1. Aleksandr CARIOW, 2. Marina POLYAKOVA
ORCID: 1.0000-0002-4513-4593; 2.0000-0001-7229-7657

DOI: 10.15199/48.2025.06.20

On the multiplications of pure quaternions

Mnożenie czystych kwaternionów

Abstract. *The need to multiply quaternions when at least one quaternion is pure arises in the number of applications related to information processing. In this case, different situations may arise. It may turn out that the multiplicand is a vector quaternion, and the multiplier is an ordinary quaternion. Another problem is when the multiplicand is an ordinary quaternion, and the multiplier is a vector one. Finally, a situation may arise where both factors are pure quaternions. In all three cases, the number of real multiplications can be reduced compared to the number of real multiplications required to compute the product of ordinary quaternions. The article discusses algorithmic solutions that allow reducing the number of real multiplications when calculating quaternion products in each of the mentioned cases.*

Streszczenie. *Potrzeba mnożenia kwaternionów, gdy przynajmniej jeden kwaternion jest czysty, pojawia się w wielu zastosowaniach związanych z przetwarzaniem informacji. W tym przypadku mogą wystąpić różne sytuacje. Może się okazać, że mnożna jest kwaternionem wektorowym, a mnożnik jest kwaternionem zwykłym. Innym problemem jest sytuacja, gdy mnożna jest kwaternionem zwykłym, a mnożnik jest kwaternionem wektorowym. Wreszcie może wystąpić sytuacja, w której oba czynniki są kwaternionami czystymi. We wszystkich trzech przypadkach liczba rzeczywistych mnożeń może zostać zmniejszona w porównaniu z liczbą rzeczywistych mnożeń wymaganych do obliczenia iloczynu kwaternionów zwykłych. W artykule omówiono rozwiązania algorytmiczne, które pozwalają na zmniejszenie liczby rzeczywistych mnożeń podczas obliczania iloczynów kwaternionów w każdym z wymienionych przypadków.*

Keywords: hypercomplex data processing, quaternion multiplication, pure quaternion
Słowa kluczowe: analiza obwodów elektrycznych, mnożenie kwaternionów, czysty kwaternion

Introduction

Currently, hypercomplex numbers, in particular quaternions [1], are used to solve data processing problems in various fields of science and technology. This concerns such fields as electrical engineering [2, 3], mechanics [4], robotics [5-7], machine vision [8, 9], computer graphics and animation [10-12], digital signal processing and image processing [13], neural networks [14], and many others [15].

It should be noted, however, that quaternion-valued arithmetic operations are, in one way or another, reduced to performing operations with real numbers. Among other arithmetic operations on quaternions, the most expensive operation is the operation of their multiplication. The quaternion multiplication is performed in quaternion-valued filtering for color-edge detection and image smoothing [13]. The quaternion principal component analysis [13] and quaternion convolution [14] also use the quaternion multiplication. In [2, 3] electrical single and three-phase quantities such as voltage, current, impedance, and power are defined as quaternions, and the multiplication of quaternions is used in the electrical context.

The computational complexity of quaternion-valued arithmetic operations is because to perform just one addition of quaternions, 4 operations of addition of real numbers are required, and the quaternion multiplier requires 16 operations of multiplication and 12 operations of addition. It is also known that among arithmetic operations on real data, multiplication is also the most time-intensive operation [16]. To reduce computation time, in [17,18] authors proposed algorithms with a minimum number of multiplications [17,18]. However, it turns out that if at least one of the quaternion-valued factors is a pure quaternion, the number of real multiplications can be further reduced. Below we propose algorithmic solutions that require the fewest number of arithmetic operations when calculating the product of quaternions if at least one quaternion is pure.

Preliminaries

A quaternion can be defined as $u = u_0 + u_1 i + u_2 j + u_3 k$, where $u_0, u_1, u_2, u_3 \in R$, R is a real number field, and i, j, k are basic imaginary units satisfying the following multiplication relations [1]:

$$i^2 = j^2 = k^2 = -1,\ ij = -ji = k,\ jk = -kj = i,$$
$$ki = -ik = j.$$

We would like to immediately make it clear that since the multiplication of quaternions is non-commutative, we will use the left-sided product for the sake of certainty. The right-sided product can be rationalized similarly.

Typically the quaternions are multiplied similarly to the multiplying the polynomials.

If $u = u_0 + u_1 i + u_2 j + u_3 k$, and $x = x_0 + x_1 i + x_2 j + x_3 k$ are quaternions, then the product ux is expressed as follows

$$\begin{aligned} ux &= (u_0 + u_1 i + u_2 j + u_3 k)(x_0 + x_1 i + x_2 j + x_3 k) \\ &= (u_0 x_0 - u_1 x_1 - u_2 x_2 - u_3 x_3) \\ &+ (u_0 x_1 + u_1 x_0 + u_2 x_3 - u_3 x_2) i \\ &+ (u_0 x_2 - u_1 x_3 + u_2 x_0 + u_3 x_1) j \\ &+ (u_0 x_3 + u_1 x_2 - u_2 x_1 + u_3 x_0) k. \end{aligned} \tag{1}$$

Applying (1) requires 16 multiplications and 12 additions of real numbers. In [17] a more efficient algorithm for calculating the product of two quaternions with reduced multiplicative complexity is proposed. Using this algorithm the two quaternions can be multiplied using 8 multiplications and 28 additions of real numbers. In the general case, this algorithm additionally requires performing 8 shifts. However, if at least one quaternion in the product is pure, the number of real multiplications can be reduced. Let us consider this possibility in more detail.

The following notations are used below [19-20]: $\mathbf{I}_N$ is an order N identity matrix; $\mathbf{H}_2$ is a 2×2 Hadamard matrix; $\otimes$ is the Kronecker product of two matrices; $\oplus$ is the direct sum of two matrices.

Algorithm for the multiplying an ordinary quaternion by a pure quaternion

If $x = x_0 + x_1 i + x_2 j + x_3 k$ is an ordinary quaternion, and $u = u_1 i + u_2 j + u_3 k$ is a pure quaternion ($u_0 = 0$), then the multiplying x by u as polynomials is given by expression

$$\begin{aligned} ux &= (u_1 i + u_2 j + u_3 k)(x_0 + x_1 i + x_2 j + x_3 k) \\ &= (-u_1 x_1 - u_2 x_2 - u_3 x_3) + (u_1 x_0 + u_2 x_3 - u_3 x_2) i \\ &+ (-u_1 x_3 + u_2 x_0 + u_3 x_1) j + (u_1 x_2 - u_2 x_1 + u_3 x_0) k. \end{aligned} \tag{2}$$

The computation of the expression (2) requires 12 multiplications and 11 additions of real numbers. Below we show the derivation of an algorithmic solution characterized by lower multiplicative complexity. If we express the ordinary quaternion x and pure quaternion u in matrix notation then the following expression is obtained:

$$\mathbf{Y}_{4\times 1} = \mathbf{U}_4 \mathbf{X}_{4\times 1},$$

where $\mathbf{Y}_{4\times 1} = [y_0, y_1, y_2, y_3]^{\mathrm{T}}$, $\mathbf{X}_{4\times 1} = [x_0, x_1, x_2, x_3]^{\mathrm{T}}$,

$$\mathbf{U}_4 = \begin{bmatrix} 0 & -u_1 & -u_2 & -u_3 \\ u_1 & 0 & -u_3 & u_2 \\ u_2 & u_3 & 0 & -u_1 \\ u_3 & -u_2 & u_1 & 0 \end{bmatrix}.$$

After altering the sign of the first row of the matrix $\mathbf{U}_4$, the obtained matrix $\mathbf{U}_4^{(a)}$ was decomposed as

$$\mathbf{U}_4^{(a)} = \mathbf{U}_4^{(b)} - \mathbf{U}_4^{(c)}, \tag{3}$$

where

$$\mathbf{U}_4^{(b)} = \begin{bmatrix} 0 & u_1 & u_2 & u_3 \\ u_1 & 0 & u_3 & u_2 \\ u_2 & u_3 & 0 & u_1 \\ u_3 & u_2 & u_1 & 0 \end{bmatrix}, \ \mathbf{U}_4^{(c)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2u_3 & 0 \\ 0 & 0 & 0 & 2u_1 \\ 0 & 2u_2 & 0 & 0 \end{bmatrix}.$$

Comparison with the matrix templates from [21] shows that the matrix $\mathbf{U}_4^{(b)}$ matches the pattern $\begin{bmatrix} \mathbf{A}_2 & \mathbf{B}_2 \\ \mathbf{B}_2 & \mathbf{A}_2 \end{bmatrix}$, where $\mathbf{A}_2 = \begin{bmatrix} 0 & u_1 \\ u_1 & 0 \end{bmatrix}$, $\mathbf{B}_2 = \begin{bmatrix} u_2 & u_3 \\ u_3 & u_2 \end{bmatrix}$.

Then we obtain the factorization

$$\begin{aligned} \mathbf{U}_4^{(b)} &= 1/2 \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right) \\ &\times \left[\left(\mathbf{A}_2 + \mathbf{B}_2\right) \oplus \left(\mathbf{A}_2 - \mathbf{B}_2\right)\right] \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right). \end{aligned} \tag{4}$$

The resulting matrices match the pattern $\begin{bmatrix} a & b \\ b & a \end{bmatrix}$, where $a = u_2$, $b = u_1 + u_3$ for matrix $\mathbf{A}_2 + \mathbf{B}_2$, and $a = -u_2$, $b = u_1 - u_3$ for the matrix $\mathbf{A}_2 - \mathbf{B}_2$. Therefore, $\mathbf{A}_2 + \mathbf{B}_2$ is factorized as

$$\mathbf{A}_2 + \mathbf{B}_2 = 1/2 \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right) \left[\left(a+b\right) \oplus \left(a-b\right)\right] \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right). \tag{5}$$

The matrix $\mathbf{A}_2 - \mathbf{B}_2$ is decomposed similarly.

Taking into account the product

$$\mathbf{U}_4^{(c)} \mathbf{X}_{4\times 1} = diag\left(0, 2u_3, 2u_1, 2u_2\right)[x_0, x_2, x_3, x_1]^{\mathrm{T}},$$

and the expressions (3), (4), and (5), we obtain the following factorization of the quaternion matrix:

$$\mathbf{Y}_{4\times 1} = \mathbf{W}_{4\times 7} \mathbf{W}_7^{(2)} \mathbf{W}_7^{(0)} \mathbf{D}_7 \mathbf{W}_7^{(0)} \mathbf{W}_7^{(1)} \mathbf{W}_{7\times 4} \mathbf{X}_{4\times 1}, \tag{6}$$

where

$$\mathbf{D}_7 = diag\left(s_0, s_1, s_2, s_3, s_4, s_5, s_6\right),$$

$$s_0 = \left(u_1 + u_2 + u_3\right)/4, \ s_1 = \left(u_2 - u_1 - u_3\right)/4,$$

$$s_2 = \left(-u_2 + u_1 - u_3\right)/4; \ s_3 = \left(-u_2 - u_1 + u_3\right)/4;$$

$$s_4 = 2u_3; \ s_5 = 2u_1; \ s_6 = 2u_2;$$

$$\mathbf{W}_7^{(1)} = \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right) \oplus \mathbf{P}_3; \ \mathbf{P}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix};$$

$$\mathbf{W}_7^{(2)} = \left(\mathbf{H}_2 \otimes \mathbf{I}_2\right) \oplus \mathbf{I}_3; \ \mathbf{W}_7^{(0)} = \mathbf{H}_2 \oplus \mathbf{H}_2 \oplus \mathbf{I}_3;$$

$$\mathbf{W}_{7\times 4} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}};$$

$$\mathbf{W}_{4\times 7} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}.$$

Based on the resulting factorization (6) of the quaternion matrix, the algorithm for multiplying an ordinary quaternion by a pure quaternion is shown by the data flow graph in Figure 1. It illustrates the space-time structure of the developed algorithm. The data flow graph is oriented from right to left, and the straight lines denote the data transfer operations. The dashed lines determine the data transfer operations by simultaneously altering the sign. Nodes with the connected lines define summations, and nodes with the diverged lines mean data transfer. Circles denote multiplications.

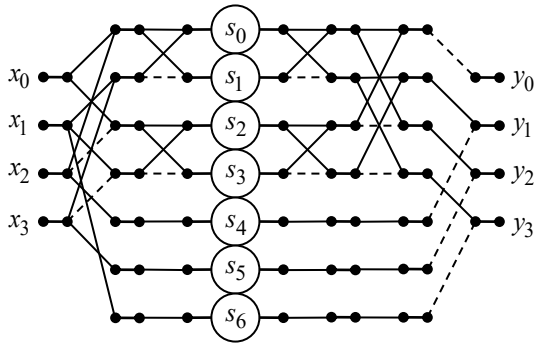

Fig. 1. Data flow graph of the algorithm for multiplying an ordinary quaternion by a pure quaternion

Now we will construct an algorithm for calculating the coefficients s_0, s_1, s_2, s_3. In matrix notation, we obtain the expression

$$\mathbf{S}_{4\times1}=\mathbf{W}_{4\times3}\mathbf{U}_{3\times1},\ \mathbf{S}_{4\times1}=\left(s_0,s_1,s_2,s_3\right)^{\mathrm{T}},$$

$$\mathbf{U}_{3\times1}=[u_1,u_2,u_3]^{\mathrm{T}},\ \mathbf{W}_{4\times3}=\frac{1}{4}\begin{bmatrix}1&1&1\\-1&1&-1\\1&-1&-1\\-1&-1&1\end{bmatrix}.$$

Having decomposed the matrix $\mathbf{W}_{4\times3}$ we obtain

$$\mathbf{S}_{4\times1}=1/4\mathbf{W}_{4\times3}^{(0)}\mathbf{W}_3\mathbf{U}_{3\times1},$$

where $\mathbf{W}_3=\begin{bmatrix}1&0&1\\0&1&0\\1&0&-1\end{bmatrix}$.

The algorithm for the calculation of the coefficients s_0, s_1, s_2, s_3 of the multiplying an ordinary quaternion by a pure quaternion is presented in the data flow graph in Figure 2. We are able to reduce the number of additions from 8 to 6 compared to the calculation of these coefficients using the expression (6).

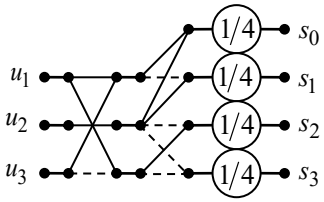

Fig. 2. The data flow graph for the calculation of the coefficients for multiplying an ordinary quaternion by a pure quaternion

From here on it is easy to see that the algorithms we synthesize contain multiplications by powers of two. These are the so-called trivial multiplications since they are reduced to simple shift operations. Some researchers do not even take these operations into account when calculating the total number of mathematical operations required to implement the algorithms. However, for the sake of completeness, we will still take them into account.

So, in general, the algorithm requires 7 multiplications, 25 additions, four double right shifts, and three left shifts. However, if the quaternion u, as is often the case, has constant coefficients, then the number of additions is reduced to 19, and the shifts disappear altogether.

Algorithm for the multiplying a pure quaternion by an ordinary quaternion

Let $u=u_0+u_1i+u_2j+u_3k$ be an ordinary quaternion, $x=x_1i+x_2j+x_3k$ be a pure quaternion ($x_0=0$). The multiplying x by u as polynomials is given as

$$\begin{aligned}ux&=(u_0+u_1i+u_2j+u_3k)(x_1i+x_2j+x_3k)\\&=(-u_1x_1-u_2x_2-u_3x_3)+(u_0x_1+u_2x_3-u_3x_2)i\\&+(u_0x_2-u_1x_3+u_3x_1)j+(u_0x_3+u_1x_2-u_2x_1)k.\end{aligned}\tag{7}$$

As one can be seen, the computation of the expression (7) requires 12 multiplications and 11 additions of real numbers. We show the derivation of an algorithmic solution characterized by lower multiplicative complexity. If we express the ordinary quaternion u and pure quaternion x in matrix notation then the following expression is obtained:

$$\mathbf{Y}_{4\times1}=\mathbf{U}_{4\times3}\mathbf{X}_{3\times1},$$

where

$$\mathbf{X}_{3\times1}=[x_1,x_2,x_3]^{\mathrm{T}},\ \mathbf{U}_{4\times3}=\begin{bmatrix}-u_1&-u_2&-u_3\\u_0&-u_3&u_2\\u_3&u_0&-u_1\\-u_2&u_1&u_0\end{bmatrix}.$$

Next, we alter the sign of the first row of the matrix $\mathbf{U}_{4\times3}$. The obtained matrix $\mathbf{U}_{4\times3}^{(a)}$ can be decomposed as

$$\mathbf{U}_{4\times3}^{(a)}=\mathbf{U}_{4\times3}^{(b)}-\mathbf{U}_{4\times3}^{(c)},\tag{8}$$

where

$$\mathbf{U}_{4\times3}^{(b)}=\begin{bmatrix}u_1&u_2&u_3\\u_0&u_3&u_2\\u_3&u_0&u_1\\u_2&u_1&u_0\end{bmatrix},\ \mathbf{U}_{4\times3}^{(c)}=\begin{bmatrix}0&0&0\\0&2u_3&0\\0&0&2u_1\\2u_2&0&0\end{bmatrix}.$$

It should be said that it is difficult to find a successful factorization for the matrix $\mathbf{U}_{4\times3}^{(b)}$. However, we can artificially supplement it with a column of the following type $[u_0,u_1,u_2,u_3]^{\mathrm{T}}$. As a result, we obtain the matrix

$$\mathbf{U}_4^{(d)}=\begin{bmatrix}u_0&u_1&u_2&u_3\\u_1&u_0&u_3&u_2\\u_2&u_3&u_0&u_1\\u_3&u_2&u_1&u_0\end{bmatrix},$$

which can be factorized instead of the matrix $\mathbf{U}_{4\times3}^{(b)}$. The matrix obtained in this way can be easily factorized [17] using decompositions (4) and (5), Then, considering that for pure quaternion $x_0=0$, the resulting decomposition and data flow graph can be corrected. Then, taking into account the fact that the resulting decomposition, as well as the algorithm graph, can be corrected.

Based on the expressions (4)-(6) and (8) we obtain the following factorization of the quaternion matrix:

$$\mathbf{Y}_{4\times1}=\mathbf{W}_{4\times7}\mathbf{W}_7^{(2)}\mathbf{W}_7^{(0)}\mathbf{D}_7^{(a)}\mathbf{W}_7^{(0)}\mathbf{W}_7^{(1)}\mathbf{W}_{7\times3}\mathbf{X}_{3\times1},\tag{9}$$

where $\mathbf{D}_7^{(a)}=diag\left(s_7,s_8,s_9,s_{10},s_4,s_5,s_6\right)$,

$s_7=\left(u_0+u_1+u_2+u_3\right)/4$; $s_8=\left(u_0+u_2-u_1-u_3\right)/4$;

$s_9=\left(u_0-u_2+u_1-u_3\right)/4$; $s_{10}=\left(u_0-u_2-u_1+u_3\right)/4$;

$$\mathbf{W}_{7\times3}=\begin{bmatrix}0&1&0&0&1&0&0\\0&0&1&0&0&1&0\\0&0&0&1&0&0&1\end{bmatrix}^{\mathrm{T}},$$

and the matrices $\mathbf{W}_{4\times7}$, $\mathbf{W}_7^{(0)}$, $\mathbf{W}_7^{(1)}$, $\mathbf{W}_7^{(2)}$ and coefficients s_4, s_5, s_6 are defined in (6).

The algorithm for multiplying a pure quaternion by a regular quaternion, based on the obtained factorization (9) of the quaternion matrix, is shown by the data flow graph in Figure 3.

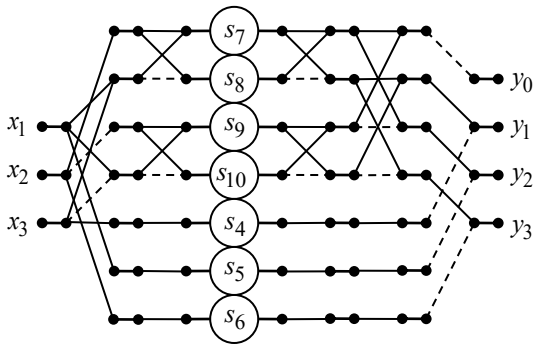

Fig. 3. Data flow graph of the algorithm of the multiplying a pure quaternion by an ordinary quaternion

Let synthesize the algorithm to calculate the coefficients s_7, s_8, s_9, s_{10}. In matrix notation we obtain the expression

$$\mathbf{S}_{4\times1}^{(a)} = \mathbf{W}_4\mathbf{U}_{4\times1}, \mathbf{S}_{4\times1}^{(a)} = \left(s_7, s_8, s_9, s_{10}\right)^{\mathrm{T}},$$

$$\mathbf{U}_{4\times1} = [u_0, u_1, u_2, u_3]^{\mathrm{T}}, \ \mathbf{W}_4 = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

The matrix $\mathbf{W}_4$ matches the pattern $\begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix}$, and we give

$$\mathbf{S}_{4\times1}^{(a)} = 1/4\,\mathbf{W}_4^{(1)}\mathbf{W}_4^{(0)}\mathbf{U}_{4\times1},$$

where $\mathbf{W}_4^{(1)} = \mathbf{H}_2 \oplus \mathbf{H}_2$, $\mathbf{W}_4^{(0)} = \mathbf{H}_2 \otimes \mathbf{I}_2$.

The algorithm for the calculation of the coefficients s_7, s_8, s_9, s_{10} during the multiplying a pure quaternion by an ordinary quaternion is presented in the data flow graph in Figure 4.

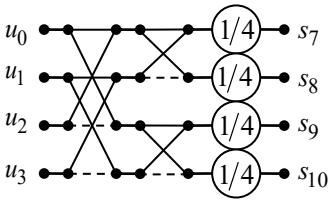

Fig. 4. The data flow graph for the calculation of the coefficients for the multiplying a pure quaternion by an ordinary quaternion

The algorithm requires 7 multiplications, 25 additions, four double right shifts and three left shifts. However, if the quaternion u, as is often the case, has constant coefficients, then the number of additions is reduced to 19, and the shifts disappear altogether.

Algorithm for the multiplying two pure quaternions

If $u = u_1 i + u_2 j + u_3 k$ and $x = x_1 i + x_2 j + x_3 k$ are pure quaternions, then the multiplying x by u as polynomials is given by expression

$$\begin{aligned} ux &= (u_1 i + u_2 j + u_3 k)(x_1 i + x_2 j + x_3 k) \\ &= (-u_1 x_1 - u_2 x_2 - u_3 x_3) + (u_2 x_3 - u_3 x_2) i \\ &+ (u_3 x_1 - u_1 x_3) j + + (u_1 x_2 - u_2 x_1) k. \end{aligned} \tag{10}$$

Calculating the product (10) requires 9 multiplications and 5 additions of real numbers. We can show a solution that requires fewer operations. If we represent the multiplication of pure quaternions u and x in matrix notation, we obtain the following expression:

$$\mathbf{Y}_{4\times1} = \mathbf{U}_{4\times3}^{(d)}\mathbf{X}_{3\times1},$$

where $\mathbf{U}_{4\times3}^{(d)} = \begin{bmatrix} -u_1 & -u_2 & -u_3 \\ 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$.

If we change the sign of the first row of the matrix $\mathbf{U}_{4\times3}^{(d)}$ then the obtained matrix $\mathbf{U}_{4\times3}^{(e)}$ can be decomposed as

$$\mathbf{U}_{4\times3}^{(e)} = \mathbf{U}_{4\times3}^{(f)} - \mathbf{U}_{4\times3}^{(c)}, \tag{11}$$

where $\mathbf{U}_{4\times3}^{(c)}$ is defined in (8), and

$$\mathbf{U}_{4\times3}^{(f)} = \begin{bmatrix} u_1 & u_2 & u_3 \\ 0 & u_3 & u_2 \\ u_3 & 0 & u_1 \\ u_2 & u_1 & 0 \end{bmatrix}.$$

It should be said that it is also difficult to find a successful factorization for matrix $\mathbf{U}_{4\times3}^{(f)}$. However, we can artificially supplement it with a column of the following type $[0, u_1, u_2, u_3]^{\mathrm{T}}$. As a result, we obtain the matrix $\mathbf{U}_4^{(b)}$. This matrix can be factorized instead of the matrix $\mathbf{U}_{4\times3}^{(f)}$, using expressions (4), (5). Then, taking into account the fact that for pure quaternion $x_0 = 0$, the resulting decomposition, as well as the data flow graph, can be corrected.

Further, based on the expressions (4)-(6) and (9), (11) we obtain the following factorization of the quaternion matrix:

$$\mathbf{Y}_{4\times1} = \mathbf{W}_{4\times7}\mathbf{W}_7^{(2)}\mathbf{W}_7^{(0)}\mathbf{D}_7\mathbf{W}_7^{(0)}\mathbf{W}_7^{(1)}\mathbf{W}_{7\times3}\mathbf{X}_{3\times1}, \tag{12}$$

where $\mathbf{D}_7, \mathbf{W}_7^{(0)}$, $\mathbf{W}_7^{(1)}$, $\mathbf{W}_7^{(2)}$ are defined as in (6), and $\mathbf{W}_{7\times3}$ is determined as in (9).

Applying the factorization (12) of the quaternion matrix, the algorithm for multiplying two pure quaternions can be presented by data flow graph in Figure 3 where s_0, s_1, s_2, s_3 are used as multipliers instead of the s_7, s_8, s_9, s_{10}. The data flow graph for computing the coefficients s_0, s_1, s_2, s_3 was shown in Figure 2.

As in the previous case, the algorithm requires 7 multiplications, 25 additions, four double right shift operations, and three left shifts. If the quaternion u is a constant quaternion, then the number of additions is also reduced to 17, and shifts are not required.

Discussion of the computational complexity and verification of the obtained solutions

The tables below present estimates of the number of arithmetic operations required to calculate a product of quaternions. The cases of multiplication of two ordinary

quaternions (a well-known algorithm) are considered, as well as new algorithmic solutions concerning the calculation of products when one or both quaternions are pure quaternions. Table 1 summarizes estimates regarding schoolbook methods for calculating quaternion products. Table 2 presents the estimates of the proposed algorithms for multiplying quaternions with variable coefficients, and Table 3 presents the estimates of the same algorithms for the case when the coefficients of the quaternion u are constant numbers.

An experimental study of the correctness of the proposed algorithms was implemented using the MATLAB R2023b software. For this purpose, fifty 4D vectors with normally distributed elements with zero mean and unit variance were generated. The resulting vectors were presented as quaternions. Then, the products of ordinary and pure quaternions, pure and ordinary quaternions, and the product of two pure quaternions were calculated using matrix-vector products. In addition, calculations were implemented with quaternion matrices factorized using expressions (6), (9), and (12). The results were found to match, indicating the correctness of the developed algorithms. In addition, the correctness of the proposed data flow graphs for the constructed algorithms was checked on the same generated 4D random vectors. The corresponding quaternions were multiplied first as usual matrix-vector products, and then using data flow graphs. The coincidence of the results showed the correctness of the data flow graphs for the proposed algorithms.

Table 1. The number of operations in schoolbook methods for the quaternion multiplying

Quaternion product:	Multiplications	Additions	Shifts
ordinary × ordinary	16	12	-
ordinary × pure	12	11	-
pure × ordinary	12	11	-
pure × pure	9	5	-

Table 2. The number of operations for the proposed quaternion multiplying algorithms

Quaternion product:	Multiplications	Additions	Shifts
ordinary × ordinary [17]	8	28	8
ordinary × pure	7	25	7
pure × ordinary	7	25	7
pure × pure	7	25	7

Table 3. The number of operations for quaternion multiplication algorithms in the case where quaternion u is a “constant” quaternion

Quaternion product:	Multiplications	Additions	Shifts
ordinary × ordinary [17]	8	20	4
ordinary × pure	7	19	4
pure × ordinary	7	17	4
pure × pure	7	17	4

Conclusions

The paper presents new algorithms for calculating the product of two quaternions, assuming that at least one quaternion is pure. Even compared to the best fast quaternion multiplication algorithm [17], we save one real multiplication. This may seem like a small thing, but it is important too. Unfortunately, the number of real additions has increased. However, as already noted, a common situation is when one of the quaternions (in our case, u) has constant number coefficients. In this case, the number of additions is reduced. In cases where the multiplication operation is more costly than the addition operation, the resulting solutions become especially useful. The noted advantages of the developed algorithms allow us to assume their practical suitability for solving applied problems in various fields of science and technology.

***Authors**: prof. dr hab. inż. Aleksandr Cariow, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Informatyki, ul. Żołnierska 49, 71-210 Szczecin, E-mail: acariow@zut.edu.pl ; dr hab. Marina Polyakova, Narodowy Uniwersytet Politechnika w Odesie, Instytut Systemów Komputerowych, al. Szewczenki 1, 65044 Odesa, E-mail: polyakova@opu.edu.ua.*

REFERENCES

[1] Kantor I., Solodovnikov A., Hypercomplex numbers, Springer-Verlag, New York, 1989.
[2] Barry N., The application of quaternions in electrical circuits, 2016 27th Irish Signals and Systems Conference (ISSC), IEEE, 2016, DOI: 10.1109/ISSC.2016.7528440.
[3] Do Prado Brazil V., de Leles Ferreira Filho A., Ishihara J. Y., "Electrical three phase circuit analysis using quaternions," 2018 18th International Conference on Harmonics and Quality of Power (ICHQP), Ljubljana, Slovenia, 2018, pp. 1-6, doi: 10.1109/ICHQP.2018.8378813.
[4] Wie B., Barba P. M., Quaternion Feedback for Spacecraft Large Angle Maneuvers, *Journal of Guidance Control and Dynamics*, vol. 8, no. 3, pp 360- 365, 1985.
[5] Yuan J. S. C, Closed Loop Manipulator Control With Quaternion Feedback, *IEEE Trans. on Robotics and Automation*, vol. 4, no. 4, pp. 434-439, 1988.
[6] Chao Hu, Meng, M. Q. H., Mandal, M., Liu, P.X., Robot Rotation Decomposition Using Quaternions, International Conference on Mechatronics and Automation, 25-28 June 2006, 1158 – 1163.
[7] Cariow A., Cariowa G., Majorkowska-Mech D., An algorithm for quaternion-based 3D rotation, *Int. J. Appl. Math. Comput. Sci*., 2020, vol. 30, No. 1, pp. 149–160
[8] Mukundan R., Malik N. K., Attitude Estimation Using Moment Invariants, *Pattern Recognition Letters*, vol 14, pp. 199-205, 1993.
[9] Horn B. K. P, Closed Form Solution of Absolute Orientation Using Unit Quaternions, Journal of the Optical Society of America, Vol. 4, No. 4 (1987), pp 629- 642.
[10] Watt A., Watt M., Advanced Animation and Rendering Techniques, Addison Wesley (1992).
[11] Vince, J. Virtual Reality Systems, Addison-Wesley (1999).
[12] Vince, J. Quaternions for Computer Graphics; Springer: London, UK, 2011.
[13] Miron S., Flamant J., Le Bihan N., Chainais P., Brie D., Quaternions in Signal and Image Processing: A comprehensive and objective overview, IEEE Signal Processing Magazine, vol. 40, no. 6, pp. 26-40, 2023, DOI: 10.1109/MSP.2023.3278071
[14] Hirose A., Shang F., Otsuka Y., Natsuaki R., Matsumoto Y, Usami N, Song Y., Chen H., Quaternion Neural Networks: A physics-incorporated intelligence framework, *IEEE Signal Processing Magazine*, vol. 41, no. 3, pp. 88 – 100, 2024, doi: 10.1109/MSP.2024.3384179
[15] Bayro-Corrochano E. A Survey on Quaternion Algebra and Geometric Algebra Applications in Engineering and Computer Science 1995-2020, *IEEE Access*, v. 9, pp. 104326-104355, 2021, Doi: 10.1109/ACCESS.2021.3097756.
[16] de Groote H. F., On the complexity of quaternion multiplication, *Infor. Process. Lett*., vol.43, no.6, pp.161-164, 1975.
[17] Ţariova G., Ţariov A., Aspekty algorytmiczne redukcji liczby bloków mnożących w układzie do obliczania iloczynu dwóch kwaternionów, *Pomiary, Automatyka, Kontrola*, Nr 7, str. 668-690, 2010.
[18] Cariow A., Cariowa G., A unified approach for developing rationalized algorithms for hypercomplex number multiplication. *Przegląd Elektrotechniczny*, ISSN 0033-2097, R. 91 NR 2/2015, pp. 36-39. doi:10.15199/48.2015.02.09
[19] Granata J., Conner M., and Tolimieri R., The tensor product: A mathematical programming language for FFTs and other fast DSP operations, *IEEE Signal Process. Mag.*, vol. 9, no. 1, pp. 40–48, Jan. 1992.
[20] Steeb W.-H., Hardy Y., Matrix Calculus and Kronecker Product: A Practical Approach to Linear and Multilinear Algebra, World Scientific Publishing Company; 2 edition, 2011.
[21] Cariow A., A simple and practical approach to development of the fast algorithms for matrix-vector multiplication. In 2019 Signal Processing Symposium (SPSympo), 17-19 September 2019, Krakow, Poland, DOI: 10.1109/SPS.2019.8882008.