Numerical Algorithms (2022) 91:1461–1481
https://doi.org/10.1007/s11075-022-01310-1

ORIGINAL PAPER

A complex structure-preserving algorithm for the full rank decomposition of quaternion matrices and its applications

Gang Wang[1,2] · **Dong Zhang**[1,2] · **Vasily. I. Vasiliev**[1] · **Tongsong Jiang**[1,2,3]

Received: 27 October 2021 / Accepted: 28 March 2022 / Published online: 19 May 2022

Abstract

In this paper, based on the Gauss transformation of a quaternion matrix, we study the full rank decomposition of a quaternion matrix, and obtain a direct algorithm and complex structure-preserving algorithm for full rank decomposition of a quaternion matrix. In addition, we expand the application of the above two full rank decomposition algorithms and give a fast algorithm to calculate the quaternion linear equations. The numerical examples show that the complex structure-preserving algorithm is more efficient. Finally, we apply the structure-preserving algorithm of the full rank decomposition to the sparse representation classification of color images, and the classification effect is well.

Keywords Quaternion matrix · Complex representation · Complex structure-preserving algorithm · Full rank decomposition · Sparse representation classification

Mathematics Subject Classification (2010) 15A33

✉ Tongsong Jiang
jiangtongsong@sina.com

✉ Vasily. I. Vasiliev
vasvasil@mail.ru

[1] Institute of Mathematics and Information Science, North-Eastern Federal University, Yakutsk, 677000, Russia

[2] School of Mathematics and Statistics, Heze University, Heze Shandong, 274015, People's Republic of China

[3] School of Mathematics and Statistics, Linyi University, Linyi Shandong 276005, People's Republic of China

1 Introduction

A quaternion, which was found in 1840 by William Rowan Hamilton, is in the form of $q = q_1 + q_2\mathrm{i} + q_3\mathrm{j} + q_4\mathrm{k}$, $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = -1$, $\mathrm{ijk} = -1$, where $q_1, q_2, q_3, q_4 \in \mathbf{R}$, and $\mathrm{ij} = -\mathrm{ji} = \mathrm{k}, \mathrm{jk} = -\mathrm{kj} = \mathrm{i}, \mathrm{ki} = -\mathrm{ik} = \mathrm{j}$. With the further development of quaternion algebras, the quaternion matrices have been widely used in computer science, quantum mechanics, signal and color image processing, control theory, geometric rotation and other fields [1–8]. Especially in recent years, after Professor Wei Musheng and others proposed the real structure-preserving algorithm of a quaternion matrix, they greatly promoted the research and application of the algorithm of a quaternion matrix. In papers [9–18], by using the real structure-preserving algorithm of quaternion matrices, the authors studied the LU decomposition problem of quaternion matrices, the singular value decomposition problem of quaternion matrices, eigenvalue problem of Hermitian quaternion matrices and the power method of eigenvalues of quaternion matrices and so on.

As a fundamental property of matrix algebras, the full rank decomposition is always a hot topic. In the field of face recognition, full rank decomposition can avoid iterative steps and solve the sparse system directly, save a lot of time and improve the classification accuracy [20–23]. In paper [19], the authors proposed a fast and robust face recognition method named enhancing sparsity via full rank decomposition. The proposed method first represented the test sample as a linear combination of the training data as the same as sparse representation, then made a full rank decomposition of the training data matrix. Finally, obtained the generalized inverse of the training data matrix and solved the general solution of the linear equation. In the field of system control, the full rank system compensator extended the internal model control to the structural rank deficient system, which effectively avoided the characteristic that the model control method is only suitable for the square systems. In paper [24], the authors proposed an internal model control method for structured rank deficient systems based on the full rank decomposition. The system converted into a column full rank system by designing a pre-compensator. Then a feedback-compensator is designed to improve the dynamic characteristics of the full rank system and decrease the controller design difficulties.

Based on the extensive application of quaternion algebra and the full rank decomposition in the above fields, and the structure-preserving algorithm greatly improves the timeliness of a quaternion matrix decomposition, and its results are better than quaternion toolbox (QTFM) to a certain extent. We find that the complex preserving structure algorithm is also efficient for the full rank decomposition of a quaternion matrix. The theoretical basis of complex structure-preserving algorithm is the same as real structure-preserving algorithm, but its algorithm writing form is relatively simple, the dimensions of the matrix are reduced by half, and scholars have not studied and given the full rank decomposition algorithm of quaternion matrices. This paper, by means of the Gauss elimination method of quaternion matrices, derives the direct algorithm and complex structure-preserving algorithm for the full

rank decomposition of a quaternion matrix, proves the full rank decomposition form of the generalized inverse matrix of a quaternion matrix. In addition, by using the above two full rank decomposition algorithms, we also give a fast algorithm for solving the quaternion linear equations. The feasibility and timeliness of the algorithm are proved by numerical examples. Finally, this algorithm can also be applied to the sparse representation classification of color images, and the classification effect is well.

Let $\mathbf{R}$ denote the real number field, $\mathbf{C}$ the complex number field, $\mathbf{Q} = \mathbf{R} \oplus \mathbf{R}\mathrm{i} \oplus \mathbf{R}\mathrm{j} \oplus \mathbf{R}\mathrm{k}$ the quaternion ring, in which $\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1, \mathrm{ij} = -\mathrm{ji} = \mathrm{k}, \mathrm{jk} = \text{-kj} = \mathrm{i}, \mathrm{ki} = \text{-ik=j}$. For any matrix $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} \in \mathbf{Q}^{m\times n}$, $\overline{A} = A_1 - A_2\mathrm{i} - A_3\mathrm{j} - A_4\mathrm{k}$, $A^T = A_1^T + A_2^T\mathrm{i} + A_3^T\mathrm{j} + A_4^T\mathrm{k}$, $A^* = A_1^T - A_2^T\mathrm{i} - A_3^T\mathrm{j} - A_4^T\mathrm{k}$, denote the conjugate, the transpose, the conjugate transpose of the matrix A, respectively. $\mathbf{F}^{m\times n}$ denotes the set of $m \times n$ matrices over the ring of $\mathbf{F}$. For any quaternion $q = q_1 + q_2\mathrm{i} + q_3\mathrm{j} + q_4\mathrm{k}$, $\overline{q} = q_1 - q_2\mathrm{i} - q_3\mathrm{j} - q_4\mathrm{k}$ is the conjugate of q, the norm of the quaternion q is defined to be $\|q\| = \sqrt{|q\overline{q}|} = \sqrt{|q_1^2 + q_2^2 + q_3^2 + q_4^2|}$.

2 Preliminaries

In this section, we first review some basic properties about quaternion matrices and their complex representation matrices, and then prove the full rank decomposition form of the generalized inverse matrix of a quaternion matrix.

For $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} = M_1 + M_2\mathrm{j} \in \mathbf{Q}^{m\times n}$, $A_1, A_2, A_3, A_4 \in \mathbf{R}^{m\times n}$, a complex representation matrix A^σ of A is defined to be

$$A^\sigma = \begin{bmatrix} M_1 & M_2 \\ -\overline{M}_2 & \overline{M}_1 \end{bmatrix} \in \mathbf{C}^{2m\times 2n}, \tag{2.1}$$

where $M_1 = A_1 + A_2\mathrm{i}$, $M_2 = A_3 + A_4\mathrm{i} \in \mathbf{C}^{m\times n}$. By (2.1), we have

$$A^\sigma = \begin{bmatrix} A_c^\sigma & Q_m^T\overline{A_c^\sigma} \end{bmatrix} = \begin{bmatrix} A_r^\sigma \\ \overline{A_r^\sigma}Q_n \end{bmatrix}, \qquad Q_m^T A^\sigma Q_n = \overline{A^\sigma}, \tag{2.2}$$

where $A_r^\sigma = \begin{bmatrix} M_1 & M_2 \end{bmatrix}$, $A_c^\sigma = \begin{bmatrix} M_1 \\ -\overline{M}_2 \end{bmatrix}$, $Q_t = \begin{bmatrix} 0 & I_t \\ -I_t & 0 \end{bmatrix}$. By (2.1) and (2.2), it is easy to prove the following results.

Proposition 2.1 [18] *Let $A, B \in \mathbf{Q}^{m\times n}$, $C \in \mathbf{Q}^{n\times p}$, $a \in \mathbf{R}$. Then*

$$(A + B)^\sigma = A^\sigma + B^\sigma, (AC)^\sigma = A^\sigma C^\sigma, (aA)^\sigma = aA^\sigma, \tag{2.3}$$

$$(A + B)_r^\sigma = A_r^\sigma + B_r^\sigma, (AC)_r^\sigma = A_r^\sigma C^\sigma, (aA)_r^\sigma = aA_r^\sigma. \tag{2.4}$$

For any matrix $A \in \mathbf{Q}^{n\times n}$, by the Frobenius norm and 2 norm of a complex matrix, define the following Frobenius norm and 2 norm of a quaternion matrix

$$\|A\|_F = \frac{1}{\sqrt{2}} \left\|A^\sigma\right\|_F, \quad \|A\|_2 = \left\|A^\sigma\right\|_2, \tag{2.5}$$

then the Frobenius norm and 2 norm are unitarily invariant norms of a quaternion matrix. From the above discussion, it is easy to prove the following results.

Proposition 2.2 *Let $A \in \mathbf{Q}^{m\times n}, b \in \mathbf{Q}^m$. Then*

$$\|A\|_F = \left\|A_r^\sigma\right\|_F, \ \|b\|_2 = \left\|b_c^\sigma\right\|_2. \tag{2.6}$$

Proposition 2.3 *For any quaternion matrices A, B, we have*

(1) $\text{rank}(A) = \frac{1}{2}\text{rank}(A^\sigma)$;
(2) $\text{rank}(AB) \leqslant \min\{\text{rank}(A), \text{rank}(B)\}$, *in which* $A \in \boldsymbol{Q}^{m\times n}, B \in \mathbf{Q}^{n\times p}$;
(3) $\text{rank}(A^*) = \text{rank}(\overline{A}) = \text{rank}(A^T)$, $\text{rank}(A^*A) = \text{rank}(A)$, *in which* $A \in \mathbf{Q}^{m\times n}$;
(4) $\text{rank}(A) = \text{rank}(B)$ *if and only if* $\text{rank}(A^\sigma) = \text{rank}(B^\sigma)$, *in which* $A, B \in \mathbf{Q}^{m\times n}$.

From [18]*, supposed that $A \in \boldsymbol{Q}^{m\times n}$, if there exists $X \in \mathbf{Q}^{n\times m}$ such that*

$$(1)\ (AX)^* = AX; \quad (2)\ (XA)^* = XA; \quad (3)\ AXA = A; \quad (4)\ XAX = X, \tag{2.7}$$

then X is called the generalized inverse matrix of matrix A, denoted by $A^\dagger$, and the matrix $A^\dagger$ satisfies the following properties,

(1) $(A^*)^\dagger = (A^\dagger)^*$;
(2) *If* $\text{rank}(A) = m$, *then* $A^\dagger = A^*(AA^*)^{-1}$;
(3) *If* $\text{rank}(A) = n$, *then* $A^\dagger = (A^*A)^{-1}A^*$.

Proposition 2.4 [18] *Supposed that $A \in \mathbf{Q}^{m\times n}, b \in \mathbf{Q}^n$. Then when the quaternion linear equation $Ax = b$ is regarded as the least squares problem, its solution sets and the compatible linear system $Ax = AA^\dagger b$ are the same. A general solution of the quaternion least squares problem have the following expression $x = A^\dagger b + (I - A^\dagger A)y$, in which $y \in \mathbf{Q}^n$ is any quaternion vector.*

Theorem 2.5 *Suppose that matrix A has the full rank decomposition $A = BC$, in which $B \in \mathbf{Q}^{m\times r}$, $C \in \mathbf{Q}^{r\times n}$, $\text{rank}(B) = \text{rank}(C) = r > 0$. Then $A^\dagger = C^\dagger B^\dagger = C^*(CC^*)^{-1}(B^*B)^{-1}B^*$ is a generalized inverse matrix of matrix A.*

Proof Since B is a column full rank matrix, C is a row full rank matrix, it is easy to get $B^{\dagger} = (B^*B)^{-1}B^*$, $C^{\dagger} = C^*(CC^*)^{-1}$. Let $X = C^{\dagger}B^{\dagger}$, easy to verify that matrix X satisfies (2.7), i.e., X is the generalized inverse matrix of A, and $A^{\dagger} = C^{\dagger}B^{\dagger} = C^*(CC^*)^{-1}(B^*B)^{-1}B^*$. □

3 Gaussian elimination algorithm for full rank decomposition of a quaternion matrix

Gauss elimination method is widely used in matrix decomposition, for example, full rank decomposition, LU decomposition, LDL^* decomposition, and Cholesky decomposition. In this section, we derive the full rank decomposition process of a quaternion matrix by the Gauss elimination method, and give a direct algorithm of the full rank decomposition.

Given a quaternion vector $x = (x_1, x_2, \cdots, x_n)^T \in \mathbf{Q}^n$, if $x_i \neq 0$, then

$$l_i = \left(0, \cdots, 0, l_{i+1,i}, l_{i+2,i}, \cdots l_{n,i}\right)^T, \tag{3.1}$$

where $l_{k,i} = x_k x_i^{-1} = \frac{x_k \overline{x}_i}{\|x_i\|^2}$, $k = i+1, \cdots, n$. Define the following quaternion Gauss transformation matrix,

$$L_i = I_n - l_i e_i^T = \begin{bmatrix} 1 \cdots 0 & 0 \cdots 0 \\ \vdots \ddots \vdots & \vdots \ddots \vdots \\ 0 \cdots 1 & 0 \cdots 0 \\ 0 \cdots -l_{i+1,i} & 1 \cdots 0 \\ \vdots \ddots \vdots & \vdots \ddots \vdots \\ 0 \cdots -l_{n,i} & 0 \cdots 1 \end{bmatrix}, \tag{3.2}$$

clearly, $L_i x = (x_1, \cdots, x_i, 0, \cdots, 0)^T$. L_i, l_i, $l_{k,i}$ denote the quaternion Gauss transformation matrix, quaternion Gauss transformation vector, quaternion multiplier, respectively.

For any matrix $A = (a_{ij}) \in \mathbf{Q}^{m\times n}$, the process of full rank decomposition of matrix A is as follows.

(1) If $a_{11} \neq 0$, construct a quaternion Gauss transformation matrix L_1 such that

$$L_1 A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{m2}^{(2)} & \cdots & a_{mn}^{(2)} \end{bmatrix} = A^{(2)}, \text{ where } L_1 = \begin{bmatrix} 1 & & & \\ -l_{21} & 1 & & \\ -l_{31} & & 1 & \\ \vdots & & & \ddots \\ -l_{m1} & & & 1 \end{bmatrix}. \tag{3.3}$$

(2) If $a_{11} = 0$. Suppose that there exists $a_{i1} \neq 0$. Interchange rows 1 and i of matrix A, i.e., there exists a permutation matrix P_1 such that

$$L_1 P_1 A = A^{(2)}. \tag{3.4}$$

(3) Similarly, the result after step $k-1$, we have the following results,

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots & \vdots & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \vdots & \vdots & \vdots \\ & & & a_{mk}^{(k)} & \cdots & a_{mn}^{(k)} \end{bmatrix}. \tag{3.5}$$

In the above process, if $a_{kk}^{(k)} = a_{kk+1}^{(k)} = \cdots = a_{mk}^{(k)} = 0$, then go to the operation on the $k+1$ column element. After step t above, convert the original quaternion matrix to the upper triangular matrix is as follows,

$$L_r P_r \cdots L_1 P_1 A = \begin{bmatrix} a_{11}^{(1)} & \cdots & a_{1r}^{(1)} & \cdots & a_{1n}^{(1)} \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & a_{rr}^{(r)} & \cdots & a_{rn}^{(r)} \\ 0 & & \cdots & & 0 \\ \vdots & & & & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} = \begin{bmatrix} C \\ 0 \end{bmatrix}$$

$$\Rightarrow A = P^{-1} L^{-1} \begin{bmatrix} C \\ 0 \end{bmatrix} = \begin{bmatrix} B & * \end{bmatrix} \begin{bmatrix} C \\ 0 \end{bmatrix} = BC, \tag{3.6}$$

in which $L = L_r \cdots L_1$, $P = Q_r \cdots Q_2 P_1$, $Q_i = (L_{i-1} \cdots L_1)^{-1} P_i (L_{i-1} \cdots L_1)$, $2 \leqslant i \leqslant r$, $L_r P_r \cdots L_1 P_1 A = L_r P_r \cdots L_2 L_1 Q_2 P_1 A = L_r \cdots L_1 Q_r \cdots Q_2 P_1 A = LPA$.

Theorem 3.1 *Let $A \in \mathbf{Q}^{m \times n}$, $rank(A) = r$, then there exist $B \in \mathbf{Q}^{m \times r}$, $C \in \mathbf{Q}^{r \times n}$ and $rank(B) = rank(C) = r$, such that $A = BC$.*

Next, we give a direct algorithm for full rank decomposition of a quaternion matrix. This algorithm is based on quaternion toolbox (QTFM), which directly decomposes the quaternion matrix.

Algorithm 1 (Gaussian elimination algorithm for the full rank decomposition): For any matrix $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} \in \mathbf{Q}^{m\times n}$, input A, output $B \in \mathbf{Q}^{m\times r}$, $C \in \mathbf{Q}^{r\times n}$, i.e., $A = BC$.

```
Function [B, C]=FR(A)
(1) Initialization
    M=A; [m, n]=size(M); p=1:m; j=1;
    W=eye(m); N=eye(m); B=eyeq(m);
(2) Cyclic body of Gauss elimination method
    while j< min {m, n}
    d = conj(M(j, j)) * M(j, j);
    if d>0
       B(j + 1:m,j) = M(j + 1:m,j) * (conj(M(j, j))/d);
       M(j + 1:m,j) = zerosq(m - j,1);
       M(j + 1:m,j + 1:n) = M(j + 1:m,j + 1:n)-B(j + 1:m,j)*M(j,j + 1:n);
    else k=j+1;
      while k<= m
       d = conj(M(j, j)) * M(j, j)
       if d > 0
          M([j,k],:) = M([k,j],:); p([j,k]) = p([k,j]);
          B(j + 1:m,j) = M(j + 1:m,j)*(conj(M(j, j))/d);
          M(j + 1:m,j) = zerosq(m - j,1);
          M(j + 1:m,j + 1:n) = M(j + 1:m,j + 1:n)-B(j + 1:m,j)*M(j,j + 1:n);
          break;
       else k=k+1;
       end end
     end  j=j+1; end
(3) Output results
       for k = 1:m
          N(k,:) = W(p(k),:);
       end
       for j = n: - 1:1
       id = find(M(:, j). * conj(M(:, j)) > 1e − 10, 1, 'last′);
       if id > 0 break;
       end
     end
     B=N*B(:,1:id); C=M(1:id,:);
```

4 Complex structure-preserving algorithm for the full rank decomposition of quaternion matrices

In this section, based on the algorithm of the previous chapter, we use the isomorphic complex representation of a quaternion matrix, transform the corresponding algorithm operation to the complex field, reduce the matrix dimension and the

amount of calculation, and obtain a structure-preserving algorithm of the full rank decomposition.

Let $l_i = l_i^{(1)} + l_i^{(2)}\mathrm{j} \in \mathbf{Q}^n$, where $l_i^{(j)} = \left(0, \cdots, 0, l_{i+1,i}^{(j)}, l_{i+1,i}^{(j)}, \cdots, l_{n,i}^{(j)}\right)^T$, $i = 1, 2, \cdots, n-1$, $j = 1, 2$, by (3.2), the complex representation matrix of Gaussian transformation matrix L_i is as follows:

$$L_i^\sigma = I_n^\sigma - l_i^\sigma \left(e_i^T\right)^\sigma = \left[\begin{array}{ccccc|ccccc} 1 & & & & & 0 & & & & \\ 0 & \ddots & & & & \vdots & \ddots & & & \\ \vdots & & 1 & & & \vdots & & 0 & & \\ \vdots & & -l_{i+1,i}^{(1)} & \ddots & & \vdots & & -l_{i+1,i}^{(2)} & \ddots & \\ \vdots & & \vdots & & \ddots & \vdots & & \vdots & & \ddots \\ 0 & \cdots & -l_{n,i}^{(1)} & \cdots & 1 & 0 & \cdots & -l_{n,i}^{(2)} & \cdots & 0 \\ \hline 0 & & & & & 1 & & & & \\ \vdots & \ddots & & & & 0 & \ddots & & & \\ \vdots & & 0 & & & \vdots & & \overline{l_{i+1,i}^{(1)}} & & \\ \vdots & & \overline{l_{i+1,i}^{(2)}} & \ddots & & \vdots & & \ddots & & \\ \vdots & & \vdots & & \ddots & \vdots & & & \ddots & \\ 0 & \cdots & \overline{l_{n,i}^{(2)}} & \cdots & 0 & 0 & \cdots & \overline{l_{n,i}^{(1)}} & \cdots & 1 \end{array}\right], \tag{4.1}$$

it is easy to get,

$$\left(L_i^\sigma\right)^{-1} = I_n^\sigma + l_i^\sigma \left(e_i^T\right)^\sigma = \left[\begin{array}{ccccc|ccccc} 1 & & & & & 0 & & & & \\ 0 & \ddots & & & & \vdots & \ddots & & & \\ \vdots & & 1 & & & \vdots & & 0 & & \\ \vdots & & l_{i+1,i}^{(1)} & \ddots & & \vdots & & l_{i+1,i}^{(2)} & \ddots & \\ \vdots & & \vdots & & \ddots & \vdots & & \vdots & & \ddots \\ 0 & \cdots & l_{n,i}^{(1)} & \cdots & 1 & 0 & \cdots & l_{n,i}^{(2)} & \cdots & 0 \\ \hline 0 & & & & & 1 & & & & \\ \vdots & \ddots & & & & 0 & \ddots & & & \\ \vdots & & 0 & & & \vdots & & \overline{l_{i+1,i}^{(1)}} & & \\ \vdots & & -\overline{l_{i+1,i}^{(2)}} & \ddots & & \vdots & & \ddots & & \\ \vdots & & \vdots & & \ddots & \vdots & & & \ddots & \\ 0 & \cdots & -\overline{l_{n,i}^{(2)}} & \cdots & 0 & 0 & \cdots & \overline{l_{n,i}^{(1)}} & \cdots & 1 \end{array}\right]. \tag{4.2}$$

For any quaternion vector, $x = x^{(1)} + x^{(2)}\mathrm{j} \in \mathbf{Q}^n$, $x^{(j)} = \left(x_1^{(j)}, \cdots, x_n^{(j)}\right)^T \in \mathbf{C}^n$, $j = 1, 2$, if we take the $k(i < k \leq n)$ member of vector and convert it to $0(x_i \neq 0)$, let

$$l_i^{(j)} = \left(0, \cdots, 0, l_{k,i}^{(j)}, 0, \cdots, 0\right)^T, i < k \leq n, j = 1, 2, \tag{4.3}$$

where $l_{k,i}^{(1)} = \left(x_k^{(1)}\overline{x_i^{(1)}} + x_k^{(2)}\overline{x_i^{(2)}}\right)/\|x_i\|^2$, $l_{k,i}^{(2)} = \left(-x_k^{(1)}x_i^{(2)} + x_k^{(2)}x_i^{(1)}\right)/\|x_i\|^2$, $\|x_i\|^2 = \left\|x_i^{(1)}\right\|^2 + \left\|x_i^{(2)}\right\|^2$, then $L_i x = (x_1, x_2, \cdots x_{k-1}, 0, x_{k+1}, \cdots x_n)^T$.

For any matrix $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} = M_1 + M_2\mathrm{j} \in \mathbf{Q}^{m\times n}$, $M_1 = \left(a_{ij}^{(1)}\right)$, $M_2 = \left(a_{ij}^{(2)}\right) \in \mathbf{C}^{m\times n}$, let the initial matrix be as follows,

$$A^{\sigma} = \left[\begin{array}{ccc:ccc} a_{11}^{(1)} & \cdots & a_{1n}^{(1)} & a_{11}^{(2)} & \cdots & a_{1n}^{(2)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}^{(1)} & \cdots & a_{mn}^{(1)} & a_{m1}^{(2)} & \cdots & a_{mn}^{(2)} \\ \hdashline -a_{11}^{(2)} & \cdots & -a_{1n}^{(2)} & a_{11}^{(1)} & \cdots & a_{1n}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -\overline{a_{m1}^{(2)}} & \cdots & -\overline{a_{mn}^{(2)}} & \overline{a_{m1}^{(1)}} & \cdots & \overline{a_{mn}^{(1)}} \end{array}\right]. \tag{4.4}$$

(1) If $\|a_{11}\|^2 = \left\|a_{11}^{(1)}\right\|^2 + \left\|a_{11}^{(2)}\right\|^2 > 0$, construct the complex representation matrix of Gaussian transformation matrix L_1,

$$L_1^{\sigma} = \left[\begin{array}{ccccc:ccccc} 1 & & & & & 0 & & & & \\ -l_{21}^{(1)} & 1 & & & & -l_{21}^{(2)} & 0 & & & \\ \vdots & \vdots & \ddots & & & \vdots & \vdots & \ddots & & \\ -l_{m1}^{(1)} & 0 & \cdots & 1 & & -l_{m1}^{(2)} & 0 & \cdots & 0 & \\ \hdashline 0 & & & & & 1 & & & & \\ \overline{l_{21}^{(2)}} & 0 & & & & \overline{-l_{21}^{(1)}} & 1 & & & \\ \vdots & \vdots & \ddots & & & \vdots & \vdots & \ddots & & \\ \overline{l_{m1}^{(2)}} & 0 & \cdots & 0 & & \overline{-l_{m1}^{(1)}} & 0 & \cdots & 1 & \end{array}\right], \tag{4.5}$$

in which, $l_{i1}^{(1)} = \left(a_{i1}^{(1)}\overline{a_{11}^{(1)}} + a_{i1}^{(2)}\overline{a_{11}^{(2)}}\right)/\|a_{11}\|^2$, $l_{i1}^{(2)} = \left(-a_{i1}^{(1)}a_{11}^{(2)} + a_{i1}^{(2)}a_{11}^{(1)}\right)/\|a_{11}\|^2$, $i = 2, ..., m$, then,

$$L_1^{\sigma}A^{\sigma} = \left[\begin{array}{cc:cc} \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} & \hat{a}_{11}^{(2)} & \hat{M}_{12}^{(2)} \\ 0 & \hat{M}_{22}^{(1)} & 0 & \hat{M}_{22}^{(2)} \\ \hdashline -\hat{a}_{11}^{(2)} & -\hat{M}_{12}^{(2)} & \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} \\ 0 & -\hat{M}_{22}^{(2)} & 0 & \hat{M}_{22}^{(1)} \end{array}\right], \tag{4.6}$$

clearly, $L_1^{\sigma}A^{\sigma}$ is the complex representation matrix of matrix $\begin{bmatrix} \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} \\ 0 & \hat{M}_{22}^{(1)} \end{bmatrix} + \begin{bmatrix} \hat{a}_{11}^{(2)} & \hat{M}_{12}^{(2)} \\ 0 & \hat{M}_{22}^{(2)} \end{bmatrix}\mathrm{j}$, and $\hat{M}_{12}^{(1)}$, $\hat{M}_{12}^{(2)} \in \mathbf{C}^{1\times(n-1)}$, $\hat{M}_{22}^{(1)}$, $\hat{M}_{22}^{(2)} \in \mathbf{C}^{(m-1)\times(n-1)}$ are submatrix after Gauss transformation of matrix M_1, M_2.

(2) If $\|a_{11}\|^2 = \left\|a_{11}^{(1)}\right\|^2 + \left\|a_{11}^{(2)}\right\|^2 = 0$, choose $i_0 (2 \le i_0 \le m)$ such that, $\left\|a_{i_0 1}\right\|^2 = \left\|a_{i_0 1}^{(1)}\right\|^2 + \left\|a_{i_0 1}^{(2)}\right\|^2 > 0$, interchange rows 1 and i_0 of A, i.e., there exists a real

permutation matrix P_1 such that,

$$P_1^\sigma A^\sigma = \left[\begin{array}{cccc:cccc} a_{i_0 1}^{(1)} & \cdots & a_{i_0 n}^{(1)} & & a_{i_0 1}^{(2)} & \cdots & a_{i_0 n}^{(2)} \\ * & \cdots & * & & * & \cdots & * \\ a_{11}^{(1)} & \cdots & a_{1n}^{(1)} & & a_{11}^{(2)} & \cdots & a_{1n}^{(2)} \\ * & \cdots & * & & * & \cdots & * \\ \hdashline -a_{i_0 1}^{(2)} & \cdots & -a_{i_0 n}^{(2)} & & a_{i_0 1}^{(1)} & \cdots & a_{i_0 n}^{(1)} \\ * & \cdots & * & & * & \cdots & * \\ -a_{11}^{(2)} & \cdots & -a_{1n}^{(2)} & & a_{11}^{(1)} & \cdots & a_{1n}^{(1)} \\ * & \cdots & * & & * & \cdots & * \end{array}\right], \tag{4.7}$$

in which case, $P_1^\sigma A^\sigma$ satisfies (1), by (4.6), construct the matrix L_1 such that,

$$L_1^\sigma P_1^\sigma A^\sigma = \left[\begin{array}{cc:cc} \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} & \hat{a}_{11}^{(2)} & \hat{M}_{12}^{(2)} \\ 0 & \hat{M}_{22}^{(1)} & 0 & \hat{M}_{22}^{(2)} \\ \hdashline -\hat{a}_{11}^{(2)} & -\hat{M}_{12}^{(2)} & \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} \\ 0 & -\hat{M}_{22}^{(2)} & 0 & \hat{M}_{22}^{(1)} \end{array}\right], \tag{4.8}$$

clearly, $L_1^\sigma P_1^\sigma A^\sigma$ is a complex representation matrix of matrix $\begin{bmatrix} \hat{a}_{11}^{(1)} & \hat{M}_{12}^{(1)} \\ 0 & \hat{M}_{22}^{(1)} \end{bmatrix} + \begin{bmatrix} \hat{a}_{11}^{(2)} & \hat{M}_{12}^{(2)} \\ 0 & \hat{M}_{22}^{(2)} \end{bmatrix}$ j, in which $\hat{M}_{12}^{(1)},\ \hat{M}_{12}^{(2)} \in \mathbf{C}^{1\times(n-1)}, \hat{M}_{22}^{(1)},\ \hat{M}_{22}^{(2)} \in \mathbf{C}^{(m-1)\times(n-1)}$ are submatrix after Gauss transformation of matrix M_1, M_2.

After the above step t, M_1, M_2 are two unit upper triangular matrices, i.e.,

$$L_r^\sigma P_r^\sigma \cdots L_1^\sigma P_1^\sigma A^\sigma = \left[\begin{array}{cc:cc} \hat{M}^{(1)} & & \hat{M}^{(2)} & \\ & 0 & & 0 \\ \hdashline -\overline{\hat{M}}^{(2)} & & \overline{\hat{M}}^{(1)} & \\ & 0 & & 0 \end{array}\right]. \tag{4.9}$$

For unified representation, $P_1, \cdots, P_r$ display all exists, $P_i (0 \le i \le r)$ not all of them exist in the actual operation, and $\hat{M}^{(1)}, \hat{M}^{(2)} \in \mathbf{C}^{r\times n}$ are submatrix after Gauss transformation of matrix M_1, M_2, then the complex representation matrix of the full rank decomposition matrix is

$$C^\sigma = \begin{bmatrix} \hat{M}^{(1)} & \hat{M}^{(2)} \\ -\overline{\hat{M}}^{(2)} & \overline{\hat{M}}^{(1)} \end{bmatrix} \in \mathbf{C}^{2r\times 2n},\ B^\sigma = P^\sigma L^\sigma(:,\ [1:r, n:n+r]) \in \mathbf{C}^{2m\times 2r}, \tag{4.10}$$

where $P^\sigma = \left(Q_r^\sigma \cdots Q_2^\sigma P_1^\sigma\right)^{-1},\ Q_i^\sigma = \left(L_{i-1}^\sigma \cdots L_1^\sigma\right)^{-1} P_i^\sigma \left(L_{i-1}^\sigma \cdots L_1^\sigma\right),\ 2 \leqslant i \leqslant r,\ L^\sigma = \left(L_r^\sigma \cdots L_1^\sigma\right)^{-1}$.

Theorem 4.1 *Supposed that* $A \in \mathbf{Q}^{m\times n}, rank(A) = r$, *then there exist* $B^\sigma \in \mathbf{C}^{2m\times 2r}$, $C^\sigma \in \mathbf{C}^{2r\times 2n}$, $rank(B^\sigma) = rank(C^\sigma) = 2r$, *satisfies* $A^\sigma = B^\sigma C^\sigma$, *i.e.,* $A = BC$.

By (2.2) *and* Proposition 2.1*, we only need to calculate the first row block of the complex representation matrix in the actual program operation. Next, we give the specific algorithm of the above structure-preserving process.*

Algorithm 2 (Complex structure-preserving algorithm for the full rank decomposition): For any matrix $A = M_1 + M_2\mathrm{j} \in \mathbf{Q}^{m\times n}$, Input $M_1, M_2 \in \mathbf{C}^{\mathrm{m}\times n}$, output $B_1, B_2 \in \mathbf{C}^{\mathrm{m}\times r}, C_1, C_2 \in \mathbf{C}^{\mathrm{r}\times n}$, i.e., $B = B_1 + B_2\mathrm{j}, C = C_1 + C_2\mathrm{j}$, such that $A = BC$.

i. Complex matrix representation function.

Function:$\mathrm{A}^{\sigma} = \mathrm{CR}(\mathrm{M}_1, \mathrm{M}_2)$

$$\mathrm{A}^{\sigma} = \begin{bmatrix} \mathrm{M}_1 & \mathrm{M}_2 \\ -\overline{\mathrm{M}_2} & \overline{\mathrm{M}_1} \end{bmatrix};$$

end

ii. Matrix decomposition function.

Function $[\mathrm{B}_1, \mathrm{B}_2, \mathrm{C}_1, \mathrm{C}_2] = \mathrm{QFC}(\mathrm{M}_1, \mathrm{M}_2)$

(1) Initialization

M = [M1, M2]; [m,n] = size(M1); j = 1; p = 1 : m;

BB=[eye(m,n), zeros(m,n)]; W=eye(m); N=eye(m);

(2) Cyclic body of Gauss elimination method

while j <= min(m,n)

d=[M(j,j),M(j,j+n)]*[M(j,j),M(j,j+n)]';

if d > 0

BB(j + 1:m, [j,j + n]) = M(j + 1:m, [j,j + n]) ∗ CR(conj(M(j,j))/d, −M(j,j + n)/d);

M(j + 1:m,[j,j + n]) = zeros(m - j,2);

M(j + 1:m,[j + 1:n,j + 1 + n:2*n]) = M(j + 1:m,[j + 1:n,j + 1 + n:2*n])

−BB(j + 1:m,[j,j + n])*CR(M(j,j + 1:n),M(j,j + 1 + n:2*n));

else k=j+1; while k<=m

d=[M(j,j),M(j,j+n)]*[M(j,j),M(j,j+n)]';

if d > 0

M([j,k],:) = M([k,j],:); p([j,k]) = p([k,j]);

BB(j + 1:m, [j,j + n]) = M(j + 1:m, [j,j + n]) ∗ CR(conj(M(j,j))/d, −M(j,j + n)/d);

M(j + 1:m,[j,j + n]) = zeros(m - j,2);

M(j + 1:m,[j + 1:n,j + 1 + n:2*n]) = M(j + 1:m,[j + 1:n,j + 1 + n:2*n])

−BB(j + 1:m,[j,j + n])*CR(M(j,j + 1:n),M(j,j + 1 + n:2*n)); break;

else k=k+1; end end

end j=j+1; end

(3) Output results

for k = 1:m

N(k,:) = W(p(k),:);

end

for j = n: - 1:1

id = find(sum(([M(:, j), M(:, j + n)].∗[conj(M(:, j)), conj(M(:, j + n))]')>1e−10, 1, 'last');

if id > 0 break; end

end

B1=N*BB(:,1:id); B2=N*BB(:,n+1:n+id); C1=M(1:id,1:n); C2=M(1:id,n+1:2*n);

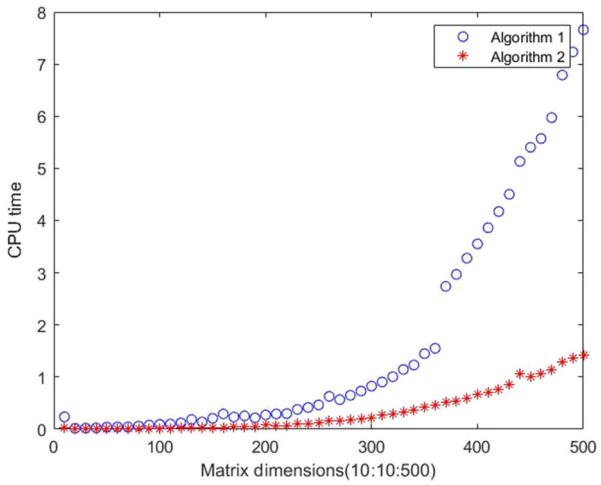

Fig. 1 CPU time scatter diagram

Example 4.1 For any quaternion matrix $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} \in \mathbf{Q}^{m\times n}$, let $m = n = 10 : 10 : 500, A_1 = \mathrm{rand}(m, n), A_2 = \mathrm{rand}(m, n), A_3 = \mathrm{rand}(m, n), A_4 = \mathrm{rand}(m, n)$. Calculate the CPU time and relative error of two algorithms for the full rank decomposition of random quaternion matrices of order 10 to 500 $\left(\eta_k = \frac{\|A_k - B_k C_k\|}{\|A\|}\right)$ as follows (Figs. 1 and 2).

Example 4.1 shows that the error of Algorithm 2 is smaller than that of Algorithm 1, and the running time of CPU is also shortened obviously. It can be seen that the result of complex structure-preserving algorithm for the full rank decomposition of quaternion matrices have better timeliness, small error and short running time, which is conducive to the analysis and discussion of a quaternion matrix.

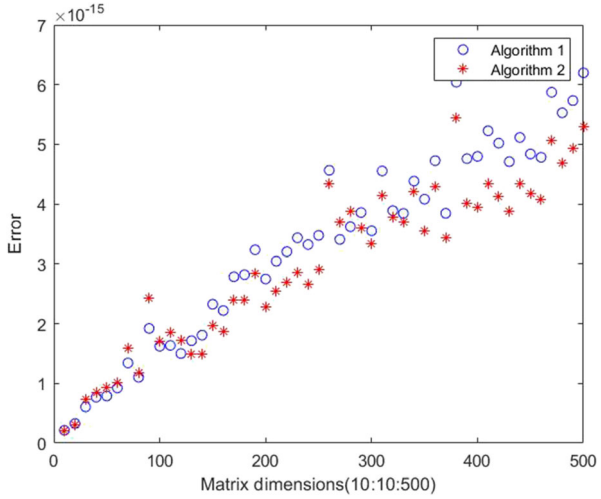

Fig. 2 Error scatter diagram of algorithms 1 and 2

5 A fast algorithm for solving the quaternion linear equations

The main application of the full rank decomposition is to solve the generalized inverse matrix, and then to solve the general solution of corresponding linear equation. At the same time, in the process of the full rank decomposition, we can also get the rank of matrix and so on. In this section, we extend the above two full rank decomposition algorithms, give two algorithms for solving quaternion linear equations, and compare the feasibility and timeliness of the two algorithms. Finally, the complex structure-preserving algorithm of the full rank decomposition is applied to the sparse representation of color images, and the test images are classified.

Algorithm 3 (A direct algorithm for solving quaternion linear equations $Ax = b$): Let $A \in \mathbf{Q}^{m\times n},\ b \in \mathbf{Q}^{m\times 1}$.

Step 1. Input the matrix A. Run the Algorithm 1, compute $B,\ C$;
Step 2. Compute the generalized inverse matrix $A^{\dagger} = C^*(CC^*)^{-1}(B^*B)^{-1}B^*$ of matrix A;
Step 3. Compute the minimum norm solution $X = A^{\dagger}b$ for linear equations $Ax = b$.

Algorithm 4 (A fast complex structure preserving algorithm for solving quaternion linear equations $Ax = b$): Let $A \in \mathbf{Q}^{m\times n},\ b \in \mathbf{Q}^{m\times 1}$.

Step 1. Compute the complex representation matrix A^{σ} and b^{σ};
Step 2. Run the Algorithm 2, compute $B_1,\ B_2,\ C_1,\ C_2$;
Step 3. Compute the generalized inverse matrix of A.
Function:$(\mathrm{A}^{\dagger})_{\mathrm{r}}^{\sigma} = \mathrm{QGI}(\mathrm{B}_1, \mathrm{B}_2, \mathrm{C}_1, \mathrm{C}_2)$
r=size(C_1, 1); CC = [C_1, C_2];

$$\mathrm{Q_n} = \begin{bmatrix} \text{zeros(n)} & \text{eye(n)} \\ -\text{eye(n)} & \text{zeros(n)} \end{bmatrix};\ \mathrm{Q}_r = \begin{bmatrix} \text{zeros(r)} & \text{eye(r)} \\ -\text{eye(r)} & \text{zeros(r)} \end{bmatrix};$$

$$(\mathrm{A}^{\dagger})_r^{\sigma} = \left[C_1^*,\ -C_2^T\right]\begin{bmatrix} [CCCC^*,\ -CCQ_nCC^T] \\ [\overline{CC}CC^T,\ -\overline{CC}Q_nCC^*]Q_r \end{bmatrix}^{-1}\begin{bmatrix} [B_1^*,\ -B_2^T]CR(B_1, B_2) \\ [B_1^T,\ -B_2^*]CR(\overline{B_1}, \overline{B_2})Q_r \end{bmatrix}^{-1} CR(B_1^*, -B_2^T);$$

end
Step 4. Compute the minimum norm solution X for linear equations $Ax = b$.

$$XX = (A^{\dagger})_r^{\sigma}b^{\sigma}, \qquad x = XX(:, 1) + XX(:, 2)\mathrm{j}.$$

In addition, the above steps 1–3 is also a fast algorithm for calculating the generalized inverse matrix of quaternion matrix.

Example 5.1 Find the general solution to the system of linear equations $Ax = b$ by the full rank decomposition of matrix A, in which

$$A = \begin{bmatrix} 1+2\mathrm{i} & 3\mathrm{j} & 2\mathrm{k} \\ 0 & 4+2\mathrm{k} & \mathrm{i+j} \\ 2+4\mathrm{i} & 8\mathrm{i+12j} & -3+3\mathrm{k} \end{bmatrix},\ b = \begin{bmatrix} -3+14\mathrm{j+5k} \\ 3+13\mathrm{i+5j+k} \\ -29+5\mathrm{i+33j-11k} \end{bmatrix}.$$

(1) By (2.1) and (2.2), we have

$$A_r^{\sigma} = \left[\begin{array}{ccc:ccc} 1+2\mathrm{i} & 0 & 0 & 0 & 3 & 2\mathrm{i} \\ 0 & 4 & \mathrm{i} & 0 & 2\mathrm{i} & 1 \\ 2+4\mathrm{i} & 8\mathrm{i} & -3 & 0 & 12 & 3\mathrm{i} \end{array}\right],$$

Run Algorithm 2, and we get

$$B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & \mathrm{i} \end{bmatrix}, \ B_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 2 \end{bmatrix}, \ C_1 = \begin{bmatrix} 1+2\mathrm{i} & 0 & 0 \\ 0 & 4 & \mathrm{i} \end{bmatrix}, \ C_2 = \begin{bmatrix} 0 & 3 & 2i \\ 0 & 2\mathrm{i} & 1 \end{bmatrix},$$

i.e., $\mathrm{rand}(A) = 2$, the full rank decomposition is $A = BC$, in which

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & \mathrm{i}+2\mathrm{j} \end{bmatrix}, \ C = \begin{bmatrix} 1+2\mathrm{i} & 3\mathrm{j} & 2\mathrm{k} \\ 0 & 4+2\mathrm{k} & \mathrm{i}+\mathrm{j} \end{bmatrix}.$$

(2) By (1) and Algorithm 4, it is easy to get

$$A^{\dagger}b = \begin{bmatrix} -0.2286+0.4929\mathrm{i}+2.8357\mathrm{j}-1.0286\mathrm{k} \\ 1.4357+2.0143\mathrm{i}-0.2429\mathrm{j}+0.0929\mathrm{k} \\ 3.2000+2.4000\mathrm{i}+0.1571\mathrm{j}+1.2571\mathrm{k} \end{bmatrix},$$

then the general solution of linear equations $Ax = b$ is $x = A^{\dagger}b + (I - A^{\dagger}A)y$, in which $y \in \mathbf{Q}^n$ is any quaternion vector.

Example 5.2 For any quaternion matrix $A = A_1 + A_2\mathrm{i} + A_3\mathrm{j} + A_4\mathrm{k} \in \mathbf{Q}^{m\times n}$, $x = x_1 + x_2\mathrm{i} + x_3\mathrm{j} + x_4\mathrm{k} \in \mathbf{Q}^{n\times 1}$, let $m = n = 10 : 10 : 500$, $A_1 = \mathrm{rand}(m, n)$, $A_2 = \mathrm{rand}(m, n)$, $A_3 = \mathrm{rand}(m, n)$, $A_4 = \mathrm{rand}(m, n)$, $x_1 = \mathrm{rand}(n, 1)$, $x_2 = \mathrm{rand}(n, 1)$, $x_3 = \mathrm{rand}(n, 1)$, $x_4 = \mathrm{rand}(n, 1)$, $b = Ax$. Calculate the CPU time and relative error of two algorithms for the solution of quaternion linear equations $Ax = b$ of order 10 to 500 $\left(\eta_k = \frac{\|A^{\dagger}b-x\|_F}{\|x\|_F}\right)$ as follows (Figs. 3 and 4).

Example 5.2 shows that in terms of calculating the linear equations, the running time and calculation error of Algorithm 4 are also far less than the result of Algorithm 3. And for the calculation of high-dimensional linear equations, Algorithm 3 is difficult to implement, but Algorithm 4 is still fast. In the above chart, in order to compare the results of the two algorithms, we only performed operations within 500 dimensions. In fact, Algorithm 4 can perform higher-dimensional operations.

Example 5.3 Given a color face recognition training data set (Fig. 5), and the corresponding test sample (Figs. 6, 7) are classified. (Color image classification for different subjects. Images of a subject from the Faces95 database.)

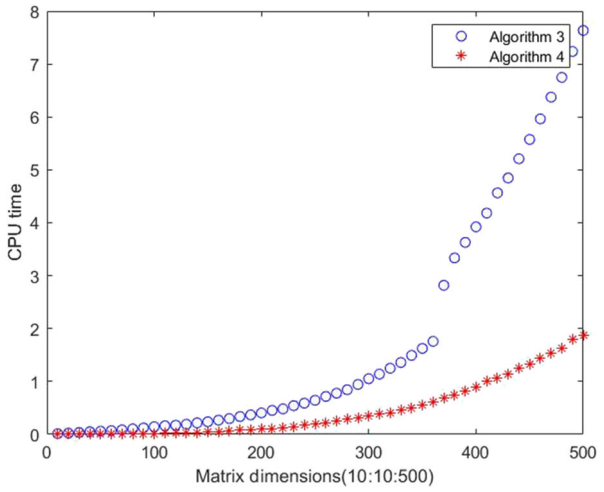

Fig. 3 CPU time scatter diagram

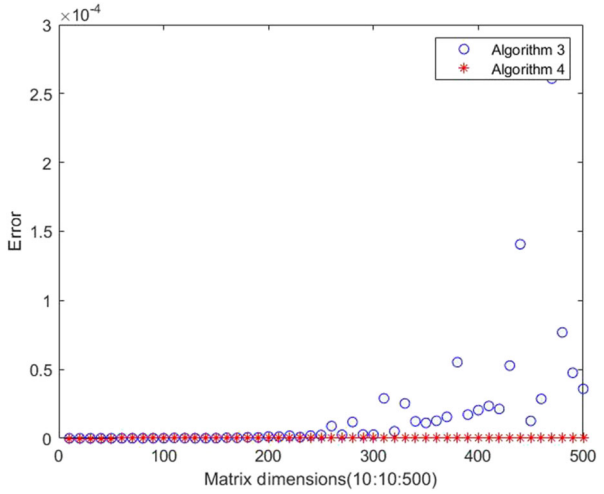

Fig. 4 Error scatter diagram of algorithms 3 and 4

Fig. 5 The set of training samples 1, c=4

Fig. 6 The test sample 1

Let $A = R\mathrm{i} + G\mathrm{j} + B\mathrm{k}$ be quaternion representation matrix of the training data set (Fig. 5), $b = R_b\mathrm{i} + G_b\mathrm{j} + B_b\mathrm{k}$ be quaternion representation matrix of the test sample (Figs. 6, 7), where R, R_b, G, G_b, B, B_b are real matrices standing for red, green, and blue colors. A training set is divided into $c(c = 4)$ categories. The goal is exactly to predict the label of b from the given c class training samples. For each class i, let $\delta_i : \mathbf{R}^n \to \mathbf{R}^n$ be the characteristic function which selects the coefficients associated with the ith class. The test sample b can be approximated by $\hat{b}_i = A\delta_i(x)$ which uses the vector δ_i from each class. By using Algorithm 4 and the reconstruction residual for ith class $r_i(b) = \|b - \hat{b}_i\|_2$, we can obtain the approximate sparse representation and classification results of the test image (Figs. 8, 9, 10 and 11).

Fig. 7 The test sample 2

Fig. 8 The approximate test sample 1

In Example 5.3, four subjects were randomly selected from Faces95 database, and the color face sparse representation was performed on the test images of the first subject and the fourth subject respectively. The results show that the classification is more accurate.

Example 5.4 Given a color face recognition training data set (Fig. 12), and the corresponding test data (Fig. 13) are classified. (Color image classification of different micro expressions of the same subject.) By using Algorithm 4, we can obtain the approximate sparse representation and classification results of the test image (Figs. 14 and 15).

Fig. 9 Figure 6 belongs to the first category of the set of training samples

Fig. 10 The approximate test sample 2

Fig. 11 Figure 7 belongs to the fourth category of the set of training samples

Fig. 12 The set of training samples 2, c=4

Fig. 13 The test sample 1

Fig. 14 The approximate test sample 1

Fig. 15 Figure 13 belongs to the first category of the set of training samples

In Example 5.4, several sets of micro expression data sets of a subject are randomly selected from the network to sparse represent the different expression images of the same subject. The results show that the classification method is more accurate.

6 Conclusions

In this paper, based on the Gauss elimination method of quaternion matrix, we discuss the direct Gauss transform (based on quaternion toolbox (QTFM) and the complex structure-preserving Gauss transform of a quaternion matrix. By using the Gauss transformation, we obtain the direct algorithm and complex structure-preserving algorithm for the full rank decomposition of a quaternion matrix. Finally, this algorithm can also be applied to the sparse representation classification of color images. The feasibility and timeliness of the proposed algorithms are demonstrated by the corresponding application examples.

Funding This paper is supported by the Shandong Natural Science Foundation (No. ZR201709250116) and Chinese Government Scholarship (CSC No. 202008370340, No. 202108370086).

Data availability All data generated or analyzed during this study are included in this published article (and its supplementary information files).

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Adler, S.: Composite leptons quark constructed as triply occupied quasi-particle in quaternionic quantum mechanics. Phys. Lett B. **332**, 358–365 (1994)
2. Adler, S.: Quaternionic Quantum Mechanics and Quantum Fields. Oxford University Press, New York (1995)
3. De Leo, S., Ducati, G.: Quaternionic groups in physics. Int. J. Theor. Phys. **38**, 2197–2220 (1999)
4. Le Bihan, N., Sangwine, S.: Quaternion principal component analysis of color images. IEEE Int. Conf. Image Process **1**, 809–812 (2003)
5. Mackey, D.S., Mackey, N., Tisseur, F.: Structured tools for structured matrices. Electron. J. Linear Algebra **10**, 106–145 (2003)
6. Sangwine, S., Le Bihan, N.: Quaternion toolbox for matlab. http://qtfm.sourceforge.net/
7. Jiang, T.: An algorithm for quaternionic linear equations in quaternionic quantum theory. J. Math. Phys. **45**(11), 4218–4222 (2004)
8. Wang, G., Guo, Z., Zhang, D., Jiang, T.: Algebraic techniques for least squares problem over generalized quaternion algebras: A unified approach in quaternionic and split quaternionic theory. Math. Meth. Appl. Sci. **43**(3), 1124–1137 (2020)
9. Wang, M., Ma, W.: A structure-preserving algorithm for the quaternion Cholesky decomposition. Appl. Math. Comput. **223**, 354–361 (2013)
10. Wang, M., Ma, W.: A structure-preserving method for the quaternion LU decomposition in quaternionic quantum theory. Comput. Phys. Commun. **184**(9), 2182–2186 (2013)
11. Sangwine, S.J.: Comment on 'A structure-preserving method for the quaternion LU decomposition in quaternionic quantum theory' by Minghui Wang and Wenhao Ma. Comput. Phys. Commun. **188**, 128–130 (2015)
12. Li, Y., Wei, M., Zhang, F., Zhao, J.: A real structure-preserving method for the quaternion LU decomposition, revisited. Calcolo **54**(4), 1553–1563 (2017)
13. Li, Y., Wei, M., Zhang, F., Zhao, J.: A fast structure-preserving method for computing the singular value decomposition of quaternion matrices. Appl. Math. Comput. **235**, 157–167 (2014)
14. Li, Y., Wei, M., Zhang, F., Zhao, J.: Real structure-preserving algorithms of Householder based transformations for quaternion matrices. Comput. Appl. Math. **305**, 82–91 (2016)
15. Li, Y., Wei, M., Zhang, F., Zhao, J.: On the power method for quaternion right eigenvalue problem. Comput. Appl. Math. **345**, 59–69 (2019)
16. Jia, Z., Wei, M., Ling, S.: A new structure-preserving method for quaternion Hermitian eigenvalue problems. Comput. Appl. Math. **239**, 12–24 (2013)
17. Ma, R., Jia, Z., Bai, Z.: A structure-preserving Jacobi algorithm for quaternion Hermitian eigenvalue problems. Comput. Math. Appl. **75**(3), 809–820 (2018)
18. Wei, M., Li, Y., Zhang, F., Zhao, J.: Quaternion Matrix Computations. Nova Science Publishers (2018)
19. Lu, Y., Cui, J., Fang, Z.: Enhancing sparsity via full rank decomposition for robust face recognition. Neural Comput. Appl. **25**, 1043–1052 (2014)
20. Malkoti, A., Vedanti, N., Tiwari, R.K.: A highly efficient implicit finite difference scheme for acoustic wave propagation. J. Appl. Geophys. **161**, 204–215 (2019)
21. Cosnard, M., Marrakchi, M., Robert, Y.: Parallel Gaussian elimination on an MIMD computer. Parallel Comput. **6**, 275–296 (1988)
22. Litvinov, G.L., Maslova, E.V.: Universal numerical algorithms and their software implementation. Program Comput. Software **26**(5), 275–280 (2000)
23. Sheng, X., Chen, G.: Full-rank representation of generalized inverse $A_{TS}^{(2)}$ and its application, Comput. Math. Appl. **54**, 1422–1430 (2007)
24. Jiang, M., Jiang, B., Cai, W., Du, X., et al.: Internal model control for structured rank deficient system based on full rank decomposition. Cluster Comput. **20**, 13–24 (2017)