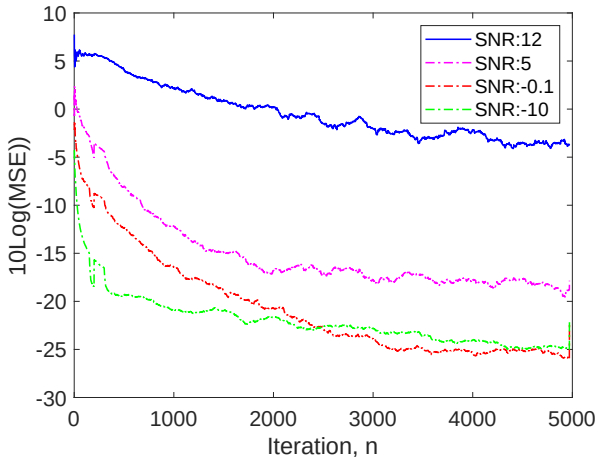


Fig. 6.4: Mean Square Errors vs. SNR using Symmetric Unimodal Density Noise

6.3.1.2 Bi-Modal Density Noise

In this experiment we used same test bench as previous section but instead of using symmetric unimodal density Gaussian Noise, we applied Bi-modal density or Gaussian mixture distribution noise as bellow [11]:

$$\begin{aligned} p_v(i) = &(0.90N(0.0, 0.01) + 0.10N(1.0, 0.01)) \\ &+i(0.70N(3.0, 0.01) + 0.30N(0.5, 0.01)) \\ &+j(0.45N(1.0, 0.01) + 0.55N(4.5, 0.01)) \\ &+k(0.80N(0.5, 0.01) + 0.20N(1.5, 0.01)) \end{aligned} \tag{6.45}$$

Fig.6.7 shows the algorithm adaptation error when algorithm converged. As shown in Fig.6.7, the algorithm error signal density is not unimodal. Therefore even after algorithm converged by minimizing the error entropy (or maximizing information potential IP), it does not guarantee that error variance (MSE) can converge to zero due to bi-modal distribution nature of error signal [46]. As shown in Fig.6.5 and Fig.6.6 when algorithm converges to maximum $IP = 2.1$ with $(\eta = 0.45, 1)$, the

variance of error (MSE) reached to -10 dB. Although the information potential of bi-modal case ($IP = 2.1, MSE = -10dB$) is slightly smaller than unimodal case ($IP = 2.9, MSE = -25dB$), it penalized the error variance MSE by +15 dB.

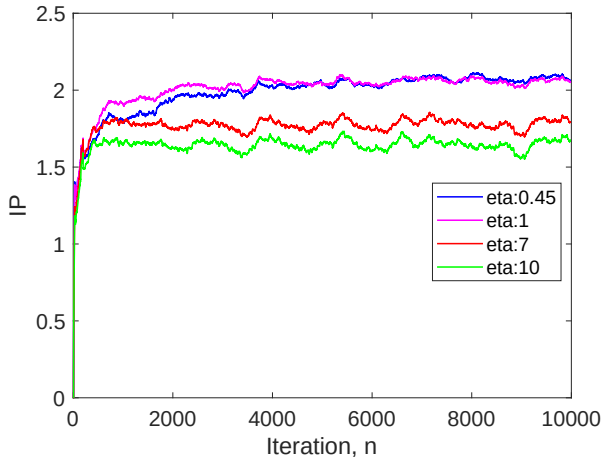


Fig. 6.5: IP using Bi-modal density Noise.

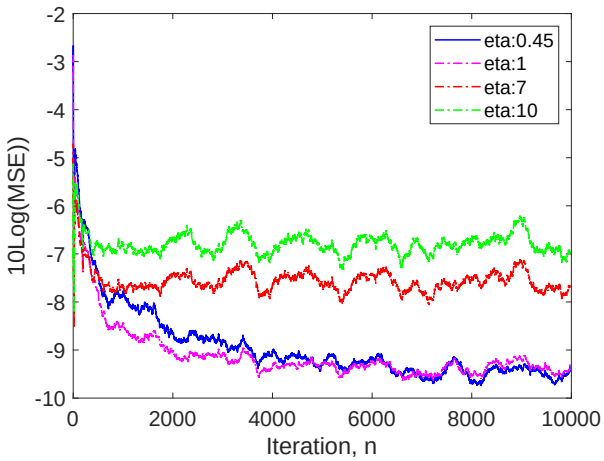
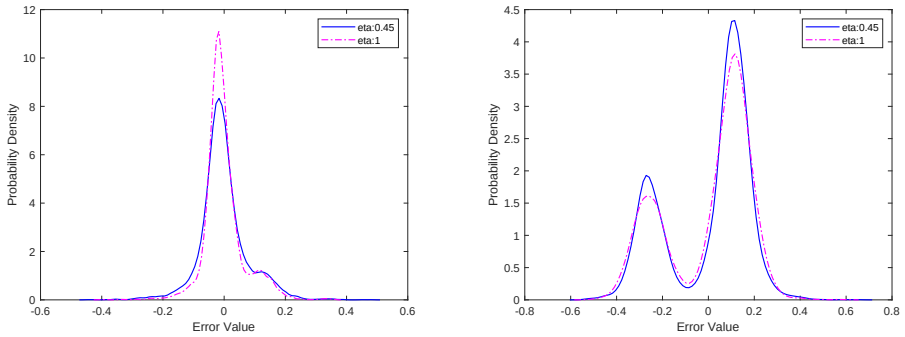
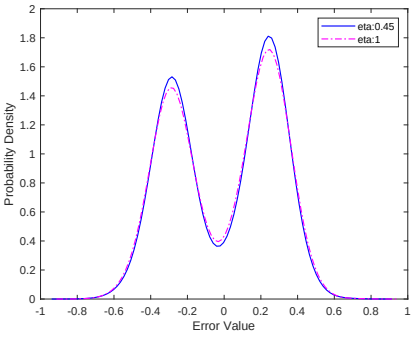


Fig. 6.6: Mean Square Errors vs. adaptation step size using Bi-modal density Noise.

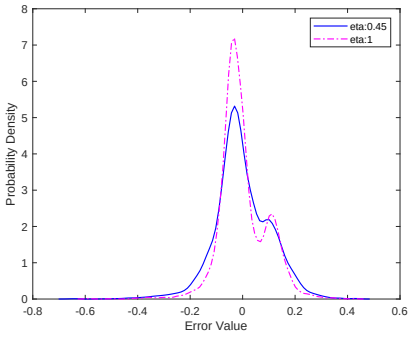


(a) Real component.

(b) imaginary i-component.



(c) imaginary j-component.



(d) imaginary k-component.

Fig. 6.7: Probability Density of Error Signal using Bi-modal density Noise.

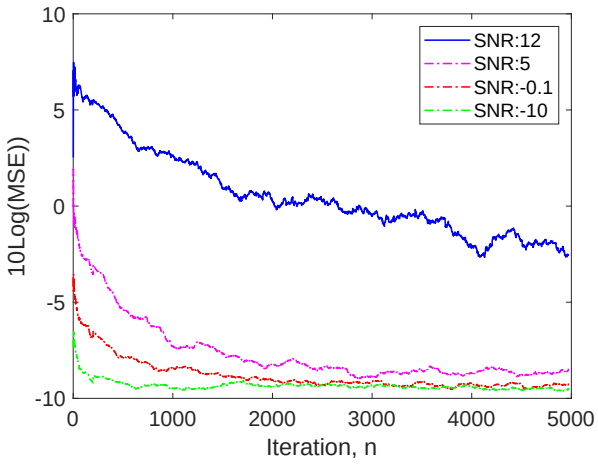


Fig. 6.8: Mean Square Errors vs. SNR using Bi-modal density Noise.

6.3.2 Four Dimensional Saito Chaotic Circuit

In this experiment we predict the 4 dimensional Saito chaotic circuit time series with one-step-ahead prediction. The process is given by

$$\begin{bmatrix} \frac{\partial x_1}{\partial \tau} \\ \frac{\partial y_1}{\partial \tau} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_1 & -\alpha_1\beta_1 \end{bmatrix} \begin{bmatrix} x_1 - \gamma\rho_1 h(z) \\ y_1 - \gamma\frac{\rho_1}{\beta_1} h(z) \end{bmatrix} \tag{6.46}$$

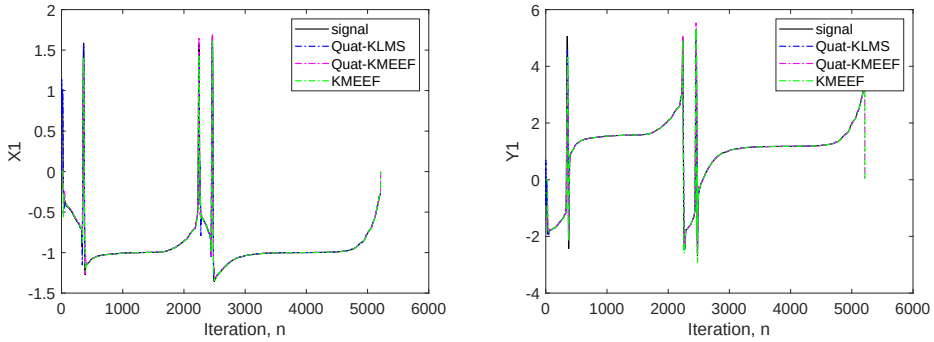
$$\begin{bmatrix} \frac{\partial x_2}{\partial \tau} \\ \frac{\partial y_2}{\partial \tau} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_2 & -\alpha_2\beta_2 \end{bmatrix} \begin{bmatrix} x_2 - \gamma\rho_2 h(z) \\ y_2 - \gamma\frac{\rho_2}{\beta_2} h(z) \end{bmatrix} \tag{6.47}$$

where $h(z)$ is the normalized hysteresis value given by

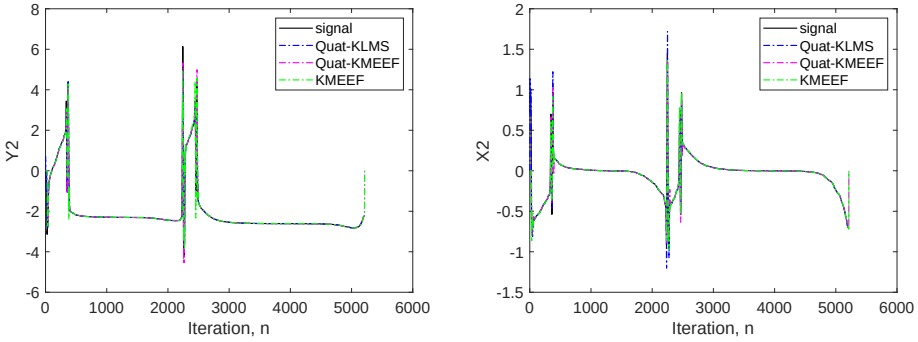
$$h(z) = \begin{cases} 1 & \text{if } z \geqslant 1 \\ -1 & \text{if } z \leqslant -1 \end{cases} \tag{6.48}$$

The parameter z, ρ_1 and ρ_2 are given as $z = x_1 + x_2$, $\rho_1 = \beta_1/1 - \beta_1$ and $\rho_2 = \beta_2/1 - \beta_2$ where $\alpha_1 = 7.5$, $\alpha_2 = 15$, $\beta_1 = 0.16$, $\beta_2 = 0.097$ and $\gamma = 1.3$ [24].

The parameters for the Quat-KMEEF were $\eta = 80$, $\bar{\sigma} = 0.07$, and $\sigma = 2.5$, and for the Quat-KLMS $\eta = 0.7$, $\bar{\sigma} = 0.07$ were used. Fig.6.9 shows the one-step prediction using Quat-KMEEF, Quat-KLMS and quadruple real-valued KMEEF algorithms. Fig.6.10 to Fig.6.13 show the dynamic transition region between iteration 2200 to 2500 of Fig.6.9 for real and imaginary components. As shown in Fig.6.10 to Fig.6.13 the Quat-KMEEF can predict and track the heavy dynamic transitions of the actual signal much closer than the Quat-KLMS and quadruple real-valued KMEEF.



(a) Real component. (b) imaginary i-component.



(c) imaginary j-component. (d) imaginary k-component.

Fig. 6.9: Four Dimensional Saito Chaotic Circuit Time Series Iteration 0 to 5000.

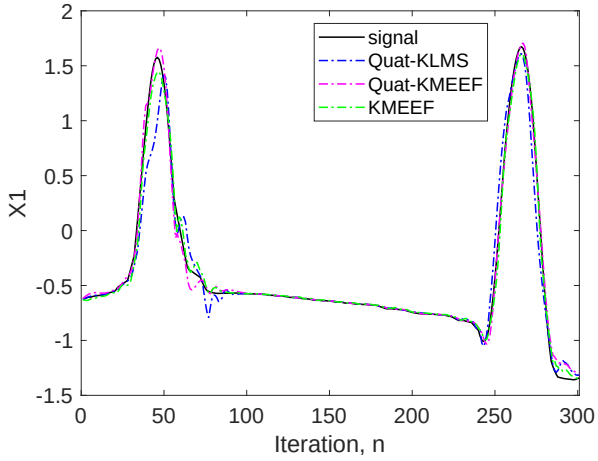


Fig. 6.10: Real component of Four Dimensional Saito Chaotic Circuit Time Series zoom in Iteration 2200 to 2500.

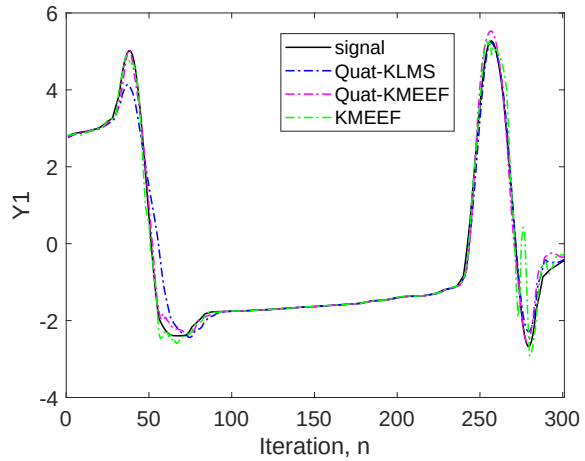


Fig. 6.11: Imaginary i-component of Four Dimensional Saito Chaotic Circuit Time Series zoom in Iteration 2200 to 2500.

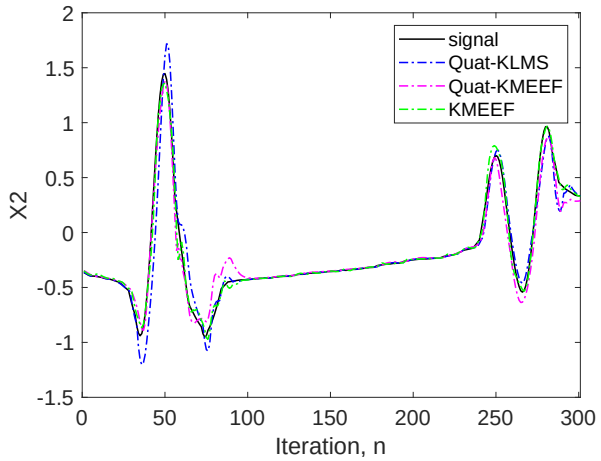


Fig. 6.12: Imaginary j-component of Four Dimensional Saito Chaotic Circuit Time Series zoom in Iteration 2200 to 2500.

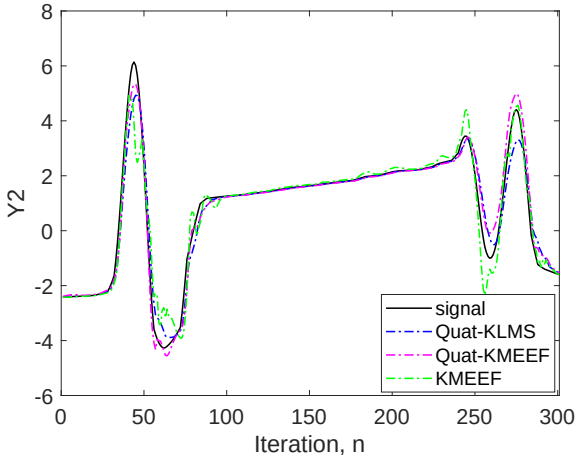
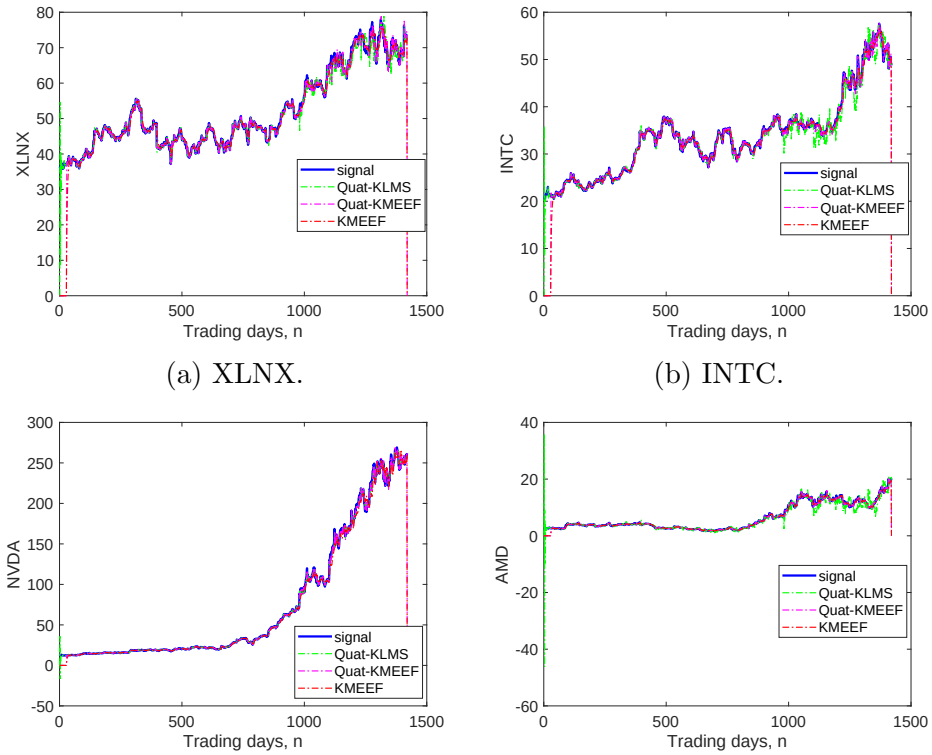


Fig. 6.13: Imaginary k-component of Four Dimensional Saito Chaotic Circuit Time Series zoom in Iteration 2200 to 2500.

6.3.3 Stock Market Prediction

In this experiment four stocks fused into quaternionic model which are XLNX, INTC, NVDA and AMD. The stock market prices obtained from "https://finance.yahoo.com" under Quotes "Historical Data" tabs for the period of Dec 9, 2012 to Dec9, 2018 with Daily frequency. We used one step ahead prediction model to predict the "High" price of each index at each trading day. To form a quaternion input, the four stocks are fused in a quaternion form as $q = XLNX + iINTC + jNVDA + kAMD$. Fig.6.14 shows the simulation results of one step ahead prediction using Quat-KMEEF, Quat-KLMS and quadruple KMEEF. The parameters for the Quat-KMEEF were $\eta = 150$, $\bar{\sigma} = 223$, and $\sigma = 54.7$, for the quadruple real-valued KMEEF were $\eta = 37.5$, $\bar{\sigma} = 223$, and $\sigma = 54.7$ and for the Quat-KLMS $\eta = 0.7$, $\bar{\sigma} = 223$ were used. To visualize each algorithm prediction, we plot the trading days from 1000 to 1200. The zoom area could be any arbitrary interval. Fig.6.15 to Fig.6.18 show XLNX, INTC, NVDA, AMD stocks "High" respectively. As shown in Fig.6.15 to Fig.6.18, the Quat-KMEEF follows the dynamic of the market (rise or decline of stock) much better than the other two algorithms especially

during the high fluctuations times. The prediction gains in (dB) of all algorithms are shown in Table 6.1. The prediction gain is defined as $R_p = 10\log(\sigma_y^2/\sigma_e^2)$ where σ_y^2 and σ_e^2 are the estimated variances of the input and the error respectively. The prediction gain of Quat-KMEEF dominates the other two algorithms Quat-KLMS and quadruple KMEEF. Therefore the Quat-KMEEF algorithm performs better if there are coupling or correlation within the components of quaternion 4-Dimensional input versus the quadruple real-valued KMEEF algorithm.



(a) XLNX. (b) INTC.

(c) NVDA. (d) AMD.

Fig. 6.14: Stocks "High" Trading days Dec9 2012 to Dec9 2018.

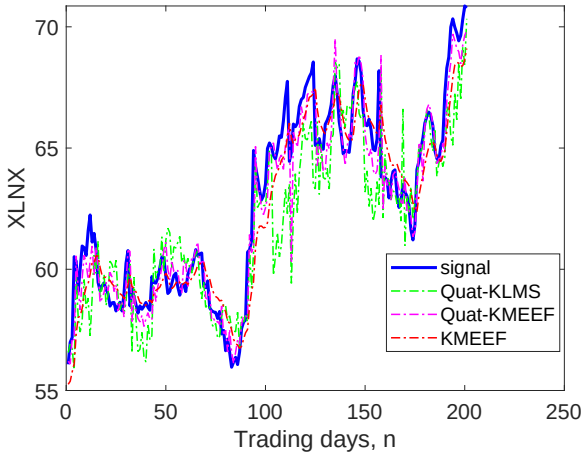


Fig. 6.15: XLNX "High" Trading days 1000 to 1200

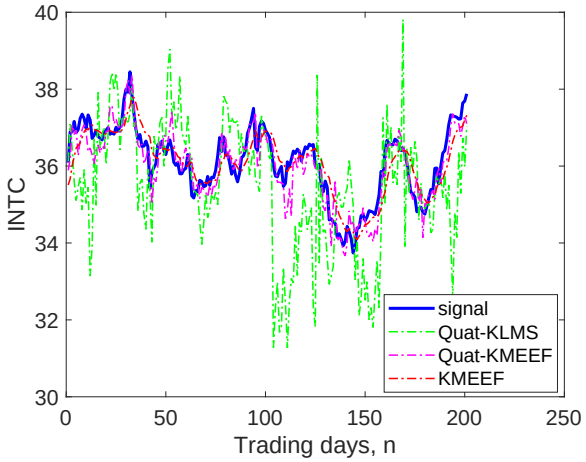


Fig. 6.16: INTC "High" Trading days 1000 to 1200

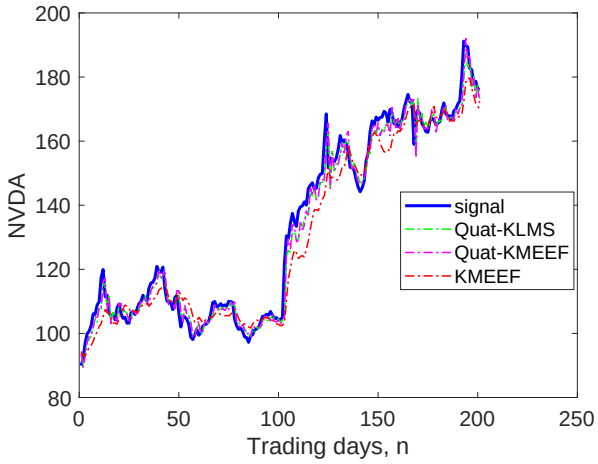


Fig. 6.17: NVDA "High" Trading days 1000 to 1200

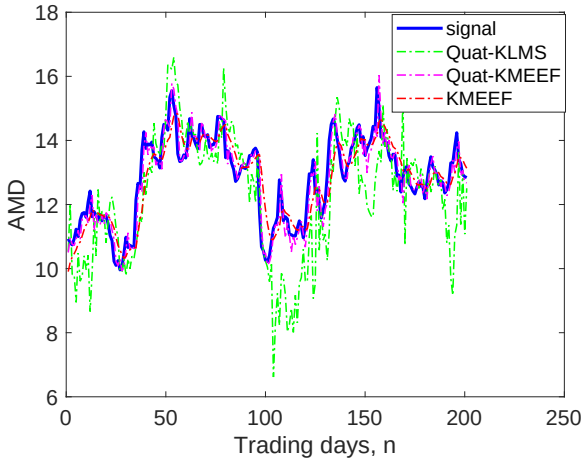


Fig. 6.18: AMD "High" Trading days 1000 to 1200

Table 6.1: Prediction Gain (dB)

	Quat-KMEEF	**KMEEF**	**Quat-KLMS**
XLNX	19.32	16.52	17.02
INTC	20.79	18.06	15.9
NVDA	30.44	25.88	29.52
AMD	27.05	23.75	10.09

6.4 Conclusion

We have shown the derivation and convergence analysis of a quaternion kernel adaptive algorithm (Quat-KMEEF) based on minimum error entropy with fiducial point [17]. The algorithm is based on information theoretic learning (ITL) cost function. The gradient is derived based on quaternion RKHS using GHR calculus. Algorithm using minimum error entropy may converge to set of optimal weights without having zero mean error. Traditionally, to make the mean of output error equal to zero, the output during testing session was biased with the mean of errors of training session. However, for non-symmetric or heavy tails error PDF the estimation of error mean is problematic. Using augmented maximum information potential cost function with fiducial point enable online learning without biasing the output during testing session. Simulation results are used to show the behavior of the new algorithm (QKMEEF) when signal is non-Gaussian in presence of unimodal noise versus bi-modal noise distributions. It shows that in case of unimodal noise distribution, minimizing error entropy (maximizing information potential) results much lower error variance (MSE) versus the bi-modal case, which confirms the theory that in unimodal Gaussian distribution case the MSE could be bounded in terms of the entropy power. Simulation results show that the Quat-KMEEF can track and predict the 4-Dimensional non stationary signals where there are correlations between components better than quadruple real-valued KMEEF and Quat-KLMS algorithms.

Chapter 7

Quaternion Stochastic Information Gradient Algorithm for Nonlinear Adaptive Systems

In this chapter, we develop a kernel adaptive filter for quaternion data, using stochastic information gradient (SIG) cost function based on the information theoretic learning (ITL) approach. The new algorithm (QKSIG) is useful for quaternion-based kernel applications of nonlinear filtering. Adaptive filtering in quaterion domain intrinsically incorporates component-wise real valued cross-correlation or the coupling within the dimensions of the quaternion input. We apply generalized Hamilton-real (GHR) calculus that is applicable to quaternion Hilbert space for evaluating the cost function gradient. The QKSIG algorithm minimizes Shannon's entropy of the error between the filter output and desired response and minimizes the divergence between the joint densities of input-desired and input-output pairs. The SIG technique reduces the computational complexity of the error entropy estimation. Here, ITL with SIG approach is applied to quaternion adaptive filtering for three different reasons. First, it reduces the algorithm computational complexity compared to our previous work quaternion kernel minimum error entropy algorithm (QKMEE). Second, it improves the filtering performance by considering the coupling within the dimensions of the quaternion input. Third, it performs better in biased or non-Gaussian signal and noise environments due to ITL approach. We present convergence analysis and steady-state performance analysis results of the new algorithm (QKSIG). Simulation results are used to show the behavior of the new

algorithm QKSIG in quaternion non-Gaussian signal and noise environments compared to the existing ones such as quadruple real-valued kernel stochastic information gradient (KSIG) and quaternion kernel LMS (QKLMS) algorithms.

To illustrate the relations of this work (QKSIG) to the previous ones (LMS, KLMS, QKLMS, MEE, KMEE, QKMEE), we summarized the algorithms characteristics in Table.7.1 based on their applications conditions. These conditions could be signal and noise environments and domains such as Gaussian or non-Gaussian, real or quaternion and linear or non-linear environments.

Table 7.1: Algorithms comparison based on their applications conditions

	criterion	**signal/noise**	**domain**	**linear/non-linear**
LMS	MSE	Gaussian	real	linear
KLMS	MSE	Gaussian	real	non-linear
QKLMS	MSE	Gaussian	quaternion	non-linear
MEE	ITL	non-Gaussian	real	linear
KMEE	ITL	non-Gaussian	real	non-linear
QKMEE	ITL	non-Gaussian	quaternion	non-linear
QKSIG	ITL	non-Gaussian	quaternion	non-linear

The QKSIG algorithm leverages the ITL and SIG properties in three different ways. First, it reduces the algorithm computational complexity compared to our previous work QKMEE due to SIG. Second, it improves the filtering performance by considering the coupling within the dimensions of the quaternion input due to quaternion augmented statistics compared to quadruple real-valued inputs. Third, it performs better in biased or non-Gaussian signal and noise environments due to ITL approach compared to MSE criteria.

7.1 Shannon's Entropy and Parzen Window

Shannon's entropy [32], for a quaternion random variable e can be defined as

$$H_S(e) = -\int_{\mathbb{H}} p_e(\varepsilon) \log p_e(\varepsilon) d\varepsilon = E_{p_e}[-\log p_e] \tag{7.1}$$

where p_e is probability distribution function (PDF) of quaternion random variable e defined as

$$\int_{\mathbb{H}} p_e(\varepsilon) d\varepsilon = 1, \quad \forall \varepsilon \in \mathbb{H} \quad p_e(\varepsilon) \geqslant 0.$$

In our previous work (QKMEE) [13], we proposed quaternion minimum error entropy algorithm based on the *order–2 Renyi's entropy* function. In practice the entropy function is not accessible since it is a function of the PDF of relative random variable e. The entropy can be estimated by using some specific method such as the *Parzen window*. Parzen window approximates the unknown PDF of the underlying samples of a random variable.

For a set of N statistically independent random samples $\{e_i\}_{i=1}^{N}$ of quaternion random variable e, the *Parzen window* [47] computes the estimate of the PDF p_e as

$$\hat{p_e}(\varepsilon) = \frac{1}{N} \sum_{l=1}^{N} \kappa_\sigma(\varepsilon - e_l) \tag{7.2}$$

where κ_σ is quaternion-extended Gaussian kernel [25].

The κ_σ can be defined as

$$\begin{aligned}\kappa_\sigma(x-y) =& \frac{4}{\sqrt{2\pi}\sigma}\exp\left[\frac{-1}{2\sigma^2}\Big((x_r-y_r)^2+(x_i-y_i)^2\right.\\ &\left.+(x_j-y_j)^2+(x_k-y_k)^2\Big)\right]\\ =& \frac{4}{\sqrt{2\pi}\sigma}\exp\left[\frac{-1}{2\sigma^2}|x-y|^2\right]\end{aligned}$$

where x and y are quaternion numbers $\in \mathbb{H}$ in forms of $x = x_r + ix_i + jx_j + kx_k$ and $y = y_r + iy_i + jy_j + ky_k$ [25].

The stochastic approximation of (7.1) can be done by dropping the expectation and evaluating its argument at the most recent sample of random variable e [48]. Therefore, the stochastic approximation of (7.1) at time n can be written as

$$\hat{H}_{S,n}(e) = -\log\left[\frac{1}{N}\sum_{l=1}^{N}\kappa_\sigma\Big(e(n)-e(n-l)\Big)\right]. \tag{7.3}$$

Training algorithm using error entropy will converge to a set of optimal weights. These optimal weights may not yield zero-mean error, since entropy does not change with the mean of the error distribution [3]. This can be fixed by biasing the system output to the desired signal to make the error mean equal to zero. However, for non-symmetric or heavy tails error PDF, estimation of error mean is problematic. In supervised learning the goal is to make most of the errors equal to zero. We can construct naturally an augmented criterion so that it minimizes the error entropy while locating the peak of the error PDF at the origin [18]. This can be done by minimizing the function (7.3) with respect to a fiducial point $e(n-N-1) = 0$, where $\{e(i)\}_{i=0}^{n}$ are the errors produced by adaptive filter [18]. Minimizing function (7.3) with respect to a fiducial point is the

same as minimizing augmented cost function $J_n(e)$ (7.4) in quaternion domain which is constructed by adding a fiducial point $e(n-N-1)=0$ to the equation (7.3). The augmented cost function $J_n(e)$ can be defined as

$$J_n(e) = -\log\left[\frac{1}{(N+1)}\left(\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))\right)\right] \tag{7.4}$$

where $\forall l : 0 \leqslant l \leqslant N \quad e(n-l) \in \mathbb{H}$.

Based on the stochastic approximation cost function in quaternion domain (7.4), we develop *Quaternion Kernel Stochastic Information Gradient Algorithm* in the next section.

7.2 Quaternion Stochastic Information Gradient Algorithm Derivation

For the quaternion kernel stochastic information gradient adaptive algorithm, the goal is to minimize the cost function $J_n(e)$ (7.4) with respect to free parameter $\mathbf{w}_n$ as

$$\begin{aligned} \min_{\forall \mathbf{w}_n \in \mathcal{H}} \quad & J_n(e) \\ \text{s.t.} \quad & e(n) = d(n) - y_n \\ & y_n =< \Phi(\mathbf{u}_n), \mathbf{w}_n >= \mathbf{w}_n^H \varphi_n \end{aligned} \tag{7.5}$$

where $d(n)$ is desired signal, $\mathbf{u}_n$ input vector and $\varphi_n = \Phi(\mathbf{u}_n)$. The $\Phi(.)$ is the kernel map to a QRKHS [25] defined as

$$\begin{aligned}\Phi(\mathbf{u}) =& \Phi(\mathbf{u}_r + i\mathbf{u}_i + j\mathbf{u}_j + k\mathbf{u}_k) \\ &= \phi(\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k) + i\phi(\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k) \\ &+ j\phi(\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k) + k\phi(\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k)\end{aligned}$$

where ϕ is the feature map of real kernel κ defined as

$$\phi(\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k) = \kappa_{\bar{\sigma}}(., (\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k))$$

and the $\kappa_{\bar{\sigma}}$ for two quaternion vectors $\mathbf{u}$ and $\mathbf{u}'$ of length d expressed as

$$\begin{aligned}&\kappa_{\bar{\sigma}}((\mathbf{u}_r, \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k), (\mathbf{u'}_r, \mathbf{u'}_i, \mathbf{u'}_j, \mathbf{u'}_k)) = \bar{\kappa}_{\bar{\sigma}}(\mathbf{u}, \mathbf{u}') \\ &= \exp\left[-\sigma^{-2}\sum_{l=1}^{d}\left|u_{r,l} - u'_{r,l}\right|^2 + \left|u_{i,l} - u'_{i,l}\right|^2\right. \\ &\left. + \left|u_{j,l} - u'_{j,l}\right|^2 + \left|u_{k,l} - u'_{k,l}\right|^2\right].\end{aligned}$$

Minimizing the cost function $J_n(e)$ (7.4) can be done by using unconstrained optimization algorithms such as gradient descent with update rule as [11]

$$\begin{aligned}\mathbf{w}_{n+1} =& \mathbf{w}_n - \eta\nabla_{\mathbf{w}_n^*} J_n(e) \\ =& \mathbf{w}_n - \eta\left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n}\right)^H\end{aligned} \tag{7.6}$$

where H is Hermitian transpose (conjugate transpose) and η is adaptation step size.

The gradient of the cost function $J_n(e)$ (7.4) can be calculated based on the following equation (the full derivation of gradient using GHR calculus can be found in Appendix A)

$$\begin{aligned}
\nabla_{\mathbf{w}_n^*} J_n(e) &= \left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right)^H \\
&= \left(\frac{-1}{4\sigma^2} \right) \times \Bigg[\Bigg(\kappa_\sigma \Big(e(n) \Big) e(n) \varphi_n^H \\
&+ \sum_{l=1}^{N} \Big[\kappa_\sigma (e(n) - e(n-l)) \Big] \\
&\times \Big[e(n) - e(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \Bigg) \\
&\times \frac{1}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N} \kappa_\sigma(e(n) - e(n-l))} \Bigg]^H
\end{aligned} \tag{7.7}$$

where $e(n-l) = d(n-l) - \mathbf{w}_n^H \varphi_{n-l}$ are *a posteriori errors* for all $l : 1 \leqslant l \leqslant N$.

By setting $\mathbf{w}_0 = 0$ in equation (7.6), we can obtain filter output weight at time n as

$$
\begin{aligned}
\mathbf{w}_n = (\eta/4\sigma^2) \sum_{p=0}^{n-1} \Bigg[\Bigg(& \kappa_\sigma\Big(e(p)\Big) e(p) \varphi_p^H \\
& + \sum_{l=1}^{N} \Big[\kappa_\sigma\Big(e(p) - e(p-l)\Big) \Big] \\
& \times \Big[e(p) - e(p-l) \Big] \Big[\varphi_p^H - \varphi_{p-l}^H \Big] \Bigg) \\
& \times \frac{1}{\kappa_\sigma(e(p)) + \sum_{l=1}^{N} \kappa_\sigma(e(p) - e(p-l))} \Bigg]^H .
\end{aligned}
\tag{7.8}
$$

The weight update equation (7.8) includes the products of errors with Hermitian transpose of the input vectors φ_{p-l} in QRKHS for $\forall l : 1 \leqslant l \leqslant N$ as

$$
\begin{aligned}
e(p-l)\varphi_{p-l}^H &= \Big[d(p-l) - \mathbf{w}_{p-l}^H \varphi_{p-l} \Big] \varphi_{p-l}^H \\
&= d(p-l)\varphi_{p-l}^H - \mathbf{w}_{p-l}^H \varphi_{p-l} \varphi_{p-l}^H \\
&= d(p-l)\Phi(\mathbf{u}_{p-l})^H - \mathbf{w}_{p-l}^H \Phi(\mathbf{u}_{p-l}) \Phi(\mathbf{u}_{p-l})^H .
\end{aligned}
\tag{7.9}
$$

The covariance term $\Phi(\mathbf{u}_{p-l})\Phi(\mathbf{u}_{p-l})^H$ in (7.9) indicates that the augmented statistics of quaternion input vector is inherent to the QKSIG algorithm.

By substituting the weight update in the equation $y_n = \mathbf{w}_n^H \varphi_n$ and using properties of QRKHS and kernel trick to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, we can simplify the equation in kernel form as

$$
\begin{aligned}
y_n = \zeta \sum_{p=0}^{n-1} \Bigg[\Bigg(& \kappa_\sigma\Big(e(p)\Big) e(p) \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) \\
& + \sum_{l=1}^{N} \Big[\kappa_\sigma(e(p) - e(p-l)) \Big] \\
& \times \Big[e(p) - e(p-l) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) \Big] \Bigg) \\
& \times \frac{1}{\kappa_\sigma(e(p)) + \sum_{l=1}^{N} \kappa_\sigma(e(p) - e(p-l))} \Bigg]
\end{aligned}
\tag{7.10}
$$

where $\zeta = \eta/\sigma^2$.

Based on equation (7.10), the pseudo code for QKSIG could be summarized in Algorithm 1 table.

Algorithm 2 QKSIG Algorithm

Input: signal and desired $\{(\mathbf{u_i}, d(i))\}_{i=1}^{\infty}$
Output: Estimate desired output $\hat{d(n)} = y_n$ at time n, Residual $e(n)$
1: **Initialization** The kernel parameters $\bar{\sigma}$,σ and $\zeta = \eta/\sigma^2$
2: **while** $(\mathbf{u_n}, d(n))$, available **do**
3: {calculate filter output at iteration n}

$$
\begin{aligned}
y_n = \zeta \sum_{p=0}^{n-1} \Bigg[\Bigg(& \kappa_\sigma\Big(e(p)\Big) e(p) \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) \\
& + \sum_{l=1}^{N} \Big[\kappa_\sigma(e(p) - e(p-l)) \Big] \\
& \times \Big[e(p) - e(p-l) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) \Big] \Bigg) \\
& \times \frac{1}{\kappa_\sigma(e(p)) + \sum_{l=1}^{N} \kappa_\sigma(e(p) - e(p-l))} \Bigg]
\end{aligned}
$$

4: $e(n) = d(n) - y_n$ {calculate error at iteration n}
5: $n = n + 1$
6: **end while**

7.3 Computational Complexity

Compared with the existing quaternion minimum error entropy algorithm QKMEE [13], the proposed QKSIG algorithm requires less computational cost. In the newly proposed QKSIG algorithm, the cost function uses linear combination of the kernel of error samples. In the previous work, QKMEE [13], the cost function used was the quadratic combination of the kernel of the error samples. To calculate the computational complexity of the proposed QKSIG algorithm, we need to calculate total number of real additions and real multiplications of the filter output equation (7.10). For this purpose, we suppose the input vector $\mathbf{u_n}$ has the length equal to L. The kernel evaluation $\bar{\kappa}_{\bar{\sigma}}$ [25] requires $4L$ real multiplications and $8L-1$ real additions.

Table 7.2: The number of real-valued operations for the QKMEE and QKSIG algorithms, for input length equal to L

	real multiplications	**real additions**
QKMEE	$n(N+1)^2(8L+9)$	$n(N+1)^2(16L+14)$
QKSIG	$n\Big[(N+1)(8L+10)+4\Big]$	$n(N+1)(16L+22)$

For a fair comparison in terms of computational complexity, we consider the computation cost of real multiplications. As one can observe from Table 7.2, the QKSIG and QKMEE algorithms have $\mathcal{O}(nNL)$ and $\mathcal{O}(nN^2L)$ computational complexity respectively.

7.4 Convergence Analysis

The goal of the convergence analysis is to find a range for learning step size η in equation

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \nabla_{\mathbf{w}_n^*} J_n(e) \tag{7.11}$$

where the QKSIG algorithm converges to optimal set of weights.

To study the convergence of QKSIG algorithm, we consider an approach using the energy conservation relation [49]. The weight error at iteration $n+1$ can be defined as

$$\begin{aligned}
\mathbf{v}_{n+1} &= \mathbf{w}^0 - \mathbf{w}_{n+1} \\
&= \mathbf{w}^0 - (\mathbf{w}_n - \Delta\mathbf{w}_n) \\
&= \mathbf{v}_n + \Delta\mathbf{w}_n \\
&= \mathbf{v}_n + \frac{\eta}{4\sigma^2}\nabla_{\mathbf{w}_n^*} J_n(e) \\
&= \mathbf{v}_n - \gamma(n)\left[\kappa_\sigma\Big(e(n)\Big)e(n)\varphi_n^H + \mathbf{\Lambda}(n){\mathbf{\Psi_d}}^H(n)\right]^H
\end{aligned} \tag{7.12}$$

where $\mathbf{w}^0$ is the optimal weight and

$$\mathbf{\Psi_d}(n) = [\varphi_n - \varphi_{n-1}, ..., \varphi_n - \varphi_{n-N}],$$

$$\mathbf{\Lambda}(n) = \Big[\kappa_\sigma(e(n)-e(n-1))(e(n)-e(n-1)), ..., \kappa_\sigma(e(n)-e(n-N))(e(n)-e(n-N))\Big]$$

and

$$\gamma(n) = \tfrac{\eta}{4\sigma^2} \times \tfrac{1}{\kappa_\sigma(e(n))+\sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))}.$$

For checking energy conservation, we initially find *a priori* and *a posteriori* errors: $e_n^a = \mathbf{v}_n^H\varphi_n$ and $e_n^p = \mathbf{v}_{n+1}^H\varphi_n$ defined as

$$
\begin{aligned}
e_n^p &= \mathbf{v}_{n+1}^H \varphi_n \\
&= e_n^a - \frac{\eta}{4\sigma^2} \\
&\times \Bigg(\kappa_\sigma \Big(e(n) \Big) e(n) \varphi_n^H \varphi_n \\
&+ \sum_{l=1}^{N} \Big[\kappa_\sigma \Big(e(n) - e(n-l) \Big) \Big] \\
&\times \Big[e(n) - e(n-l) \Big] \Big[\varphi_n^H \varphi_n - \varphi_{n-l}^H \varphi_n \Big] \Bigg) \\
&\times \frac{1}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N} \kappa_\sigma(e(n) - e(n-l))}.
\end{aligned}
\tag{7.13}
$$

Based on energy conservation relation, both sides of equation (7.12) should have same energy, thus the energy can be expressed as

$$
\|\mathbf{v}_{n+1}\|_{\mathbb{F}}^2 = \left\| \mathbf{v}_n - \gamma(n) \Big[\kappa_\sigma \Big(e(n) \Big) e(n) \varphi_n^H + \mathbf{\Lambda}(n) {\mathbf{\Psi_d}}^H(n) \Big] \right\|_{\mathbb{F}}^2 \tag{7.14}
$$

where $\|.\|_{\mathbb{F}}^2$ is the norm in QRKHS defined as $\|\mathbf{v}\|_{\mathbb{F}}^2 =< \mathbf{v}, \mathbf{v} >= \mathbf{v}^H \mathbf{v}$.

By expanding equation (7.14) we can write

$$
\begin{aligned}
&\mathbf{v}_{n+1}^H \mathbf{v}_{n+1} \\
&= \left(\mathbf{v}_n^H - \gamma(n) \left[\kappa_\sigma \left(e(n) \right) e(n) \varphi_n^H + \mathbf{\Lambda}(n) {\mathbf{\Psi_d}}^H(n) \right] \right) \\
&\times \left(\mathbf{v}_n - \gamma(n) \left[\kappa_\sigma \left(e(n) \right) \varphi_n e^*(n) + \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right] \right) \\
&= \mathbf{v}_n^H \mathbf{v}_n \\
&- 2\gamma(n) \mathfrak{Re} \left(\kappa_\sigma \left(e(n) \right) \mathbf{v}_n^H \varphi_n e^*(n) + \mathbf{v}_n^H \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right) \\
&+ \gamma^2(n) \left(\kappa_\sigma^2 \left(e(n) \right) e(n) \varphi_n^H \varphi_n e(n)^* \right. \\
&+ 2\mathfrak{Re} \left(\kappa_\sigma \left(e(n) \right) e(n) \varphi_n^H \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right) \\
&\left. + \mathbf{\Lambda}(n) {\mathbf{\Psi_d}}^H(n) \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right).
\end{aligned}
\tag{7.15}
$$

By taking expectation of both sides of equation (7.15) we can obtain

$$
\begin{aligned}
&E \left[\| \mathbf{v}_{n+1} \|_{\mathbb{F}}^2 \right] = E \left[\| \mathbf{v}_n \|_{\mathbb{F}}^2 \right] \\
&- 2E \left[\gamma(n) \mathfrak{Re} \left(\kappa_\sigma \left(e(n) \right) \mathbf{v}_n^H \varphi_n e^*(n) + \mathbf{v}_n^H \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right) \right] \\
&+ E \left[\gamma^2(n) \kappa_\sigma^2 \left(e(n) \right) e(n) \varphi_n^H \varphi_n e(n)^* \right. \\
&+ 2\gamma^2(n) \mathfrak{Re} \left(\kappa_\sigma \left(e(n) \right) e(n) \varphi_n^H \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right) \\
&\left. + \gamma^2(n) \mathbf{\Lambda}(n) {\mathbf{\Psi_d}}^H(n) \mathbf{\Psi_d}(n) \mathbf{\Lambda}^H(n) \right].
\end{aligned}
\tag{7.16}
$$

For convergence, the energy of the weight error vector should gradually reduce per

iteration or

$$E\left[\begin{array}{c} \\ \|\mathbf{v}_{n+1}\|_{\mathbb{F}}^2 \\ \\ \end{array}\right] < E\left[\begin{array}{c} \\ \|\mathbf{v}_{n}\|_{\mathbb{F}}^2 \\ \\ \end{array}\right]. \tag{7.17}$$

Therefore, the algorithm converges if the following inequality could be satisfied

$$\begin{aligned} &- 2E\left[\gamma(n)\mathfrak{Re}\left(\kappa_\sigma\Big(e(n)\Big)\mathbf{v}_n^H\varphi_n e^*(n) + \mathbf{v}_n^H\mathbf{\Psi_d}(n)\mathbf{\Lambda}^H(n)\right)\right] \\ &+ E\Bigg[\gamma^2(n)\kappa_\sigma^2\Big(e(n)\Big)e(n)\varphi_n^H\varphi_n e(n)^* \\ &+ 2\gamma^2(n)\mathfrak{Re}\left(\kappa_\sigma\Big(e(n)\Big)e(n)\varphi_n^H\mathbf{\Psi_d}(n)\mathbf{\Lambda}^H(n)\right) \\ &+ \gamma^2(n)\mathbf{\Lambda}(n)\mathbf{\Psi_d}^H(n)\mathbf{\Psi_d}(n)\mathbf{\Lambda}^H(n)\Bigg] < 0. \end{aligned} \tag{7.18}$$

After algorithm convergence, $\mathbf{\Lambda}$ converged to $\mathbf{0}$. Thus, we can simplify (7.18) as

$$\begin{aligned} &\frac{-2\eta}{4\sigma^2}E\left[\frac{\kappa_\sigma\Big(e(n)\Big)|e_n^a|^2}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))}\right] \\ &+ \frac{\eta^2}{16\sigma^4}E\left[\frac{\kappa_\sigma^2\Big(e(n)\Big)|e(n)|^2\|\varphi_n\|_{\mathbb{F}}^2}{(\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l)))^2}\right] < 0 \end{aligned} \tag{7.19}$$

or

$$0 < \eta < 8\sigma^2(N+1)\frac{E\left[\,|e(n)|^2\,\right]}{E\left[|e(n)|^2\|\varphi_n\|_{\mathbb{F}}^2\right]}. \tag{7.20}$$

By assuming that $\|\varphi_n\|_{\mathbb{F}}^2$ is independent of $|e_n^a|^2$, we next obtain

$$0 < \eta < 8\sigma^2(N+1)\frac{E\left[\,|e(n)|^2\,\right]}{E\left[|e(n)|^2\right]E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}. \tag{7.21}$$

Thus, using kernel trick $\varphi_n^H\varphi_n = 4\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})$ and substituting it in (7.21) we can conclude that in order to guarantee the algorithm convergence the step size η should be

$$0 < \eta < \frac{8\sigma^2(N+1)}{4\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})}. \tag{7.22}$$

7.5 Steady-state Performance

To analyze the steady state performance of the QKSIG algorithm using the energy conservation, we employed the same method of analysis described in [49]. The estimation error of QKSIG is given by $e(n) = e_n^a + v$ where e_n^a is *a priori* defined in previous section as $e_n^a = \mathbf{v}_n^H\varphi_n$ and $e_n^p = \mathbf{v}_{n+1}^H\varphi_n$.

The steady-state MSE can be defined as follows

$$MSE = lim_{n\to\infty} E\left[|e(n)|^2 \right] = lim_{n\to\infty} E\left[|e_n^a|^2 \right] + {\sigma_v}^2$$

where $lim_{n\to\infty} E\left[|e_n^a|^2 \right]$ is the excess MSE (EMSE) resulting from a mismatch between the estimated value and true value of the system weight vector [49].

Based on the *a priori* and *a posteriori* errors relation described in previous section (7.13), at the the steady state ($n \to \infty$) we can obtain the following

$$\begin{aligned} e_n^p &= e_n^a - \frac{\eta}{4(N+1)^2\sigma^2} \times e(n)\|\varphi_n\|_{\mathbb{F}}^2 \\ &= e_n^a - \frac{\eta}{4(N+1)^2\sigma^2} \times (e_n^a + v)\|\varphi_n\|_{\mathbb{F}}^2 \\ &= e_n^a - \theta(e_n^a + v)\|\varphi_n\|_{\mathbb{F}}^2 \end{aligned} \tag{7.23}$$

where $\theta = \frac{\eta}{4(N+1)^2\sigma^2}$. Therefore, by evaluating the energies on both sides of (7.23) we can obtain

$$\begin{aligned} |e_n^p|^2 =& |e_n^a|^2 - \theta|e_n^a|^2\|\varphi_n\|_{\mathbb{F}}^2 - \theta e_n^a v^*\|\varphi_n\|_{\mathbb{F}}^2 \\ &- \theta|e_n^a|^2\|\varphi_n\|_{\mathbb{F}}^2 - \theta v e_n^{a*}\|\varphi_n\|_{\mathbb{F}}^2 \\ &+ \theta^2\left(|e_n^a|^2 + e_n^a v^* + v e_n^{a*} + \|\varphi_n\|_{\mathbb{F}}^2 \right)\|\varphi_n\|_{\mathbb{F}}^4 . \end{aligned} \tag{7.24}$$

Noting noise v is independent of φ_n. By taking expectation of both sides of (7.24), and

assuming $E\left[|e_n^p|^2\right] = E\left[|e_n^a|^2\right]$ at steady state ($n \to \infty$) we can obtain

$$\begin{aligned} 2E\left[|e_n^a|^2\right] &= \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2 |e_n^a|^2\right] + \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2 \|v\|_{\mathbb{H}}^2\right] \\ &= \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2 |e_n^a|^2\right] + \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right] E\left[\|v\|_{\mathbb{H}}^2\right] \\ &= \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2 |e_n^a|^2\right] + \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right] {\sigma_v}^2. \end{aligned} \tag{7.25}$$

By assuming that $\|\varphi_n\|_{\mathbb{F}}^2$ is independent of $|e_n^a|^2$, we next obtain

$$E\left[|e_n^a|^2\right] = \frac{\theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right] {\sigma_v}^2}{2 - \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}. \tag{7.26}$$

Therefore, the EMSE and MSE in this case can be calculated as

$$\begin{aligned} EMSE =& lim_{n\to\infty} E\left[|e_n^a|^2\right] \\ &= \left(\frac{\theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}{2 - \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}\right) {\sigma_v}^2 \\ &= \left(\frac{\theta E\left[4\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})\right]}{2 - \theta E\left[4\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})\right]}\right) {\sigma_v}^2 \\ &= \left(\frac{2\theta\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})}{1 - 2\theta\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n}, \mathbf{u_n})}\right) {\sigma_v}^2 \end{aligned} \tag{7.27}$$

$$
\begin{aligned}
MSE =& lim_{n\to\infty} E\left[|e_n^a|^2\right] + {\sigma_v}^2 \\
&= \left(\frac{2-\theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right] + \theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}{2-\theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}\right){\sigma_v}^2 \\
&= \left(\frac{2}{2-\theta E\left[\|\varphi_n\|_{\mathbb{F}}^2\right]}\right){\sigma_v}^2 \\
&= \left(\frac{2}{2-\theta E\left[4\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n},\mathbf{u_n})\right]}\right){\sigma_v}^2 \\
&= \left(\frac{1}{1-2\theta\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_n},\mathbf{u_n})}\right){\sigma_v}^2.
\end{aligned}
\tag{7.28}
$$

7.6 Simulation Results

7.6.1 Quaternion Nonlinear Channel Estimation

In these experiments, we show the behavior of QKSIG algorithm when signal is non-Gaussian in presence of symmetric unimodal and bi-modal noises in quaternion domain. Also, we present comparison between new QKSIG algorithm and the previous one QK-MEE in terms of performance, computational complexity and execution time.

7.6.1.1 Symmetric Unimodal Gaussian Noise

The QKSIG algorithm was simulated for a nonlinear channel with symmetric unimodal Gaussian noise. The channel consists of the quaternion filter, i.e.,

$$
\begin{aligned}
z(n) &= g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n) \\
&+ h_1^* u(n-1) + h_2^* u^i(n-1) + h_3^* u^j(n-1) + h_4^* u^k(n-1)
\end{aligned}
\tag{7.29}
$$

and non-linearity, i.e.,

$$
y(n) = z(n) + az^2(n) + bz^3(n) + v(n) \tag{7.30}
$$

where $v(n)$ is Symmetric unimodal density Gaussian noise described later. Coefficients $g1, ..., g4, h1, ..., h4, a, b$, and noise $v(n)$ are all quaternion valued. The coefficients used were [23]

$a = 0.075 + i0.35 + j0.1 - k0.05,$

$b = -0.025 - i0.25 - j0.05 + k0.03,$

$g1 = -0.40 + i0.30 + j0.15 - k0.45,$

$h1 = 0.175 - i0.025 + j0.1 + k0.15,$

$g2 = -0.35 - i0.15 - j0.05 + k0.20,$

$h2 = 0.15 - i0.225 + j0.125 - k0.075,$

$g3 = -0.10 - i0.40 + j0.20 - k0.05,$

$h3 = +0.025 + i0.075 - j0.05 - k0.05,$

$g4 = +0.35 + i0.10 - j0.10 - k0.15,$

$h4 = -0.05 - i0.075 - j0.075 + k0.175.$

For the tests, the input $u(n)$ was formed using impulsive Gaussian mixture models to form non-Gaussian signals as follows [23]:

$$\begin{aligned} p_u = &(0.85N(1.0, 0.01) + 0.15N(3.0, 0.01)) \\ &+i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01)) \\ &+j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01)) \\ &+k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01)) \end{aligned} \tag{7.31}$$

And noise $v(n)$ was formed using symmetric unimodal Gaussian distributions as:

$$\begin{aligned} p_v = &(N(0.0, 0.01)) \\ &+i(N(3.0, 0.01)) \\ &+j(N(1.0, 0.01)) \\ &+k(N(0.5, 0.01)) \end{aligned} \tag{7.32}$$

where $N(m_N, \sigma_N)$ denotes the normal (Gaussian) PDF with mean m_N and variance σ_N. The kernel parameters $\bar{\sigma} = 2.24$; $\sigma = 0.55$ for the Parzen window with size $N = 10$, are estimated using Silverman's rule $1.06 \times min\{\sigma_Y, R/1.34\} \times N^{1/5D}$ [50] where σ_Y is the data standard deviation, D is data dimension, R is the interquartile and N is the number of samples. The simulation results of the QKSIG algorithm for the nonlinear channel are shown in Fig.1 to Fig.4. The theoretical upper bound of adaptation step size could be approximated using inequality (7.22) as $\eta < \lceil \frac{8*0.55^2*(10+1)}{4} \rceil = \lceil 6.65 \rceil = 7 < 8$.

Fig.7.1 and Fig.7.2 show the convergence of the normalized entropy (NEntropy) and mean-squared-error (MSE) of the QKSIG algorithm with different convergence step size η respectively. The results are ensemble-averaged over 20 realizations with 60-samples

moving average window. As shown in Fig.7.2, the algorithm using step size $\eta = 0.1$ generates higher MSE compared to $\eta = 1$. Therefore, higher MSE yields higher variance in error probability distributions. The higher variance also can be seen in Fig.7.3 where all error probability distributions are uni-modal with higher variance with $\eta = 0.1$.

Fig.7.4 shows the impact of signal to noise ratio (SNR) on convergence misadjustment with step size $\eta = 1$. The results are ensemble-averaged over 20 realizations with 60-samples moving average window. As shown in Fig.7.4 by increasing the SNR the convergence misadjustment degrades.

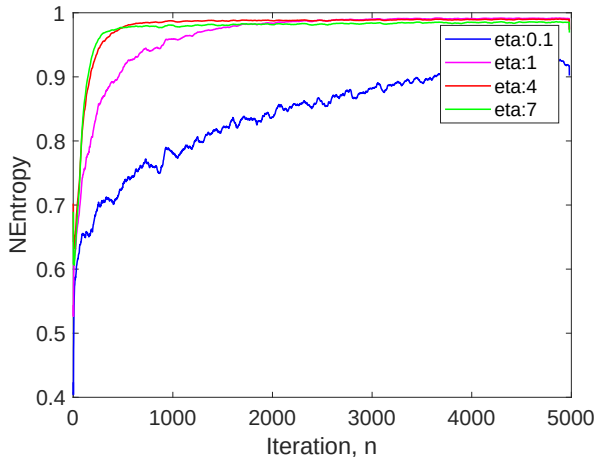


Fig. 7.1: Normalized Entropy using Symmetric Unimodal Gaussian Noise.

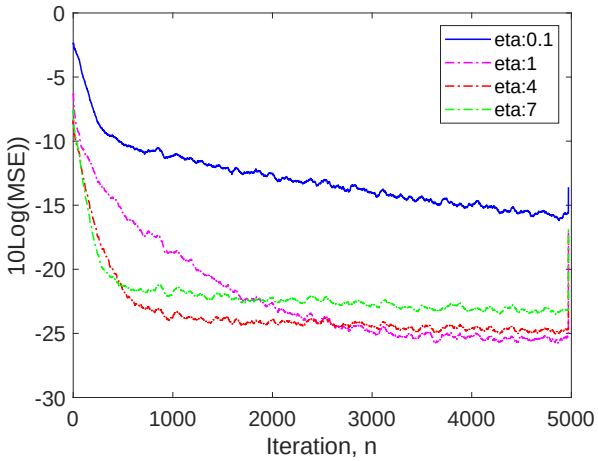
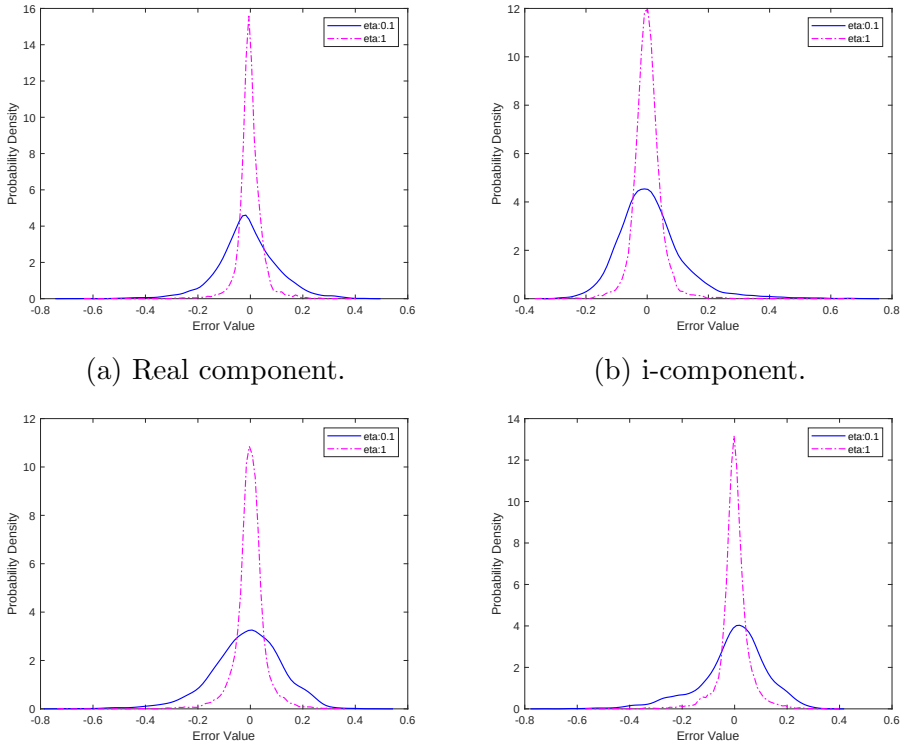


Fig. 7.2: Mean Squared Error vs. adaptation step size using Symmetric Unimodal Gaussian Noise.



(a) Real component.

(b) i-component.

(c) j-component.

(d) k-component.

Fig. 7.3: Probability Density of Error Signal using Symmetric Unimodal Gaussian Noise.

Table 7.3 shows the comparison between new QKSIG algorithm and the previous one

QKMEE in terms of performance, computational complexity and execution time. For this purpose, the first 4000 out of 5000 samples are used for training and another 1000 samples are used for testing session with different convergence step sizes. As one can observe, both QKSIG and QKMEE algorithms approximately have same performance based on testing mean-squared-errors (MSE). The QKSIG algorithm average running time for one iteration is almost 10 times faster than QKMEE algorithm which is consistent with computational complexity order with $N = 10$ and input vector size $L = 5$. As a result, QKSIG has same performance as QKMEE with lower computational complexity and average execution time.

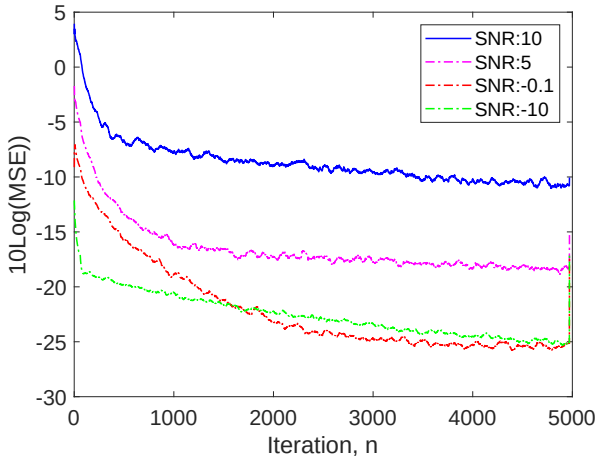


Fig. 7.4: Mean Squared Error vs. SNR using Symmetric Unimodal Density Noise

Table 7.3: Performance comparison of QKSIG and QKMEE with different step sizes in nonlinear channel Estimation

	Parameter	Testing MSE	Computation complexity	Average running time (one iteration) (ms)
QKSIG	$\eta = 0.1$	0.6898 ± 0.0278	$\mathcal{O}(nNL)$	36.398
	$\eta = 1$	0.0329 ± 0.0029		
	$\eta = 4$	0.0338 ± 0.0033		
	$\eta = 7$	0.0477 ± 0.005		
QKMEE	$\eta = 0.1$	0.6230 ± 0.0383	$\mathcal{O}(nN^2L)$	331.48
	$\eta = 1$	0.0332 ± 0.0029		
	$\eta = 4$	0.0340 ± 0.0033		
	$\eta = 7$	0.0469 ± 0.0049		

7.6.1.2 Bi-Modal Density Noise

In this experiment we used same test bench as previous section but instead of using symmetric unimodal density Gaussian noise, we applied Bi-modal density or Gaussian mixture distribution noise as below [23]:

$$\begin{aligned} p_v = &(0.90N(0.0, 0.01) + 0.10N(1.0, 0.01)) \\ &+i(0.70N(3.0, 0.01) + 0.30N(0.5, 0.01)) \\ &+j(0.45N(1.0, 0.01) + 0.55N(4.5, 0.01)) \\ &+k(0.80N(0.5, 0.01) + 0.20N(1.5, 0.01)) \end{aligned} \tag{7.33}$$

Fig.7.5 and Fig.7.6 show the convergence of the normalized entropy (NEntropy) and mean-squared-error (MSE) of the QKSIG algorithm with different convergence step size η respectively. The results are ensemble-averaged over 20 realizations with 60-samples moving window average. Fig.7.7 shows the algorithm adaptation error when algorithm converged. As shown in Fig.7.7, the algorithm error signal density is bi-modal with heavy tails. As shown in Fig.7.2 and Fig.7.6 when algorithm converges to minimum entropy, the unimodal case has better mean-squared-error $MSE = -25$dB compared with bi-modal case which has $MSE = -10$dB.

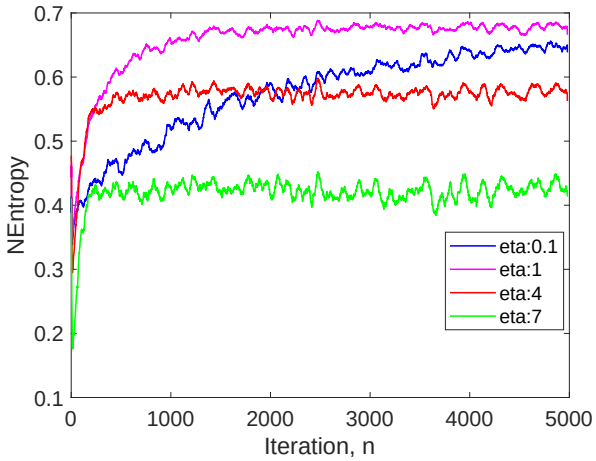


Fig. 7.5: Normalized Entropy using Bi-modal density Noise.

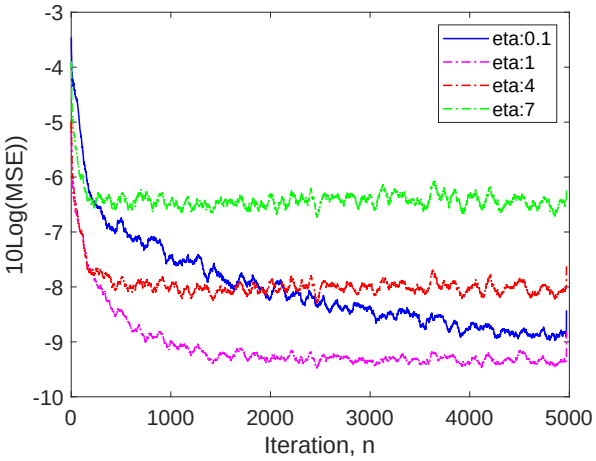
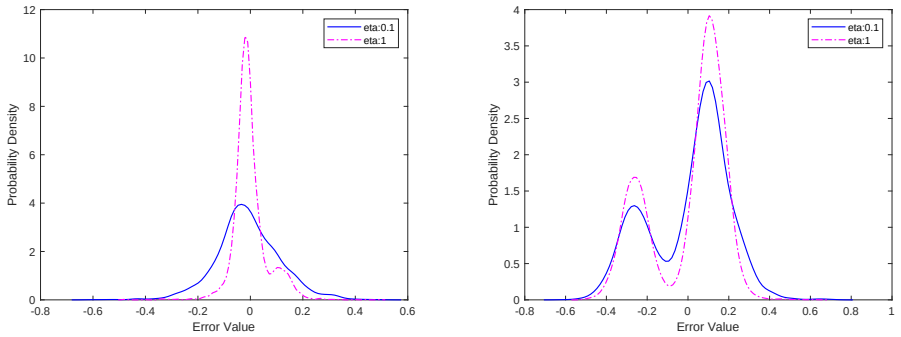
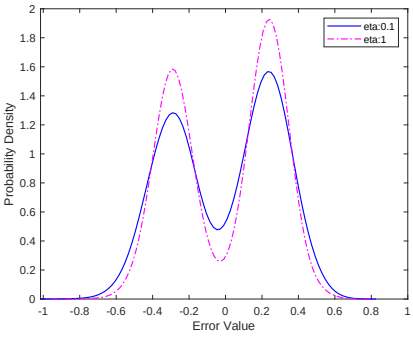


Fig. 7.6: Mean Squared Error vs. adaptation step size using Bi-modal density Noise.

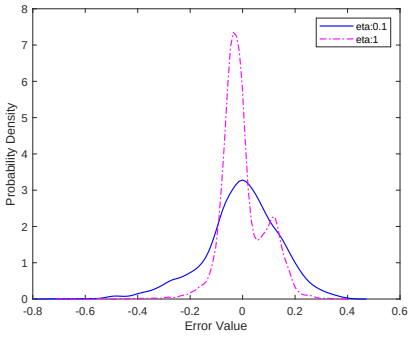


(a) Real component.

(b) i-component.



(c) j-component.



(d) k-component.

Fig. 7.7: Probability Density of Error Signal using Bi-modal density Noise.

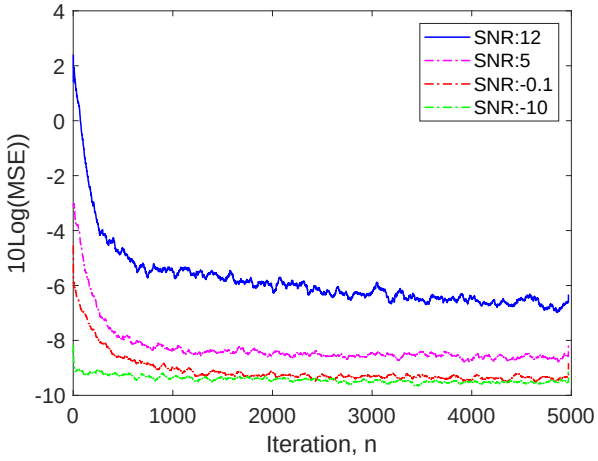


Fig. 7.8: Mean Squared Error vs. SNR using Bi-modal density Noise.

7.6.2 Adaptive Line Enhancement

In this experiment, we show the advantage of using ITL over MSE criterion in non-Gaussian environment such as real-world electroencephalogram (EEG) data in quaternion domain. For this purpose, one-step ahead linear prediction is used to enhance a real-world electroencephalogram (EEG) data, corrupted with 50Hz power-line interference. The data was recorded from 8 electrodes for 30 seconds and sampled at 256Hz according to the 10-20 system. The electrode positions used were $Fp1, Fp2, C3, C4, O1, O2, vEOG$, $hEOG$. The data was collected for raising the eyebrows. The signals were grouped based on pattern similarity (i.e., $Fp1$ and $Fp2$ are symmetrically located, signals $C3$ and $C4$, etc.), and eight signals were considered a tuple of quaternion inputs.
The eight input signals are divided into two consecutive quaternion inputs: $(Fp1 + iC3 + jO1 + kvEOG)$ and $(Fp2 + iC4 + jO2 + khEOG)$ [5].
The QKSIG algorithm was simulated with Parzen window size $N = 10$. The parameters for the QKSIG $\eta = 0.7$, $\bar{\sigma} = 2.24$ and $\sigma = 0.736$ were chosen and for the QKLMS $\eta = 0.35$, $\bar{\sigma} = 2.24$ were used.

Fig.7.9 shows the performance of algorithms measured based on the prediction gain. The prediction gain can be described as $R_p = 10\log(\sigma_y^2/\sigma_e^2)$ where σ_y^2 and σ_e^2 are the average power of the input and output error respectively.

As shown in Fig.7.9, the QKSIG has better steady state prediction gain (around 6 to 7 dB more) than QKLMS. Even though, both QKSIG and QKLMS algorithms are operating in quaternion domain, the QKSIG are using ITL criterion for filtering instead of MSE criterion used by QKLMS. Therefore, QKSIG minimizes the divergence between input-output and input-desired joint PDF and yields better performance. QKMEE

algorithm gives similar performance as QKSIG as shown in Fig.7.9.

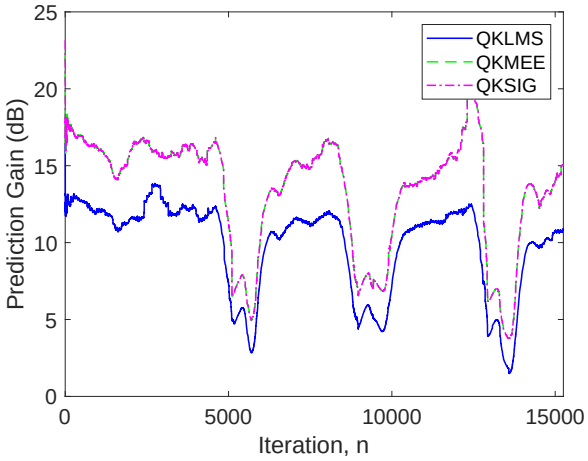


Fig. 7.9: Prediction gain

7.6.3 Stock Market Prediction

In this experiment, we demonstrate that why it is important to simultaneously consider non-Gaussian distributions and quaternion domains. For this reason, we use stock market indices time series which have non-stationary and non-Gaussian nature. This experiment shows first, the advantage of using quaternion over the quadruple real-valued filtering and the second, the advantage of using ITL over MSE criterion in non-Gaussian environment such as stock market prediction. For this purpose, four US Stock Indices fused into quaternionic model which are NASDAQ Composite ($IXIC$), Dow Jones Industrial Average (DJI), Russell 2000 (RUT) and SNP ($GSPC$). The US Indices obtained from "https://finance.yahoo.com" under Quotes "Historical Data" tabs for the period of Feb 8, 2012 to Feb9, 2018 with Daily frequency. We used one-step-ahead prediction model to predict the "High" price of each index at each trading day. To form a quaternion input, the four indices are fused in a quaternion form [51] as $q = IXIC + iDJI + jRUT + kGSPC$. Table IV shows the simulation results of one-step-ahead prediction using QKSIG, QKLMS and quadruple real-valued KSIG.

As shown in Table 7.4, the prediction gain of QKSIG dominates the quadruple real-valued KSIG for each index. The quadruple real-valued KSIG are using four separate real domain KSIG filters, one for each index, without considering the cross-correlation or coupling among the four indices (or dimensions). By combining all four indices as a quaternion and using QKSIG algorithm, intrinsically takes into account the cross-correlation or coupling among them and yields better prediction gain for each index.

Table 7.4 also shows the advantage of using ITL instead of MSE criterion in non-Gaussian environment. The prediction gain of QKSIG dominates the QKLMS for each index. Even though, both QKSIG and QKLMS algorithms are operating in quaternion domain, the QKSIG are using ITL criterion for filtering instead of MSE used by QKLMS.

Therefore, it is advantages for better prediction to use QKSIG instead of the QKLMS and quadruple real-valued KSIG algorithms.

Table 7.4: Prediction Gain (dB)

	QKSIG	**QKMEE**	**KSIG**	**QKLMS**
IXIC	34.87	32.77	31.15	28.05
DJI	33.83	31.15	30.12	26.44
RUT	33.98	32.01	30.24	13.07
GSPC	34.38	32.25	30.72	18.39

7.7 Conclusion

We have shown the derivation and convergence analysis of a quaternion kernel adaptive algorithm. The resulting algorithm is the QKSIG algorithm. The new algorithm (QK-

SIG) minimizes the error stochastic information gradient (SIG) cost function which is based on the information theoretic learning (ITL). A gradient of cost function is derived using GHR calculus in quaternion RKHS. The new algorithm (QKSIG) is beneficial in three different ways. First, reduces computational complexity as compared to our previous work QKMEE. Second, improves the filtering performance by taking into account the coupling within the dimensions of the quaternion input compared to quadruple real-valued inputs. Third, performs better in biased or non-Gaussian signal and noise environments compared to MSE criteria. Minimizing error entropy, minimizes the divergence between the joint densities of input-desired and input-output pairs (Appendix B). Simulation results show that there are advantages for better prediction to use QKSIG instead of the QKLMS and quadruple real-valued KSIG algorithms in simultaneously non-Gaussian distributions and quaternion environments. Future research will extend QKSIG algorithm for widely-linear case.

Chapter 8

Quaternion Stochastic Information Gradient Algorithm with Self Adjusting Step-size for Nonlinear Adaptive Systems

In this chapter, we develop a kernel adaptive filter for quaternion data, based on stochastic information gradient (SIG) cost function with self adjusting step-size. The new algorithm (QKSIG-SAS) is based on the information theoretic learning (ITL) approach and could be useful for quaternion based kernel applications of nonlinear filtering. In chapter 7, we developed the quaternion stochastic information gradient algorithm (QKSIG) [20], which minimizes Shannon's entropy of the error between the filter output and desired response and reduces the entropy estimation computational complexity. The new algorithm (QKSIG-SAS) has faster speed of convergence as compared to our previous work algorithm QKSIG in non-stationary environments.

8.1 The Quaternion Stochastic Information Gradient Algorithm with Self Adjusting Step-size Derivation

For the quaternion kernel adaptive filter based on minimum error entropy, the goal is to minimize the cost function $J_{SAS,n}(e)$ [52] with respect to free parameter $\mathbf{w}_n$ as

$$\begin{aligned} \min_{\forall \mathbf{w}_n \in \mathcal{H}} \quad & J_{SAS,n}(e) = (J_n(0) - J_n(e))^2 \\ \text{s.t.} \quad & e(n) = d(n) - y_n \\ & y_n =< \Phi(\mathbf{u}_n), \mathbf{w}_n >= \mathbf{w}_n^H \varphi_n \end{aligned} \tag{8.1}$$

where $d(n)$ is desired signal, $\mathbf{u}_n$ input vector, φ_n= $\Phi(\mathbf{u}_n)$ and $\Phi(.)$ is the kernel map to a QRKHS [25] defined as:

$$\begin{aligned} \Phi(\mathbf{u}) =& \Phi(\mathbf{u}_r + i\mathbf{u}_i + j\mathbf{u}_j + k\mathbf{u}_k) \\ &= \phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{q}_k^T]^T) + i\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) \\ &+ j\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) + k\phi([\mathbf{u}_r^T \mathbf{u}_i^T \mathbf{u}_j^T \mathbf{u}_k^T]^T) \end{aligned}$$

where ϕ is the feature map of real kernel κ defined as:

$$\phi(u_r, u_i, u_j, u_k) = \kappa(., u_r, u_i, u_j, u_k).$$

Minimizing the cost function $J_{SAS,n}(e)$ can be done with unconstrained optimization

algorithm such as gradient descent algorithm as

$$\begin{aligned}\mathbf{w}_{n+1} =& \mathbf{w}_n - \eta \nabla_{\mathbf{w}_n^*} J_{SAS,n}(e) \\ =& \mathbf{w}_n + 2\eta(J_n(0) - J_n(e)) \nabla_{\mathbf{w}_n^*} J_n(e) \\ =& \mathbf{w}_n + 2\eta(J_n(0) - J_n(e)) \left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right)^H \\ =& \mathbf{w}_n + 2\eta(n) \left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right)^H \end{aligned} \tag{8.2}$$

where $\eta(n) = 2\eta(J_n(0) - J_n(e))$ is adaptation step size.

By setting $\mathbf{w}_0 = 0$, we can obtain filter output weight as

$$\begin{aligned}\mathbf{w}_n = \zeta(n) \sum_{p=0}^{n-1} \Bigg[\Bigg(& \kappa_\sigma\Big(e(p)\Big) e(p) \varphi_p^H \\ & + \sum_{l=1}^{N} \Big[\kappa_\sigma\Big(e(p) - e(p-l)\Big) \Big] \\ & \times \Big[e(p) - e(p-l) \Big] \Big[\varphi_p^H - \varphi_{p-l}^H \Big] \Bigg) \\ & \times \frac{1}{\kappa_\sigma(e(p)) + \sum_{l=1}^{N} \kappa_\sigma(e(p) - e(p-l))} \Bigg]^H \end{aligned} \tag{8.3}$$

where $\zeta(n) = \eta(n)/4\sigma^2$ and $e(n-l) = d(n-l) - \mathbf{w}_n^H \varphi_{n-l}$ are *a posteriori errors* for all $l : 1 \leqslant l \leqslant N$.

By substituting the weight update in the $y_n = \mathbf{w}_n^H \varphi_n$ and using properties of Quaternion

Reproducing Kernel Hilbert Space (QRKHS) and the 'kernel trick' to replace the inner product of two vectors with quaternion kernel $\bar{\kappa}_{\bar{\sigma}}$, we can simplify the equation in kernel form as

$$
\begin{aligned}
y_n = \zeta(n) \sum_{p=0}^{n-1} \Bigg[\Bigg(& \kappa_\sigma\Big(e(p)\Big) e(p) \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) \\
& + \sum_{l=1}^{N} \Big[\kappa_\sigma(e(p) - e(p-l)) \Big] \\
& \times \Big[e(p) - e(p-l) \Big] \Big[\bar{\kappa}_{\bar{\sigma}}(\mathbf{u_p}, \mathbf{u_n}) - \bar{\kappa}_{\bar{\sigma}}(\mathbf{u_{p-l}}, \mathbf{u_n}) \Big] \Bigg) \\
& \times \frac{1}{\kappa_\sigma(e(p)) + \sum_{l=1}^{N} \kappa_\sigma(e(p) - e(p-l))} \Bigg]
\end{aligned}
\tag{8.4}
$$

8.2 Switching Scheme Between Adaptive Algorithms

The QKSIG-SAS algorithm gradient update step size converge to zero near the optimum point of error entropy surface [52]. This property stall the gradient descent algorithm and tracking of weight update of the QKSIG-SAS algorithm. The combination of QKSIG-SAS and QKSIG algorithms enable the system to update the weight vector due to small changes in error entropy surface which is crucial for tracking signals in non-stationary environments. In order to determine to switching time between two algorithms to maximize the speed of convergence an analytical criterion needs to be developed. The dynamics of adaptation can be understood in terms of energy minimization in the context of Lyapunov stability theory [53]. Simply, the faster the Lyapunov energy decreases, the faster we are getting towards the optimal solution, especially since our energy function is based on the criterion that needs to be optimized. The general

switching time is determined as:

$$\frac{\partial J_{SAS,n}(e)}{\partial \mathbf{w}_n}\Delta \mathbf{w}_n = \frac{\partial J_{SIG,n}(e)}{\partial \mathbf{w}_n}\Delta \mathbf{w}_n \tag{8.5}$$

Or

$$\begin{aligned} &- 4\eta(J_n(0) - J_n(e))^2 \left\| \frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right\|_{\mathbb{F}}^2 \\ &= -\eta_{SIG} \left\| \frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right\|_{\mathbb{F}}^2 \end{aligned} \tag{8.6}$$

The QKSIG-SAS should be used when the following condition satisfied:

$$\begin{aligned} &- 4\eta(J_n(0) - J_n(e))^2 \left\| \frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right\|_{\mathbb{F}}^2 \\ &> -\eta_{SIG} \left\| \frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right\|_{\mathbb{F}}^2 \end{aligned} \tag{8.7}$$

or

$$\begin{aligned} &J_n(e) > J_n(0) + 0.5\sqrt{(}\eta_{SIG}/\eta) \\ &\textit{or} \\ &J_n(e) < J_n(0) - 0.5\sqrt{(}\eta_{SIG}/\eta) \end{aligned} \tag{8.8}$$

8.3 Simulation Results

8.3.1 Quaternion Nonlinear Channel Estimation

8.3.1.1 Symmetric Unimodal Density Noise

The QKSIG-SAS algorithm was simulated for a nonlinear channel with symmetric unimodal Gaussian noise. The channel consisted of the quaternion filter, i.e.,

$$\begin{aligned} z(n) &= g_1^* u(n) + g_2^* u^i(n) + g_3^* u^j(n) + g_4^* u^k(n) \\ &+ h_1^* u(n-1) + h_2^* u^i(n-1) + h_3^* u^j(n-1) + h_4^* u^k(n-1) \end{aligned} \tag{8.9}$$

and nonlineraity, i.e.,

$$y(n) = z(n) + az^2(n) + bz^3(n) + v(n) \tag{8.10}$$

where $v(n)$ is Symmetric unimodal density Gaussian noise described later. Coefficients $g1, ..., g4, h1, ..., h4, a, b$, and noise $v(n)$ are all quaternion valued.The coefficients used were

$a = 0.075 + i0.35 + j0.1 - k0.05,$

$b = -0.025 - i0.25 - j0.05 + k0.03,$

$$g1 = -0.40 + i0.30 + j0.15 - k0.45,$$

$$h1 = 0.175 - i0.025 + j0.1 + k0.15,$$

$$g2 = -0.35 - i0.15 - j0.05 + k0.20,$$

$$h2 = 0.15 - i0.225 + j0.125 - k0.075,$$

$$g3 = -0.10 - i0.40 + j0.20 - k0.05,$$

$$h3 = +0.025 + i0.075 - j0.05 - k0.05,$$

$$g4 = +0.35 + i0.10 - j0.10 - k0.15,$$

$$h4 = -0.05 - i0.075 - j0.075 + k0.175.$$

For the tests, the input $u(n)$ was formed using impulsive Gaussian mixture models to form non-Gaussian signals as follows [23]:

$$\begin{aligned} p_u(i) = &(0.85N(1.0, 0.01) + 0.15N(3.0, 0.01)) \\ &+i(0.40N(0.5, 0.01) + 0.60N(2.5, 0.01)) \\ &+j(0.65N(3.5, 0.01) + 0.35N(1.5, 0.01)) \\ &+k(0.25N(2.0, 0.01) + 0.75N(5.5, 0.01)) \end{aligned} \tag{8.11}$$

And noise $v(n)$ was formed using symmetric unimodal Gaussian distributions as:

$$\begin{aligned} p_v(i) = &(N(0.0, 0.01)) \\ &+i(N(3.0, 0.01)) \\ &+j(N(1.0, 0.01)) \\ &+k(N(0.5, 0.01)) \end{aligned} \tag{8.12}$$

where $N(m_N, \sigma_N)$ denotes the normal (Gaussian) PDF with mean m_N and variance σ_N. The kernel parameters $\bar{\sigma} = 2.24$; $\sigma = 0.55$ for the Parzen window with size $N = 10$,

are estimated using Silverman's rule $1.06 \times min\{\sigma_Y, R/1.34\} \times N^{1/5L}$ [46] where σ_Y is the data standard deviation, L is data dimension, R is the interquartile and N is the number of samples. The simulation results of the QKSIG-SAS algorithm for the nonlinear channel are shown in Fig.8.1 to Fig.8.2.

Fig.8.1 and Fig.8.2 show the convergence of the normalized entropy (NEntropy) and mean-squared-error (MSE) of the QKSIG-SAS, QKSIG and Switching (combination of the QKSIG-SAS and QKSIG algorithms).

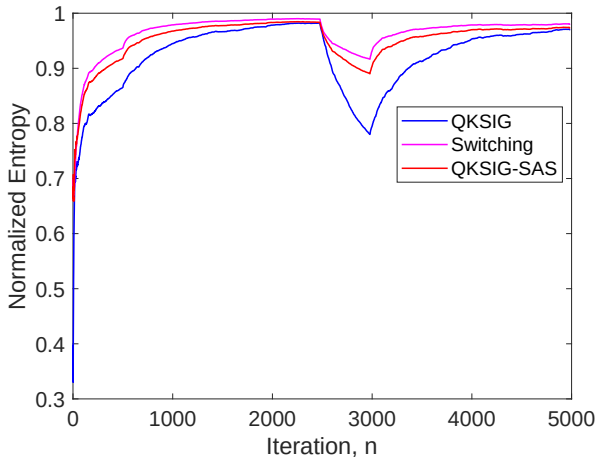


Fig. 8.1: Normalized Entropy of QKSIG-SAS, Switching and QKSIG algorithms

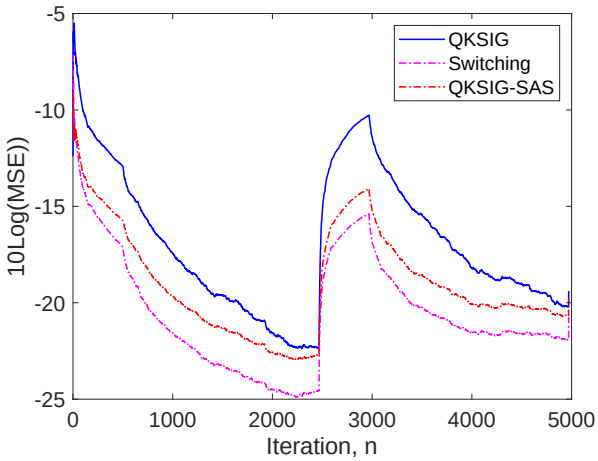


Fig. 8.2: Mean Squared Error of QKSIG-SAS, Switching and QKSIG algorithms

8.4 Conclusion

We have shown the derivation and convergence analysis of a quaternion kernel adaptive algorithm based on stochastic information gradient cost function with self-adjusting step size. The algorithm is based on information theoretic learning (ITL) cost function minimizing error entropy. The resulting algorithm is the QKSIG-SAS algorithm. The GHR calculus was used to derive the gradient of error entropy in quaternion domain. Simulation results show that the new algorithm (QKSIG-SAS) has faster speed of convergence as compared to QKSIG algorithm in non-stationary environments.

Chapter 9

Summary and Future Work

In this thesis we performed a study as well as developed algorithms for the kernel adaptive filters for quaternion domain data, based on information theoretic learning cost functions which could be useful for quaternion based kernel applications of nonlinear system filtering. The initial chapters (Chapter 2 and 3) describe the background theories of information theoretic learning, kernel adaptive filter theory and quaternion data and related properties. The later chapters (Chapter 4 to 8) presented the new research results and algorithms from the work performed.

Chapter 4 showed the derivation and demonstration of convergence of quaternion kernel minimum error entropy algorithm (QKMEE). The algorithm is based on information theoretic learning (ITL) cost function. A gradient was derived based on GHR calculus applied on quaternion RKHS. Algorithm using minimum error entropy may converge to set of optimal weights without having zero mean error. Therefore, to make the mean of output error equal to zero, the output during testing session was biased with the mean of errors of training session. It was shown that the QKMEE algorithm performed better with non-Gaussian signal and noise environment compared to QKLMS which is based on the MSE criteria adaptive filter. Also the results showed that minimizing error entropy could result in more concentrated error probability distribution PDF compared to MSE criterion.

Chapter 5 showed the derivation and convergence analysis of a quaternion kernel normalized minimum error entropy adaptive algorithm (QKNMEE). The algorithm is based on

information theoretic learning (ITL) cost function. We showed that the convergence of QKNMEE is very fast and outperforms the existing QKMEE and QKLMS algorithms. The convergence analysis showed that convergence step-size is independent of kernel size. Also it was shown that the convergence rate of the QKNMEE is independent of the input power and the kernel size.

Chapter 6 showed the derivation and convergence analysis of quaternion kernel minimum error entropy with fiducial point (QKMEEF) algorithm. Algorithm using minimum error entropy may converge to a set of optimal weights without having zero mean error. Traditionally, to make the mean of output error equal to zero, the output during testing session was biased with the mean of errors of training session. However, for non-symmetric or heavy tails error PDF the estimation of error mean is problematic. Using augmented maximum information potential cost function with fiducial point enable online learning without biasing the output during testing session. It was shown the behavior of the new algorithm (QKMEEF) when signal is non-Gaussian in presence of unimodal and bi-modal noise distributions. It showed that in case of unimodal noise distribution, minimizing error entropy (maximizing information potential) results in much lower error variance (MSE) versus the bi-modal case, which confirms the theory that in unimodal Gaussian distribution case the MSE could be bounded in terms of the entropy power. Also it was shown that the QKMEEF can track and predict the 4-Dimensional non stationary signals where there are correlations between components better than quadruple real-valued KMEEF and QKLMS algorithms.

Chapter 7 showed the derivation and convergence analysis of quaternion kernel stochastic information gradient (QKMEE-SIG) algorithm. The new algorithm (QKMEE-SIG) minimizes the error stochastic information gradient cost function which is based on information theoretic learning (ITL) approach. A gradient of cost function is derived using GHR calculus in quaternion RKHS. The new algorithm (QKSIG) is beneficial in

three different ways. First, reduces computational complexity as compared to our previous work QKMEE. Second, improves the filtering performance by taking into account the coupling within the dimensions of the quaternion input compared to quadruple real-valued inputs. Third, performs better in biased or non-Gaussian signal and noise environments compared to MSE criteria. Minimizing error entropy, minimizes the divergence between the joint densities of input-desired and input-output pairs (Appendix B). Simulation results show that there are advantages for better prediction to use QKSIG instead of the QKLMS and quadruple real-valued KSIG algorithms in simultaneously non-Gaussian distributions and quaternion environments.

Chapter 8 showed the derivation and convergence analysis of quaternion kernel stochastic information gradient algorithm with self-adjusting step size (QKSIG-SAS) algorithm. It was shown that the QKSIG-SAS algorithm has faster speed of convergence compared to QKSIG algorithm in non-stationary environments.

Future work could involve applying more optimization techniques to speed up the convergence rate of adaptive filters such as fixed-point technique. Further, we can modify the developed algorithms by applying the adaptive kernel size method to make the algorithm adjust the kernel size adaptively and as a result speed up the convergence rate of the algorithms.

All the newly developed algorithms could be applied in areas such as Machine Learning and Deep Learning in quaternion domain. Recently, an increasing interest has been shown on quaternion neural networks (QNN) such as feed forward neural networks (QFFNN), quaternion recurrent neural networks (QRNN) and quaternion convolutional neural network (QCNN) [54, 55, 56, 57]. Based on our knowledge, the cost functions for all the existing QNN, in supervised learning are based on MSE criterion which minimizes the mean-squared-error of neural network. Alternatively, by applying the information theoretic learning cost functions one can develop new algorithms in QNN area.

Appendix A

To derive the gradient of cost function $J_n(e)$ (7.4), we define functions $f\colon \mathbb{H} \to \mathbb{H}$ and $g_l\colon \mathbb{H} \to \mathbb{R}$ as

$$f(x) = \exp(x) \tag{A.1}$$

$$g_l(\mathbf{w}_n) = -\frac{|e(n) - e(n-l)|^2}{2\sigma^2}. \tag{A.2}$$

To simplify the notation for function g_l in our derivative for a given l, $1 \leqslant l \leqslant N$, we define $g(\mathbf{w}_n) = g_l(\mathbf{w}_n) = -\frac{\left|e(n)-e(n-l)\right|^2}{2\sigma^2}$.

With the above notation the gradient of cost function $J_n(e)$ (7.4) can be written as

$$\begin{aligned}
&\frac{\partial J_n(e)}{\partial \mathbf{w}_n} = \\
&-\left(\frac{\frac{\partial\left[\kappa_\sigma\left(e(n)\right)\right]}{\partial \mathbf{w}_n} + \frac{\partial\left[\sum_{l=1}^{N}\kappa_\sigma\left(e(n)-e(n-l)\right)\right]}{\partial \mathbf{w}_n}}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))} \right) \\
&= \left(\frac{\frac{-4}{\sqrt{2\pi}\sigma}\partial\left[\exp\{-\frac{|e(n)|^2}{2\sigma^2}\}\right]}{\partial \mathbf{w}_n} + \frac{\frac{-4}{\sqrt{2\pi}\sigma}\partial\left[\sum_{l=1}^{N}\exp\{-\frac{|e(n)-e(n-l)|^2}{2\sigma^2}\}\right]}{\partial \mathbf{w}_n}}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))} \right) \\
&= \frac{\frac{-4}{\sqrt{2\pi}\sigma}\frac{\partial\exp(\frac{-|e(n)|^2}{2\sigma^2})}{\partial \mathbf{w}_n} + \frac{-4}{\sqrt{2\pi}\sigma}\sum_{l=1}^{N}\frac{\partial\left[f(g_l(\mathbf{w}_n))\right]}{\partial \mathbf{w}_n}}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))}.
\end{aligned} \tag{A.3}$$

For a given l, the partial derivative can be calculated with GHR chain rule as

$$\begin{aligned}\frac{\partial\Big[f(g_l(\mathbf{w}_n))\Big]}{\partial\mathbf{w}_n} &= \frac{\partial\Big[f(g(\mathbf{w}_n))\Big]}{\partial\mathbf{w}_n} \\ &= \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial\mathbf{w}_n}.\end{aligned} \tag{A.4}$$

Using properties of the left GHR derivatives and exponential function properties, we can simplify (A.4) as follow

$$\sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial\mathbf{w}_n} = \exp(g)\frac{\partial g}{\partial\mathbf{w}_n}. \tag{A.5}$$

To calculate the derivative of function g, we can expand it as follow

$$\begin{aligned}g(\mathbf{w}_n) &= -\frac{|e(n)-e(n-l)|^2}{2\sigma^2} \\ &= \frac{-1}{2\sigma^2}|e(n)-e(n-l)|^2 \\ &= \frac{-1}{2\sigma^2}\Big[(e(n)-e(n-l))(e(n)-e(n-l))^*\Big] \\ &= \frac{-1}{2\sigma^2}\Big[(e(n)-e(n-l))(e^*(n)-e^*(n-l))\Big] \\ &= \frac{-1}{2\sigma^2}\Big[|e(n)|^2+|e(n-l)|^2-e(n)e^*(n-l) \\ &\quad -e(n-l)e^*(n)\Big].\end{aligned} \tag{A.6}$$

Therefore, we can find partial derivative of g using GHR calculus as below

$$\frac{\partial g}{\partial \mathbf{w}_n} = \left(\frac{-1}{2\sigma^2}\right)\Bigg[\frac{\partial |e(n)|^2}{\partial \mathbf{w}_n} + \frac{\partial |e(n-l)|^2}{\partial \mathbf{w}_n} - \frac{\partial e(n)e^*(n-l)}{\partial \mathbf{w}_n} - \frac{\partial e(n-l)e^*(n)}{\partial \mathbf{w}_n}\Bigg]. \tag{A.7}$$

By substituting $e(n-l) = d(n-l) - \mathbf{w}_n^H \varphi_{n-l}$ in (A.7) and using GHR calculus, we can compute each partial derivative of (A.7) as

$$\begin{aligned}\frac{\partial |e(n-l)|^2}{\partial \mathbf{w}_n} &= \frac{\partial e(n-l)e^*(n-l)}{\partial \mathbf{w}_n} \\ &= e(n-l)\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n} + \frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l)\end{aligned} \tag{A.8}$$

where

$$\frac{\partial e^*(n-l)}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H \frac{\partial \mathbf{w}_n}{\partial \mathbf{w}_n} = -\varphi_{n-l}^H. \tag{A.9}$$

Using the left GHR derivatives rules summarized in [58], $\frac{\partial q}{\partial q^{v*}} v = \frac{-1}{2} v^*$, the second term of (A.8) can be calculated as

$$\begin{aligned}\frac{\partial e(n-l)}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) &= -\frac{\partial \mathbf{w}_n^H \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) \\ &= -\mathbf{w}_n^H \frac{\partial \varphi_{n-l}}{\partial \mathbf{w}_n^{e^*(n-l)}} e^*(n-l) - \frac{\partial \mathbf{w}_n^H}{\partial \mathbf{w}_n^{\varphi_{n-l}e^*(n-l)}} \varphi_{n-l} e^*(n-l) \\ &= 0 + \frac{1}{2}\Big(\varphi_{n-l} e^*(n-l)\Big)^H \\ &= \frac{1}{2} e(n-l)\varphi_{n-l}^H.\end{aligned} \tag{A.10}$$

By substituting (A.9) and (A.10) in (A.8) we can obtain

$$\frac{\partial |e(n-l)|^2}{\partial \mathbf{w}_n} = -\frac{1}{2} e(n-l)\varphi_{n-l}^H. \tag{A.11}$$

Using the same method, the other terms of (A.7) can be calculated. By substituting all partial derivatives, we can simplify (A.7) as below

$$\begin{aligned} \frac{\partial g}{\partial \mathbf{w}_n} &= \left(\frac{-1}{2\sigma^2}\right)\Bigg(-\frac{1}{2} e(n)\varphi_n^H + -\frac{1}{2} e(n-l)\varphi_{n-l}^H \\ &\quad + \frac{1}{2} e(n)\varphi_{n-l}^H + \frac{1}{2} e(n-l)\varphi_n^H \Bigg) \\ &= \left(\frac{1}{4\sigma^2}\right)\Big[e(n) - e(n-l)\Big]\Big[\varphi_n^H - \varphi_{n-l}^H\Big]. \end{aligned} \tag{A.12}$$

Therefore, by substituting (A.12) in (A.4) we can obtain

$$\begin{aligned} \frac{\partial\Big[f(g_l(\mathbf{w}_n))\Big]}{\partial \mathbf{w}_n} &= \sum_{v\in\{1,i,j,k\}} \frac{\partial f}{\partial g^v}\frac{\partial g^v}{\partial \mathbf{w}_n} \\ &= \left(\frac{1}{4\sigma^2}\right) \times \Bigg(\exp(g_l(\mathbf{w}_n)) \\ &\quad \times \Big[e(n) - e(n-l)\Big]\Big[\varphi_n^H - \varphi_{n-l}^H\Big]\Bigg). \end{aligned} \tag{A.13}$$

Without loss of generality with replacing (A.6) by

$$g(\mathbf{w}_n) = -\frac{|e(n)|^2}{2\sigma^2} \tag{A.14}$$

we can show the following equation

$$\frac{\partial \exp(\frac{-|e(n)|^2}{2\sigma^2})}{\partial \mathbf{w}_n} = \left(\frac{1}{4\sigma^2}\right)\left(\exp(\frac{-|e(n)|^2}{2\sigma^2})e(n)\varphi_n^H\right). \tag{A.15}$$

By substituting (A.13) and (A.15) in (A.3) we can obtain

$$\begin{aligned}
\frac{\partial J_n(e)}{\partial \mathbf{w}_n} &= \frac{\frac{-4}{\sqrt{2\pi}\sigma}\frac{\partial \exp(\frac{-|e(n)|^2}{2\sigma^2})}{\partial \mathbf{w}_n} + \frac{-4}{\sqrt{2\pi}\sigma}\sum_{l=1}^{N}\frac{\partial\left[f(g_l(\mathbf{w}_n))\right]}{\partial \mathbf{w}_n}}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))} \\
&= \left(\frac{-1}{4\sigma^2}\right) \times \frac{4}{\sqrt{2\pi}\sigma} \times \left(\exp(\frac{-|e(n)|^2}{2\sigma^2})e(n)\varphi_n^H \right. \\
&\left. + \sum_{l=1}^{N}\exp(g_l(\mathbf{w}_n)) \times \left[e(n)-e(n-l)\right]\left[\varphi_n^H - \varphi_{n-l}^H\right]\right) \\
&\times \frac{1}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N}\kappa_\sigma(e(n)-e(n-l))}
\end{aligned} \tag{A.16}$$

Therefore the gradient of the cost function $J_n(e)$ can be calculated based on the following

equation

$$
\begin{aligned}
\nabla_{\mathbf{w}_n^*} J_n(e) &= \left(\frac{\partial J_n(e)}{\partial \mathbf{w}_n} \right)^H \\
&= \left(\frac{-1}{4\sigma^2} \right) \times \Bigg[\Bigg(\kappa_\sigma \Big(e(n) \Big) e(n) \varphi_n^H \\
&+ \sum_{l=1}^{N} \Big[\kappa_\sigma (e(n) - e(n-l)) \Big] \\
&\times \Big[e(n) - e(n-l) \Big] \Big[\varphi_n^H - \varphi_{n-l}^H \Big] \Bigg) \\
&\times \frac{1}{\kappa_\sigma(e(n)) + \sum_{l=1}^{N} \kappa_\sigma (e(n) - e(n-l))} \Bigg]^H
\end{aligned}
\tag{A.17}
$$

Appendix B

Lemma 4. Minimizing Shannon's entropy of quaternion random variable e from quaternion adaptive system (B.1), minimizes Kullback-Leibler divergence [59] between the joint probability distribution of the input-desired and input-output of adaptive system.

Proof. Suppose we have the following minimization problem:

$$\begin{aligned} \min_{\forall \mathbf{w} \in \mathcal{H}} \quad & H_S(e) = -\int_{\mathbb{H}} p_e(\varepsilon) \log p_e(\varepsilon) d\varepsilon = E_e[-\log p_e] \\ \text{s.t.} \quad & e = d - y \\ & y = <\mathbf{u}, \mathbf{w}> = \mathbf{w}^H \mathbf{u} \end{aligned} \tag{B.1}$$

where $\mathbf{u}$ is the system input vector, y is the system output, d is desired signal and p_e is probability distribution function of quaternion random variable e. Based on the problem definition (B.1), we can deduce the following PDF equation

$$p_{e|\mathbf{w}}(e) = p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(d - e|\mathbf{u}). \tag{B.2}$$

Therefore, minimizing Shannon entropy of error can be written as

$$
\begin{aligned}
&\min_{\forall \mathbf{w}\in\mathcal{H}} \quad -\int_{\mathbb{H}} p_{e|\mathbf{w}}(\varepsilon)\log p_{e|\mathbf{w}}(\varepsilon)d\varepsilon \\
&= \min_{\forall \mathbf{w}\in\mathcal{H}} -\int_{\mathbb{H}} p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(d-\varepsilon|\mathbf{u})\log p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(d-\varepsilon|\mathbf{u})d\varepsilon \\
&= \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}} p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|\mathbf{u})\log p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|\mathbf{u})dy \\
&\equiv \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}} p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|\mathbf{u})\log p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|\mathbf{u})dy.\int_{\mathbb{H}} p_{\mathbf{u}}(x)dx \\
&= \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}}\int_{\mathbb{H}} p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|x)p_{\mathbf{u}}(x)\log p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|x)dydx \\
&= \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}}\int_{\mathbb{H}} p_{\mathbf{y},\mathbf{u}|\mathbf{w}}(y,x)\log p_{\mathbf{y}|\mathbf{u},\mathbf{w}}(y|x)dydx \\
&\equiv \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}}\int_{\mathbb{H}}\int_{\mathbb{H}} p_{\mathbf{y},\mathbf{u}|\mathbf{w}}(y,x)\log \frac{p_{\mathbf{y},\mathbf{u}|\mathbf{w}}(y,x)}{p_{\mathbf{u},\mathbf{d}}(x,\zeta)}dydxd\zeta
\end{aligned}
\tag{B.3}
$$

where $p_{\mathbf{y},\mathbf{u}}$ is the input-output joint PDF and $p_{\mathbf{u},\mathbf{d}}$ is the input-desired joint PDF. Thus, from (B.3) we can conclude that minimizing Shannon entropy of error minimizes the Kullback-Leibler divergence between the joint probability distribution of the input-desired and input-output as:

$$
\begin{aligned}
&\min_{\forall \mathbf{w}\in\mathcal{H}} \quad -\int_{\mathbb{H}} p_{e}(\varepsilon)\log p_{e}(\varepsilon)d\varepsilon \\
&\equiv \min_{\forall \mathbf{w}\in\mathcal{H}} \int_{\mathbb{H}}\int_{\mathbb{H}}\int_{\mathbb{H}} p_{\mathbf{y},\mathbf{u}|\mathbf{w}}(y,x)\log \frac{p_{\mathbf{y},\mathbf{u}|\mathbf{w}}(y,x)}{p_{\mathbf{u},\mathbf{d}}(x,\zeta)}dydxd\zeta \\
&= \min_{\forall \mathbf{w}\in\mathcal{H}} D_{KL}(p_{\mathbf{y},\mathbf{u}|\mathbf{w}}||p_{\mathbf{u},\mathbf{d}}).
\end{aligned}
\tag{B.4}
$$

□

List of Publication

Refereed Journals:

T. Ogunfunmi and C. Safarian, "The Quaternion Stochastic Information Gradient Algorithm for Nonlinear Adaptive Systems", *IEEE Transactions on Signal Processing."* vol. 67, no. 23, pp. 5909-5921, 1 Dec.1, 2019.

Carlo Safarian, Tokunbo Ogunfunmi, "The quaternion minimum error entropy algorithm with fiducial point for nonlinear adaptive systems," *Signal Processing,* Volume 163, 2019, Pages 188-200.

Conference Proceedings:

T. Ogunfunmi and C. Safarian, "A Quaternion Kernel Minimum Error Entropy Adaptive Filter," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Calgary, AB, 2018, pp. 4149-4153.

C. Safarian and T. Ogunfunmi, "Quaternion Kernel Normalized Minimum Error Entropy Adaptive Algorithms," *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Honolulu, HI, USA, 2018, pp. 572-578.

C. Safarian, T. Ogunfunmi, W. J. Kozacky and B. K. Mohanty, "FPGA implementation of LMS-based FIR adaptive filter for real time digital signal processing applications," *2015 IEEE International Conference on Digital Signal Processing (DSP)*, Singapore, 2015, pp. 1251-1255.

Bibliography

[1] D. Erdogmus and J. C. Principe, "From linear adaptive filtering to nonlinear information processing - the design and analysis of information processing systems," *IEEE Signal Processing Magazine.*, vol. 23, no. 6, pp. 14–33, 2006. iii, 1, 5, 7, 8

[2] W. Liu, I. Park, and J. C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950–1961, Dec 2009. iii, 1

[3] D. Erdogmus and J. Principe, "An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems," *IEEE Transactions on Signal Processing.*, vol. 50, no. 7, pp. 1950–1961, 2002. iii, iv, 1, 5, 77, 78, 113

[4] C. C. Took and D. P. Mandic, "A quaternion widely linear adaptive filter," *IEEE Transactions on Signal Processing.*, vol. 58, no. 8, pp. 4427–4431, 2010. iii

[5] S. Javidi, C. C. Took, C. Jahanchahi, N. L. Bihan, and D. P. Mandic, "Blind extraction of improper quaternion sources," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* iii, 136

[6] D. P. Mandic, C. Jahanchahi, and C. C. Took, "A quaternion gradient operator and its applications," *IEEE Signal Processing Letters.*, vol. 18, no. 1, pp. 47–50, 2011. iii

[7] A. Shilton and D. T. H. Lai, "Quaternionic and complex-valued support vector regression for equalization and function approximation," *2007 International Joint Conference on Neural Networks.* iii

[8] E. J. Bayro-Corrochano and N. Arana-Daniel, "Clifford support vector machines for classification, regression, and recurrence," *IEEE Transactions on Neural Networks.*, vol. 21, no. 11, pp. 1731–1746, 2010. iii

[9] C. C. Took and D. P. Mandic, "Augmented second-order statistics of quaternion random signals," *Signal Processing.*, vol. 91, no. 2, pp. 214–224, 2010. iii, 2, 23, 31, 32, 34

[10] D. Xu, C. Jahanchahi, C. C. Took, and D. P. Mandic, "Quaternion derivatives: The GHR calculus," *Submitted exclusively to the London Mathematical Society.*, 2014. iv, 2, 24, 42

[11] D. Xu, Y. Xia, and D. P. Mandic, "Optimization in quaternion dynamic systems:gradient, hessian, and learning algorithms," *IEEE Transactions on Neural Networks and Learning Systems.*, vol. 27, no. 2, pp. 249–261, 2016. iv, 2, 21, 25, 26, 28, 29, 30, 42, 99, 115

[12] F. A. Tobar and D. P. Mandic, "Quaternion reproducing kernel hilbert spaces: Existence and uniqueness conditions," *IEEE Transactions on Information Theory.*, vol. 60, no. 9, pp. 5736–5749, 2014. iv, 2, 35, 36, 37, 38, 39, 42

[13] T. Ogunfunmi and C. Safarian, "A quaternion kernel minimum error entropy adaptive filter," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4149–4153, April 2018. iv, v, 60, 77, 78, 112, 119

[14] W. A. Martins, P. S. R. Diniz, and Y.-F. Huang, "On the normalized minimum error-entropy adaptive algorithm: Cost function and update recursion," *2010 First IEEE Latin American Symposium on Circuits and Systems (LASCAS).*, 2010. iv, 60, 61

[15] P. P. Brahma, "Prediction of sunspot number using minimum error entropy cost based kernel adaptive filters," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA).*, 2015. iv

[16] C. Safarian and T. Ogunfunmi, "Quaternion kernel normalized minimum error entropy adaptive algorithms," *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 572–578, Nov 2018. iv

[17] C. Safarian and T. Ogunfunmi, "The quaternion minimum error entropy algorithm with fiducial point for nonlinear adaptive systems," *Signal Processing*, vol. 163, pp. 188 – 200, 2019. iv, 109

[18] W. Liu, P. P. Pokharel, and J. C. Principe, "Error entropy, correntropy and m-estimation," *2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing.*, 2006. v, 78, 113

[19] D. Erdogmus and J. Principe, "An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems," *IEEE Transactions on Signal Processing.*, vol. 50, no. 7, pp. 1950–1961, 2002. v

[20] T. Ogunfunmi and C. Safarian, "The quaternion stochastic information gradient algorithm for nonlinear adaptive systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5909–5921, Dec 2019. v, 140

[21] W. Liu, J. C. Principe, and S. Haykin, "Kernel adaptive filtering: A comprehensive introduction," *Wiley.*, 2010. xi, 10, 14, 15, 19

[22] C. C. Took and D. P. Mandic, "The quaternion LMS algorithm for adaptive filtering of hypercomplex processes," *IEEE Transactions on Signal Processing.*, vol. 57, no. 4, pp. 1316–1327, 2009. 1, 2

[23] T. Ogunfunmi and T. Paul, "The quarternion maximum correntropy algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs.*, vol. 62, no. 6, pp. 598–602, 2015. 2, 53, 72, 73, 95, 96, 128, 129, 133, 146

[24] F. A. Tobar and D. P. Mandic, "The quaternion kernel least squares," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing.* 2, 57, 102

[25] T. K. Paul and T. Ogunfunmi, "A kernel adaptive algorithm for quaternion-valued inputs," *IEEE Transactions on Neural Networks and Learning Systems.*, vol. 26, no. 10, pp. 2422–2439, 2015. 2, 44, 45, 61, 67, 80, 81, 97, 112, 113, 115, 119, 141

[26] Q. Barthelemy, A. Larue, and J. I. Mars, "About QLMS derivations," *IEEE Signal Processing Letters.*, vol. 21, no. 2, pp. 240–243, 2014. 2

[27] M. D. Jiang, Y. Li, and W. Liu, "Properties of a general quaternion-valued gradient operator and its application to signal processing," *Frontiers of Information Technology and Electronic Engineering.*, vol. 17, no. 2, pp. 83–95, 2016. 2, 24, 80

[28] S. Haykin, "Information theoretic learning in unsupervised adaptive filtering," *New York:Wiley.*, vol. 1, pp. 265–3193, 2000. 5

[29] J. C. Principe, "Information theoretic learning: Renyi's entropy and kernel perspectives," *Springer.*, 2010. 5, 6

[30] X. Feng, K. Loparo, and Y. Fang, "Optimal state estimation for stochastic systems: An information theoretic approach," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 771–785, June 1997. 5

[31] D. Erdogmus and J. C. Principe, "Comparison of entropy and mean square error criteria in adaptive system training using higher order statistics," *Proc. ICA.*, 2000. 5, 6

[32] C. E. Shannon., "A mathematical theory of communication," *The Bell System Technical Journal.*, vol. 27, no. 3, pp. 379 – 423, 1948. 5, 6, 112

[33] A. Renyi, "Probability theory," *New York: Elsevier.*, 1970. 5, 42, 79

[34] T. Cover and J. Thomas, "Elements of information theory," *New York: Wiley.*, 1991. 6

[35] R. Duda, P. Hart, and D. Stork, "Pattern classification," *New York: Wiley.*, 2000. 7

[36] A. S. Hardy., "Elements of quaternions," *Boston, Ginn, Heath, Co.*, 1887. 20

[37] B. Silverman, "Density estimation for statistics and data analysis," *New York: Routledge.*, 1998. 54

[38] S. Han, S. Rao, K. H. Jeong, and J. Principe, "A normalized minimum error entropy stochastic algorithm," *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings.*, 2006. 60

[39] S. Han, "A family of minimum renyis error entropy algorithm for information processing," *Ph.D. Thesis, University of Florida, FL, USA.*, 2007. 60

[40] S. Zhao, B. Chen, and J. C. Principe, "Kernel adaptive filtering with maximum correntropy criterion," *The 2011 International Joint Conference on Neural Networks.*, 2011. 68, 89, 96

[41] L. Martino and V. Elvira, "Compressed monte carlo for distributed bayesian inference," *viXra:1811.0505.*, 2018. 77, 78

[42] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE.*, vol. 92, no. 3, pp. 401–422, 2004. 78

[43] Y. Chen, M. Welling, and A. Smola, "Super-samples from kernel herding," *In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence.*, pp. 1–8, 2010. 78

[44] S. Lacoste-Julien, F. Lindsten, and F. Bach, "Sequential kernel herding: Frank-wolfe optimization for particle filtering," *In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence.*, pp. 544–552, 2015. 78

[45] P. Fearnhead, "Using random quasi-monte carlo within particle filters, with application to financial time series," *Journal of Computational and Graphical Statistics.*, vol. 14, no. 4, pp. 751–769, 2005. 78

[46] H. W. Chung, B. M. Sadler, and A. O. Hero, "Bounds on variance for unimodal distributions," *IEEE Transactions on Information Theory.*, vol. 63, no. 11, pp. 6936 – 6949, 2017. 99, 147

[47] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, 09 1962. [Online]. Available: https://doi.org/10.1214/aoms/1177704472 112

[48] D. Erdogmus, K. Hild, and J. Principe, "Online entropy manipulation: stochastic information gradient," *IEEE Signal Processing Letters.*, vol. 10, no. 8, pp. 242 – 245, 2003. 113

[49] M. Xiang, C. C. Took, and D. P. Mandic, "Cost effective quaternion minimum mean square error estimation: From widely linear to four channel processing," *Signal Processing.*, vol. 136, pp. 81 – 91, 2017. 120, 124, 125

[50] B. Silverman, "Density estimation for statistics and data analysis," *New York: Routledge.*, 1998. 129

[51] C. C. Took and D. Mandic, "Fusion of heterogeneous data sources: A quaternionic approach," *2008 IEEE Workshop on Machine Learning for Signal Processing.*, pp. 456 – 461, 2008. 137

[52] S. Han, S. Rao, D. Erdogmus, and J. Principe., "An improved minimum error entropy criterion with self adjusting step-size," *IEEE Workshop on Machine Learning for Signal Processing.*, 2005. 141, 143

[53] H. Khalil, "Nonlinear systems," *Macmillan, New York.*, 1992. 143

[54] T. Minemoto, T. Isokawa, H. Nishimura, and NobuyukiMatsui, "Feed forward neural network with random quaternionic neurons," *Signal Processing.*, vol. 136, pp. 59–68, July 2017. 151

[55] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," *2018 International Joint Conference on Neural Networks (IJCNN).*, pp. 1–8, 2018. 151

[56] T. Parcollet, M. Ravanelli, M. Morchid, G. Linares, C. Trabelsi, R. D. Mori, and Y. Bengio, "Quaternion recurrent neural networks," *arXiv preprint arXiv:1806.04418v2.*, July 2018. 151

[57] D. Comminiello, M. Lella, S. Scardapane, and A. Uncini, "Quaternion convolutional neural networks for detection and localization of 3d sound events," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*, pp. 8533–8537, 2019. 151

[58] D. Xu, C. Jahanchahi, C. C. Took, and D. P. Mandic, "Enabling quaternion derivatives: The generalized HR calculus," *Roy. Soc. Open Sci.*, vol. 2, no. 8, 2015. 154

[59] S. Kullback., "Information theory and statistics," *New York: Dover.*, 1968. 158