A Fast Algorithm for Quaternion Multiplication

Janusz P. Papliński, Aleksandr Cariow
West Pomeranian University of Technology in Szczecin, Faculty of Computer Science and Information Technology, Żołnierska 49, 71-210 Szczecin, Poland

Marina Polyakova
Odesa Polytechnic National University, Institute of Computer Systems, 1 Shevchenko Ave., Odesa, 65044, Ukraine

Abstract: Quaternion algebra is used in many fields, including electrical engineering, robotics, computer graphics, navigation systems, and image and signal processing. In these applications, reducing computational complexity is particularly important, especially by limiting the number of real multiplications required to compute quaternion products. This paper proposes fast quaternion multiplication algorithms that reduce the number of real multiplications compared with the classical direct method while preserving computational correctness.

Keywords: quaternion multiplication, fast algorithms, computational complexity, real multiplication reduction, Hadamard matrices, signal processing, three-phase electrical signal

1. Introduction

Quaternion algebra is widely used to model and process data in numerous fields, including electrical engineering [1, 2], quantum mechanics [3], robotics [4], engineering and computer science [5], signal and image processing [6, 7], neural networks [8], image inpainting [9], and many other applications [10–15]. In electrical engineering, quaternion algebra has important applications in the analysis and processing of multiphase power systems, signal polarization, and three-dimensional rotations arising in electromagnetic fields and control systems [16, 17]. For example, quaternions provide a natural framework for representing and analyzing three-phase electrical quantities, such as voltages, currents, and power, as quaternion-valued signals [16]. This approach extends classical complex-power theory to unbalanced and delta-connected loads, enabling unbalanced systems to be decomposed into balanced components and thereby improving power-system analysis. Furthermore, quaternion power expressions generalize complex power by replacing the imaginary unit with quaternion units, allowing a more comprehensive representation of instantaneous power in three-wire systems [16].

Among the basic operations of quaternion algebra, multiplying two quaternions is the most computationally expensive. As noted in [18–23], direct computation of a quaternion product requires 16 real multiplications and 12 real additions. Numerous techniques have been proposed to reduce this computational complexity, including distributed arithmetic [19], the Coordinate Rotation Digital Computer (CORDIC) algorithm [18, 20], the lifting scheme [21], and combinations of the latter two approaches. A lifting-based quaternion multiplier requires 12 real multiplications and 12 real additions, representing a 25 % reduction in the number of multiplications compared with the direct implementation. Similarly, the four-dimensional CORDIC algorithm requires 12 real multiplications and only eight real additions.

Although the latency of floating-point multiplication on modern processors is often comparable to that of addition, minimizing the number of multiplications remains an important objective. In Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) implementations, multipliers are still considerably more expensive than adders: they require more hardware resources, consume more power, and frequently constitute the critical computational resource in embedded systems. In an early paper [22], the authors proposed decomposing the quaternion product into a weighted sum of Hadamard matrices. Their algorithm requires only 8 real multiplications and 40 real additions. By further transforming the resulting Hadamard-matrix decomposition, they derived another implementation requiring the same 8 real multiplications but only 32 real additions. The algorithms reported in [23] and [24] are descendants of the eight-multiplication algorithm originally proposed by Howell and Lafon [22]. Their approach exploits the structural properties of the quaternion multiplication matrix to reduce the computational complexity to eight real multiplications and only 20 real additions. However, an alternative factorization can be obtained that preserves the same number of multiplications while slightly reducing the number of real additions.

Motivated by the approach presented in [22], this paper develops an alternative quaternion multiplication algorithm that further reduces the number of real additions while preserving the same number of real multiplications. Although the proposed algorithm has a computational complexity comparable to that of the algorithm in [22], it differs substantially in its derivation, computational implementation, and space–time structure.

Autor korespondujący:
Janusz P. Papliński, janusz.paplinski@zut.edu.pl

Artykuł recenzowany
nadesłany 18.03.2026 r., przyjęty do druku 11.08.2026 r.

Howell and Lafon [22] obtained their algorithm through tensor-rank decomposition of the quaternion product and expressed the result as eight bilinear products. In contrast, the proposed method starts from the matrix–vector representation of quaternion multiplication and derives an equivalent eight-multiplication scheme through explicit matrix factorization. Consequently, although both approaches attain the same minimal multiplicative complexity, the sequences of additions, intermediate variables, and data-flow structures are different.

2. Preliminary remarks

Let $\mathbb{H}$ be a four-dimensional vector space over the field of real numbers $\mathbb{R}$, and let i, j, and k be the basic imaginary units whose multiplication rules are given in Table 1.

Table 1. Multiplication rules for the basic imaginary units
Tabela 1. Zasady mnożenia podstawowych jednostek urojonych

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

A quaternion $q \in \mathbb{H}$ is a hypercomplex number that can be written as $q = q_0 + q_1 i + q_2 j + q_3 k$ where $q_l \in \mathbb{R}$, $l = 0, 1, 2, 3$; and i, j, k are the quaternion imaginary basis elements satisfying the well-known Hamilton multiplication rules $i^2 = j^2 = k^2 = ijk = -1$, $ij = k$, $jk = i$, $ki = j$ [5–7].

Now suppose that we are given two quaternions $q = q_0 + q_1 i + q_2 j + q_3 k$, $p = p_0 + p_1 i + p_2 j + p_3 k$ and we wish to compute their product. It is important to note that quaternion multiplication is non-commutative, that is, $qp \neq pq$ in general, which has specific algebraic and computational consequences. The product $y = qp$ can be computed using the following expression, which follows directly from the Hamilton multiplication rules and the bilinearity of quaternion multiplication [5–7]:

$$\begin{aligned} y = qp &= (q_0 + q_1 i + q_2 j + q_3 k)(p_0 + p_1 i + p_2 j + p_3 k) \\ &= (q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3) \\ &\quad + (q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2) i \\ &\quad + (q_0 p_2 + q_2 p_0 + q_3 p_1 - q_1 p_3) j \\ &\quad + (q_0 p_3 + q_3 p_0 + q_1 p_2 - q_2 p_1) k. \end{aligned}$$

We introduce the two column vectors $\mathbf{X}_{4\times1} = [p_0, p_1, p_2, p_3]^{\mathrm{T}}$ and $\mathbf{Y}_{4\times1} = [y_0, y_1, y_2, y_3]^{\mathrm{T}}$ together with the square matrix

$$\mathbf{Q}_4 = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}.$$

Then, quaternion multiplication can be expressed in the matrix–vector form as

$$\mathbf{Y}_{4\times1} = \mathbf{Q}_4 \mathbf{X}_{4\times1}. \tag{1}$$

3. Fast quaternion multiplication algorithm

Let $\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ denote the 2×2 Hadamard matrix and let $\mathbf{Q}_{4\times1} = [q_0, q_1, q_2, q_3]^{\mathrm{T}}$. Furthermore, let $\mathbf{I}_N$ denote the identity matrix of order N. An empty cell in a matrix denotes a zero element. We use the symbol "$\otimes$" to denote the Kronecker (tensor) product of two matrices and the symbol "$\oplus$" to denote their direct sum.

The fast quaternion multiplication algorithm is constructed in three stages. First, based on an analytical representation of the quaternion product, we define an algorithm-forming vector that serves as the basis for the subsequent matrix factorization. Next, we present the key factorization of the corresponding quaternion multiplication matrix. Finally, we translate the resulting computational scheme into an implementable form using data-flow graphs that illustrate the sequence of arithmetic operations.

To determine an algorithm-forming vector for the quaternion multiplication algorithm, we formulate the following auxiliary lemma. We introduce the matrix $\mathbf{W}_8^{(1)} = \mathbf{H}_2 \oplus \mathbf{H}_2 \oplus \mathbf{H}_2 \oplus \mathbf{H}_2$, which is a direct sum of four 2×2 Hadamard submatrices, thereby reducing the number of additions. We also define two auxiliary matrices:

$$\mathbf{W}_8^{(0)} = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix} \oplus \begin{bmatrix} & & & 1 \\ & & 1 & \\ 1 & & & \\ & 1 & & \end{bmatrix},$$

$$\mathbf{W}_{8\times4} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ & & 1 & \\ 1 & & & \\ & & 1 & \\ & & & 1 \\ & 1 & & \end{bmatrix}.$$

Based on the algorithm presented in [22], we formulate the following lemma.

Lemma. Let

$$\mathbf{S}_{8\times1} = [q_3 - q_2, q_0 + q_1, q_0 - q_1, q_3 + q_2, q_3 - q_1, q_3 + q_1, q_0 + q_2, q_0 - q_2]^{\mathrm{T}}.$$

denote the algorithm-forming vector. Then, the following factorization holds:

$$\mathbf{S}_{8\times1} = \mathbf{W}_8^{(0)} \mathbf{W}_8^{(1)} \mathbf{W}_{8\times4} \mathbf{Q}_{4\times1}. \tag{2}$$

Proof. To prove the lemma, we evaluate the matrix products in (2) sequentially from right to left. First, we obtain the product

$$\mathbf{W}_{8\times4} \mathbf{Q}_{4\times1} = [q_0, q_1, q_3, q_2, q_0, q_2, q_3, q_1]^{\mathrm{T}} \tag{3}$$

Next, the resulting product is multiplied by the matrix $\mathbf{W}_8^{(1)}$ as follows:

$$\mathbf{W}_8^{(1)}\mathbf{W}_{8\times4}\mathbf{Q}_{4\times1}=\begin{bmatrix}1&1&&&&&&\\1&-1&&&&&&\\&&1&1&&&&\\&&1&-1&&&&\\&&&&1&1&&\\&&&&1&-1&&\\&&&&&&1&1\\&&&&&&1&-1\end{bmatrix}\times\begin{bmatrix}q_0\\q_1\\q_3\\q_2\\q_0\\q_2\\q_3\\q_1\end{bmatrix}=\begin{bmatrix}q_0+q_1\\q_0-q_1\\q_3+q_2\\q_3-q_2\\q_0+q_2\\q_0-q_2\\q_3+q_1\\q_3-q_1\end{bmatrix}. \tag{4}$$

Thus, based on the matrices defined in (3) and (4), we obtain the vector $\mathbf{S}_{8\times1}$ as follows:

$$\mathbf{S}_{8\times1}=\mathbf{W}_8^{(0)}\mathbf{W}_8^{(1)}\mathbf{W}_{8\times4}\mathbf{Q}_{4\times1}=$$

$$=\begin{bmatrix}&&1&&&&&\\1&&&&&&&\\&1&&&&&&\\&&1&&&&&\\&&&&&&1&\\&&&&&1&&\\&&&1&&&&\\&&&&1&&&\end{bmatrix}\times\begin{bmatrix}q_0+q_1\\q_0-q_1\\q_3+q_2\\q_3-q_2\\q_0+q_2\\q_0-q_2\\q_3+q_1\\q_3-q_1\end{bmatrix}=\begin{bmatrix}q_3-q_2\\q_0+q_1\\q_0-q_1\\q_3+q_2\\q_3-q_1\\q_3+q_1\\q_0+q_2\\q_0-q_2\end{bmatrix}.$$

This completes the proof of the lemma. ■

Let us introduce the following matrices:

$$\mathbf{D}_9=\mathrm{diag}(1,1,1,1,0.5,1,1,1,1), \tag{5}$$

$$\mathbf{W}_9=\begin{bmatrix}1&&&&&&&&\\&1&&&&&&&\\&&1&&&&&&\\&&&1&&&&&\\&&&&1&&&&1\\&&&&&1&&&\\&&&&&&1&&\\&&&&&&&1&\\&&&&&&&&1\end{bmatrix}, \tag{6}$$

$$\mathbf{W}_{9\times8}=\mathbf{I}_5\oplus\begin{bmatrix}1&&\\&1&\\&&1\\1&1&1\end{bmatrix}, \tag{7}$$

$$\mathbf{W}_{4\times9}=\begin{bmatrix}1&&&1&-1&&&&\\&1&&1&&&&&-1\\&&1&1&&&-1&&\\&&&1&1&-1&&&\end{bmatrix},$$

$$\mathbf{P}_8=\begin{bmatrix}&&&1\\1&&&\\&&1&\\&1&&\end{bmatrix}\oplus\begin{bmatrix}&&1&\\&1&&\\1&&&\end{bmatrix},$$

$$\mathbf{P}_{8\times4}=\begin{bmatrix}1&&&\\&1&&\\&&1&\\&&&1\\1&&&\\&&&1\\&1&&\\&&1&\end{bmatrix}.$$

Based on (2), we formulate the following theorem.

Theorem.
The matrix $\mathbf{Q}_4$ can be factorized as follows:

$$\mathbf{Q}_4=\mathbf{W}_{4\times9}\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times8}\mathrm{diag}(\mathbf{S}_{8\times1})\mathbf{P}_8\mathbf{W}_8^{(1)}\mathbf{P}_{8\times4}. \tag{8}$$

Proof. As in the proof of the lemma, we multiply the matrices in the decomposition (8) sequentially from right to left, using the associativity of matrix multiplication at each step. We begin with the rightmost product $\mathbf{W}_8^{(1)}\mathbf{P}_{8\times4}$, in which $\mathbf{P}_{8\times4}$ rearranges the input components and $\mathbf{W}_8^{(1)}$ forms the required pairwise combinations. After reordering, summing, and collecting like terms, we obtain the following intermediate representation:

$$\mathbf{W}_8^{(1)}\mathbf{P}_{8\times4}=\begin{bmatrix}1&1&&&&&&\\1&-1&&&&&&\\&&1&1&&&&\\&&1&-1&&&&\\&&&&1&1&&\\&&&&1&-1&&\\&&&&&&1&1\\&&&&&&1&-1\end{bmatrix}\times\begin{bmatrix}1&&&\\&1&&\\&&&1\\&&&1\\1&&&\\&&&1\\&1&&\\&&1&\end{bmatrix}.$$

Next, we multiply the resulting matrix by the permutation matrix, which rearranges its rows according to the required computational structure and transforms the intermediate terms into the final arrangement required by the quaternion multiplication algorithm:

$$\mathbf{P}_8\mathbf{W}_8^{(1)}\mathbf{P}_{8\times 4} = \begin{bmatrix} & & 1 & & & & & \\ 1 & & & & & & & \\ & & & 1 & & & & \\ & 1 & & & & & & \\ & & & & & & & 1 \\ & & & & & & 1 & \\ & & & & & 1 & & \\ & & & & 1 & & & \end{bmatrix} \times \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \\ 1 & & & 1 \\ 1 & & & -1 \\ & 1 & 1 & \\ & 1 & -1 & \end{bmatrix} = \begin{bmatrix} & & 1 & -1 \\ 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & 1 & -1 & \\ & 1 & 1 & \\ 1 & & & -1 \\ 1 & & & 1 \end{bmatrix}. \tag{9}$$

Next, using (5)–(7), we extract the matrices $\mathbf{W}_9$, $\mathbf{W}_{9\times 8}$ and the sparse matrix $\mathbf{D}_9$ and compute their product:

$$\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times 8} = \operatorname{diag}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0.5 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & 1 \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{bmatrix} \times \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & 1 & 1 \end{bmatrix}$$

$$= \mathbf{I}_4 \oplus \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ & 1 & 1 & 1 \end{bmatrix}$$

The resulting matrix is then multiplied by the matrix $\mathbf{W}_{4\times 9}$:

$$\mathbf{W}_{4\times 9}\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times 8} = \begin{bmatrix} 1 & & & 1 & -1 & & & & -1 \\ & 1 & & 1 & & & & & \\ & & 1 & 1 & & & -1 & & \\ & & & 1 & 1 & -1 & & & \end{bmatrix} \times \mathbf{I}_4 \oplus \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & & 0.5 & -0.5 & 0.5 & 0.5 \\ & 1 & & 0.5 & -0.5 & -0.5 & -0.5 \\ & & 1 & 0.5 & 0.5 & 0.5 & -0.5 \\ & & & 1 & 0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix}.$$

We construct the diagonal matrix from the vector $\mathbf{S}_{8\times 1}$ defined in the lemma. Finally, we multiply the resulting matrix $\mathbf{W}_{4\times 9}\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times 8}$ by this diagonal matrix $\operatorname{diag}\mathbf{S}_{8\times 1}$ and by the matrix $\mathbf{P}_8\mathbf{W}_8^{(1)}\mathbf{P}_{8\times 4}$ obtained in (9):

$$\mathbf{W}_{4\times 9}\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times 8}\mathrm{diag}(\mathbf{S}_{8\times 1})\mathbf{P}_8\mathbf{W}_8^{(1)}\mathbf{P}_{8\times 4} =$$

$$= \begin{bmatrix} 1 & & & 0.5 & -0.5 & 0.5 & 0.5 \\ & 1 & & 0.5 & -0.5 & -0.5 & -0.5 \\ & & 1 & 0.5 & 0.5 & 0.5 & -0.5 \\ & & & 1 & 0.5 & 0.5 & -0.5 & 0.5 \end{bmatrix} \mathrm{diag} \begin{bmatrix} q_3 - q_2 \\ q_0 + q_1 \\ q_0 - q_1 \\ q_3 + q_2 \\ q_3 - q_1 \\ q_3 + q_1 \\ q_0 + q_2 \\ q_0 - q_2 \end{bmatrix} \begin{bmatrix} & & 1 & -1 \\ 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & 1 & -1 & \\ & 1 & 1 & \\ 1 & & & -1 \\ 1 & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} q_3 - q_2 & & & & 0.5(q_3 - q_1) & -0.5(q_3 + q_1) & 0.5(q_0 + q_2) & 0.5(q_0 - q_2) \\ & q_0 + q_1 & & & 0.5(q_3 - q_1) & -0.5(q_3 + q_1) & -0.5(q_0 + q_2) & -0.5(q_0 - q_2) \\ & & q_0 - q_1 & & 0.5(q_3 - q_1) & 0.5(q_3 + q_1) & 0.5(q_0 + q_2) & -0.5(q_0 - q_2) \\ & & & q_3 + q_2 & 0.5(q_3 - q_1) & 0.5(q_3 + q_1) & -0.5(q_0 + q_2) & 0.5(q_0 - q_2) \end{bmatrix} \times \begin{bmatrix} & & 1 & -1 \\ 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & 1 & -1 & \\ & 1 & 1 & \\ 1 & & & -1 \\ 1 & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} = \mathbf{Q}_4.$$

This completes the proof of the theorem. ■

Based on the matrix factorization (8), we obtain the following algorithm for multiplying two quaternions:

$$\mathbf{Y}_{4\times 1} = \mathbf{W}_{4\times 9}\mathbf{D}_9\mathbf{W}_9\mathbf{W}_{9\times 8}\mathrm{diag}(\mathbf{S}_{8\times 1})\mathbf{P}_8\mathbf{W}_8^{(1)}\mathbf{P}_{8\times 4}\mathbf{X}_{4\times 1}. \qquad (10)$$

Figure 1a shows the data-flow graph of the proposed quaternion multiplication algorithm. The algorithm takes as input the vector $\mathbf{X}_{4\times 1}$, which contains the components of one quaternion. The scaling factors are $s_0 = q_3 - q_2$, $s_1 = q_0 + q_1$, $s_2 = q_0 - q_1$, $s_3 = q_3 + q_2$, $s_4 = q_3 - q_1$, $s_5 = q_3 + q_1$, $s_6 = q_0 + q_2$, and $s_7 = q_0 - q_2$. The outputs are the components of the vector $\mathbf{Y}_{4\times 1}$, representing the product of two quaternions expressed by the vectors $\mathbf{Q}_{4\times 1}$ and $\mathbf{X}_{4\times 1}$. Therefore, the data-flow graph in Figure 1a implements quaternion multiplication with constant coefficients.

Figure 1b presents the auxiliary data-flow graph that implements expression (2). The input to this algorithm is the vector $\mathbf{Q}_{4\times 1}$, which contains the components of the second quaternion. The outputs are the scaling factors s_0, s_1, s_2, s_3, s_4, s_5, s_6, and s_7 used in the matrix factorization given by (8). Thus the proposed quaternion multiplication algorithm with variable coefficients is implemented by combining the data-flow graphs shown in Figures 1a and 1b.

To justify the derivation of the matrices used in the factorization (8), we first note that two fast quaternion multiplication algorithms were proposed in [22]. Both algorithms require eight real multiplications and 40 and 32 real additions, respectively. However, these algorithms are presented only in the form of mathematical expressions, which is insufficient for efficient computational organization and hardware-oriented implementation.

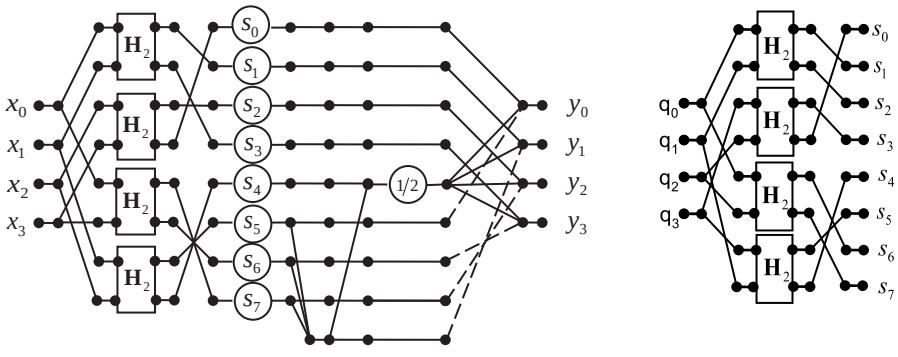

Fig. 1. Data-flow graph of the basic quaternion multiplication algorithm (a) and auxiliary data-flow graph for computing the algorithm-forming vector (b)
Rys. 1. Wykres przepływu danych podstawowego algorytmu mnożenia kwaternionów (a) oraz pomocniczy wykres przepływu danych realizujący obliczanie wektora współczynników mnożenia (b)

Therefore, the algorithms were first represented as data-flow graphs similar to those shown in Fig. 1. For each subgraph at every hierarchical level of the data-flow graph, the corresponding adjacency matrix was constructed. This procedure is described in detail in [25]. The resulting adjacency matrices were then transformed to reduce the number of additions. The final optimized matrices obtained through this process are given in (5)–(7).

4. Computational complexity of the proposed algorithms

The correctness of the proposed algorithms was verified numerically. For this purpose, 50 four-dimensional vectors were generated, with each component drawn from a normal distribution with zero mean and unit variance. The vectors were represented as quaternions. Quaternion products were computed using direct matrix-vector multiplication and, independently, by multiplying the matrices in the factorizations (2) and (8). Agreement between the results confirmed the correctness of the proposed algorithm.

The correctness of the proposed data-flow graphs was also verified using the same set of randomly generated four-dimensional vectors. Quaternion multiplication was first performed using direct matrix-vector products and then repeated using the proposed algorithm. Agreement between the results confirmed the correctness of the data-flow graphs.

We next compare the numbers of arithmetic operations required by the existing and proposed quaternion multiplication algorithms. Table 2 lists the numbers of multiplications, additions, and shifts required by the algorithms with constant and variable coefficients. As shown, the proposed algorithms reduce the number of multiplications by 33–50 %, while the corresponding change in the number of additions depends on the reference algorithm and may exceed a factor of two.

Although [22] presents formulas for quaternion multiplication, these formulas constitute a set of arithmetic expressions rather than a complete computational algorithm. Specifically, they do not define the organization of the computations, the input and output data formats, or the sequence of operations. Their derivation is also not discussed.

In contrast, the proposed method is based on the established matrix-vector representation of quaternion multiplication. By exploiting a novel factorization of the transformation matrix, the algorithm halves the number of multiplications compared with the conventional approach. This factorization yields a complete matrix-vector computational procedure that explicitly defines the data-flow and execution order. The corresponding data-flow graphs illustrate the space-time organization of the computations and provide a direct basis for efficient hardware implementation.

The proposed algorithm offers advantages similar to those of the algorithm presented in [22], but differs in its computational implementation and space-time structure. Furthermore, unlike the algorithm in [22], the proposed algorithm does not require three fast Walsh-Hadamard transforms and is therefore simpler. The two algorithms differ in the order of operations and in their intermediate results. However, the proposed algorithm was derived independently using a fundamentally different approach based on matrix factorization.

The pseudocode for the proposed quaternion multiplication algorithm is presented in Algorithm 1. The inputs to the variable-coefficient algorithm are the quaternion component vectors $Q=[q_0, q_1, q_2, q_3]^T$ and $X=[x_0, x_1, x_2, x_3]^T$. The quantities $s_0,\ldots,s_7$ are the algorithm-forming coefficients, whereas $p_0,\ldots,p_7$ denote the intermediate products. The outputs of the algorithm are y_0, y_1, y_2, and y_3.

Algorithm 1. Pseudocode of the proposed fast quaternion multiplication algorithm
Algorytm 1. Pseudokod proponowanego szybkiego algorytmu mnożenia kwaternionów

Input:
$\mathbf{Q} = [q_0, q_1, q_2, q_3]^T$, $\mathbf{X} = [x_0, x_1, x_2, x_3]^T$
Output:
$\mathbf{Y} = [y_0, y_1, y_2, y_3]^T = \mathbf{QX}$
Step 1. Compute the algorithm-forming coefficients
$s_0 \leftarrow q_3 - q_2$, $s_1 \leftarrow q_0 + q_1$,
$s_2 \leftarrow q_0 - q_1$, $s_3 \leftarrow q_3 + q_2$,
$s_4 \leftarrow q_3 - q_1$, $s_5 \leftarrow q_3 + q_1$,
$s_6 \leftarrow q_0 + q_2$, $s_7 \leftarrow q_0 - q_2$.
Step 2. Compute the intermediate products
$p_0 \leftarrow (x_0 + x_1)s_1$, $p_1 \leftarrow (x_0 - x_1)s_3$,
$p_2 \leftarrow (x_2 + x_3)s_2$, $p_3 \leftarrow (x_2 - x_3)s_0$,
$p_4 \leftarrow (x_0 + x_3)s_7$, $p_5 \leftarrow (x_0 - x_3)s_6$,
$p_6 \leftarrow (x_1 + x_2)s_5$, $p_7 \leftarrow (x_1 - x_2)s_4$.
Step 3. Compute the common intermediate terms
$r \leftarrow p_4 + p_5 + p_6$,
$h \leftarrow (p_7 + r)/2$.
Step 4. Compute the output components
$y_0 \leftarrow p_3 + h - p_6$,
$y_1 \leftarrow p_0 + h - r$,
$y_2 \leftarrow p_2 + h - p_4$,
$y_3 \leftarrow p_1 + h - p_5$.
Return $\mathbf{Y} = [y_0, y_1, y_2, y_3]^T$.

Table 2. Numbers of operations required by the quaternion multiplication algorithms
Tabela 2. Liczba operacji w algorytmach mnożenia kwaternionów

Algorithm	Reference, year of publication	Mults.	Adds.	Shifts
Matrix-vector product	–	16	12	–
Lifting-based	[21], 2011	12	12	–
4D CORDIC	[20], 2016	12	8	–
Howell, Lafon	[22], 1975	8	32	4
Structural-approach-based algorithm with constant coefficients	[23], 2010 [24], 2024	8	20	4
Structural-approach-based algorithm with variable coefficients	[23], 2010 [24], 2024	8	28	8
Proposed algorithm with constant coefficients	–	8	19	1
Proposed algorithm with variable coefficients	–	8	27	1

5. Conclusions

This paper has presented novel quaternion multiplication algorithms that significantly reduce the required number of real multiplications. Using a structural approach, we derived sparse matrix factorizations and represented the algorithms with constant and variable coefficients as data-flow graphs. Rigorous mathematical proofs and numerical verification using randomly generated quaternion inputs confirm the correctness of the proposed algorithms.

Compared with conventional direct quaternion multiplication and recently developed techniques, such as lifting-based and four-dimensional CORDIC algorithms, the proposed solutions reduce the number of multiplications by up to 50 %. Although this reduction comes at the cost of additional additions, the resulting trade-off may be advantageous in hardware implementations in which multipliers are typically more resource-intensive than adders. Future research will focus on the implementation and evaluation of the proposed algorithms on hardware platforms.

References

1. Barry N., *The application of quaternions in electrical circuits*, 27th Irish Signals and Systems Conference (ISSC), Londonderry, UK, 2016, DOI: 10.1109/ISSC.2016.7528440.
2. Brasil V.P., Filho A.L.F., Ishihara J.Y., *Electrical three phase circuit analysis using quaternions*, 18th International Conference on Harmonics and Quality of Power (ICHQP), Ljubljana, Slovenia, 2018, DOI: 10.1109/ICHQP.2018.8378813.
3. Danielewski M., Sapa L., *Foundations of the Quaternion Quantum Mechanics*, "Entropy", Vol. 22, No. 12, 2020, DOI: 10.3390/e22121424.
4. Ahmed A., Ju H., Yang Y., Xu H., *An improved unit quaternion for attitude alignment and inverse kinematic solution of the robot arm wrist*, "Machines", Vol. 11, No. 7, 2023, DOI: 10.3390/machines11070669.
5. Bayro-Corrochano E., *A survey on quaternion algebra and geometric algebra applications in engineering and computer science 1995–2020*, "IEEE Access", Vol. 9, 2021, 104326–104355, DOI: 10.1109/ACCESS.2021.3097756.
6. Huang C., Li J., Gao G., *Review of Quaternion-Based Color Image Processing Methods*, "Mathematics", Vol. 11, No. 9, 2023, DOI: 10.3390/math11092056.
7. 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, 2023, 26–40, DOI: 10.1109/MSP.2023.3278071.
8. Altamirano-Gomez G., Gershenson C., *Quaternion Convolutional Neural Networks: Current Advances and Future Directions*, "Advances in Applied Clifford Algebras", Vol. 34, 2024, DOI: 10.1007/s00006-024-01350-x.
9. Chen J., Ng M.K., *Color Image Inpainting via Robust Pure Quaternion Matrix Completion: Error Bound and Weighted Loss*, "SIAM Journal on Imaging Sciences", Vol. 15, No. 3, 2022, 1469–1498, DOI: 10.1137/22M1476897.
10. BenYoussef N., Bouzid A., *Color Edge Detection using Quaternion Convolution and Vector Gradient*, VISIGRAPP 2017, Porto, 135–139, DOI: 10.5220/0006229601350139.
11. Shi L., Funt B., *Quaternion color texture segmentation*, "Computer Vision and Image Understanding", Vol. 107, No. 1, 2007, 88–96, DOI: 10.1016/j.cviu.2006.11.014.
12. Zeng R., Wu J., Shao Z., Chen Y., Chen B., Senhadji L., Shu H., *Color image classification via quaternion principal component analysis network*, "Neurocomputing", Vol. 216, 2016, 416–428, DOI: 10.1016/j.neucom.2016.08.006.
13. Zhang H., Qi T., Zeng T., *Quaternion-Aware Low-Rank Prior for Blind Color Image Deblurring*, "Journal of Scientific Computing", Vol. 101, 2024, DOI: 10.1007/s10915-024-02671-6.
14. Voronin V., Semenishchev E., Zelensky A., Agaian S., *Quaternion-based local and global color image enhancement algorithm*, Proceedings of SPIE Defense + Security, Vol. 10993, 2019, 13–20, DOI: 10.1117/12.2519574.
15. Wu T., Huang C., Jin Z., Jia Z., Ng M.K., *Total Variation Based Pure Quaternion Dictionary Learning Method for Color Image Denoising*, "International Journal of Numerical Analysis and Modelling", Vol. 19, No. 5, 2022, 709–737.
16. Komeno A.S.F., Filho A.L.F., Ishihara J.Y., Brasil V.P., *Three-phase Quaternion Power in Three-wire Systems*, International Conference on Renewable Energies and Power Quality (ICREPQ '21), Almeria, 2021, Vol. 19, 493–498, DOI: 10.24084/repqj19.327.
17. Taylor M.G., *Application of quaternions to obtain analytic solutions to systems of polarization components*, "Optics Communications", Vol. 607, 2026, DOI: 10.1016/j.optcom.2026.132904.
18. Biswas D., Ye Z., Mazomenos E.B., Jöbges M., Maharatna K., *CORDIC Framework for Quaternion-based Joint Angle Computation to Classify Arm Movements*, IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018, DOI: 10.1109/ISCAS.2018.8350967.
19. Parfieniuk M., Park S.Y., *Versatile quaternion multipliers based on distributed arithmetic*, "Circuits Systems and Signal Processing", Vol. 37, 2018, 4880–4906, DOI: 10.1007/s00034-018-0789-5.
20. Parfieniuk M., Park S.Y., *Sparse-iteration 4D CORDIC algorithms for multiplying quaternions*, "IEEE Transactions on Computers", Vol. 65, No. 9, 2016, 2859–2871, DOI: 10.1109/TC.2015.2506572.
21. Parfieniuk M., Petrovsky N.A., Petrovsky A.A., *Rapid Prototyping of Quaternion Multiplier: From Matrix Notation to FPGA-Based Circuits*, [In:] Rapid Prototyping Technology – Principles and Functional Requirements, Rijeka, 2011, 227–246, DOI: 10.5772/20939.
22. Howell T.D., Lafon J.-C., *The complexity of the quaternion product*, Technical Report TR 75-245, Department of Computer Science, Cornell University, Ithaca, NY, June 1975, [https://hdl.handle.net/1813/6458].
23. Ţariova G., Ţariov A., *Aspekty algorytmiczne redukcji liczby bloków mnożących w układzie do obliczania iloczynu dwóch kwaternionów*, „Pomiary Automatyka Kontrola", Nr 7, 2010, 668–690.
24. Cariow A., Naumowicz M., Handkiewicz A., *Structure and Principles of Operation of a Quaternion VLSI Multiplier*, "Applied Sciences", Vol. 14, No. 18, 2024, DOI: 10.3390/app14188123.
25. Kitsela V., Polyakova M., Cariow A., *Fast algorithms for short-length type VI discrete cosine transform*, "Electronics", Vol. 15, No. 3, 2026, DOI: 10.3390/electronics15030699.

Szybki algorytm mnożenia kwaternionów

Streszczenie: Mnożenie kwaternionów znajduje zastosowanie w wielu dziedzinach, takich jak elektrotechnika, robotyka, grafika komputerowa, systemy nawigacyjne oraz przetwarzanie obrazów i sygnałów. W obszarach tych istotne znaczenie ma ograniczenie złożoności obliczeniowej algorytmów, w szczególności redukcja liczby mnożeń liczb rzeczywistych podczas obliczania iloczynów kwaternionów. W artykule zaproponowano algorytmy umożliwiające zmniejszenie liczby operacji mnożenia rzeczywistego w procesie obliczania iloczynów kwaternionów w porównaniu z klasyczną metodą bezpośrednią, przy zachowaniu poprawności i efektywności obliczeń.

Słowa kluczowe: mnożenie kwaternionów, szybkie algorytmy, złożoność obliczeniowa, redukcja liczby mnożeń liczb rzeczywistych, przetwarzanie sygnałów, macierze Hadamarda, trójfazowy sygnał elektryczny

Janusz P. Papliński, PhD Eng.
janusz.paplinski@zut.edu.pl
ORCID: 0000-0002-6100-3913

He received his master's degree in technical computer science and telecommunications in 1990 and earned his doctoral degree in 1996. In 2016, he joined the Department of Computer Science and Information Technology at the West Pomeranian University of Technology in Szczecin, where he now works in the Department of Computer Architectures and Telecommunications. His research interests include optimising digital signal processing and image processing algorithms, and implementing algorithms into programmable circuits.

Prof. Aleksandr Cariow, DSc PhD Eng.
atariov@wi.zut.edu.pl
ORCID: 0000-0002-4513-4593

He received his MSc, PhD and DSc degrees in computer science in 1976, 1984 and 2001, respectively. In 1999, he joined the Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, Poland, where he currently works in the Department of Computer Architectures and Telecommunications. His research interests include digital signal and image processing algorithms, VLSI architectures, and data processing parallelization.

Prof. Marina Polyakova, DSc PhD Eng.
polyakova@op.edu.ua
ORCID: 0000-0001-7229-7657

She received her MSc, PhD and DSc degrees in computer science in 1994, 2004 and 2014, respectively. In 2015, she joined the Institute of Computer Systems of Odesa Polytechnic National University, where she is currently a professor at the Department of Applied Mathematics and Information Technologies. Her research interests include digital signal processing algorithms, image processing, neural networks, and time series analysis.