Quaternion Filtering Based on Quaternion Involutions and Its Application in Signal Processing

Gang Wang¹, Rui Xue², Member, IEEE

¹School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China
²School of Electronics and Information Engineering, Beihang University, Beijing, P.R. China

Corresponding author: Rui Xue (e-mail: xuerui@buaa.edu.cn).

This work was supported by the National Natural Science Foundation of China under Grants 61371182 and 41301459.

ABSTRACT The quaternion gradient plays an important role in quaternion signal processing, and has undergone several modifications. Recently, three methods for obtaining the quaternion gradient have been proposed based on generalized HR calculus, the quaternion product, and quaternion involutions, respectively. The first method introduces the quaternion rotation, which is difficult to calculate and often relies on lookup tables; the second is cumbersome because it depends on all the real and imaginary parts of the variables and parameters; the third transforms the gradient from the quaternion domain to the real domain, and uses real derivatives and the real chain rule. In this paper, we generalize the quaternion involutions method to calculate the quaternion gradient of the quaternion matrix function. Several examples are presented in which the proposed gradient is applied to the adaptive filter, Kalman filter, and kernel filters to solve the associated optimization problems.

INDEX TERMS Least mean squares, mean square error, quaternion filter, Wirtinger calculus.

I. INTRODUCTION

Quaternions provide a compact model of mutual information between data channels, and have been used in fields such as traditional navigation [1-2], Kalman filtering [3-5], neural networks [6-7], spectral estimation [8-9], communication [10], system control [11], motion tracking [5, 12], biomedical engineering [13], and adaptive filtering [14-30]. The quaternion domain is a non-commutative extension of the complex domain. In the derivation of quaternion signal processing algorithms, the mean square error (MSE) criterion is commonly used as the cost function. The cost function is real-valued and nonanalytic, and the variables are quaternion-valued. The real-valued cost function based on MSE for the quaternion linear filter in the discrete time domain i is given by

\[ J_{\text{MSE}}(w) = |e(i)|^2 = e(i)e^*(i) = e^*(i)e(i). \]  \hspace{1cm} (1)

where \( e(i) \) is the error signal and \( w \) is the estimated parameter.

Most gradient algorithms can be generalized from the real domain to the complex domain [31], but cannot be directly generalized to the quaternion domain because the quaternion product is non-commutative. Although the chain rule works well in real and complex variables, it cannot be used for quaternion variables.

The quaternion gradient plays an important role in quaternion signal processing, and has undergone several modifications. From 2009 to 2012, three types of gradient algorithms were proposed: quaternion least mean squares (QLMS) [14, 15], HR-QLMS [23], and iQLMS [24-25]. The performance of QLMS was analyzed in [26]. These gradients have subsequently been applied in many studies. For instance, a quaternion maximum correntropy algorithm [27] and an echo network [29] used QLMS, whereas a quaternion Kalman filter [3], distributed quaternion Kalman filter [5], and kernel adaptive algorithm [30] have been developed based on HR-QLMS. A diffusion quaternion algorithm for a sensor network employed iQLMS [32].

However, we cannot obtain the same gradient from different expressions of the same cost function using these three gradient algorithms.

Let
\[
J_1(w) = e(i)e'(i), \\
J_2(w) = e'(i)e(i). 
\]

Clearly, we have \( J_1(w) = J_2(w) \).

For QLMS [14], the gradients of \( J_1 \) and \( J_2 \) are given as
\[
\frac{\partial J_1}{\partial w^1} = -e(i)u'(i)+0.5u(i)e'(i), \\
\frac{\partial J_2}{\partial w^1} = -u'(i)e(i)+0.5e'(i)u'(i). 
\]

where \( u(i) \in \mathbb{H}^N \) is the quaternion input vector.

For HR-QLMS [23], the gradients of \( J_1 \) and \( J_2 \) are
\[
\frac{\partial J_1}{\partial w^1} = e(i)\frac{\partial e'(i)}{\partial w^1} + \frac{\partial e'(i)}{\partial w^1} = -e(i)u'(i)+0.5u(i)e'(i), \\
\frac{\partial J_2}{\partial w^1} = e'(i)\frac{\partial e(i)}{\partial w^1} + \frac{\partial e(i)}{\partial w^1} = 0.5e'(i)u(i) - u'(i)e(i). 
\]

For iQLMS [24-25], the gradients of \( J_1 \) and \( J_2 \) are given by
\[
\frac{\partial J_1}{\partial w^1} = \delta_{11} \frac{\partial J_1}{\partial w^1} + \delta_{21} \frac{\partial J_1}{\partial w^1} + \delta_{12} \frac{\partial J_1}{\partial w^1} = -\frac{3}{2}e(i)u'(i), \\
\frac{\partial J_2}{\partial w^1} = \delta_{11} \frac{\partial J_2}{\partial w^1} + \delta_{21} \frac{\partial J_2}{\partial w^1} + \delta_{12} \frac{\partial J_2}{\partial w^1} = -\frac{3}{2}u'(i)e(i). 
\]

It follows from (3)–(5) that the gradient of \( J_1 \) is different from that of \( J_2 \) for the three algorithms, because the quaternion product is non-commutative.

Actually, \( J_1 \) is equal to \( J_2 \), and it is unreasonable to obtain different gradients from the same objective function. Clearly, in the complex domain, the gradient of \( J_1 \) is equal to that of \( J_2 \) [31]. Hitzer [33] noticed a similar mistake in a split quaternion nonlinear adaptive filter [28], and obtained corrections based on the Clifford calculus.

From 2014 to 2019, three methods for obtaining the correct quaternion gradient were proposed: generalized HR (GHR) calculus [34-36], quaternion product [37], and quaternion involutions [38]. The gradients given by these three methods are the same, and produce the same expression for \( J_1 \) and \( J_2 \) [38].

Took and Mandic [14] introduced the GHR calculus, and derived a systematic framework for calculating the derivatives of a quaternion matrix function with respect to quaternion variables [34-36]. Recently, it has been pointed out [37-38] that the full derivation of the gradient in [14] is incorrect, and the correct quaternion gradient is obtained using the product method in [37] and the quaternion involutions method in [38].

In all three algorithms, the product method [37] is simple but cumbersome, because it depends on all the real and imaginary parts of the variables and parameters, and it is difficult to calculate the derivatives of a quaternion matrix function. The GHR method can calculate the derivatives of a quaternion matrix function, but the quaternion rotation introduced is very difficult to compute, and lookup tables are often required [34]. In the quaternion involutions method, the quaternion gradient is transformed from the quaternion domain to the real domain of four times the dimension, and the real derivatives and real chain rules can be easily applied.

As mentioned above, many quaternion applications in signal processing domain were influenced by the early incorrect methods [3, 27, 29, 30, 32]. In this paper, we applied the correct method to adaptive filters, Kalman filters, and kernel filters to solve the associated optimization problems.

The remainder of this paper is organized as follows. In Section II, we describe the quaternion algebra. In Section III, we present the quaternion involutions method for obtaining the quaternion gradient. Section IV presents applications to an adaptive linear filter, a Kalman filter, and a kernel filter. We present our conclusions in Section IV.

II. PRELIMINARIES

A. QUATERNION ALGEBRA

The quaternion domain is a non-commutative extension of the complex domain. A quaternion variable \( q \in \mathbb{H} \) consists of a real part \( R \{ \} \) and a pure quaternion \( T \{ \} \), which comprises three imaginary components and can be expressed as
\[
q = R \{ q \} + T \{ q \} \\
= R \{ q \} + iT \{ q \} + jT \{ q \} + kT \{ q \} \\
= q_a + i q_b + j q_c + k q_d \in \mathbb{H},
\]

where \( q_a, q_b, q_c, q_d \in \mathbb{R} \). The relationships among the orthogonal unit vectors \( i, j, k \) are given by
\[
i j = k, \ j k = i, \ k i = j, \ i j k = i^2 = j^2 = k^2 = -1.
\]

The quaternion product is non-commutative, i.e., \( i j = -j k \), but \( i j = -k i \). Given \( q_1, q_2 \in \mathbb{H} \), we have
\[
q_1 q_2 = R \{ q_1 q_2 \} + T \{ q_1 q_2 \}, \qquad IJ = K, \ K J = I, \ K I = J, \ I J K = I^2 = j^2 = k^2 = -1.
\]

The symbols “+” and “×” represent the dot product and vector product, respectively.

If the three perpendicular quaternion involutions (self-inverse mappings) are given by
\[
q' = -i q t = q_a + i q_b - j q_c - k q_d, \\
q' = -j q t = q_a - i q_b + j q_c - k q_d, \\
q' = -k q t = q_a - i q_b - j q_c + k q_d,
\]

then
\[q_a = \frac{1}{2}\left(q + q^*\right), \quad q_b = \frac{1}{2i}(q - q^*) \]
\[q_c = \frac{1}{2j}(q - q^*), \quad q_d = \frac{1}{2\kappa}(q - q^{*\kappa})\]

where the quaternion conjugate operation can be expressed as

\[q^* = q_a - iq_b - jq_c - \kappa q_d = \frac{1}{2}(q + q^* - q)\]  
(11)

During the derivation of the quaternion gradient, the quaternion involutions (9) and property (11) are often used.

The norm of a quaternion is defined as

\[\|q\|_f = q q^* = q_a^2 + q_b^2 + q_c^2 + q_d^2\]  
(12)

**B. QUATERNION VECTOR**

For one vector \(q \in \mathbb{H}^{4 \times 1}\), the augmented vector \(q^a \in \mathbb{H}^n\) can be expressed as

\[q^a = \left[q^T q^T q^T q^T q^T q^T\right]^T\]  
(13)

which contains the necessary second-order statistical information, and the augmented covariance matrix is given by

\[R_{q^a} = E\left[q^a q^{a^H}\right] = \begin{bmatrix} C_{qq} & C_{qq} & C_{qq} & C_{qq} \\ C_{qq} & C_{qq} & C_{qq} & C_{qq} \\ C_{qq} & C_{qq} & C_{qq} & C_{qq} \\ C_{qq} & C_{qq} & C_{qq} & C_{qq} \end{bmatrix}\]  
(14)

where

\[C_{\theta \phi} = E\left[q^\theta q^{\phi H}\right], \quad \theta, \phi \in \{1, i, j, k\} \]  
(15)

**C. HR CALCULUS**

HR calculus is an extension of Wirtinger calculus, which is also called CR calculus. CR calculus provides a simple and straightforward approach to calculating derivatives with respect to complex parameters.

HR calculus comprises the HR derivatives [35]

\[
\begin{align*}
\frac{\partial f(q, q^* q^* q^*)}{\partial q} &= \left[1 - i - j - \kappa\right]\frac{\partial f}{\partial q_a} \\
\frac{\partial f(q, q^* q^* q^*)}{\partial q^*} &= \left[1 - i + j - \kappa\right]\frac{\partial f}{\partial q_a} \\
\frac{\partial f(q, q^* q^* q^*)}{\partial q^*} &= \frac{1}{4}\left[1 + i - j - \kappa\right]\frac{\partial f}{\partial q_b} \\
\frac{\partial f(q, q^* q^* q^*)}{\partial q^{*\kappa}} &= \frac{1}{4}\left[1 + i + j - \kappa\right]\frac{\partial f}{\partial q_b}
\end{align*}
\]  
(16)

and the HR* derivatives [35]

\[
\begin{bmatrix}
\frac{\partial f(q, q^* q^* q^*)}{\partial q} \\
\frac{\partial f(q, q^* q^* q^*)}{\partial q^*} \\
\frac{\partial f(q, q^* q^* q^*)}{\partial q^{*\kappa}} \\
\end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & i & j & -\kappa \\ 1 & i & j & \kappa \\ 1 & i & j & \kappa \\ \end{bmatrix}
\begin{bmatrix}
\partial f/\partial q_a \\
\partial f/\partial q_b \\
\partial f/\partial q_b \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\partial f/\partial q_a \\
\partial f/\partial q_b \\
\partial f/\partial q_b \\
\end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & i & j & \kappa \\ 1 & i & j & \kappa \\ 1 & i & j & \kappa \\ \end{bmatrix}
\begin{bmatrix}
\partial f/\partial q_a \\
\partial f/\partial q_b \\
\partial f/\partial q_b \\
\end{bmatrix}
\]

\(17\)

**III. QUATERNION GRADIENT**

**A. QUATERNION GRADIENT UPDATES**

To derive the expression for gradient descent in the quaternion domain, we construct three related vectors from the real vectors \(w_a, w_b, w_c, w_d \in \mathbb{R}^N\). The first is the quaternion vector \(w = w_a + w_iw_j + w_jw_k + w_kw_i \kappa \in \mathbb{H}^N\). The second is the real composite 4N-dimensional vector \(w_h = [w_a, w_b, w_c, w_d]^T \in \mathbb{R}^{4N}\). The third is the quaternion augmented vector \(w^a = [w, w^T, w^T, w^{*T}]^T \in \mathbb{H}^{4N}\). The quaternion augmented vector \(w^a\) is related to the real composite vector \(w_h \in \mathbb{R}^{4N}\) through \(w^a = U_H^w w_h\) and \(w_h = U_H^w w^a / 4\), where the real-to-quaternion transformation

\[U_N = \begin{bmatrix} I & iI & jI & kI \\ I & iI & -jI & -kI \\ I & -iI & jI & -kI \\ I & -iI & -jI & kI \end{bmatrix} \in \mathbb{H}^{4N \times 4N}
\]  
(18)

is unitary up to a factor of four, that is, \(U_N U_N^H = 4I\). Consider a function \(f(w) : \mathbb{H}^N \to \mathbb{R}\) that is real differentiable up to second order. If we write the function as \(f(w_h) : \mathbb{R}^{4N} \to \mathbb{R}\) using the definition given above, we can establish the following relationship:

\[
\frac{\partial f}{\partial w_h} = U_H^w \frac{\partial f}{\partial w^a}.
\]  
(19)

The real gradient update rule is \(\Delta w_h = -\mu(\partial f/\partial w_h)\), and we obtain the quaternion update relationship

\[
\Delta w^a = U_N^H \Delta w_h = -\mu U_N^H \frac{\partial f}{\partial w^a} = -\mu U_N^H \frac{\partial f}{\partial w^a}.
\]  
(20)

where \(\mu\) is the step size. The dimension of the update equation can be further reduced as

\[
\Delta w^a = \begin{bmatrix} \Delta w \\ \Delta w' \\ \Delta w' \\ \Delta w'' \end{bmatrix} = -4\mu \frac{\partial f}{\partial w^a} = -4\mu \frac{\partial f}{\partial w^a}.
\]  
(21)

Thus, we obtain the quaternion gradient update rule \(\Delta w = -4\mu \frac{\partial f}{\partial w^a}\). If the factor 4 is absorbed into the step size, then we obtain the following gradient descent algorithm:

\[\]
\[ w(i) = w(i-1) + \Delta w = w(i-1) - \mu \frac{\partial f}{\partial w^*}. \] (22)

The expression \( \frac{\partial f}{\partial w^*} \) also defines the direction of the maximum rate of change of \( f \) with respect to the vector \( w \). This is consistent with the complex case. During the following derivations, we use the gradient expression of the cost function \( J \) with respect to the vector variable \( w \) and the matrix variable \( Q \).

\[
\begin{align*}
\frac{\partial J}{\partial w^*} &= \frac{1}{4} \left( \frac{\partial J}{\partial w_a} + \frac{\partial J}{\partial w_b} + \frac{\partial J}{\partial w_c} + \frac{\partial J}{\partial w_d} \right), \\
\frac{\partial J}{\partial Q^a} &= \frac{1}{4} \left( \frac{\partial J}{\partial Q_{a,b}} + \frac{\partial J}{\partial Q_{a,c}} + \frac{\partial J}{\partial Q_{a,d}} + \frac{\partial J}{\partial Q_{a,b}} \right) + \frac{\partial J}{\partial Q_{a,b}} \kappa.
\end{align*}
\] (23)

**B. PROPERTIES OF REAL VECTOR AND MATRIX DERIVATIVES**

Let us denote
\[
q = q_a + i q_b + j q_c + k q_d \in \mathbb{H},
q_a, q_b, q_c, q_d \in \mathbb{R},
w = w_a + w_b i + w_c j + w_d k \in \mathbb{H}^{N \times 1},
\]
\[
w_a, w_b, w_c, w_d \in \mathbb{R}^{N \times 1},
Q = Q_a + Q_b i + Q_c j + Q_d k \in \mathbb{H}^{N \times M},
\]
\[
Q_a, Q_b, Q_c, Q_d \in \mathbb{R}^{N \times M}.
\]

Some properties of the quaternion vector and matrix transpose are
\[
(u^T v)^* = v^H u^*,
\]
\[
(uv^T)^* = (v^* u^H)^T,
\]
\[
(AQ)^* = (Q^H A^H)^T,
\]
\[
(AQ_q)^* = Q^*_q A^T,
\]
\[
(AQ)^H = Q^H A^H,
\]
\[
(AQ)^T \neq Q^T A^T.
\] (25)

Multiplication between a quaternion value and a real scalar is commutative. Some properties of the quaternion matrix trace are
\[
\begin{align*}
Tr(Q_a A) &= Tr(AQ_a), \\
Tr(QA) &= Tr(A^T Q^T), \\
Tr(QA) &\neq Tr(QA).
\end{align*}
\] (26)

Quaternion matrix derivatives have the following properties:
\[
\frac{\partial Tr(Q A)}{\partial Q} = \frac{1}{2} A^T,
\]
\[
\frac{\partial Tr(Q A)}{\partial Q} = -\frac{1}{2} A^H.
\] (27)

Finally, quaternion matrix derivatives have the following properties:
\[
\begin{align*}
\frac{\partial Tr(Q A)}{\partial Q_a} &= \frac{\partial Tr(Q^T A^T)}{\partial Q_a} = A^T, \\
\frac{\partial Tr(AQ)C}{\partial Q_a} &= \frac{\partial Tr(A^T C^T)}{\partial Q_a} = A^T C^T.
\end{align*}
\] (28)

**C. QUATERNION GRADIENT**

Given the above properties, we can easily obtain the following quaternion gradients with respect to the quaternion vector and matrix variables.

Some gradients of quaternion vectors are
\[
\begin{align*}
\frac{\partial w^H w \alpha}{\partial w^*} &= -\frac{1}{2} w^* \alpha^* , \\
\frac{\partial w^H u}{\partial w^*} &= R \{ u \} \alpha^* ,
\end{align*}
\] (29)

Similarly, the gradients of quaternion matrices are
\[
\begin{align*}
\frac{\partial Tr(AQ)}{\partial Q_a} &= -\frac{1}{2} A^T , \\
\frac{\partial Tr(QA)}{\partial Q_a} &= -\frac{1}{2} A^H.
\end{align*}
\] (30)

and
\[
\begin{align*}
\frac{\partial Tr(BQ^H)}{\partial Q_a} &= B ,
\end{align*}
\]
\[
\frac{\partial Tr(Q^H B)}{\partial Q_a} = R \{ B \} ,
\]
\[
\frac{\partial w^H D w}{\partial w^*} = R \{ D w \} - \left( w^H D \right)^T / 2.
\]
\[
\frac{\partial \text{Tr}(AQ)}{\partial Q} = -\frac{1}{2} A^T c^u,
\]
\[
\frac{\partial \text{Tr}(AQ')}{\partial Q'} = R \{ C^T \} A^T,
\]
\[
\frac{\partial \text{Tr}(Q^H AQ)}{\partial Q'} = R \{ AQ \} - \frac{1}{2} (Q^H A)^T.
\]  
(31)

**Remark 1:** The full derivations of (29), (30), and (31) are given in Appendices A, B, and C, respectively. The gradients are obtained using quaternion involutions, and are the same as in [35] where quaternion rotations are used. In our method, the real derivatives and real chain rules can be easily used.

**IV. APPLICATIONS IN SIGNAL PROCESSING**

Quaternion gradients are used in adaptive filters, Kalman filters, and kernel filters to solve the associated optimization problems.

**A. QUATERNION GRADIENT FOR ADAPTIVE FILTERS**

When the quaternion linear filtering problem is considered in the discrete time domain \(i\), there is a quaternion input vector \(u(i) \in H^n\) with the unknown original quaternion parameter \(w^o \in H^n\) and the desired response \(d(i) \in H^1\). It is assumed that \(d(i)\) is generated by the linear regression model

\[
d(i) = w^o u(i) + v(i),
\]  
(32)

where \(v(i)\) is the additive noise.

The error signal for the quaternion linear filter is defined as

\[
e(i) = d(i) - w^o u(i),
\]

where \(w\) is the estimation of \(w^o\). In [38], we derived the gradient descent formula for \(e(i) = d(i) - w^o u(i)\).

The real-valued cost function based on MSE is given by (1) and (2). For the error signal model (33),

\[
e(i) = d(i)\left[ w^a u(i) - i w^b u(i) - j w^c u(i) - k w^d u(i) \right],
\]

and

\[
e'(i) = d'(i)\left[ w^a u'(i) + w^b u'(i) + w^c u'(i) + w^d u'(i) \right].
\]

Then, the gradients of \(J_1\) and \(J_2\) are obtained as follows:

\[
\frac{\partial J_1}{\partial w} = \frac{\partial J_2}{\partial w} = -\frac{1}{2} u(i) e'(i).
\]  
(36)

The full derivation of (36) is given in Appendix D. The update rule for the quaternion gradient descent algorithm is

\[
w(i+1) = w(i) + \eta u(i) e'(i),
\]

where \(\eta\) is the step size.

**Remark 2:** During the derivations, the relationship between the quaternion conjugate and three perpendicular quaternion involutions is used. The obtained gradient of \(J_1\) is equivalent to that of \(J_2\). This result is consistent with the GHR method in [26, 34, 35, 36], and the quaternion product method [37].

**Remark 3:** The quaternion adaptive filter can also be generalized to the diffusion case for the collaborative processing of quaternion signals over distributed networks. In [32], iQLMS [24-25] was applied to a diffusion network. However, the gradient of \(J_1\) is different from that of \(J_2\) in the iQLMS algorithm, because the quaternion product is non-commutative. A correct approach is to employ (37) in diffusion networks.

**B. QUATERNION GRADIENT FOR KALMAN FILTERS**

In [3] and [5], HR calculus is used for quaternion Kalman filters (KFs) and distributed quaternion KFs, respectively. In this paper, the gradients based on quaternion involutions are used to derive the quaternion KF.

In a quaternion KF, the linear state space model is given by [21]

\[
\begin{align*}
\dot{x}_k &= A_x x_{k-1} + w_k, \\
y_k &= C_x x_k + v_k,
\end{align*}
\]  
(38)

where \(x_k\) is the noisy state (of dimension \(p \times 1\)) to be estimated at instant \(k\), \(y_k\) is the observation (of dimension \(q \times 1\)), \(w_k\) is the state noise vector (of dimension \(p \times 1\)), \(v_k\) is the observation noise vector (of dimension \(q \times 1\)), \(A_x\) is a \(p \times p\) matrix called the state transition matrix, and \(C_x\) is a \(q \times p\) matrix called the observation transition matrix. The state noise \(w_k\) and the observation noise \(v_k\) are zero-mean, discrete, white-noise processes with covariance matrices of \(Q_k\) and \(R_k\).

Given the previous state \(\hat{x}_{k-1|k-1}\), the KF estimates the current state \(\hat{x}_{k|k}\) as a function of the observations \(y_i\), \(0 \leq i \leq k\).

\[
\hat{x}_{k|k} = A_k \hat{x}_{k-1|k-1} + G_k \left( y_k - C_k A_k \hat{x}_{k-1|k-1} \right).
\]  
(39)

We then determine the Kalman gain \(G_k\), which is estimated by minimizing the following cost function based on the MSE criterion:

\[
J_{\text{MSE}}(G_k) = E \left[ e_k^H e_k \right] = Tr \left( P_{k|k} \right),
\]

\[
e_k = x_k - \hat{x}_{k|k} = [e_{k1}, e_{k2}, \ldots, e_{kp}]^T,
\]

\[
P_{k|k} = E \left[ e_k e_k^H \right],
\]

where \(e_k\) is the state error.

The Kalman gain \(G_k\) of (39) is obtained from

\[
\frac{\partial \text{Tr} \left( P_{k|k} \right)}{\partial G_k} = 0.
\]  
It follows from [3] that
\[
\begin{align*}
  P_{j,k} &= P_{j,k-1} - G_k C_j P_{j,k-1} - P_{j,k-1} C_k^H G_k^H + G_k S_k G_k^H, \\
  S_k &= C_j P_{j,k-1} C_k^H + R_k, \quad (41) \\
  P_{j,k-1} &= A_k P_{j,k-1} A_k^H + Q_{j,k-1}^{-1}.
\end{align*}
\]

From (30) and (31), it follows that
\[
\frac{\partial \text{Tr}(G_k S_k G_k^H)}{\partial G_k^*} = -\frac{1}{2} (C_j P_{j,k-1})^H, \quad (42)
\]
\[
\frac{\partial \text{Tr}(P_{j,k-1} C_k^H G_k^H)}{\partial G_k^*} = P_{j,k-1} C_k^H, \quad (43)
\]
\[
\frac{\partial \text{Tr}(G_k S_k G_k^H)}{\partial G_k} = G_k S_k - \frac{1}{2} G_k S_k^H. \quad (44)
\]

The results of (42)–(44) are consistent with the GHR method in [35].

As
\[
\begin{align*}
  S_k &= S_k^H, \\
  P_{j,k-1} &= P_{j,k-1}^H,
\end{align*} \quad (45)
\]

we have
\[
\frac{\partial \text{Tr}(P_{j,k})}{\partial G_k^*} = \frac{1}{2} (C_j P_{j,k-1})^H - P_{j,k-1} C_k^H G_k S_k - \frac{1}{2} G_k S_k^H
\]
\[
= (G_k S_k - P_{j,k-1} C_k^H)/2 = 0,
\]
which gives
\[
G_k = P_{j,k-1} C_k^H (S_k)^{-1}. \quad (47)
\]

Substituting (47) into (41) yields
\[
P_{j,k} = P_{j,k-1} - G_k C_j P_{j,k-1}. \quad (48)
\]

**Remark 4:** In [3], HR calculus was used to calculate the Kalman gain \( G_k \) as follows:
\[
(C_j P_{j,k-1})^T - \frac{1}{2} P_{j,k-1} C_k^H (S_k G_k^H)^T - \frac{1}{2} G_k S_k^H,
\]
which also yields (47). Thus, we obtain the same quaternion KF algorithm as that in [3].

However, the following HR gradients were used in [3]:
\[
\begin{align*}
  \frac{\partial \text{Tr}(G_k S_k G_k^H)}{\partial G_k^*} &= (C_j P_{j,k-1})^T, \quad (50) \\
  \frac{\partial \text{Tr}(P_{j,k-1} C_k^H G_k^H)}{\partial G_k^*} &= P_{j,k-1} C_k^H, \quad (51)
\end{align*}
\]

The results of (50)–(52) based on the HR method are incorrect, and are different from those given by the GHR method in [35] and by our method in (42)–(44).

**C. QUATERNION GRADIENT FOR KERNEL FILTERS**

Kernel methods work well in coping with the nonlinear problems using the kernel trick [39]. In [40], HR calculus is used for quaternion kernel maximum correntropy (KMC). In this paper, the gradients of quaternion involutions are used to derive the quaternion KMC.

For a set of \( n \) input–output pairs \(((u_1, d_1), (u_2, d_2), \cdots, (u_n, d_n))\), \( u \in \mathbb{U} \subseteq \mathbb{H}^m \) is mapped into a potentially infinite-dimensional feature space \( \mathcal{F} \). The feature-space parametric model becomes
\[
y(i) = w^H \varphi(i), \quad (53)
\]
where \( \varphi(i) = \psi(u_i) \), and \( \psi(u) \) is a kernel map to a quaternion reproducing kernel Hilbert space (RKHS) [39]. We have
\[
K_o(u_i, u_j) = \langle \psi(u_i), \psi(u_j) \rangle. \quad (54)
\]

In quaternion KMC, the cost function is given by [39]
\[
J(n) = \frac{1}{N} \sum_{i=0}^{N-1} K_o[d(i), y(i)], \quad (55)
\]

where \( d_o \) is the desired response, \( y_o \) is the filter output, \( N \) is the number of samples, \( \sigma \) is the kernel size, and the Gaussian kernel for quaternion inputs [40] is
\[
K_o(d, y) = \frac{4}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(d - y)^2}{2\sigma^2}\right)
\]
\[
= \frac{4}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(d - y)(d - y)^T\right), \quad (56)
\]

\[
= \frac{4}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}((d_x - y_x)^2 + (d_y - y_y)^2 + (d_z - y_z)^2\right)\right)
\]

The KMC aims to maximize the correntropy between \( d_o \) and \( y_o \) using gradient ascent. The weight update rule of the quaternion gradient descent algorithm is
\[
\begin{align*}
  w(i+1) &= w(i) + \eta \frac{\partial f(i)}{\partial w} \\
  &= w(i) + \eta \frac{\partial}{\partial w} K_o[d(i) - y(i)] \quad (57) \\
  &= w(i) + \eta \frac{\partial}{\partial w} \exp\left(-\frac{1}{2\sigma^2}|d(i) - w^H \varphi(i)|^2\right).
\end{align*}
\]
where the scaling factor \( 4/\sqrt{2\pi \sigma^2} \) is absorbed into the step size \( \eta \).

Let us denote

\[
e(i) = d(i) - w^H_i \varphi(i) \]

\[
\phi_e(i) = \exp \left[ -\frac{e(i)e^*(i)}{2\sigma^2} \right].
\]

The quaternion gradient can be calculated as

\[
\frac{\partial \phi_e(i)}{\partial w^*} = \frac{\partial}{\partial w^*} \left[ \frac{-e(i)e^*(i)}{2\sigma^2} \right] \phi_e(i).
\]

and it follows from (36) that

\[
\frac{\partial \phi_e(i)}{\partial w^*} = \frac{\partial}{\partial w^*} \left[ \frac{-e(i)e^*(i)}{2\sigma^2} \right] \phi_e(i)
\]

The update rule for the quaternion gradient ascent algorithm is

\[
w(i+1) = w(i) + \eta \phi_e(i) \varphi(i) e^*(i).
\]

The kernel filter output is

\[
y(n) = w^H_n \varphi(n)
\]

\[
= \eta \sum_{i=0}^{n-1} \phi_e(i) \left[ \varphi(i)e^*(i) \right]^H \varphi(n)
\]

\[
= \eta \sum_{i=0}^{n-1} \phi_e(i) \left[ \varphi^H(i) \varphi(n) \right]
\]

\[
= \eta \sum_{i=0}^{n-1} \phi_e(i) e(i) \psi(u_i, u_n)
\]

Using the kernel trick, we obtain a new quaternion KMC algorithm

\[
y(n) = \eta \sum_{i=0}^{n-1} \phi_e(i) e(i) K_n(u_i, u_n),
\]

which has a similar expression to the real KMC algorithm in [41].

**Remark 5:** Compared with the KMC algorithms in the real domain, the proposed corresponding algorithms in the quaternion domain are generic extensions. In contrast, previous quaternion KMC algorithms [40] used the HR gradients, which have different expressions for \( J_1 \) and \( J_2 \).

When HR gradients are used, we have

\[
\frac{\partial J_1}{\partial \omega^*} = \frac{-\varphi(i)e^*(i) + 0.5e(i)\varphi^*(i)}{w^*},
\]

\[
\frac{\partial J_2}{\partial \omega^*} = \frac{-e^*(i)\varphi(i) + 0.5\varphi^*(i)e(i)}{w^*}.
\]

Substituting (65) into (53) yields

\[
y(n) = w^H_n \varphi(n)
\]

\[
= \eta \sum_{i=0}^{n-1} \phi_e(i) \left[ \varphi(i)e^*(i) - 0.5e(i)\varphi^*(i) \right]^H \varphi(n).
\]

The quaternion KMC [40] algorithm is based on (66). Thus, there will be two different expressions for the quaternion KMC [40]. Moreover, we can easily use the kernel trick using the proposed quaternion gradient. On the other side, it is difficult for the early proposed quaternion KMC [40] to use the kernel trick, and its expression is very cumbersome.

**V. SIMULATION**

We tested the proposed quaternion KMC and quat-KMC derived in [40] using both Q-proper and Q-improper input sequence. The input was generated as

\[
x(n) = g_u(n) + g_{v'}(n) + g_{v''}(n) + g_{u'}(n)
\]

\[
+ h_u(n-1) + h_{v'}(n-1) + h_{v''}(n-1) + h_{u'}(n-1),
\]

and the output was nonlinear,

\[
y(n) = x(n) + ax^2(n) + bx^3(n).
\]

The quaternion coefficients in (68) and (69) were
The Q-proper input sequence can be expressed as follows

\[ u(n) = 0.5u_a(n) + 0.5u_b(n) + 0.5u_c(n) + 0.5u_d(n) \kappa; \quad (71) \]

while Q- improper input sequence is

\[ u(t) = \sqrt{28/32}u_a(n) + \sqrt{1/16}u_b(n) + \sqrt{1/32}u_c(n) + \sqrt{1/32}u_d(n) \kappa; \quad (72) \]

where \( u_a(n), u_b(n), u_c(n), u_d(n) \) are all formed using uniform distribution models. Gauss noise was added to the output whose power was -17dBm. We set the step size \( \eta = 0.1 \) and band width \( \sigma = 1 \) for quat-KMC [40] while \( \eta = 0.7 \) and \( \sigma = 1 \) for proposed QKMC for the best performance. The simulation result was shown as follows.

![FIGURE 1. Comparison between quaternion KMC and quat-KMC in Q-proper input.](image1)

![FIGURE 2. Comparison between quaternion KMC and quat-KMC in Q-improper input.](image2)

\[
\begin{align*}
g_1 &= -0.45 - 0.4t + 0.3j + 0.15k \\
h_1 &= 0.15 + 0.175t - 0.025j + 0.1k \\
g_2 &= 0.2 - 0.35r - 0.15j - 0.05k \\
h_2 &= -0.075 + 0.15t - 0.22j + 0.125k \\
g_3 &= -0.05 - 0.4r - 0.4j - 0.2k \\
h_3 &= -0.05 + 0.025t + 0.075j - 0.05k \\
g_4 &= 0.15 + 0.35r + 0.1j - 0.1k \\
h_4 &= 0.175 - 0.05t - 0.07j - 0.075k \\
a &= -0.05 + 0.075t + 0.35j + 0.1k \\
b &= 0.03 - 0.025t - 0.25j + 0.05k.
\end{align*}
\] (70)

VI. CONCLUSION

In this paper, we have generalized the quaternion involutions method to calculate the quaternion gradient of the quaternion vector and matrix function, and derived gradient expressions in the quaternion domain. In the quaternion involutions method, the quaternion gradient is transformed from the quaternion domain to the real domain of four times the dimension, and the real derivatives and real chain rules can easily be used.

The proposed gradient operators have been applied to adaptive filters, Kalman filters, and kernel filters to solve the associated optimization problems. In the adaptive filter, the obtained gradient is the same as that of the GHR method, and can be generalized to diffusion networks in place of iQLMS. For Kalman filters, the obtained gradient of the trace of the covariance matrix for a state is different from that of the HR method [3], but the resulting Kalman gain is the same as that of the HR method. For kernel filters, previous quaternion KMC algorithms used the HR gradients, which have different expressions for J1 and J2; the proposed KMC algorithms have similar expressions to the corresponding algorithms in the real domain, and can be regarded as generic extensions of real-valued KMC algorithms.

APPENDIX A: DERIVATION OF (29)

The cost function \( J(w) = u^H w \alpha \) can be written as

\[ J(w) = u^H w \alpha = u^H (w_a + w_b t + w_c j + w_d k) \alpha. \] (A1)

Then, we have
\[
\frac{\partial J}{\partial w_a} = \frac{\partial u^H w_a u}{\partial w_a} = u^\ast_a, \tag{A2}
\]
\[
\frac{\partial J}{\partial w_b} = \frac{\partial u^H w_b u}{\partial w_b} = \frac{\partial w^T b u}{\partial w_b} = u^\ast b, \tag{A3}
\]
\[
\frac{\partial J}{\partial w_c} = \frac{\partial u^H w_c u}{\partial w_c} = u^\ast c, \tag{A4}
\]
\[
\frac{\partial J}{\partial w_d} = \frac{\partial u^H w_d u}{\partial w_d} = \frac{\partial w^T d u}{\partial w_d} = u^\ast d. \tag{A5}
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(w) = u^H w^a u \).

\[
\frac{\partial J}{\partial w^a} = \left( u^\ast + u^3 a^1 + u^1 a^3 + u^3 a^1 a^3 \right) / 4 = u^\ast / 2. \tag{A6}
\]

The cost function \( J(w) = u^H w^a u = u^\ast \left( w^T_a - w^T_b j - w^T_c k - w^T_d \right) u \). \tag{A7}

Then, we have

\[
\frac{\partial J}{\partial w^a} = \frac{\partial u^H w^a u}{\partial w^a} = \frac{\partial w^T a u}{\partial w^a} = u^\ast u, \tag{A8}
\]
\[
\frac{\partial J}{\partial w^b} = \frac{\partial u^H w^b u}{\partial w^b} = \frac{\partial w^T b u}{\partial w^b} = u^\ast b, \tag{A9}
\]
\[
\frac{\partial J}{\partial w^c} = \frac{\partial u^H w^c u}{\partial w^c} = \frac{\partial w^T c u}{\partial w^c} = u^\ast c, \tag{A10}
\]
\[
\frac{\partial J}{\partial w^d} = \frac{\partial u^H w^d u}{\partial w^d} = \frac{\partial w^T d u}{\partial w^d} = u^\ast d. \tag{A11}
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(w) = u^H w^a u \).

\[
\frac{\partial J}{\partial w^a} = \left( u^\ast + u^3 a^1 + u^1 a^3 + u^3 a^1 a^3 \right) / 4 = u^\ast / 2. \tag{A12}
\]

The cost function \( J(w) = u^H w^a u = u^\ast \left( w^T_a - w^T_b j - w^T_c k - w^T_d \right) u \) can be written as

\[
J(w) = u^H \bar{w} u = u^H \left( w^T_a - w^T_b j + w^T_c k - w^T_d \right) u. \tag{A13}
\]

Then, we have

\[
\frac{\partial J}{\partial w^a} = \frac{\partial u^H \bar{w} u}{\partial w^a} = \frac{\partial w^T a u}{\partial w^a} = u^\ast a, \tag{A14}
\]
\[
\frac{\partial J}{\partial w^b} = \frac{\partial u^H \bar{w} u}{\partial w^b} = \frac{\partial w^T b u}{\partial w^b} = u^\ast b, \tag{A15}
\]
\[
\frac{\partial J}{\partial w^c} = \frac{\partial u^H \bar{w} u}{\partial w^c} = \frac{\partial w^T c u}{\partial w^c} = u^\ast c, \tag{A16}
\]
\[
\frac{\partial J}{\partial w^d} = \frac{\partial u^H \bar{w} u}{\partial w^d} = \frac{\partial w^T d u}{\partial w^d} = u^\ast d. \tag{A17}
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(w) = u^H w^a u \).

\[
\frac{\partial J}{\partial w^a} = \left( u^\ast + u^3 a^1 + u^1 a^3 + u^3 a^1 a^3 \right) / 4 = u^\ast / 2. \tag{A18}
\]

The cost function \( J(w) = u^H D w^a u = w^T a D \left( w^T_a + w^T_b j + w^T_c k \right) D w^a u \) can be written as

\[
J(w) = u^H D w^a u = u^H \left( w^T_a + w^T_b j + w^T_c k \right) D w^a u = u^H \left( w^T_a - w^T_b j - w^T_c k \right) D w^a u. \tag{A19}
\]

Then, we have

\[
\frac{\partial J}{\partial w^a} = \frac{\partial u^H D w^a u}{\partial w^a} = \frac{\partial w^T a u}{\partial w^a} = u^\ast a, \tag{A20}
\]
\[
\frac{\partial J}{\partial w^b} = \frac{\partial u^H D w^a u}{\partial w^b} = \frac{\partial w^T b u}{\partial w^b} = u^\ast b, \tag{A21}
\]
\[
\frac{\partial J}{\partial w^c} = \frac{\partial u^H D w^a u}{\partial w^c} = \frac{\partial w^T c u}{\partial w^c} = u^\ast c, \tag{A22}
\]
\[
\frac{\partial J}{\partial w^d} = \frac{\partial u^H D w^a u}{\partial w^d} = \frac{\partial w^T d u}{\partial w^d} = u^\ast d. \tag{A23}
\]
The gradient of \( \mathbf{J} \) is
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{w}} = \mathbf{D} \mathbf{w} - \mathbf{1} \mathbf{D} \mathbf{w} \mathbf{1},
\]
where \( \mathbf{D} \) is the Jacobian matrix of \( \mathbf{J} \) and \( \mathbf{w} \) is the input vector.

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( \mathbf{J} = \mathbf{w}^H \mathbf{D} \mathbf{w} \).

The cost function \( \mathbf{J}(\mathbf{Q}) = \mathbf{Tr}(\mathbf{A} \mathbf{Q}) \) can be written as
\[
\mathbf{J}(\mathbf{Q}) = \mathbf{Tr}(\mathbf{A} \mathbf{Q}) = \mathbf{Tr}[(\mathbf{Q}_a + \mathbf{Q}_d + \mathbf{Q}_j + \mathbf{Q}_k) \mathbf{A}].
\]

Then, we have
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_a} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_a \mathbf{A})}{\partial \mathbf{Q}_a} = \mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_b} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_b \mathbf{A})}{\partial \mathbf{Q}_b} = -\mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_c} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_c \mathbf{A})}{\partial \mathbf{Q}_c} = -\mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_d} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_d \mathbf{A})}{\partial \mathbf{Q}_d} = -\mathbf{A}.
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( \mathbf{J} = \mathbf{w}^H \mathbf{D} \mathbf{w} \).

The cost function \( \mathbf{J}(\mathbf{Q}) = \mathbf{Tr}(\mathbf{A} \mathbf{Q}) \) can be written as
\[
\mathbf{J}(\mathbf{Q}) = \mathbf{Tr}(\mathbf{A} \mathbf{Q}) = \mathbf{Tr}[(\mathbf{Q}_a + \mathbf{Q}_d + \mathbf{Q}_j + \mathbf{Q}_k) \mathbf{A}].
\]

Then, we have
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_a} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_a \mathbf{A})}{\partial \mathbf{Q}_a} = \mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_b} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_b \mathbf{A})}{\partial \mathbf{Q}_b} = -\mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_c} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_c \mathbf{A})}{\partial \mathbf{Q}_c} = -\mathbf{A},
\]
\[
\frac{\partial \mathbf{J}}{\partial \mathbf{Q}_d} = \frac{\partial \mathbf{Tr}(\mathbf{Q}_d \mathbf{A})}{\partial \mathbf{Q}_d} = -\mathbf{A}.
\]
\[
\frac{\partial J}{\partial Q_b} = -\frac{\partial \text{Tr}(Q_b^tB)}{\partial Q_b} - t = -B_t, \quad (B21)
\]

\[
\frac{\partial J}{\partial Q_c} = -\frac{\partial \text{Tr}(Q_c^tJ)}{\partial Q_c} = -J B_c, \quad (B22)
\]

\[
\frac{\partial J}{\partial Q_d} = -\frac{\partial \text{Tr}(Q_d^t\kappa B)}{\partial Q_d} = -\kappa B_d. \quad (B23)
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(Q^TB) \).

\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left( \frac{\partial J}{\partial Q_a} + \frac{\partial J}{\partial Q_b} t + \frac{\partial J}{\partial Q_c} J + \frac{\partial J}{\partial Q_d} \kappa \right) = (B24)
\]

**APPENDIX C: DERIVATION OF (31)**

The cost function \( J(Q) = \text{Tr}(AQC) \) can be written as

\[
J(Q) = \text{Tr}(AQC) = \text{Tr}[A(Q_a + Q_b t + Q_c J + Q_d \kappa) C]. \quad (C1)
\]

Then, we have

\[
\frac{\partial J}{\partial Q_a} = \frac{\partial \text{Tr}(AQ_a C)}{\partial Q_a} = A^T C^T. \quad (C2)
\]

\[
\frac{\partial J}{\partial Q_b} = \frac{\partial \text{Tr}(AQ_b C)}{\partial Q_b} t = A^T C^T t, \quad (C3)
\]

\[
\frac{\partial J}{\partial Q_c} = \frac{\partial \text{Tr}(AQ_c C)}{\partial Q_c} J = A^T J C^T J, \quad (C4)
\]

\[
\frac{\partial J}{\partial Q_d} = \frac{\partial \text{Tr}(AQ_d C)}{\partial Q_d} \kappa = A^T \kappa C^T J. \quad (C5)
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(AQC) \).

\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left( \frac{\partial J}{\partial Q_a} + \frac{\partial J}{\partial Q_b} t + \frac{\partial J}{\partial Q_c} J + \frac{\partial J}{\partial Q_d} \kappa \right) = (C6)
\]

The cost function \( J(Q) = \text{Tr}(AQ' C) \) can be written as

\[
J(Q) = \text{Tr}(AQ'C) = \text{Tr}[A(Q_a - Q_b t - Q_c J - Q_d \kappa) C]. \quad (C7)
\]

Then, we have

\[
\frac{\partial J}{\partial Q_a} = \frac{\partial \text{Tr}(AQ_a C)}{\partial Q_a} = A^T C^T, \quad (C8)
\]

\[
\frac{\partial J}{\partial Q_b} = \frac{\partial \text{Tr}(AQ_b C)}{\partial Q_b} t = -A^T C^T t, \quad (C9)
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(AQ'C) \).

\[
\frac{\partial J}{\partial Q_a} = \frac{\partial \text{Tr}(AQ_a C)}{\partial Q_a} = A^T C^T, \quad (C10)
\]

\[
\frac{\partial J}{\partial Q_b} = \frac{\partial \text{Tr}(AQ_b C)}{\partial Q_b} t = -A^T \kappa C^T J. \quad (C11)
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(AQ'C) \).

\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left( \frac{\partial J}{\partial Q_a} + \frac{\partial J}{\partial Q_b} t + \frac{\partial J}{\partial Q_c} J + \frac{\partial J}{\partial Q_d} \kappa \right) = (C12)
\]

The cost function \( J(Q) = \text{Tr}(Q^TAQ') \) can be written as

\[
J(Q) = \text{Tr}(Q^TAQ') = \text{Tr}[Q(AQ_a + Q_b t + Q_c J + Q_d \kappa)AQ'] \quad (C13)
\]

Then, we have

\[
\frac{\partial J}{\partial Q_a} = \frac{\partial \text{Tr}(Q_a AQ')}{\partial Q_a} + \frac{\partial \text{Tr}(QAQ')}{\partial Q_a} = (C14)
\]

\[
\frac{\partial J}{\partial Q_b} = \frac{\partial \text{Tr}(Q_b AQ')}{\partial Q_b} t + \frac{\partial \text{Tr}(QAQ')}{\partial Q_b} t = (C15)
\]

\[
\frac{\partial J}{\partial Q_c} = \frac{\partial \text{Tr}(Q_c AQ')}{\partial Q_c} J + \frac{\partial \text{Tr}(QAQ')}{\partial Q_c} J = (C16)
\]

\[
\frac{\partial J}{\partial Q_d} = \frac{\partial \text{Tr}(Q_d AQ')}{\partial Q_d} \kappa + \frac{\partial \text{Tr}(QAQ')}{\partial Q_d} \kappa = (C17)
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(Q^TAQ') \).

\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left( \frac{\partial J}{\partial Q_a} + \frac{\partial J}{\partial Q_b} t + \frac{\partial J}{\partial Q_c} J + \frac{\partial J}{\partial Q_d} \kappa \right) = (C18)
\]

The cost function \( J(Q) = \text{Tr}(Q^TAQ) \) can be written as
Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(Q^H A Q) \). Then, we have

\[
\frac{\partial J}{\partial Q_a} = \frac{\partial \text{Tr}(Q^H AQ)}{\partial Q_a} + \frac{\partial \text{Tr}(Q^t AQ)}{\partial Q_a}
= (Q^H A)^T + iAQ,
\]

\[
\frac{\partial J}{\partial Q_b} = \frac{\partial \text{Tr}(Q^H AQ)}{\partial Q_b} - \frac{\partial \text{Tr}(Q^t AQ)}{\partial Q_b}
= -(Q^H A)^T - iAQ t,
\]

\[
\frac{\partial J}{\partial Q_c} = \frac{\partial \text{Tr}(Q^H AQ)}{\partial Q_c} - \frac{\partial \text{Tr}(Q^t AQ)}{\partial Q_c}
= -(Q^H A)^T - iAQ t
\]

Using the quaternion involutions (9) and property (11), we obtain the gradient of \( J(Q) = \text{Tr}(Q^H A Q) \).

\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left[ \frac{\partial J}{\partial Q_a} + \frac{\partial J}{\partial Q_b} + \frac{\partial J}{\partial Q_c} + \frac{\partial J}{\partial Q_d} \right]
= \left( (Q^H A)^T / 2 + [AQ + (AQ)^T + (AQ)^T + (AQ)^T] / 4 \right)
= R\{AQ\} - (Q^H A)^T / 2.
\]

**APPENDIX D: DERIVATION OF (36)**

The cost function \( J_1(w) = e(i)e^*(i) \) can be written as

\[
J_1(w) = (d(i) - [w_0^Tw(i) - jw_1^Tw(i) - jw_2^Tw(i) - jw_3^Tw(i)]) \ e^*(i)
= e(i) \left[ d^*(i) - \left[ w_0^Tw^*(i) + w_1^Tw^*(i) + jw_2^Tw^*(i) + jw_3^Tw^*(i) \right] \right].
\]

Then, we have

\[
\frac{\partial J}{\partial w_a} = \frac{\partial e(i)w_a^Tw^*(i) - e(i)w_a^Tu(i)e^*(i)}{\partial w_a}
= -e(i)u^*(i) - u(i)e^*(i),
\]

\[
\frac{\partial J}{\partial w_b} = \frac{\partial e(i)w_b^Tw^*(i) - e(i)w_b^Tu(i)e^*(i)}{\partial w_b}
= e(i)u^*(i) + u(i)e^*(i),
\]

\[
\frac{\partial J}{\partial w_c} = \frac{\partial e(i)w_c^Tw^*(i) - e(i)w_c^Tu(i)e^*(i)}{\partial w_c}
= e(i)u^*(i) + u(i)e^*(i),
\]

\[
\frac{\partial J}{\partial w_d} = \frac{\partial e(i)w_d^Tw^*(i) - e(i)w_d^Tu(i)e^*(i)}{\partial w_d}
= e(i)u^*(i) + u(i)e^*(i).
\]

The cost function \( J_2(w) = e^*(i)e(i) \) can be written as

\[
J_2(w) = e^*(i) \left[ d(i) - [w_0^Tw(i) - jw_1^Tw(i) - jw_2^Tw(i) - jw_3^Tw(i)] \right] e(i)
= e^*(i) \left[ d^*(i) - \left[ w_0^Tw^*(i) + w_1^Tw^*(i) + jw_2^Tw^*(i) + jw_3^Tw^*(i) \right] \right] e(i).
\]

Then, we have

\[
\frac{\partial J}{\partial w_a} = \frac{\partial e^*(i)w_a^Tu(i) - \partial e^*(i)w_a^Tu(i)e^*(i)}{\partial w_a}
= -e(i)u(i) - u^*(i)e(i),
\]

\[
\frac{\partial J}{\partial w_b} = \frac{\partial e^*(i)w_b^Tu(i) - \partial e^*(i)w_b^Tu(i)e^*(i)}{\partial w_b}
= e(i)u(i) - u^*(i)e(i),
\]

\[
\frac{\partial J}{\partial w_c} = \frac{\partial e^*(i)w_c^Tu(i) - \partial e^*(i)w_c^Tu(i)e^*(i)}{\partial w_c}
= e(i)u(i) - u^*(i)e(i),
\]

\[
\frac{\partial J}{\partial w_d} = \frac{\partial e^*(i)w_d^Tu(i) - \partial e^*(i)w_d^Tu(i)e^*(i)}{\partial w_d}
= e(i)u(i) - u^*(i)e(i).
\]
\[
\frac{\partial J}{\partial Q} = \frac{1}{4} \left( \frac{\partial J}{\partial Q_1} + \frac{\partial J}{\partial Q_2} + \frac{\partial J}{\partial Q_3} + \frac{\partial J}{\partial Q_4} \right) \\
= -e^e \left[ u + u^* + u^* \right] / 4 \\
- u^* \left[ e - e^* - e^* - e^* \right] / 4 \\
= -e^e \left[ (u(i) + u^*(i)) / 2 + u^*(i) e^e(i) / 2 \right] \\
= -u^* e^e(i) / 2.
\]
(D13)

REFERENCES


