electronics

MDPI

Article

One-Dimensional Quaternion Discrete Fourier Transform and an Approach to Its Fast Computation

Dorota Majorkowska-Mech *,† **and Aleksandr Cariow** †

Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, Zolnierska 49, 71-210 Szczecin, Poland; atariov@zut.edu.pl

* Correspondence: dmajorkowska@zut.edu.pl

† These authors contributed equally to this work.

Abstract: This paper proposes a new method for calculating the quaternion discrete Fourier transform for one-dimensional data. Although the computational complexity of the proposed method still belongs to the $O(N \log_2 N)$ class, it allows us to reduce the total number of arithmetic operations required to perform it compared to other known methods for computing this transform. Moreover, compared to the method using symplectic decomposition, the presented method does not require changing the basis in the subspace of pure quaternions and, consequently, calculating the new basis vectors and change-of-basis matrix.

Keywords: one-dimensional quaternion discrete Fourier transform; symplectic decomposition; fast Fourier transform

Citation: Majorkowska-Mech, D.; Cariow, A. One-Dimensional Quaternion Discrete Fourier Transform and an Approach to Its Fast Computation. *Electronics* **2023**, *12*, 4974. https://doi.org/10.3390/electronics12244974

Academic Editor: Nakkeeran Kaliyaperumal

Received: 5 November 2023
Revised: 8 December 2023
Accepted: 10 December 2023
Published: 12 December 2023

1. Introduction

The discrete Fourier transform (DFT) is a fundamental tool for data processing and analysis in many fields of science and technology [1]. The DFT is especially used in digital signal and image processing. If a one-dimensional discrete Fourier transform is used in the processing and analysis of one-dimensional signals, such as speech and other audio signals [2,3], then in the case of image processing and recognition, we are dealing with a two-dimensional DFT [4–9].

The rapid development of signal and image processing technologies has led to advanced data processing techniques using data representation and processing in the hypercomplex domain. This led to the proposal to represent each pixel of a color image as a quaternion [10–14].

The old approach to using color channels to process color images was to use separate channels for the primary colors red, green, and blue. This approach divided the color image into three separate channels. In fact, three 2D matrices were formed, as a result of the presence of three channels for color images, and the computations with each matrix were carried out independently in grayscale. With this approach, due to channel separation, the cross-correlation between channels was lost at the start of the process because the color image was not considered as a whole [14,15].

In the quaternion-based method, a color image was considered as one matrix consisting of pure quaternion numbers. Such a matrix is processed as a whole. In order to convert an RGB image into a quaternion matrix, the RGB channels were simply inserted into the vector part of the quaternion matrix. In addition, such a representation of the pixel matrix made it possible to significantly reduce the computational complexity of color image processing methods compared to three-channel processing.

A natural development of research within the framework of the quaternion-valued representation of pixels in color images was the definition of a two-dimensional quaternion discrete Fourier transform (2-D QDFT) and the development of fast algorithms for 2-D QDFT implementation [12,16–19].

As for the one-dimensional QDFT, this issue has not been studied in detail, since many computer scientists simply did not see the use of such a transform. That is why there are relatively few research studies on this topic [16,19–24]. In this article, we would like to fill this gap and give a detailed definition of 1-D QDFT, as well as propose a fast method for its implementation.

2. Quaternions

Quaternions were introduced by Hamilton in 1843 [25]. They fulfill the usual rules of algebra, but their multiplication is not commutative. Quaternions have four components, one real and three imaginary. In usual notation, which comes from complex numbers, the quaternion q is written in the form

$$q = a + bi + cj + dk \tag{1}$$

where a, b, c, and d are real numbers, and i, j, k are imaginary units that obey the following rules of multiplication:

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

$$ij = k, \quad ji = -k, \quad jk = i, \quad kj = -i, \quad ki = j, \quad ik = -j \tag{3}$$

Quaternions are also called hypercomplex numbers (together with coquaternions, biquaternions, tessarines, octonions, sedenions, etc.). A quaternion has a real part—a in (1)—and an imaginary part, which has three components and is usually denoted by $V(q) = bi + cj + dk$. The imaginary part $V(q)$ of a quaternion may be associated with a 3-space vector $[b, c, d]^T$, and it is often called the vector part of the quaternion. For this reason, the real part is often called the scalar part of the quaternion q, and denoted by $S(q)$. The whole quaternion may be represented as $q = S(q) + V(q)$, i.e., the sum of its scalar and vector parts. A quaternion in which the real (or scalar) part is equal to zero is called a pure quaternion.

The conjugate of a quaternion q is designed by $\overline{q}$ and obtained, as in complex numbers, by negating the imaginary part, so

$$\overline{q} = a - bi - cj - dk \tag{4}$$

The modulus of a quaternion q follows the definition for complex numbers, so

$$|q| = \sqrt{q\overline{q}} = \sqrt{a^2 + b^2 + c^2 + d^2} \tag{5}$$

A quaternion with a unit modulus is called a unit quaternion. The inverse of a quaternion q, denoted by q^{-1}, such that $qq^{-1} = q^{-1}q = 1$ is given by

$$q^{-1} = \frac{\overline{q}}{|q|^2} \tag{6}$$

Euler's formula for the complex exponential is also true for quaternions, so

$$e^{\mu\phi} = \cos\phi + \mu \sin\phi \tag{7}$$

where μ is a unit pure quaternion and ϕ is a real number–angle. Any quaternion q may be represented in the polar form as

$$q = |q|e^{\mu\phi} \tag{8}$$

where μ is called the eigenaxis, and ϕ is called the eigenangle of the quaternion q.

A quaternion q may be also represented as a generalized complex number [26]

$$q = w + rj \tag{9}$$

where its real and imaginary parts are ordinary complex numbers

$$w = a + bi \qquad r = c + di \tag{10}$$

The imaginary units i and j should be different and orthogonal (where the orthogonality of pure quaternions is understood as for vectors in $\mathbb{R}^3$ vector space). Definition (9) of a quaternion is known as the Cayley–Dickson form. Using this form, the quaternion q may be written as

$$q = (a + bi) + (c + di)j \tag{11}$$

where it can be seen by multiplying out, using the rules (3), that the result is the same as in the definition (1). The two parts $w = a + bi$ and $r = c + dj$ are called the simplex and perplex parts of the quaternion q, respectively, and the decomposition of the quaternion into these parts is known as symplectic decomposition [26]. Such a decomposition of a quaternion can be used to calculate the QDFT, as will be shown in Section 4.

The product of two quaternions $q = a + bi + cj + dk = S(q) + V(q)$ and $\hat{q} = \hat{a} + \hat{b}i + \hat{c}j + \hat{d}k = S(\hat{q}) + V(\hat{q})$ can be represented by the products of their scalar and vector parts

$$q\hat{q} = a\hat{a} - (b\hat{b} + c\hat{c} + d\hat{d}) + a(\hat{b}i + \hat{c}j + \hat{d}k) +$$

$$\hat{a}(bi + cj + dk) + (c\hat{d} - d\hat{c})i + (d\hat{b} - b\hat{d})j + (b\hat{c} - c\hat{b})k =$$

$$S(q)S(\hat{q}) - V(q) \cdot V(\hat{q}) + S(q)V(\hat{q}) + S(\hat{q})V(q) + V(q) \times V(\hat{q}) \tag{12}$$

where "$\cdot$" denotes the scalar product and "$\times$" denotes the cross product of vectors. Generally, quaternion multiplication is neither commutative nor anti-commutative because the dot product is commutative and the cross product is anti-commutative. When q and $\hat{q}$ are pure quaternions, then $S(q) = S(\hat{q}) = 0$, so $q\hat{q} = V(q) \cdot V(\hat{q}) + V(q) \times V(\hat{q})$. If, moreover, $V(q) \cdot V(\hat{q}) = 0$, we say that pure quaternions q and $\hat{q}$ are perpendicular ($q \perp \hat{q}$). Then $q\hat{q} = V(q) \times V(\hat{q})$ and $q\hat{q} = -\hat{q}q$. In general, two quaternions whose vector parts are parallel, in the usual vector sense, are called parallel quaternions. Likewise, if their vector parts are perpendicular, they are called perpendicular quaternions [11].

3. Two Types of Quaternion Discrete Fourier Transform

Although the quaternion Fourier transform is mainly used for two-dimensional data, i.e., images, it has been specified for one-dimensional data as well [19,21]. We will only deal with the discrete version of the one-dimensional quaternion Fourier transform.

Let $x^{(r)}(n)$, $x^{(i)}(n)$, $x^{(j)}(n)$, and $x^{(k)}(n)$ be the components of a discrete quaternion signal $x(n)$ with the number of samples N. Then

$$x(n) = x^{(r)}(n) + x^{(i)}(n)i + x^{(j)}(n)j + x^{(k)}(n)k \tag{13}$$

for $n = 0, \ldots, N-1$, where $x^{(r)}(n)$, $x^{(i)}(n)$, $x^{(j)}(n)$, and $x^{(k)}(n)$ are real numbers.

Let μ be a unit pure quaternion and μ_i, μ_j, μ_k be its coefficients associated with the imaginary units i, j, k, respectively:

$$\mu = \mu_i i + \mu_j j + \mu_k k \tag{14}$$

where

$$|\mu|^2 = \mu\overline{\mu} = (\mu_i)^2 + (\mu_j)^2 + (\mu_k)^2 = 1 \tag{15}$$

The choice of μ is arbitrary, but it matters. In applications, the case $\mu_i = \mu_j = \mu_k$ is often used; thus, taking into account (15), we obtain $\mu = (i + j + k)/\sqrt{3}$.

Since quaternion multiplication is not commutative, two different 1-D quaternion discrete Fourier transforms were defined, which are referred to as right-side and left-side QDFTs.

The right-side 1-D quaternion discrete Fourier transform is defined as follows:

$$y(m) = \sum_{n=0}^{N-1} x(n) e^{-\frac{\mu 2\pi mn}{N}} \tag{16}$$

for $m = 0, \ldots, N-1$, where the quaternion exponents are to the right of $x(n)$, and μ is any unit pure quaternion.

It should be noted that for any choice of μ, $\mu^2 = -1$, so this transform can be regarded as a generalization of the standard complex discrete Fourier transform in which the imaginary unit i has been generalized to a vector imaginary unit μ. This means that if $\mu = i$ and the input signal is complex-valued, that is all samples have values $(x^{(r)}(n) + x^{(i)}i)$, then this transform is reduced to the usual complex-valued discrete Fourier transform.

According to (7), the quaternion exponent can be written in the form

$$e^{-\frac{\mu 2\pi mn}{N}} = \cos\frac{2\pi mn}{N} - \mu \sin\frac{2\pi mn}{N} \tag{17}$$

or, if we introduce the denotations

$$a_{m,n} = \cos\frac{2\pi mn}{N}, \qquad b_{m,n} = \sin\frac{2\pi mn}{N} \tag{18}$$

and take into account (14), the quaternion exponent can by written as

$$e^{-\frac{\mu 2\pi mn}{N}} = a_{m,n} - (\mu_i i + \mu_j j + \mu_k k) b_{m,n} \tag{19}$$

Putting expressions (13) and (19) into (16), the right-side quaternion discrete Fourier transform can be written in the form

$$y(m) = \sum_{n=0}^{N-1} [x^{(r)}(n) + x^{(i)}(n)i + x^{(j)}(n)j + x^{(k)}(n)][a_{m,n} - (\mu_i i + \mu_j j + \mu_k k) b_{m,n}] \tag{20}$$

After multiplying the expressions in square brackets, according to the rules (2) and (3) of multiplication of imaginary units, we can write the result as

$$y(m) = y^{(r)}(m) + y^{(i)}(m)i + y^{(j)}(m)j + y^{(k)}(m)k \tag{21}$$

where $y^{(r)}(m)$, $y^{(i)}(m)$, $y^{(j)}(m)$, and $y^{(k)}(m)$ are real numbers. It is easy to check that they take the following forms:

$$y^{(r)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(r)}(n) a_{m,n} + \mu_i x^{(i)}(n) b_{m,n} + \mu_j x^{(j)}(n) b_{m,n} + \mu_k x^{(k)}(n) b_{m,n} \right\} \tag{22}$$

$$y^{(i)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(i)}(n) a_{m,n} - \mu_i x^{(r)}(n) b_{m,n} + \mu_j x^{(k)}(n) b_{m,n} - \mu_k x^{(j)}(n) b_{m,n} \right\} \tag{23}$$

$$y^{(j)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(j)}(n) a_{m,n} - \mu_i x^{(k)}(n) b_{m,n} - \mu_j x^{(r)}(n) b_{m,n} + \mu_k x^{(i)}(n) b_{m,n} \right\} \tag{24}$$

$$y^{(k)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(k)}(n) a_{m,n} + \mu_i x^{(j)}(n) b_{m,n} - \mu_j x^{(i)}(n) b_{m,n} - \mu_k x^{(r)}(n) b_{m,n} \right\} \tag{25}$$

The left-side 1-D quaternion discrete Fourier transform is defined as

$$z(m) = \sum_{n=0}^{N-1} e^{-\frac{\mu 2\pi mn}{N}} x(n) \tag{26}$$

for $m = 0, \ldots, N-1$, where the quaternion exponents are on the left side in multiplications. Taking into account (13) and (19), the right-side quaternion discrete Fourier transform (26) can be written in the form

$$z(m) = \sum_{n=0}^{N-1} [a_{m,n} - (\mu_i i + \mu_j j + \mu_k k) b_{m,n}][x^{(r)}(n) + x^{(i)}(n)i + x^{(j)}(n)j + x^{(k)}(n)] \quad (27)$$

After multiplying the expressions in square brackets, according to the rules (2) and (3) of multiplication of imaginary units, we can write the result as

$$z(m) = z^{(r)}(m) + z^{(i)}(m)i + z^{(j)}(m)j + z^{(k)}(m)k \quad (28)$$

where $z^{(r)}(m)$, $z^{(i)}(m)$, $z^{(j)}(m)$, and $z^{(k)}(m)$ are real numbers. It is easy to check that they take the following forms:

$$z^{(r)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(r)}(n) a_{m,n} + \mu_i x^{(i)}(n) b_{m,n} + \mu_j x^{(j)}(n) b_{m,n} + \mu_k x^{(k)}(n) b_{m,n} \right\} \quad (29)$$

$$z^{(i)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(i)}(n) a_{m,n} - \mu_i x^{(r)}(n) b_{m,n} - \mu_j x^{(k)}(n) b_{m,n} + \mu_k x^{(j)}(n) b_{m,n} \right\} \quad (30)$$

$$z^{(j)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(j)}(n) a_{m,n} + \mu_i x^{(k)}(n) b_{m,n} - \mu_j x^{(r)}(n) b_{m,n} - \mu_k x^{(i)}(n) b_{m,n} \right\} \quad (31)$$

$$z^{(k)}(m) = \sum_{n=0}^{N-1} \left\{ x^{(k)}(n) a_{m,n} - \mu_i x^{(j)}(n) b_{m,n} + \mu_j x^{(i)}(n) b_{m,n} - \mu_k x^{(r)}(n) b_{m,n} \right\} \quad (32)$$

Unlike the complex-valued DFT, the left-side and right-side QDFTs are slightly different transforms. From (22) and (29), we can see that the real parts of both transforms are the same but the imaginary parts differ from each other in the signs of some components of the sums (compare (23) with (30), (24) with (31), and (25) with (32)). Obviously, this follows from the non-commutative nature of quaternions multiplication. However, if we choose $\mu = i$ (i.e., $\mu_i = 1$ and $\mu_j = \mu_k = 0$) and the complex-valued input signal $x(n)$ (i.e., $x^{(j)}(n) = x^{(k)}(n) = 0$), then both of these transforms will be reduced to the usual complex-valued DFT.

Moreover, by analyzing (22)–(25) and (29)–(32), it is easy to see that all components of both QDFTs are linear combinations of the real and imaginary parts of four ordinary Fourier transforms for the real signals $x^{(r)}(n)$, $x^{(i)}(n)$, $x^{(j)}(n)$, and $x^{(k)}(n)$—components of the quaternion signal $x(n)$. This will be further explained in Section 5. Therefore, for determining the right-side or left-side QDFT, the calculation of four discrete Fourier transforms for the four real-valued components of the quaternion signal is needed.

Figure 1 shows graphical representations of four components of the right-side and the left-side QDFTs for the quaternion signal $x(n) = x^{(r)}(n) + x^{(i)}(n)i + x^{(j)}(n)j + x^{(k)}(n)k$ with $N = 16$ samples, where

$$x^{(r)}(n) = \sin \frac{n\pi}{N} - n \quad (33)$$

$$x^{(i)}(n) = -4 + n \quad (34)$$

$$x^{(j)}(n) = 10 + \sin \frac{2n\pi}{N} \quad (35)$$

$$x^{(k)}(n) = 5 + 2\cos \frac{n\pi}{N} \quad (36)$$

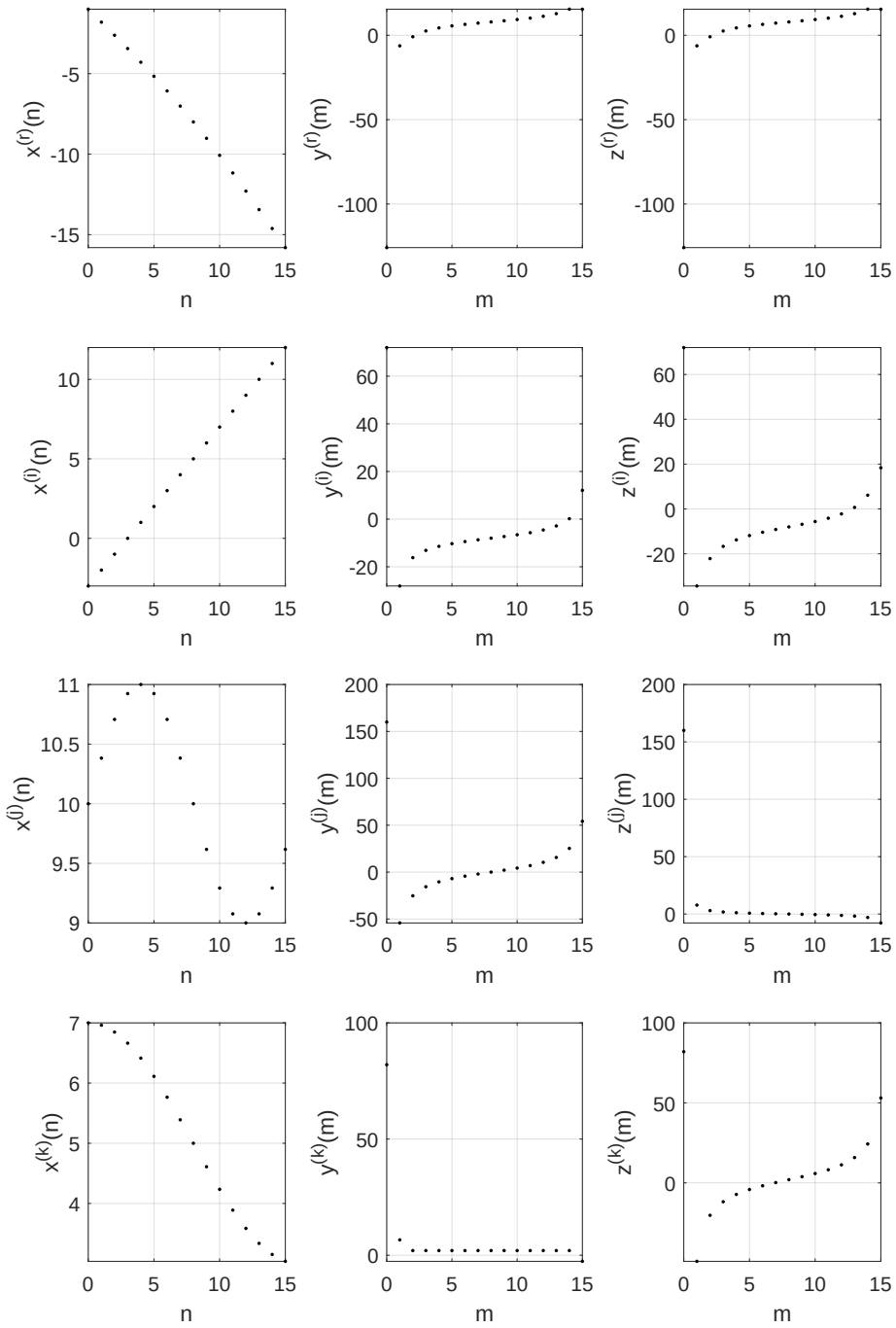

Figure 1. Four components of the quaternion input signal $x(n)$ (**the left column**), their right−side QDFT $y(n)$ (**the middle column**), and left−side QDFT $z(n)$ (**the right column**).

4. Calculation of QDFTs by Symplectic Decomposition

The use of the symplectic decomposition method allows both QDFTs to be computed by performing only two complex-valued discrete Fourier transforms instead of four. This approach was used in [26] to calculate the two-dimensional left-side QDFT. In this work, we will show how to also use the symplectic decomposition method to reduce the computational complexity of calculating one-dimensional right-side and left-side QDFTs.

If we take arbitrary two unit pure quaternions μ_1 and μ_2, such that $\mu_1 \perp \mu_2$ (pure quaternions are perpendicular when their dot product is equal to zero), and the third unit pure quaternion $\mu_3 = \mu_1\mu_2$, we can represent the quaternion signal

$$x(n) = x^{(r)}(n) + x^{(i)}(n)i + x^{(j)}(n)j + x^{(k)}(n)k \tag{37}$$

as

$$x(n) = x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1 + x^{(\mu_2)}(n)\mu_2 + x^{(\mu_3)}(n)\mu_3 \tag{38}$$

The change of the imaginary units from i, j, k to μ_1, μ_2, μ_3 corresponds to the change of the basis in $\mathbb{R}^3$ vector space from $[1,0,0]^T$, $[0,1,0]^T$, $[0,0,1]^T$ to another basis of three mutually orthogonal unit vectors; so for each sample, we obtain the new coordinates by multiplying the matrix inverse to the change-of-basis matrix by the old coordinates vector

$$\begin{bmatrix} x^{(\mu_1)}(n) \\ x^{(\mu_2)}(n) \\ x^{(\mu_3)}(n) \end{bmatrix} = \mathbf{S}_3^{-1} \begin{bmatrix} x^{(i)}(n) \\ x^{(j)}(n) \\ x^{(k)}(n) \end{bmatrix} \tag{39}$$

The change-of-basis matrix $\mathbf{S}_3$ depends on the choice of new imaginary units. For example, if we take the imaginary units μ_1, μ_2, μ_3, as in [26]

$$\mu_1 = \frac{1}{\sqrt{3}}i + \frac{1}{\sqrt{3}}j + \frac{1}{\sqrt{3}}k \tag{40}$$

$$\mu_2 = \frac{1}{\sqrt{2}}j - \frac{1}{\sqrt{2}}k \tag{41}$$

$$\mu_3 = -\frac{2}{\sqrt{6}}i + \frac{1}{\sqrt{6}}j + \frac{1}{\sqrt{6}}k \tag{42}$$

then the change-of-basis matrix has the form

$$\mathbf{S}_3 = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{2}{\sqrt{6}} \end{bmatrix} \tag{43}$$

In Cayley–Dickson form (38) has the form

$$x(n) = [x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1] + [x^{(\mu_2)}(n) + x^{(\mu_3)}(n)\mu_1]\mu_2 \tag{44}$$

This is a symplectic decomposition of each sample $x(n)$ into a simplex part

$$x^{(s)}(n) = x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1 \tag{45}$$

and a perplex part

$$x^{(p)}(n) = x^{(\mu_2)}(n) + x^{(\mu_3)}(n)\mu_1 \tag{46}$$

both isomorphic to the standard complex number. The result is a pair of complex signals. They may be stored using the same space as the quaternion signal $x(n)$. The symplectic decomposition splits a quaternion into two perpendicular planes. These planes intersect only at the origin (in 4-space). Each of these planes may be regarded as an Argand plane. The first is the Argand plane of the simplex part; its real axis is identical to the scalar axis of quaternion space and its imaginary axis is μ_1. The second is the Argand plane of the perplex part; this plane is perpendicular to both axes of the Argand plane of the simplex part. The real axis of this plane is identical to μ_2 and its imaginary axis is μ_3.

Let the axis of transform kernel of the right-side QDFT in (16) be chosen as μ_1, i.e., $\mu = \mu_1$. Then (16), can be rewritten as

$$y(m) = \sum_{n=0}^{N-1} [x^{(s)}(n) + x^{(p)}(n)\mu_2] e^{-\frac{\mu_1 2\pi mn}{N}} \tag{47}$$

The sum in (47) can be written as the sum of two terms. The first term, which is the simplex part of $y(m)$, has the form

$$y^{(s)}(m) = \sum_{n=0}^{N-1} x^{(s)}(n) e^{-\frac{\mu_1 2\pi mn}{N}} = \sum_{n=0}^{N-1} [x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1] e^{-\frac{\mu_1 2\pi mn}{N}} \tag{48}$$

This is isomorphic to the discrete Fourier transform for a complex input. The only difference is that the imaginary unit is called μ_1 instead of i. Hence, calculation of $y^{(s)}(m) = y^{(r)}(n) + y^{(\mu_1)}(n)\mu_1$ comes down to the computation of a complex DFT for the simplex part $x^{(s)}(n)$ of $x(n)$. This can be performed using the FFT algorithm. The second term

$$y^{(p)}(m)\mu_2 = \sum_{n=0}^{N-1} x^{(p)}(n)\mu_2 e^{-\frac{\mu_1 2\pi mn}{N}} = \sum_{n=0}^{N-1} x^{(p)}(n)\mu_2 \left(\cos \frac{2\pi mn}{N} - \mu_1 \sin \frac{2\pi mn}{N} \right) =$$

$$\sum_{n=0}^{N-1} x^{(p)}(n) \left(\cos \frac{2\pi mn}{N} + \mu_1 \sin \frac{2\pi mn}{N} \right) \mu_2 = \left(\sum_{n=0}^{N-1} [x^{(\mu_2)}(n) + x^{(\mu_3)}(n)\mu_1] e^{\frac{\mu_1 2\pi mn}{N}} \right) \mu_2 \tag{49}$$

where, during the transformation, we used the equality $\mu_2\mu_1 = -\mu_1\mu_2$, which is true for pure, perpendicular quaternions. It is easy to see that $y^{(p)}(n) = y^{(\mu_2)}(n) + y^{(\mu_3)}(n)\mu_1$ is the perplex part of $y(n)$. So, the calculation of $y^{(p)}(m)$ comes down to the computation of a complex inverse discrete Fourier transform (IDFT) for the perplex part $x^{(p)}(n)$ of $x(n)$, but without multiplying the result by $1/N$. This can be performed using the IFFT algorithm.

After calculating the simplex and perplex parts of the output signal $y(m) = y^{(r)}(n) + y^{(\mu_1)}(n)\mu_1 + y^{(\mu_2)}(n)\mu_3 + y^{(\mu_3)}(n)\mu_3$, we want to write $y(m)$ using the imaginary units i, j, and k instead of μ_1, μ_2, and μ_3, i.e.,

$$y(m) = y^{(r)}(m) + y^{(i)}(m)i + y^{(j)}(m)j + y^{(k)}(m)k \tag{50}$$

In the associated $\mathbb{R}^3$ vector space, this corresponds to returning to the old base $[1,0,0]^T$, $[0,1,0]^T$, $[0,0,1]^T$, so it can be done by the operation inverse to (39), i.e.,

$$\begin{bmatrix} y^{(i)}(m) \\ y^{(j)}(m) \\ y^{(k)}(m) \end{bmatrix} = \mathbf{S}_3 \begin{bmatrix} y^{(\mu_1)}(m) \\ y^{(\mu_2)}(m) \\ y^{(\mu_3)}(m) \end{bmatrix} \tag{51}$$

Summarizing the above considerations, in order to calculate the right-side QDFT using symplectic decomposition, the input signal should be represented by new imaginary units such that the first becomes the axis of the transform kernel. This corresponds to a change of the basis in the associated $\mathbb{R}^3$ vector space (for each sample of the quaternion-valued input signal). Then, a single complex DFT and a single complex IDFT should be calculated for the simplex and perplex parts of the quaternion-valued input signal, respectively. Finally, we have to return to the standard imaginary units i, j, and k, which corresponds to a return to the old basis in the associated $\mathbb{R}^3$ vector space.

A similar, and even simpler, consideration may be given to the left-side QDFT. Assuming that the axis of the transform kernel of the left-side QDFT in (26) is chosen as μ_1, and using the symplectic decomposition of the quaternion input signal $x(n)$, (26), can be rewritten as

$$z(m) = \sum_{n=0}^{N-1} e^{-\frac{\mu_1 2\pi mn}{N}} [x^{(s)}(n) + x^{(p)}(n)\mu_2] \tag{52}$$

The sum in (52) can be written as the sum of two terms. The first term, which is the simplex part of $z(m)$ has the form

$$z^{(s)}(m) = \sum_{n=0}^{N-1} e^{-\frac{\mu_1 2\pi mn}{N}} x^{(s)}(n) = \sum_{n=0}^{N-1} e^{-\frac{\mu_1 2\pi mn}{N}} [x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1] =$$

$$\sum_{n=0}^{N-1} [x^{(r)}(n) + x^{(\mu_1)}(n)\mu_1] e^{-\frac{\mu_1 2\pi mn}{N}} \tag{53}$$

Calculation of $z^{(s)}(m)$ comes down to the computation of complex-valued DFTs for the simplex part $x^{(s)}(n)$ of $x(n)$. This can be performed using the FFT algorithm. In addition, $z^{(s)}(n) = y^{(s)}(n)$.

The second term has the form

$$z^{(p)}(m)\mu_2 = \sum_{n=0}^{N-1} e^{-\frac{\mu_1 2\pi mn}{N}} x^{(p)}(n)\mu_2 = \left(\sum_{n=0}^{N-1} [x^{(\mu_2)}(n) + x^{(\mu_3)}(n)\mu_1] e^{-\frac{\mu_1 2\pi mn}{N}} \right) \mu_2 \tag{54}$$

It is easy to see that $z^{(p)}(n)$ is the perplex part of $z(n)$. Calculation of $z^{(p)}(m)$ also comes down to the computation of complex-valued DFTs for the perplex part $x^{(p)}(n)$ of $x(n)$. This can be performed using the FFT algorithm. Summarizing the above considerations, it can be argued that in order to calculate the left-side QDFT using a symplectic decomposition, the input signal should be represented by new imaginary units, the first of which will be equal to the axis of the transform kernel. This step is the same as in calculating the right-side QDFT. Then two complex DFTs should be calculated—for the simplex and perplex parts of the quaternion-valued input signal. In the end, as with the right-side QDFT calculation, the return to the standard imaginary units i, j, and k is needed.

5. The Simplest Method of QDFT Calculation

We will use the matrix notation to describe the QDFTs. Let $\mathbf{x}_N = [x_0, x_1, \ldots, x_{N-1}]^T$ be a vector of samples of the quaternion input signal, where $x_n = x(n)$ for $n = 0, 1, \ldots, N-1$, so it can be written in the form

$$\mathbf{x}_N = \begin{bmatrix} x_0^{(r)} \\ x_1^{(r)} \\ \vdots \\ x_{N-1}^{(r)} \end{bmatrix} + \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_{N-1}^{(i)} \end{bmatrix} i + \begin{bmatrix} x_0^{(j)} \\ x_1^{(j)} \\ \vdots \\ x_{N-1}^{(j)} \end{bmatrix} j + \begin{bmatrix} x_0^{(k)} \\ x_1^{(k)} \\ \vdots \\ x_{N-1}^{(k)} \end{bmatrix} k \tag{55}$$

Similarly $\mathbf{y}_N = [y_0, y_1, \ldots, y_{N-1}]^T$ and $\mathbf{z}_N = [z_0, z_1, \ldots, z_{N-1}]^T$ are the quaternion output vectors of the right-side and left-side QDFTs, respectively, where $y_n = y(n)$ and $z_n = z(n)$, so they can be written as

$$\mathbf{y}_N = \begin{bmatrix} y_0^{(r)} \\ y_1^{(r)} \\ \vdots \\ y_{N-1}^{(r)} \end{bmatrix} + \begin{bmatrix} y_0^{(i)} \\ y_1^{(i)} \\ \vdots \\ y_{N-1}^{(i)} \end{bmatrix} i + \begin{bmatrix} y_0^{(j)} \\ y_1^{(j)} \\ \vdots \\ y_{N-1}^{(j)} \end{bmatrix} j + \begin{bmatrix} y_0^{(k)} \\ y_1^{(k)} \\ \vdots \\ y_{N-1}^{(k)} \end{bmatrix} k \tag{56}$$

$$\mathbf{z}_N = \begin{bmatrix} z_0^{(r)} \\ z_1^{(r)} \\ \vdots \\ z_{N-1}^{(r)} \end{bmatrix} + \begin{bmatrix} z_0^{(i)} \\ z_1^{(i)} \\ \vdots \\ z_{N-1}^{(i)} \end{bmatrix} i + \begin{bmatrix} z_0^{(j)} \\ z_1^{(j)} \\ \vdots \\ z_{N-1}^{(j)} \end{bmatrix} j + \begin{bmatrix} z_0^{(k)} \\ z_1^{(k)} \\ \vdots \\ z_{N-1}^{(k)} \end{bmatrix} k \tag{57}$$

Let square matrices of cosine and sine coefficients be denoted by $\mathbf{A}_N$ and $\mathbf{B}_N$, respectively, so

$$\mathbf{A}_N = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \dots & a_{N-1,N-1} \end{bmatrix} \tag{58}$$

$$\mathbf{B}_N = \begin{bmatrix} b_{0,0} & b_{0,1} & \dots & b_{0,N-1} \\ b_{1,0} & a_{1,1} & \dots & b_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N-1,0} & b_{N-1,1} & \dots & b_{N-1,N-1} \end{bmatrix} \tag{59}$$

where the entries $a_{m,n}$ and $b_{m,n}$ of the matrices $\mathbf{A}_N$ and $\mathbf{B}_N$, respectively, are defined in (18). Note that the matrix $(\mathbf{A}_N - \mathbf{B}_N i) = \mathbf{F}_N$, where $\mathbf{F}_N$ is the DFT matrix.

In matrix notation, we can write (22)–(25), which describe the right-side QDFT, in the forms

$$\mathbf{y}_N^{(r)} = \mathbf{A}_N \mathbf{x}_N^{(r)} + \mu_i \mathbf{B}_N \mathbf{x}_N^{(i)} + \mu_j \mathbf{B}_N \mathbf{x}_N^{(j)} + \mu_k \mathbf{B}_N \mathbf{x}_N^{(k)} \tag{60}$$

$$\mathbf{y}_N^{(i)} = \mathbf{A}_N \mathbf{x}_N^{(i)} - \mu_i \mathbf{B}_N \mathbf{x}_N^{(r)} + \mu_j \mathbf{B}_N \mathbf{x}_N^{(k)} - \mu_k \mathbf{B}_N \mathbf{x}_N^{(j)} \tag{61}$$

$$\mathbf{y}_N^{(j)} = \mathbf{A}_N \mathbf{x}_N^{(j)} - \mu_i \mathbf{B}_N \mathbf{x}_N^{(k)} - \mu_j \mathbf{B}_N \mathbf{x}_N^{(r)} + \mu_k \mathbf{B}_N \mathbf{x}_N^{(i)} \tag{62}$$

$$\mathbf{y}_N^{(k)} = \mathbf{A}_N \mathbf{x}_N^{(k)} + \mu_i \mathbf{B}_N \mathbf{x}_N^{(j)} - \mu_j \mathbf{B}_N \mathbf{x}_N^{(i)} - \mu_k \mathbf{B}_N \mathbf{x}_N^{(r)} \tag{63}$$

Figure 2 shows a data flow diagram for calculating the right-side QDFT. The straight lines in the figures indicate the data transmission buses, so that each bus simultaneously transmits all elements of a real-valued vector. By default, we assume that data flows from left to right. Therefore, we do not use arrows, so as not to clutter up the drawings. The points at which the lines converge denote the element-wise summation of the input vectors. The rectangles show the operation of multiplying a vector by a matrix, the designation of which is inscribed inside the rectangle, and the circles show the operation of multiplying all elements of the input vector by a constant inscribed in the circle.

Now, we will develop a similar algorithm for the left-side QDFT. In matrix notation, the equivalents of (29)–(32), which describe the left-side QDFT, are as follows:

$$\mathbf{z}_N^{(r)} = \mathbf{A}_N \mathbf{x}_N^{(r)} + \mu_i \mathbf{B}_N \mathbf{x}_N^{(i)} + \mu_j \mathbf{B}_N \mathbf{x}_N^{(j)} + \mu_k \mathbf{B}_N \mathbf{x}_N^{(k)} \tag{64}$$

$$\mathbf{z}_N^{(i)} = \mathbf{A}_N \mathbf{x}_N^{(i)} - \mu_i \mathbf{B}_N \mathbf{x}_N^{(r)} - \mu_j \mathbf{B}_N \mathbf{x}_N^{(k)} + \mu_k \mathbf{B}_N \mathbf{x}_N^{(j)} \tag{65}$$

$$\mathbf{z}_N^{(j)} = \mathbf{A}_N \mathbf{x}_N^{(j)} + \mu_i \mathbf{B}_N \mathbf{x}_N^{(k)} - \mu_j \mathbf{B}_N \mathbf{x}_N^{(r)} - \mu_k \mathbf{B}_N \mathbf{x}_N^{(i)} \tag{66}$$

$$\mathbf{z}_N^{(k)} = \mathbf{A}_N \mathbf{x}_N^{(k)} - \mu_i \mathbf{B}_N \mathbf{x}_N^{(j)} + \mu_j \mathbf{B}_N \mathbf{x}_N^{(i)} - \mu_k \mathbf{B}_N \mathbf{x}_N^{(r)} \tag{67}$$

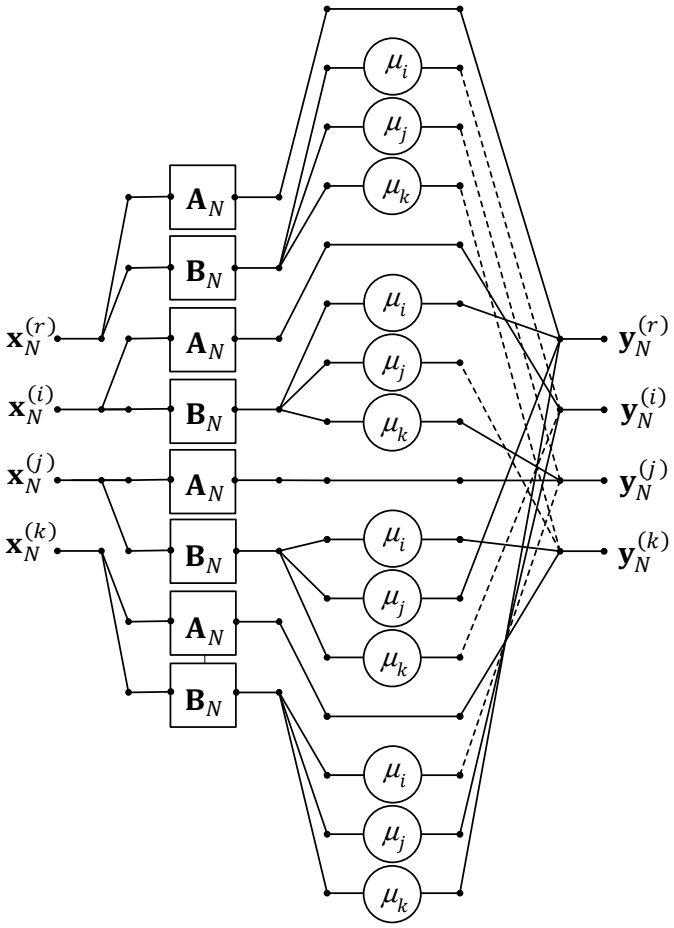

Figure 2. Data flow diagrams for calculation of the right-side QDFT.

Figure 3 shows the data flow diagram for calculation of the left-side QDFT.

To calculate both QDFTs—the right-side and the left-side—you need to multiply the matrices $\mathbf{A}_N$ and $\mathbf{B}_N$ by each of the four components of the quaternion-valued input vector $\mathbf{x}_N$. Since the components of the input vectors are real values, these operations can be performed by computing the DFT for each component of the input vector and dividing the real and imaginary parts of the result, i.e.,

$$\mathbf{A}_N \mathbf{x}_N^{(r)} = Re(\mathbf{F}_N \mathbf{x}_N^{(r)}), \quad \mathbf{B}_N \mathbf{x}_N^{(r)} = -Im(\mathbf{F}_N \mathbf{x}_N^{(r)}) \tag{68}$$

$$\mathbf{A}_N \mathbf{x}_N^{(i)} = Re(\mathbf{F}_N \mathbf{x}_N^{(i)}), \quad \mathbf{B}_N \mathbf{x}_N^{(i)} = -Im(\mathbf{F}_N \mathbf{x}_N^{(i)}) \tag{69}$$

$$\mathbf{A}_N \mathbf{x}_N^{(j)} = Re(\mathbf{F}_N \mathbf{x}_N^{(j)}), \quad \mathbf{B}_N \mathbf{x}_N^{(j)} = -Im(\mathbf{F}_N \mathbf{x}_N^{(j)}) \tag{70}$$

$$\mathbf{A}_N \mathbf{x}_N^{(k)} = Re(\mathbf{F}_N \mathbf{x}_N^{(k)}), \quad \mathbf{B}_N \mathbf{x}_N^{(k)} = -Im(\mathbf{F}_N \mathbf{x}_N^{(k)}) \tag{71}$$

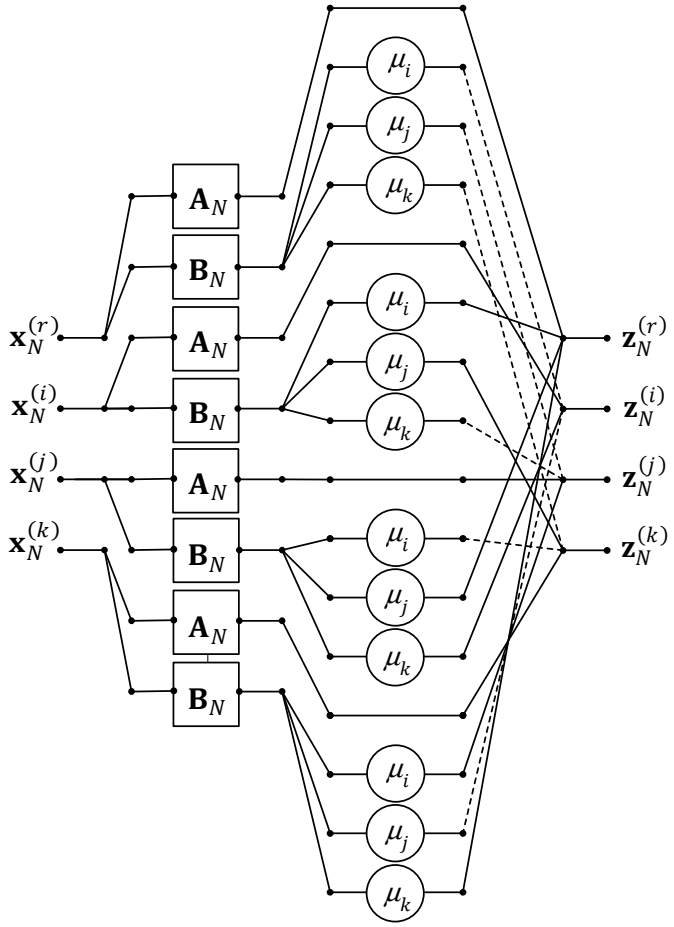

Figure 3. Data flow diagrams for calculation of the left-side QDFT.

6. 1-D QDFT Proposal

By computing both QDFTs, the main computational cost is related to multiplying the four components of the quaternion-valued input vector by the matrices $\mathbf{A}_N$ and $\mathbf{B}_N$. This is equivalent to calculating the DFTs for each of the four components and separating the real and imaginary parts, as was described in Section 5. It seems that in order to calculate any QDFT, it is necessary to compute four DFTs. However, it is known that the same operation can be done by performing only two Fourier transforms for two complex-valued input vectors, since each of the two DFTs for real-valued input vectors can be computed with one DFT for the complex-valued input vector. To explain this, let us introduce two real-valued vectors $\mathbf{x}_N^{(1)}$ and $\mathbf{x}_N^{(2)}$ formed from the complex-valued vector $\mathbf{x}_N = \mathbf{x}_N^{(1)} + i\mathbf{x}_N^{(2)}$. Let $\mathbf{c}_N^{(1)}$ be the DFT output for input $\mathbf{x}_N^{(1)}$, $\mathbf{c}_N^{(2)}$—the DFT output for input $\mathbf{x}_N^{(2)}$, and $\mathbf{c}_N$—the DFT output for complex-valued input $\mathbf{x}_N$. Although the DFTs outputs for the real-valued inputs are complex-valued, their real and imaginary parts have special symmetries, namely

$$Re(c_n^{(1)}) = Re(c_{N-n}^{(1)}), \quad Im(c_n^{(1)}) = -Im(c_{N-n}^{(1)}) \tag{72}$$

$$Re(c_n^{(2)}) = Re(c_{N-n}^{(2)}), \quad Im(c_n^{(2)}) = -Im(c_{N-n}^{(2)}) \tag{73}$$

for $n = 0, 1, \ldots, N-1$. It is said that for a real-valued input signal, the real part of its DFT is symmetric and the imaginary part is antisymmetric.

For each coefficient c_n of DFT output for complex-valued signal $\mathbf{x}_N = \mathbf{x}_N^{(1)} + i\mathbf{x}_N^{(2)}$, we can write

$$c_n = Re(c_n) + Im(c_n)i \tag{74}$$

However, the DFT is a linear transform, so it can be also written as

$$c_n = c_n^{(1)} + ic_n^{(2)} = Re(c_n^{(1)}) + iIm(c_n^{(1)}) + i[Re(c_n^{(2)}) + iIm(c_n^{(2)})] \tag{75}$$

Thus,

$$Re(c_n) = Re(c_n^{(1)}) - Im(c_n^{(2)}) \tag{76}$$

and

$$Im(c_n) = Im(c_n^{(1)}) + Re(c_n^{(2)}) \tag{77}$$

Analogously,

$$Re(c_{N-n}) = Re(c_{N-n}^{(1)}) - Im(c_{N-n}^{(2)}) \tag{78}$$

and

$$Im(c_{N-n}) = Im(c_{N-n}^{(1)}) + Re(c_{N-n}^{(2)}) \tag{79}$$

Taking into account properties (72) and (73), (78) can be written in the form

$$Re(c_{N-n}) = Re(c_n^{(1)}) + Im(c_n^{(2)}) \tag{80}$$

and (79) as

$$Im(c_{N-n}) = -Im(c_n^{(1)}) + Re(c_n^{(2)}) \tag{81}$$

By adding and subtracting the sides of (80) and (76), we obtain, respectively

$$Re(c_n^{(1)}) = \frac{Re(c_n) + Re(c_{N-n})}{2} \tag{82}$$

and

$$Im(c_n^{(2)}) = \frac{Re(c_{N-n}) - Re(c_n)}{2} \tag{83}$$

Similarly, by adding and subtracting the sides of (77) and (81) we obtain, respectively

$$Re(c_n^{(2)}) = \frac{Im(c_n) + Im(c_{N-n})}{2} \tag{84}$$

and

$$Im(c_n^{(1)}) = \frac{Im(c_n) - Im(c_{N-n})}{2} \tag{85}$$

We introduce the denotation $^{\uparrow}$, which means the signal reversed in time. So, for $\mathbf{c}_N = [c_0, c_1, \ldots, c_{N-1}]^T$, we have

$$\mathbf{c}_N^{\uparrow} = \begin{bmatrix} c_0 \\ c_{N-1} \\ \vdots \\ c_1 \end{bmatrix} = \mathbf{P}_N \mathbf{c}_N \tag{86}$$

where the matrix $\mathbf{P}_N$, which is responsible for reversing signal in time, has the form

$$\mathbf{P}_N = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & 1 & \ldots & 0 & 0 \end{bmatrix} \tag{87}$$

Then, (82) written in vector notation has the form

$$Re(\mathbf{c}_N^{(1)}) = \frac{1}{2}[Re(\mathbf{c}_N) + Re(\mathbf{c}_N^{\uparrow})] \tag{88}$$

Analogously, (85) written in vector notation has the form

$$Im(\mathbf{c}_N^{(1)}) = \frac{1}{2}[Im(\mathbf{c}_N) - Im(\mathbf{c}_N^{\uparrow})] \tag{89}$$

Similarly, (84) written in vector notation has the form

$$Re(\mathbf{c}_N^{(2)}) = \frac{1}{2}[Im(\mathbf{c}_N) + Im(\mathbf{c}_N^{\uparrow})] \tag{90}$$

In the end, (83) written in vector notation has the form

$$Im(\mathbf{c}_N^{(2)}) = \frac{1}{2}[Re(\mathbf{c}_N^{\uparrow}) - Re(\mathbf{c}_N)] \tag{91}$$

Going back to QDFT transforms, the quaternion-valud input vector $\mathbf{x}_N$ has four real-valued component $\mathbf{x}_N^{(r)}$, $\mathbf{x}_N^{(i)}$, $\mathbf{x}_N^{(j)}$, $\mathbf{x}_N^{(k)}$. Let us denote their complex Fourier transforms by $\mathbf{c}_N^{(r)}$, $\mathbf{c}_N^{(i)}$, $\mathbf{c}_N^{(j)}$, $\mathbf{c}_N^{(k)}$. They are all complex-valued, and

$$\mathbf{A}_N\mathbf{x}_N^{(r)} = Re(\mathbf{c}_N^{(r)}), \quad \mathbf{B}_N\mathbf{x}_N^{(r)} = -Im(\mathbf{c}_N^{(r)}) \tag{92}$$

$$\mathbf{A}_N\mathbf{x}_N^{(i)} = Re(\mathbf{c}_N^{(i)}), \quad \mathbf{B}_N\mathbf{x}_N^{(i)} = -Im(\mathbf{c}_N^{(i)}) \tag{93}$$

$$\mathbf{A}_N\mathbf{x}_N^{(j)} = Re(\mathbf{c}_N^{(j)}), \quad \mathbf{B}_N\mathbf{x}_N^{(j)} = -Im(\mathbf{c}_N^{(j)}) \tag{94}$$

$$\mathbf{A}_N\mathbf{x}_N^{(k)} = Re(\mathbf{c}_N^{(k)}), \quad \mathbf{B}_N\mathbf{x}_N^{(k)} = -Im(\mathbf{c}_N^{(k)}) \tag{95}$$

From the components of the quaternion input vector $\mathbf{x}_N$, we can create two complex inputs vectors $\tilde{\mathbf{x}}_N = \mathbf{x}_N^{(r)} + \mathbf{x}_N^{(i)}i$ and $\hat{\mathbf{x}}_N = \mathbf{x}_N^{(j)} + \mathbf{x}_N^{(k)}i$. Then, we calculate the complex discrete Fourier transforms for these, i.e., $\tilde{\mathbf{c}}_N$ and $\hat{\mathbf{c}}_N$, using the fast algorithm, and separate their real and imaginary parts. Then, according to (88)–(91), for the first complex vector $\tilde{\mathbf{x}}_N$, we obtain

$$Re(\mathbf{c}_N^{(r)}) = \frac{1}{2}[Re(\tilde{\mathbf{c}}_N) + Re(\tilde{\mathbf{c}}_N^{\uparrow})] \tag{96}$$

$$Im(\mathbf{c}_N^{(r)}) = \frac{1}{2}[Im(\tilde{\mathbf{c}}_N) - Im(\tilde{\mathbf{c}}_N^{\uparrow})] \tag{97}$$

$$Re(\mathbf{c}_N^{(i)}) = \frac{1}{2}[Im(\tilde{\mathbf{c}}_N) + Im(\tilde{\mathbf{c}}_N^{\uparrow})] \tag{98}$$

$$Im(\mathbf{c}_N^{(i)}) = \frac{1}{2}[Re(\tilde{\mathbf{c}}_N^{\uparrow}) - Re(\tilde{\mathbf{c}}_N)] \tag{99}$$

and, similarly, for the second complex-valued vector $\hat{\mathbf{x}}_N$, we obtain

$$Re(\mathbf{c}_N^{(j)}) = \frac{1}{2}[Re(\hat{\mathbf{c}}_N) + Re(\hat{\mathbf{c}}_N^{\uparrow})] \tag{100}$$

$$Im(\mathbf{c}_N^{(j)}) = \frac{1}{2}[Im(\hat{\mathbf{c}}_N) - Im(\hat{\mathbf{c}}_N^{\uparrow})] \tag{101}$$

$$Re(\mathbf{c}_N^{(k)}) = \frac{1}{2}[Im(\hat{\mathbf{c}}_N) + Im(\hat{\mathbf{c}}_N^{\uparrow})] \tag{102}$$

$$Im(\mathbf{c}_N^{(k)}) = \frac{1}{2}[Re(\hat{\mathbf{c}}_N^{\uparrow}) - Re(\hat{\mathbf{c}}_N)] \tag{103}$$

The data flow diagrams for our method of QDFT calculation are shown in Figure 4 for the right-side QDFT and in Figure 5 for the left-side QDFT.

It should be noted that the proposed method, unlike the method using symplectic decomposition, does not require changing the basis in the pure quaternions subspace and, consequently, calculating the new basis vectors and change-of-basis matrix.

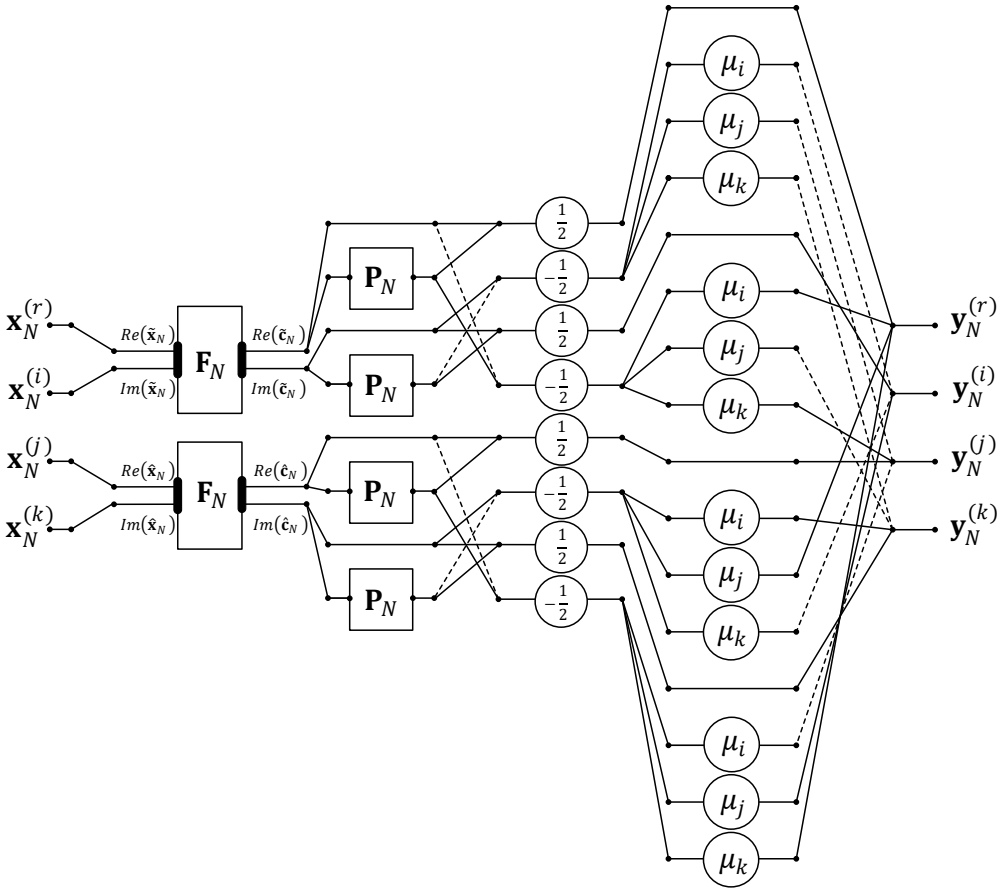

Figure 4. Data flow diagram for our method of calculation of the right-side QDFT.

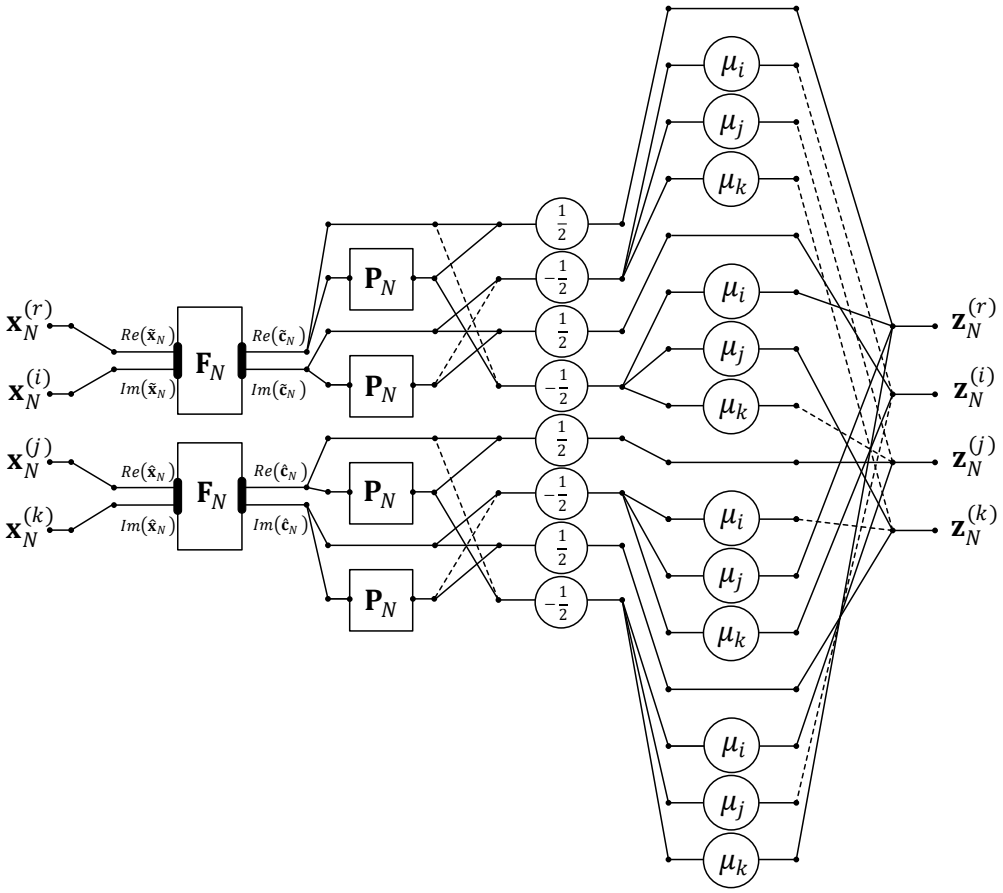

Figure 5. Data flow diagram for our method of calculation of the left-side QDFT.

7. Computational Complexity Discussion

We will now compare the computational complexity of the different ways to calculate a QDFT, taking into account the number of multiplications and additions of real numbers needed to compute this transform (these numbers are the same for the right-side and left-side transforms). Since, in each case, the ordinary FFT algorithm is used, we denote by $m_{FFT}(N)$ the number of real multiplications necessary to compute the DFT for a complex/real signal with N samples. Similarly, let $a_{FFT}(N)$ denote the number of additions of real numbers needed to determine the FFT of such a signal. In this comparison, we will omit the method of calculating QDFTs resulting directly from the definitions (16) or (26) because its computational complexity is much higher.

Let us start with the most basic calculation method resulting from (60)–(63) or (64)–(67), respectively. Since, according to (68), calculating the products of $\mathbf{A}_N\mathbf{x}_N^{(r)}$ and $\mathbf{B}_N\mathbf{x}_N^{(r)}$ is equivalent to finding the FFT for the $\mathbf{x}_N^{(r)}$ part of the quaternion vector $\mathbf{x}_N$, and similarly, for the other $\mathbf{x}_N^{(i)}$, $\mathbf{x}_N^{(j)}$, $\mathbf{x}_N^{(k)}$ parts of the quaternion input vector, we need to calculate four FFTs, which requires $4m_{FFT}(N)$ multiplications and $4a_{FFT}(N)$ additions of real numbers. Then, each of the products $\mathbf{B}_N\mathbf{x}_N^{(r)}$, $\mathbf{B}_N\mathbf{x}_N^{(i)}$, $\mathbf{B}_N\mathbf{x}_N^{(j)}$, $\mathbf{B}_N\mathbf{x}_N^{(k)}$ has to be multiplied by three real coefficients μ_i, μ_j, and μ_k, which gives $3 \cdot 4 \cdot N = 12N$ multiplications of real numbers. The last step is to add the four components of the sum. Since each sum consists of four vectors, it takes $3 \cdot N$ additions to calculate the sum. There are four such sums, so we have $4 \cdot 3 \cdot N = 12N$ additions of real numbers. Summarizing the above considerations, in order to calculate the DQFT according to this method, $m_1(N)$ multiplications and $a_1(N)$ additions of real numbers should be performed, where

$$m_1(N) = 4m_{FFT}(N) + 12N \tag{104}$$

and

$$a_1(N) = 4a_{FFT}(N) + 12N \tag{105}$$

The next method in the comparison is the method based on symplectic decomposition. First, all samples of the quaternion signal should be expressed using the new imaginary units μ_1, μ_2, μ_3, which corresponds to the multiplication of the matrix inverse to the change-of-basis matrix by the vector containing the imaginary components of the sample according to (39). Such an operation for one sample requires nine multiplications and six additions of real numbers. For N samples of the quaternion signal, we get $9N$ multiplications and $6N$ additions of real numbers. Then, a single ordinary complex FFT and a single IFFT (without dividing the result by N), or two complex FFTs, should be calculated, according to (48), (49) or (53), (54), which require $2m_{FFT}(N)$ multiplications and $2a_{FFT}(N)$ additions of real numbers. The resulting quaternion output signal is expressed using the new imaginary units μ_1, μ_2, μ_3, and we want it to be represented by standard imaginary units i, j, k. This corresponds to the multiplication of the change-of-basis matrix by the vector containing the imaginary components of the resulting vector according to (51). To do this for N samples of the quaternion output signal, $9N$ multiplications and $6N$ additions of real numbers is needed. To recap the method based on symplectic decomposition, calculating the QDFT according to this method requires $m_2(N)$ multiplications and $a_2(N)$ additions of real numbers, where

$$m_2(N) = 2m_{FFT}(N) + 18N \tag{106}$$

and

$$a_2(N) = 2a_{FFT}(N) + 12N \tag{107}$$

Finally, we evaluate the numbers of arithmetic operations needed for our method of QDFT calculation. Our method is also based on (60)–(63) for the right-side QDFT or (64)–(67) for the left-side QDFT, but taking into account relations (92)–(95), the products $\mathbf{A}_N\mathbf{x}_N^{(r)}$, $\mathbf{B}_N\mathbf{x}_N^{(r)}$, $\mathbf{A}_N\mathbf{x}_N^{(i)}$, $\mathbf{B}_N\mathbf{x}_N^{(i)}$, $\mathbf{A}_N\mathbf{x}_N^{(j)}$, $\mathbf{B}_N\mathbf{x}_N^{(j)}$ and $\mathbf{A}_N\mathbf{x}_N^{(k)}$, $\mathbf{B}_N\mathbf{x}_N^{(k)}$ are calculated from (96)–(103). This method requires the calculation of two FFTs, hence $2m_{FFT}(N)$ multiplications and $2a_{FFT}(N)$ additions of real numbers is needed. Then, the real/imaginary parts of the FFT results are added/subtracted (some after reordering their elements, which corresponds to multiplying the matrix $\mathbf{P}_N$ by these elements, as was shown in Figures 3 and 4—this does not require any arithmetic operations) and divided by 2. This may seem to require $8N$ additions and $8N$ multiplications of real numbers. In fact, $\mathbf{c}_N^{(r)}$, $\mathbf{c}_N^{(i)}$, $\mathbf{c}_N^{(j)}$, $\mathbf{c}_N^{(k)}$ are Fourier transforms of real-valued vectors, so they have symmetry, as in (72)–(73), and only half of their entries have to be calculated. Thus, the number of real additions reduces to $4N$. Multiplying by the factor $1/2$ in (96)–(103) does not require any multiplication operation. It is just a simple bit shift to the right. Then, as in the first method, each of the products $\mathbf{B}_N\mathbf{x}_N^{(r)}$, $\mathbf{B}_N\mathbf{x}_N^{(i)}$, $\mathbf{B}_N\mathbf{x}_N^{(j)}$, $\mathbf{B}_N\mathbf{x}_N^{(k)}$ has to be multiplied by three real coefficients μ_i, μ_j, and μ_k, which gives $12N$ multiplications of real numbers. The last step is to add the four components of each sum in (60)–(63) for the right-side QDFT or (64)–(67) for the left-side QDFT, as in the first method. This requires $12N$ additions of real numbers. Summarizing the above considerations, in order to calculate any of the QDFT according to our method $m_3(N)$ multiplications and $a_3(N)$ additions of real numbers have to be performed, where

$$m_3(N) = 2m_{FFT}(N) + 12N \tag{108}$$

and

$$a_3(N) = 2a_{FFT}(N) + 16N \tag{109}$$

Table 1 presents estimates for the numbers of multiplications and additions of real numbers necessary to determine any of the QDFTs in the three presented methods of QDFT calculation. It is easy to see that although our method has more additions than method 2, it requires fewer multiplications. Furthermore, the total number of arithmetic operations in our method is the smallest among the three compared methods for calculating the QDFT.

Table 1. Number of multiplications and additions of real numbers needed to calculate any 1-D QDFT for a quaternion signal of N samples.

Type of Operation	Simplest Method	Symplectic Method	Proposed Method
$\times$	$4m_{FFT}(N)+12N$	$2m_{FFT}(N)+18N$	$2m_{FFT}(N)+12N$
$+$	$4a_{FFT}(N)+12N$	$2a_{FFT}(N)+12N$	$2a_{FFT}(N)+16N$

In terms of the number of arithmetic operations, our method is very similar to the symplectic decomposition method. To compare the 1-D QDFT computation times of these two methods, we implemented them in Matlab R2022b on a computer with an Intel(R) Core(TM) i5-7400 CPU and 8 GB RAM. For quaternion signals of different lengths N, which are powers of 2, and 100,000 repetitions of transform calculation, we obtained the times, in seconds, presented in Table 2.

Table 2. Calculation times of 1-D QDFTs for signals of length N for the method using symplectic decomposition and the proposed method, for 100,000 repetitions.

N	Time for Simplectic Method [s]	Time for Proposed Method [s]
128	1.43	1.19
256	2.32	1.87
512	3.33	2.60
1024	5.40	4.49
2048	9.14	7.64
4096	19.36	14.90
8192	35.63	28.38

The dependencies in Table 2 are not an exact reflection of the dependencies in Table 1, because the calculation time also includes the time needed to transfer data from and to memory, which Table 1 does not take into account. Despite this, it can be observed from Table 2 that the times needed to calculate the 1-D QDFT with the proposed method are shorter than for the method using symplectic decomposition. However, it should be remembered that the time of computation depends very much on the way the code was written and on the implementation platform.

8. Conclusions

The article presents a new method of 1-D QDFT calculation. The proposed method allows a reduction in the number of multiplications in the calculation of the QDFT. Reducing the number of multipliers is especially important when designing dedicated on-board VLSI processors, since minimizing the number of multipliers required also reduces power dissipation and lowers the cost of implementing the entire system.

Furthermore, although the number of additions of real numbers in our method increases, the total number of arithmetic operations is still lower than in other ways of calculating this transform. This leads to a reduction in computational costs, which makes the proposed solution also preferable for its implementation on a general-purpose computer.

It should be added that the newly proposed method can be used to compute both the left-side and right-side QDFTs.

Author Contributions: Conceptualization, A.C. and D.M.-M.; methodology, A.C. and D.M.-M.; validation, D.M.-M.; formal analysis, D.M.-M.; investigation, D.M.-M.; writing—original draft preparation, D.M.-M.; writing—review and editing, A.C. and D.M.-M.; supervision, A.C. and D.M.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DFT	Discrete Fourier transform
IDFT	Inverse discrete Fourier Transform
FFT	Fast Fourier transform
IFFT	Inverse fast Fourier Transform
QDFT	Quaternion Fourier Transform
1-D QDFT	One-dimensional quaternion discrete Fourier transform
2-D QDFT	Two-dimensional quaternion discrete Fourier transform
VLSI	Very Large-Scale Integration

References

1. Briggs, W.L.; Henson, V.E. *The DFT: An Owners' Manual for the Discrete Fourier Transform*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1987.
2. Rabiner, L.R.; Schafer R.W. *Digital Processing of Speech Signals*; Prentice Hall: Englewood Cliffs, NJ, USA, 1978.
3. Zolzer, U.; Holters, M.; Great, E.; Nowak, P.; Bhattacharya, P.; Koper, L.; Ahlers, D. *Digital Audio Signal Processing*; John Wiley & Sons: Hoboken, NJ, USA, 2022.
4. Pratt, W.K. *Digital Image Processing*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
5. Huang, T.S. (ed.) *Two-Dimensional Digital Signal Processing II: Transform and Median Filters*; Springer: Berlin/Heidelberg, Germany, 1981.
6. Li, L.; Bai, R.; Lu, J.; Zhang, S.; Chang, C.-C. A Watermarking Scheme for Color Image Using Quaternion Discrete Fourier Transform and Tensor Decomposition. *Appl. Sci.* **2021**, *11*, 5006. [CrossRef]
7. Chen, B.; Coatrieux, G.; Chen, G.; Sun, X.; Coatrieux, J.L.; Shu, H. Full 4-D quaternion discrete Fourier transform based watermarking for color images. *Digit. Signal Process.* **2014**, *28*, 106–119. [CrossRef]
8. Li, M.; Yuan, X.; Chen, H.; Li, J. Quaternion Discrete Fourier Transform-Based Color Image Watermarking Method Using Qurternion QR Decomposition. *IEEE Access* **2020**, *8*, 72308–72315. [CrossRef]
9. Bahri, M.; Azis, M.I.; Firman; Lande, C. Discrete Double-Sided Quaternionic Fourier Transform and Application. *J. Phys. Conf. Ser.* **2019**, *1341*, 06200. [CrossRef]
10. Schütte, H.D.; Wenzel, J. Hypercomplex numbers in digital signal processing. In Proceedings of the ISCAS '90, New Orleans, LA, USA, 1–3 May 1990; pp. 1557–1560.
11. Ell, T. A. Hypercomplex Spectral Transformation. Ph.D. Thesis, University of Minnesota, Minneapolis, MN, USA, 1992.
12. Sangwine, S.J. Fourier transforms of colour images using quaternion or hypercomplex numbers. *Electron. Lett.* **1996**, *32*, 1979–1980. [CrossRef]
13. Witten, B.; Shragge, J. Quaternion-based Signal Processing. In Proceedings of the New Orleans 2006 SEG Annual Meeting, New Orleans, LA, USA, 1 October 2006; pp. 2862–2865.
14. Ell, T.A.; Sangwine, S.J. Hypercomplex Fourier transforms of color images. *IEEE Trans. Image Process.* **2007**, *16*, 22–35. [CrossRef] [PubMed]
15. Parchami, A.; Mahdavi, M. Full Quaternion Representation of Color images: A Case Study on QSVD-based Color Image Compression. *arXiv* **2007**, arXiv:2007.09758.
16. Ell, T.A.; Bihan, N.L.; Sangwine, S.J. *Quaternion Fourier Transforms for Signal and Image Processing*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2014.
17. Pei, S.C.; Ding J.J.; Chang J.H. Efficient implementation of quaternion Fourier transform, convolution, and correlation by 2-D complex FFT. *IEEE Trans. Signal Process.* **2001**, *49*, 2783–2797.
18. Bahri, M. Discrete Quaternion Fourier Transform and Properties. *Int. J. Math. Anal.* **2013**, *7*, 1207–1215. [CrossRef]
19. Grigoryan, A. M.; Agaian, S. S. Tensor transform-based quaternion Fourier transform algorithm. *Inf. Sci.* **2015**, *320*, 62–74. [CrossRef]
20. Grigoryan, A.M.; Agaian, S.S. 2-D Left-Side Quaternion Discrete Fourier Transform: Fast Algorithm. In Proceedings of the IS&T Int'l. Symp. on Electronic Imaging: Image Processing: Algorithms and Systems XIV, San Francisco, CA, USA, 14 February 2016; pp. 192.1–192.8.
21. Bahri, M.; Toaha, S.; Rahim, A.; Azis, M.I. On one-dimensional quaternion Fourier transform. *J. Phys. Conf. Ser.* **2019**, *1341*, 062004. [CrossRef]
22. Ouyang, J.; Coatrieux, G.; Shua, H. Robust hashing for image authentication using quaternion discrete Fourier transform and log-polar transform. *Digit. Signal Process.* **2015**, *41*, 98–109. [CrossRef]
23. Ribeiro, G.B.; Lima, J.B. Eigenstructure and fractionalization of the quaternion discrete Fourier transform. *Opt. Int. J. Light Electron Opt.* **2020**, *208*, 163957. [CrossRef]

24. Ekasasmita, W.; Bahri, M.; Bachtiar, N.; Rahim, A.; Nur, M. One-Dimensional Quaternion Fourier Transform with Application to Probability Theory. *Symmetry* **2023**, *15*, 815. [CrossRef]
25. Hamilton, W.R. *Elements of Quaternions*; Longmans, Green and Co.: London, UK, 1866.
26. Ell, T.A.; Sangwine, S.J. Decomposition of 2D hypercomplex Fourier transforms into pairs of complex Fourier transforms. In Proceedings of the 2000 10th European Signal Processing Conference, Tampere, Finland, 1 September 2000; pp. 1–4.