AIMS Mathematics, 10(7): 15663–15697.
DOI: 10.3934/math.2025702
Received: 12 March 2025
Revised: 28 May 2025
Accepted: 06 June 2025
Published: 09 July 2025

https://www.aimspress.com/journal/Math

Research article

Solving two-sided Sylvester quaternionic matrix equations: Theoretical insights, computational implementation, and practical applications

Abdur Rehman[1], Cecilia Castro[2,*], Víctor Leiva[3,*], Muhammad Zia Ur Rahman[4] and Carlos Martin-Barreiro[5]

[1] Department of Basic Sciences and Humanities, University of Engineering and Technology, Lahore, Faisalabad Campus, Faisalabad, Pakistan
[2] Centre of Mathematics, Universidade do Minho, Braga, Portugal
[3] School of Industrial Engineering, Universidad Catolica de Valparaíso, Valparaíso, Chile
[4] Department of Electrical Electronics and Telecommunication Engineering, University of Engineering and Technology, Lahore, Faisalabad Campus, Faisalabad, Pakistan
[5] Facultad de Ingeniería, Universidad Espíritu Santo, Samborondón, Ecuador

* **Correspondence:** Email: cecilia@math.uminho.pt, victorleivasanchez@gmail.com.

Abstract: In this article, we investigate a class of quaternionic functional equations of the two-sided Sylvester type, which arise in areas such as control theory, robotics, signal processing, and image analysis. Although quaternionic matrix equations have been extensively studied, two-sided Sylvester systems remain particularly challenging due to the noncommutativity of quaternion multiplication and the increased structural complexity they entail. We derive general solutions to these systems by establishing the necessary and sufficient conditions for solvability, unifying and by extending previous theoretical results. We propose an efficient algorithm to compute the general solution in both full-rank and rank-deficient cases, using Moore-Penrose inverses and projection operators The practical interest of our method is demonstrated through applications in perturbation theory, image processing, and robust control. We present numerical examples to validate the proposed approach, including a case study involving a multi-joint robotic manipulator. These results highlight both the theoretical relevance and computational advantages of the proposed method.

Keywords: functional equations in operator algebras; Moore-Penrose inverse; noncommutative functional equations; quaternionic matrix solver; rank properties; solvability conditions; singular value decomposition; Sylvester equations
Mathematics Subject Classification: 15A33, 15A09, 65F05

1. Introduction, motivating example, and objectives

1.1. Introduction

Quaternions, introduced by Hamilton [1], have evolved from pure mathematics into essential tools in areas such as quantum physics, mechanics, and signal processing [2,3]. One of their key advantages is the ability to represent three-dimensional (3D) rotations compactly, making them invaluable in aerospace engineering, computer graphics, and related fields [4,5].

More recently, the concept of quaternionic matrices has gained attention in control systems, robotics, and signal processing [6, 7], preserving the noncommutative nature of quaternions and offering distinct advantages in multidimensional transformation problems [8]. Such noncommutativity is particularly helpful in 3D rotation algorithms for robotics [9] and in control theory with multiple dynamic components.

One of the fundamental topics in this context is obtaining the solution of quaternionic matrix equations, which underpin practical tasks in engineering and physics. The Sylvester equation has been studied extensively in real and complex domains [10, 11], whereas iterative and neural network methods have also been proposed [12, 13]. Extending these approaches to quaternionic matrices [14, 15] introduces challenges due to noncommutativity, particularly in the case of two-sided Sylvester equations such as $AX + XB = C$. Standard techniques are not applicable when attempting to reverse the order quaternionic multiplication [16], which has prompted the development of specialized methods [17–20].

In many applications, it is necessary to perform multiple matrix products efficiently. Fast algorithms and modern hardware features, such as Intel advanced matrix extensions (AMX), are increasingly essential for handling large-scale or real-time scenarios [21]. Future research could explore the integration of quaternionic matrix methods with unstructured data analysis and big data frameworks [22], particularly in computer vision and robotic sensing applications. Since quaternionic operations can be more computationally intensive than their real or complex counterparts, these developments are highly relevant to the implementation of two-sided Sylvester solvers in practical contexts. Consequently, our aim is to contribute to the theory and practice of quaternionic Sylvester equations by examining solvability conditions, proposing an explicit solution framework, and illustrating its effectiveness in real-world scenarios such as control and robotics.

1.2. Applications of quaternionic matrices, and motivating example

While quaternions are typically used to represent individual 3D rotations, quaternionic matrices enable the coupling of multiple transformations within larger linear systems. This usage is especially helpful in multi-link robotic manipulators, where each joint has a known initial orientation and an unknown incremental rotation that must be determined simultaneously.

For example, consider a simplified robotic system consisting of two revolute joints. Let $A \in \mathbb{H}^{2\times 2}$ represent the rotation of the first link, let $C \in \mathbb{H}^{2\times 2}$ be that of the second link, and let E be the desired orientation of the end effector. Then, the incremental rotations X and Y, which are also 2×2 quaternionic matrices, satisfy the equation given by

$$AX + CY = E. \tag{1.1}$$

While the expression in (1.1) is sufficient for systems with two joints, more complex configurations (such as those involving four or more joints) naturally lead to higher-dimensional analogues of this equation, possibly involving additional unknown blocks such as Z and W. Such extended systems can be formulated as two-sided Sylvester-type equations, which require robust solution strategies capable of handling quaternionic noncommutativity.

Systems similar to the one presented in (1.1) arise whenever multiple 3D rotations and transformations need to be composed into a unified operation. Quaternionic matrices offer a unified framework for handling such rotations, as they preserve the multiplication order inherent to noncommutative quaternions and thereby avoid problems such as gimbal lock, while facilitating efficient linear-algebraic solution methods. More advanced robotic applications may impose additional constraints (such as collision avoidance or orientation alignment), which give rise to two-sided Sylvester-type equations involving multiple unknown matrices. In Subsection 5.2, we present a more detailed example and demonstrate how the proposed algorithm from later sections can be applied to such scenarios.

1.3. Contributions and structure of the article

Although quaternionic matrix equations have been extensively studied, two-sided Sylvester systems remain challenging and have important implications for control theory, signal processing, and image analysis [15, 19, 20]. In this article, we derive general solutions to such problems by:

- Deriving the necessary and sufficient conditions for the solvability of two-sided Sylvester quaternionic matrix equations, thereby unifying and extending previous theoretical contributions;
- Proposing an efficient algorithm to compute the general solution for both full-rank and rank-deficient cases, using quaternionic inverses and projection operators;
- Demonstrating the practical relevance of the proposed methods through applications in fields such as perturbation theory [11], image processing [23], and robust control [24].

The rest of this article is structured as follows. Section 2 provides the necessary preliminaries on quaternionic algebra and matrix operations, followed by the problem formulation and main contributions. In Section 3, we establish the theoretical framework and present the lemmas used to derive the solvability conditions. In Section 4, the general algorithm for solving quaternionic matrix equations is presented. In Section 5, we illustrate the practical implementation of our method through numerical examples and an application in robotics, including the Python code for quaternionic matrix operations and the corresponding error analysis. In Section 6, we conclude with a discussion of the results and outline directions for future research. The appendices contain the proof of the main theorem (Appendix A) and additional details on quaternionic matrix operations (Appendix B).

2. Preliminaries

2.1. Quaternion algebra and matrix operations

Quaternions, introduced by Hamilton [1], constitute a four-dimensional algebra $\mathbb{H}$ over $\mathbb{R}$, generated by the basis $\{1, \mathrm{i}, \mathrm{j}, \mathrm{k}\}$. A generic quaternion $q = e_0 + e_1\mathrm{i} + e_2\mathrm{j} + e_3\mathrm{k}$ consists of the real coefficients e_0, e_1, e_2, and e_3.

The imaginary units satisfy

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = -1, \quad \mathrm{ij} = \mathrm{k},\ \mathrm{jk} = \mathrm{i},\ \mathrm{ki} = \mathrm{j}, \tag{2.1}$$

and their multiplication is noncommutative. This noncommutative property distinguishes $\mathbb{H}$ from $\mathbb{C}$ and often provides advantages in 3D applications, such as representing rotations [2, 3].

Use $\mathbb{H}^{m\times n}$ to denote the set of $m\times n$ matrices with entries in the quaternionic algebra $\mathbb{H}$. For $A \in \mathbb{H}^{m\times n}$, the conjugate transpose, denoted $A^* \in \mathbb{H}^{n\times m}$, is defined by transposing A and applying quaternionic conjugation to each element. A quaternionic matrix may not be invertible in the conventional sense. Thus, we employ the Moore-Penrose inverse $A^\dagger \in \mathbb{H}^{n\times m}$ when handling rank-deficient or non-square matrices.

Following [8, 25], $A^\dagger$ is the unique matrix satisfying the conditions stated as

$$AA^\dagger A = A, \quad A^\dagger AA^\dagger = A^\dagger, \quad (AA^\dagger)^* = AA^\dagger, \quad (A^\dagger A)^* = A^\dagger A. \tag{2.2}$$

Then, we define left and right projectors as

$$L_A = I - A^\dagger A, \quad R_A = I - AA^\dagger, \tag{2.3}$$

which are essential in quaternionic matrix decompositions and in solving noncommutative linear systems [7, 19].

In the real or complex setting, the Sylvester equation given by $AX + XB = C$ can be solved using well-established techniques such as the Bartels-Stewart method or Kronecker-product formulations [10–13, 26, 27]. These techniques rely on commutative scalar multiplication, which allows the reordering of factors and the reformulation of the problem into a linear system. However, in the quaternionic setting, one cannot interchange AX and XB due to noncommutativity, since $\mathrm{ij} \neq \mathrm{ji}$.

Consequently, additional mathematical tools are required, often involving the Moore-Penrose inverse, defined in (2.3), to handle noncommutativity [28, 29]. This article extends these ideas to the two-sided Sylvester equation over $\mathbb{H}$, where the unknown matrix is influenced on both sides by known matrices, by increasing the complexity of the solution.

2.2. Notation overview

To streamline the presentation and facilitate the reader's understanding, Table 1 provides a concise yet comprehensive reference for the core symbols, matrix dimensions, and linear algebraic operations employed throughout the article. This includes not only the notation for standard matrix and vector operations, but also specialized constructs used in the formulation and solution of quaternionic and noncommutative systems.

Table 1 is intended to serve as a quick-access reference, allowing readers to easily recall the definitions and conventions adopted across the article without interrupting the flow of the main arguments. Such notational clarity is particularly important given the abstract algebraic structures and the operator-based techniques utilized in our methodology.

In particular, the matrix projectors LA and RA, together with the Moore-Penrose inverse $A^\dagger$, appear repeatedly in later sections where we develop the theoretical framework and computational strategies for solving generalized matrix equations in noncommutative settings. Their properties and algebraic behavior play a central role in characterizing solution spaces and ensuring the consistency of the proposed algorithms.

Table 1. Summary of the main notation used in this article.

Symbol / Expression	Definition / Meaning
$\mathbb{R}$	Field of real numbers
i, j, k	Imaginary units in the quaternionic algebra $\mathbb{H}$, satisfying the expression given in (2.1), being noncommutative
q	Generic quaternion $q = e_0 + e_1\mathrm{i} + e_2\mathrm{j} + e_3\mathrm{k}$
$\mathbb{H}$	Quaternion algebra over $\mathbb{R}$
$\mathbb{H}^{m\times n}$	Set of all $m \times n$ matrices with quaternionic entries, with $q_{ij} \in \mathbb{H}$
$A^* \in \mathbb{H}^{n\times m}$	Conjugate transpose of $A \in \mathbb{H}^{m\times n}$, defined by transposing A and taking the quaternionic conjugate of each entry
$A^\dagger \in \mathbb{H}^{n\times m}$	Moore-Penrose inverse of $A \in \mathbb{H}^{m\times n}$ satisfying the expression given in (2.2)
$L_A = I - A^\dagger A$	Left projector for A satisfying $L_A = L_A^* = L_A^2 = L_A^\dagger$, where $L_A \in \mathbb{H}^{m\times m}$ if $A \in \mathbb{H}^{m\times n}$
$R_A = I - AA^\dagger$	Right projector for A satisfying $R_A = R_A^* = R_A^2 = R_A^\dagger$, where $R_A \in \mathbb{H}^{n\times n}$ if $A \in \mathbb{H}^{m\times n}$
$\|\cdot\|_F$	The Frobenius norm of a matrix, corresponding to the square root of the sum of squared moduli of all entries
$AX + XB = C$	Two-sided Sylvester equation given in $\mathbb{H}$, where the unknown X is multiplied by known A on the left and by B on the right, with noncommutativity implying $AX \neq XA$
C_i, D_i, E_i, F_i, G_i	Known quaternionic matrices, with the dimensions suitably matched, appearing in the main Sylvester system defined in Subsection 2.3
X, Y, Z, W	Unknown quaternionic matrices, with the dimensions matched to the above, to be solved for in the main Sylvester-type system presented in Subsection 2.3

2.3. Problem statement and main contributions

We focus on solving the following system of quaternionic matrix equations, which constitutes a two-sided Sylvester-type problem, defined as

$$\begin{aligned} &\text{(Set A) } C_4X = G_4, XD_4 = G_5, C_5Y = G_6, YD_5 = G_7, C_6Z = G_8, ZD_6 = G_9, C_7W = G_{10}, WD_7 = G_{11}, \\ &\text{(Set B) } C_1XD_1 + E_1YF_1 = G_1, C_2YD_2 + E_2ZF_2 = G_2, C_3ZD_3 + E_3WF_3 = G_3, \end{aligned} \tag{2.4}$$

where C_i, D_i, E_i, F_i, and G_i are the known quaternionic matrices, while X, Y, Z, and W are the unknown quaternionic matrices to be determined. The equations stated in (2.4) are generally noncommutative, reflecting the intrinsic noncommutative nature of quaternion multiplication.

In the system formulated in (2.4), the first group of equations (labeled "Set A") imposes linear relationships involving C_4, D_4, C_5, and D_5 (as well as similar matrices), together with the unknowns X, Y, Z, and W. These equations must be solved simultaneously while preserving quaternionic noncommutativity. The second set of equations presented in (2.4) (labeled "Set B") is more intricate, as each equation involves a sum of quaternionic matrix products, thereby coupling the unknown matrices X, Y, Z, and W. As a result, solving one equation can influence the others, complicating the pursuit of a closed-form solution.

As discussed in Subsection 1.2, a robotic manipulator with multiple joints may require the concurrent determination of several incremental rotations. In a four-link scenario, for instance, the matrices C_4, D_4, C_5, and D_5 (and so on) encode constraints or feasible ranges for each joint, while unknowns X, Y, Z, W correspond to the quaternionic rotations to be applied in tandem.

Accordingly, Set A, stated in (2.4), can represent the local feasibility conditions (such as the permissible rotations at each joint), while Set B enforces global consistency (such as the final orientation of the end effector as a function of all joint rotations). Likewise, tasks such as multi-channel signal processing or color image registration may involve coupling multiple quaternionic blocks to impose global alignment or filtering constraints. Therefore, such systems of equations naturally emerge in real-world scenarios involving multiple noncommutative transformations, highlighting the necessity of a rigorous analysis of their solvability and associated solution methods.

Our main objective are the following:

(i) To establish the necessary and sufficient criteria under which the expression stated in (2.4) admits a solution over $\mathbb{H}$.

(ii) To propose an explicit procedure that systematically produces the general solution (X, Y, Z, W) whenever those criteria are satisfied.

By addressing both objectives, we develop a comprehensive theoretical and computational framework that not only expands the foundational results on two-sided Sylvester equations in $\mathbb{H}$ but also demonstrates tangible benefits for real-world scenarios.

Section 3 presents the fundamental lemmas on quaternionic matrices, which establish the solvability conditions and underlie the proof of the main theorem, culminating in the general solution.

3. Theoretical framework

3.1. Fundamental lemmas

The first lemma, adapted from [30], presents the rank identities for quaternionic matrices. These identities are essential for proving the main theorem and are applied repeatedly in the derivation.

Lemma 3.1. *Let $K \in \mathbb{H}^{m\times n}$, $P \in \mathbb{H}^{m\times t}$, and $Q \in \mathbb{H}^{l\times n}$. The rank equalities are given by*

$$\operatorname{rank}\begin{bmatrix} K \\ Q \end{bmatrix} - \operatorname{rank}(QL_K) = \operatorname{rank}(K), \operatorname{rank}\begin{bmatrix} K & P \end{bmatrix} - \operatorname{rank}(R_P K) = \operatorname{rank}(P), \operatorname{rank}\begin{bmatrix} K & P \\ Q & 0 \end{bmatrix} - \operatorname{rank}(P) - \operatorname{rank}(Q) = \operatorname{rank}(R_P K L_Q).$$

Next, we present a lemma obtained from [31] that states the necessary and sufficient conditions for the existence of a solution to the quaternionic matrix system established as

$$A_1 X_1 = C_1, X_1 B_1 = C_2, \tag{3.1}$$

where $A_1 \in \mathbb{H}^{m_1\times n_1}$, $B_1 \in \mathbb{H}^{r_1\times s_1}$, $C_1 \in \mathbb{H}^{m_1\times r_1}$, and $C_2 \in \mathbb{H}^{n_1\times s_1}$ are the given matrices, while $X_1 \in \mathbb{H}^{n_1\times r_1}$ is unknown.

Lemma 3.2. *The system defined in* (3.1) *is consistent if and only if following conditions hold:* $R_{A_1}C_1 = 0$, $C_2L_{B_1} = 0$, *and* $A_1C_2 = C_1B_1$. *Then the solution to the system presented in* (3.1) *is stated as* $X_1 = A_1^{\dagger}C_1 + L_{A_1}C_2B_1^{\dagger} + L_{A_1}U_1R_{B_1}$, *where* U_1 *is a free matrix over* $\mathbb{H}$ *of appropriate dimensions, which introduces degrees of freedom in the solution.*

Now, we introduce a lemma obtained from [32], which plays a fundamental role in the proof of the main theorem of this article. This lemma addresses the solvability conditions for the following quaternionic matrix system formulated as

$$A_1U + VB_1 + C_3WD_3 + C_4ZD_4 = E_1, \tag{3.2}$$

where $A_1, B_1, C_3, D_3, C_4, D_4$, and E_1 are known, whereas U, V, W, and Z are the unknowns to be determined. The lemma presents the equivalent conditions for the existence of solutions to the equation shown in (3.2).

Lemma 3.3. *Let* $A_1, B_1, C_3, D_3, C_4, D_4$, *and* E_1 *be the given quaternionic matrices. Define the intermediate matrices given by* $A = R_{A_1}C_3, B = D_3L_{B_1}, C = R_{A_1}C_4, D = D_4L_{B_1}, E = R_{A_1}E_1L_{B_1}, F = R_AC, G = DL_B$, *and* $H = CL_F$. *Then, the following statements are equivalent:*

(i) The system stated in (3.2) *admits a solution.*

(ii) The conditions formulated as $R_FR_AE = 0, EL_BL_G = 0, R_AEL_D = 0$, *and* $R_CEL_B = 0$ *are satisfied.*

(iii) The following rank conditions hold, which are presented as

$$\operatorname{rank}\begin{bmatrix} E_1 & C_4 & C_3 & A_1 \\ B_1 & 0 & 0 & 0 \end{bmatrix} = \operatorname{rank}(B_1) + \operatorname{rank}[C_4C_3A_1], \operatorname{rank}\begin{bmatrix} E_1 & A_1 \\ D_3 & 0 \\ D_4 & 0 \\ B_1 & 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} D_3 \\ D_4 \\ B_1 \end{bmatrix} + \operatorname{rank}(A_1),$$

$$\operatorname{rank}\begin{bmatrix} E_1 & C_3 & A_1 \\ D_4 & 0 & 0 \\ B_1 & 0 & 0 \end{bmatrix} = \operatorname{rank}[A_1C_3] + \operatorname{rank}\begin{bmatrix} D_4 \\ B_1 \end{bmatrix}, \operatorname{rank}\begin{bmatrix} E_1 & C_4 & A_1 \\ D_3 & 0 & 0 \\ B_1 & 0 & 0 \end{bmatrix} = \operatorname{rank}[A_1C_4] + \operatorname{rank}\begin{bmatrix} D_3 \\ B_1 \end{bmatrix}.$$

Under the conditions above, the general solution to the system presented in (3.2) *is given by*

$$\begin{aligned} U &= A_1^{\dagger}(E_1 - C_3WD_3 - C_4ZD_4) - A_1^{\dagger}S_7B_1 + L_{A_1}S_6, \\ V &= R_{A_1}(E_1 - C_3WD_3 - C_4ZD_4)B_1^{\dagger} + A_1A_1^{\dagger}S_7 + S_8R_{B_1}, \\ W &= A^{\dagger}EB^{\dagger} - A^{\dagger}CF^{\dagger}EB^{\dagger} - A^{\dagger}HC^{\dagger}EG^{\dagger}DB^{\dagger} - A^{\dagger}HS_2R_GDB^{\dagger} + L_AS_4 + S_5R_B, \\ Z &= F^{\dagger}ED^{\dagger} + H^{\dagger}HC^{\dagger}EG^{\dagger} + L_FL_HS_1 + L_FS_2R_G + S_3R_D, \end{aligned}$$

where $S_1, \ldots, S_8$ *are free quaternionic parameters (with appropriate dimensions) representing the degrees of freedom in the solution. These matrices can be selected arbitrarily, and their dimensions depend on the particular structure of the system.*

3.2. Main theorem

We now proceed to state and prove the main theorem, which yields the general solution to the system of quaternionic matrix equations defined in (2.4).

Theorem 3.1. *Let the matrices $C_1, \ldots, C_7, D_1, \ldots, D_7, E_1, E_2, E_3, F_1, F_2, F_3, G_1, \ldots, G_{11}$ be given, all with compatible dimensions over $\mathbb{H}$. Assign the intermediate matrices and equations formulated as*

$$P_1 = C_1L_{C_4}, P_2 = E_1L_{C_5}, P_3 = C_2L_{C_5}, P_4 = E_2L_{C_6}, P_5 = C_3L_{C_6}, P_6 = E_3L_{C_7},$$
$$Q_1 = R_{D_4}D_1, Q_2 = R_{D_5}F_1, Q_3 = R_{D_5}D_2, Q_4 = R_{D_6}F_2, Q_5 = R_{D_6}D_3, Q_6 = R_{D_7}F_3,$$
$$\widehat{X}_{01} = C_4^\dagger G_4 + L_{C_4}G_5D_4^\dagger, \widehat{Y}_{01} = C_5^\dagger G_6 + L_{C_5}G_7D_5^\dagger, \widehat{Z}_{01} = C_6^\dagger G_8 + L_{C_6}G_9D_6^\dagger, \widehat{W}_{01} = C_7^\dagger G_{10} + L_{C_7}G_{11}D_7^\dagger,$$
$$R_1 = G_1 - C_1\widehat{X}_{01}D_1 - E_1\widehat{Y}_{01}F_1, R_2 = G_2 - C_2\widehat{Y}_{01}D_2 - E_2\widehat{Z}_{01}F_2, R_3 = G_3 - C_3\widehat{Z}_{01}D_3 - E_3\widehat{W}_{01}F_3,$$
$$M_1 = R_{P_1}P_2, M_2 = R_{P_3}P_4, M_3 = R_{P_5}P_6, N_1 = Q_2L_{Q_1}, N_2 = Q_4L_{Q_3}, N_3 = Q_6L_{Q_5},$$
$$S_1 = P_2L_{M_1}, S_2 = P_4L_{M_2}, S_3 = P_6L_{M_3}, A_{11} = \left[-L_{P_3}L_{M_1}L_{S_1}\right], B_{11} = \left[-R_{Q_3}R_{Q_2}\right], C_{11} = L_{M_1},$$
$$D_{11} = R_{N_1}, F_{11} = P_3^\dagger S_2, G_{11} = R_{N_1}Q_4Q_3^\dagger, Y_{01} = M_1^\dagger R_1Q_2^\dagger + S_1^\dagger S_1P_2^\dagger R_1N_1^\dagger,$$
$$Y_{02} = P_3^\dagger R_2Q_3^\dagger - P_3^\dagger P_4M_2^\dagger R_2Q_3^\dagger - P_3^\dagger S_2P_4^\dagger R_2N_2^\dagger Q_4Q_3^\dagger, S_{11} = C_{33}L_{M_{11}}, A_{22} = \left[-L_{P_5}L_{M_2}L_{S_2}\right], B_{22} = \left[-R_{Q_5}R_{Q_4}\right],$$
$$E_{11} = Y_{02} - Y_{01}, A_{33} = R_{A_{11}}C_{11}, B_{33} = D_{11}L_{B_{11}}, C_{33} = R_{A_{11}}F_{11}, D_{33} = G_{11}L_{B_{11}},$$
$$E_{22} = R_{A_{33}}E_{11}R_{A_{33}}, E_{33} = R_{A_{11}}E_{11}L_{B_{11}}, M_{11} = R_{A_{33}}C_{33}, N_{11} = D_{33}L_{B_{33}},$$
$$C_{22} = L_{M_2}, D_{22} = R_{N_2}, F_{22} = P_5^\dagger S_3, G_{22} = R_{N_3}Q_6Q_5^\dagger, A_{44} = R_{A_{22}}C_{22}, B_{44} = D_{22}L_{B_{22}}, C_{44} = R_{A_{22}}F_{22}, D_{44} = G_{22}L_{B_{22}},$$
$$E_{44} = R_{A_{22}}E_{22}L_{B_{22}}, M_{22} = R_{A_{44}}C_{44}, N_{22} = D_{44}L_{B_{44}}, S_{22} = C_{44}L_{M_{22}}, A_{55} = \left[-L_{A_{44}}L_{M_{11}}L_{S_{11}}\right], B_{55} = \left[-R_{B_{44}}R_{D_{33}}\right],$$
$$W_{01} = M_3^\dagger R_3Q_6^\dagger + S_3^\dagger S_3P_6^\dagger R_3N_3^\dagger, X_{01} = P_1^\dagger R_1Q_1^\dagger - P_1^\dagger P_2M_1^\dagger R_1Q_1^\dagger - P_1^\dagger S_1P_2^\dagger R_1N_1^\dagger Q_2Q_1^\dagger,$$
$$C_{55} = L_{M_{11}}, D_{55} = R_{N_{11}}, F_{55} = A_{44}^\dagger S_{22}, G_{55} = R_{N_{22}}D_{44}Q_{44}^\dagger, A_{66} = R_{A_{55}}C_{55}, B_{66} = D_{55}L_{B_{55}}, C_{66} = R_{A_{55}}F_{55}, D_{66} = G_{55}L_{B_{55}},$$
$$S_3 = A_5L_{M_3}, E_{66} = R_{A_{55}}E_{55}L_{B_{55}}, M_{33} = R_{A_{66}}C_{66}, N_{33} = D_{66}L_{B_{66}},$$
$$S_{33} = C_{66}L_{M_{33}}, E_{22} = Z_{02} - Z_{01}, Z_{02} = P_5^\dagger R_3Q_5^\dagger - P_5^\dagger P_6M_3^\dagger R_3Q_5^\dagger - P_5^\dagger S_3P_6^\dagger R_3N_3^\dagger Q_6Q_5^\dagger, Z_{01} = M_2^\dagger R_2Q_4^\dagger + S_2^\dagger S_2P_4^\dagger R_2N_2^\dagger.$$

The following conditions are equivalent for the consistency of the system:

(i) The system given in (2.4) *is consistent.*

(ii) The following equalities hold, which are defined as:

$$R_{C_4}G_4 = 0, R_{C_5}G_6 = 0, R_{C_6}G_8 = 0, R_{C_7}G_{10} = 0, G_5L_{D_4} = 0, \tag{3.3}$$
$$G_7L_{D_5} = 0, G_9L_{D_6} = 0, G_{11}L_{D_7} = 0, R_{P_1}R_1L_{Q_2} = 0, C_4G_5 = G_4D_4,$$
$$C_5G_7 = G_6D_5, C_6G_9 = G_8D_6, C_7G_{11} = G_{10}D_7, R_{P_2}R_1L_{Q_1} = 0, \tag{3.4}$$
$$R_{M_1}R_{P_1}R_1 = 0, R_1L_{Q_1}L_{N_1} = 0, R_{P_3}R_2L_{Q_4} = 0, R_{P_4}R_2L_{Q_3} = 0, \tag{3.5}$$
$$R_{M_2}R_{P_3}R_2 = 0, R_2L_{Q_3}L_{N_2} = 0, R_{P_5}R_3L_{Q_6} = 0, R_{P_6}R_3L_{Q_5} = 0, R_{M_3}R_{P_5}R_3 = 0, R_3L_{Q_5}L_{N_3} = 0,$$
$$R_{A_{33}}E_{33}L_{D_{33}} = 0, R_{C_{33}}E_{33}L_{B_{33}} = 0, R_{M_{11}}R_{A_{33}}E_{33} = 0, E_{33}L_{B_{33}}L_{N_{11}} = 0, \tag{3.6}$$
$$R_{A_{44}}E_{44}L_{D_{44}} = 0, R_{C_{44}}E_{44}L_{B_{44}} = 0, R_{M_{22}}R_{A_{44}}E_{44} = 0, E_{44}L_{B_{44}}L_{N_{22}} = 0, \tag{3.7}$$
$$R_{A_{66}}E_{66}L_{D_{66}} = 0, R_{C_{66}}E_{66}L_{B_{66}} = 0, R_{M_{33}}R_{A_{66}}E_{66} = 0, E_{66}L_{B_{66}}L_{N_{33}} = 0. \tag{3.8}$$

(iii) The following rank equalities hold, which are established as:

$$\operatorname{rank}\begin{bmatrix} C_4 & G_4 \end{bmatrix} = \operatorname{rank}(C_4), \operatorname{rank}\begin{bmatrix} C_5 & G_6 \end{bmatrix} = \operatorname{rank}(C_5),$$
$$\operatorname{rank}\begin{bmatrix} G_8 & C_6 \end{bmatrix} = \operatorname{rank}(C_6), \operatorname{rank}\begin{bmatrix} C_7 & G_{10} \end{bmatrix} = \operatorname{rank}(C_7), \tag{3.9}$$
$$\operatorname{rank}\begin{bmatrix} D_4 \\ G_5 \end{bmatrix} = \operatorname{rank}(G_5), \operatorname{rank}\begin{bmatrix} D_5 \\ G_7 \end{bmatrix} = \operatorname{rank}(D_5), \operatorname{rank}\begin{bmatrix} D_6 \\ G_9 \end{bmatrix} = \operatorname{rank}(G_9), \operatorname{rank}\begin{bmatrix} D_7 \\ G_{11} \end{bmatrix} = \operatorname{rank}(D_7),$$
$$\operatorname{rank}\begin{bmatrix} C_4 & G_5 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} G_4 & D_4 \end{bmatrix}, \operatorname{rank}\begin{bmatrix} C_5 & G_7 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} G_6 & D_5 \end{bmatrix},$$

$$\text{rank}\begin{bmatrix} C_6 & G_9 \end{bmatrix} = \text{rank}\begin{bmatrix} G_8 & D_6 \end{bmatrix}, \text{rank}\begin{bmatrix} C_7 & G_{11} \end{bmatrix} = \text{rank}\begin{bmatrix} G_{10} & D_7 \end{bmatrix}, \tag{3.10}$$

$$\text{rank}\begin{bmatrix} G_1 & E_1 & C_1 \\ G_6F_1 & C_5 & 0 \\ C_4G_4 & 0 & C_4 \end{bmatrix} = \text{rank}\begin{bmatrix} E_1 & C_1 \\ C_5 & 0 \\ 0 & C_4 \end{bmatrix}, \tag{3.11}$$

$$\text{rank}\begin{bmatrix} G_1 & C_1G_5 & E_1G_7 \\ F_1 & 0 & D_5 \\ D_1 & D_4 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} F_1 & D_5 & 0 \\ D_1 & 0 & D_4 \end{bmatrix}, \text{rank}\begin{bmatrix} G_2 & C_2 & E_2G_9 \\ F_2 & 0 & D_6 \\ G_6D_2 & C_5 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} C_2 \\ C_5 \end{bmatrix} + \text{rank}[D_6F_2],$$

$$\text{rank}\begin{bmatrix} G_2 & E_2 & C_2G_7 \\ D_2 & 0 & D_2 \\ G_8F_2 & C_6 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} E_2 \\ C_6 \end{bmatrix} + \text{rank}[D_5D_2], \text{rank}\begin{bmatrix} G_2 & E_2 & C_2 \\ G_6D_2 & 0 & C_5 \\ G_8F_2 & C_6 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} E_2 & C_2 \\ C_5 & 0 \\ 0 & C_6 \end{bmatrix},$$

$$\text{rank}\begin{bmatrix} G_2 & E_2G_9 & C_2G_7 \\ F_2 & D_6 & 0 \\ D_2 & 0 & D_5 \end{bmatrix} = \text{rank}\begin{bmatrix} F_2 & D_6 & 0 \\ D_2 & 0 & D_5 \end{bmatrix}, \text{rank}\begin{bmatrix} G_3 & C_3 & E_3G_{11} \\ F_3 & 0 & D_7 \\ G_8D_3 & C_6 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} C_3 \\ C_6 \end{bmatrix} + \text{rank}[F_3D_7],$$

$$\text{rank}\begin{bmatrix} G_3 & E_3 & C_3G_9 \\ D_3 & 0 & D_6 \\ G_{10}F_3 & C_7 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} E_3 \\ C_7 \end{bmatrix} + \text{rank}\begin{bmatrix} D_3 & D_6 \end{bmatrix}, \text{rank}\begin{bmatrix} G_3 & E_3 & C_3 \\ 0 & C_7 & 0 \\ 0 & 0 & C_6 \end{bmatrix} = \text{rank}\begin{bmatrix} E_3 & C_3 \\ C_7 & 0 \\ 0 & C_6 \end{bmatrix},$$

$$\text{rank}\begin{bmatrix} G_3 & E_3G_{11} & C_3G_9 \\ F_3 & D_7 & 0 \\ D_3 & 0 & D_6 \end{bmatrix} = \text{rank}[D_7F_3] + \text{rank}[D_3D_6], \text{rank}\begin{bmatrix} 0 & F_2 & 0 & 0 & D_6 & 0 \\ 0 & D_2 & F_1 & 0 & 0 & D_5 \\ E_1 & 0 & -G_1 & C_1 & 0 & 0 \\ C_2 & G_2 & 0 & 0 & E_2G_9 & C_2G_7 \\ C_5 & 0 & -G_6F_1 & 0 & 0 & 0 \\ 0 & 0 & -G_4D_1 & C_4 & 0 & 0 \end{bmatrix} =$$

$$\text{rank}\begin{bmatrix} C_2 & 0 \\ E_1 & C_1 \\ C_5 & 0 \\ 0 & C_4 \end{bmatrix} + \text{rank}\begin{bmatrix} F_2 & 0 & D_6 & 0 \\ D_2 & F_1 & 0 & D_5 \end{bmatrix}, \tag{3.12}$$

$$\text{rank}\begin{bmatrix} 0 & 0 & F_1 & D_2 & D_5 & 0 \\ E_2 & -C_2 & 0 & -G_2 & -C_2G_7 & 0 \\ 0 & E_1 & -G_1 & 0 & 0 & -C_1G_5 \\ 0 & 0 & D_1 & 0 & 0 & D_4 \\ C_6 & 0 & 0 & -G_8F_2 & 0 & 0 \\ 0 & C_5 & -G_6F_1 & 0 & 0 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} -C_2 & E_2 \\ E_1 & 0 \\ C_5 & 0 \\ 0 & C_6 \end{bmatrix} + \text{rank}\begin{bmatrix} F_1 & D_2 & 0 & D_5 \\ D_1 & 0 & D_4 & 0 \end{bmatrix}, \tag{3.13}$$

$$\text{rank}\begin{bmatrix} 0 & 0 & D_2 & F_1 & 0 & D_5 \\ 0 & E_1 & 0 & -G_1 & -C_1 & 0 \\ E_2 & C_2 & G_2 & 0 & 0 & C_2G_7 \\ 0 & 0 & D_1 & 0 & 0 & D_4 \\ C_6 & 0 & G_8F_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -G_4D_1 & C_4 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} 0 & E_1 & C_1 \\ E_2 & C_2 & 0 \\ C_6 & 0 & 0 \\ 0 & C_5 & 0 \\ 0 & 0 & C_4 \end{bmatrix} + \text{rank}\begin{bmatrix} D_2 & D_1 & D_5 \end{bmatrix},$$

$$\text{rank}\begin{bmatrix} 0 & F_2 & 0 & F_2 & D_6 & 0 & 0 & 0 \\ 0 & 0 & -F_1 & 0 & 0 & D_5 & 0 & 0 \\ C_2 & -G_2 & 0 & G_2 & E_2G_4 & 0 & 0 & 0 \\ E_1 & 0 & -G_1 & 0 & -E_1G_7 & E_1G_7 & 0 & 0 \\ 0 & D_2 & 0 & 0 & 0 & 0 & D_5 & -C_1G_5 \\ 0 & 0 & D_1 & 0 & 0 & 0 & 0 & D_4 \\ C_5 & -G_6D_2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} F_2 & 0 & F_2 & D_6 & 0 & 0 & 0 \\ 0 & F_1 & D_2 & 0 & D_5 & 0 & 0 \\ D_2 & 0 & 0 & 0 & 0 & D_5 & 0 \\ D_2 & 0 & 0 & 0 & 0 & D_5 & 0 \\ 0 & D_1 & 0 & 0 & 0 & 0 & D_4 \end{bmatrix} + \text{rank}\begin{bmatrix} C_2 \\ E_1 \\ C_5 \end{bmatrix}, \tag{3.14}$$

$$\operatorname{rank}\begin{bmatrix} 0 & F_3 & 0 & 0 & D_7 & 0 \\ 0 & D_3 & -F_2 & 0 & 0 & D_6 \\ E_2 & 0 & -G_2 & C_2 & 0 & 0 \\ C_3 & -G_3 & 0 & 0 & -E_3G_{11} & C_3G_9 \\ C_6 & 0 & -G_8F_2 & 0 & 0 & 0 \\ 0 & 0 & -G_6D_2 & C_5 & 0 & 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} C_3 & 0 \\ E_2 & C_2 \\ C_6 & 0 \\ 0 & C_5 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} F_3 & 0 & D_7 & 0 \\ D_3 & F_2 & 0 & D_6 \end{bmatrix}, \tag{3.15}$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & F_2 & D_3 & D_6 & 0 \\ E_3 & C_3 & 0 & G_3 & C_3G_9 & 0 \\ 0 & E_2 & -G_2 & 0 & 0 & -C_2G_7 \\ 0 & 0 & D_2 & 0 & 0 & D_5 \\ C_7 & 0 & 0 & G_{10}F_3 & 0 & 0 \\ 0 & C_6 & -G_8F_2 & 0 & 0 & 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} C_3 & E_3 \\ E_2 & 0 \\ C_6 & 0 \\ 0 & C_7 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} F_2 & D_3 & D_6 & 0 \\ D_2 & 0 & 0 & D_5 \end{bmatrix}, \tag{3.16}$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & D_3 & F_2 & 0 & D_6 \\ 0 & E_2 & 0 & G_2 & C_2 & E_2G_9 \\ E_3 & C_3 & -G_3 & 0 & 0 & 0 \\ C_7 & 0 & -G_{10}F_3 & 0 & 0 & 0 \\ 0 & C_6 & -G_8D_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_6D_2 & C_5 & 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} E_2 & 0 & C_2 \\ E_2 & E_2 & 0 \\ C_6 & 0 & 0 \\ 0 & C_7 & 0 \\ 0 & 0 & C_5 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} D_3 & F_2 & D_6 \end{bmatrix},$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & F_3 & D_7 & 0 \\ 0 & F_1 & D_3 & 0 & D_6 \\ C_3 & 0 & G_3 & 0 & C_3G_9 \\ E_2 & -G_1 & 0 & 0 & -C_2G_7 \\ 0 & D_2 & 0 & 0 & D_5 \\ C_6 & -G_8F_2 & 0 & 0 & 0 \end{bmatrix} = \operatorname{rank}\begin{bmatrix} 0 & F_3 & D_7 & 0 & 0 \\ F_2 & D_3 & 0 & D_6 & 0 \\ D_2 & 0 & 0 & 0 & D_5 \\ D_2 & 0 & 0 & 0 & D_5 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} C_3 \\ E_2 \\ C_6 \end{bmatrix}, \tag{3.17}$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & F_3 & 0 & 0 & 0 & 0 & D_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & -F_2 & F_2 & 0 & 0 & 0 & D_6 & 0 & 0 \\ 0 & 0 & D_3 & F_2 & 0 & 0 & 0 & 0 & 0 & D_6 & 0 \\ 0 & 0 & 0 & 0 & D_2 & 0 & 0 & 0 & 0 & 0 & D_5 \\ C_2 & E_2 & 0 & 0 & G_2 & 0 & 0 & 0 & E_2G_9 & E_2G_9 & C_2G_7 \\ 0 & C_3 & G_3 & 0 & 0 & 0 & 0 & E_3G_{11} & 0 & 0 & 0 \\ E_1 & 0 & 0 & 0 & 0 & -G_1 & C_1 & 0 & 0 & 0 & 0 \\ C_5 & 0 & 0 & 0 & 0 & G_6F_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_6 & G_8D_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & G_4D_1 & C_4 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} P_5 & 0 & 0 \\ P_4 & P_3 & 0 \\ 0 & P_2 & P_1 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} Q_6 & 0 & 0 & 0 \\ 0 & Q_4 & Q_4 & 0 \\ Q_5 & Q_4 & 0 & 0 \\ 0 & 0 & Q_3 & Q_2 \end{bmatrix}, \tag{3.18}$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & 0 & 0 & F_2 & 0 & 0 & -F_2 & D_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & F_1 & 0 & -F_1 & 0 & 0 & 0 & D_5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & D_2 & F_1 & 0 & 0 & 0 & 0 & D_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & D_3 & -F_2 & 0 & 0 & 0 & D_6 & 0 \\ E_3 & C_3 & 0 & 0 & 0 & 0 & G_3 & 0 & C_3G_9 & 0 & 0 & C_3G_9 & 0 \\ 0 & -E_2 & C_2 & 0 & G_2 & 0 & 0 & 0 & 0 & 0 & C_2G_7 & 0 & 0 \\ 0 & 0 & 0 & D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_4 \\ 0 & 0 & E_1 & G_1 & 0 & 0 & 0 & 0 & 0 & E_1G_7 & E_1G_7 & 0 & C_1G_5 \\ C_7 & 0 & 0 & 0 & 0 & 0 & G_{10}F_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_6 & 0 & 0 & G_8F_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_5 & 0 & 0 & G_6D_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} F_1 & 0 & -F_1 & 0 & 0 & D_5 & 0 & 0 & 0 & 0 \\ 0 & D_2 & F_1 & 0 & 0 & 0 & D_5 & 0 & 0 & 0 \\ 0 & F_2 & 0 & D_3 & 0 & 0 & 0 & D_6 & 0 & 0 \\ 0 & -F_2 & 0 & 0 & F_2 & 0 & 0 & 0 & D_6 & 0 \\ D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & D_4 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} E_3 & C_3 & 0 \\ 0 & -E_2 & C_2 \\ 0 & 0 & E_1 \\ C_7 & 0 & 0 \\ 0 & C_6 & 0 \\ 0 & 0 & C_5 \end{bmatrix}, \tag{3.19}$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & F_2 & 0 & 0 & D_6 & 0 & 0 \\ 0 & 0 & 0 & D_3 & F_2 & 0 & 0 & 0 & 0 & D_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & D_2 & F_1 & 0 & 0 & 0 & D_5 \\ 0 & E_1 & 0 & 0 & 0 & 0 & G_1 & C_1 & 0 & 0 & -C_2G_7 \\ 0 & C_2 & E_2 & 0 & 0 & -G_2 & 0 & 0 & -E_2G_9 & 0 & 0 \\ E_3 & 0 & C_3 & G_3 & 0 & 0 & 0 & 0 & 0 & C_3G_9 & 0 \\ C_7 & 0 & 0 & G_{10}F_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_5 & 0 & 0 & 0 & 0 & G_6F_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_6 & G_8D_3 & G_8F_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & G_4D_1 & C_4 & 0 & 0 & 0 \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} 0 & -F_2 & F_2 & 0 & D_6 & 0 & 0 \\ D_3 & F_2 & 0 & 0 & 0 & D_6 & 0 \\ 0 & 0 & D_2 & F_1 & 0 & 0 & D_5 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} E_3 & C_3 & 0 \\ 0 & E_2 & C_2 \\ 0 & 0 & E_1 \\ C_7 & 0 & 0 \\ 0 & C_6 & 0 \\ 0 & 0 & C_5 \end{bmatrix},$$

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & 0 & 0 & F_3 & D_7 & 0 & 0 & 0 \\ 0 & 0 & F_1 & D_2 & 0 & 0 & D_5 & 0 & 0 \\ 0 & 0 & 0 & F_2 & D_3 & 0 & 0 & D_6 & 0 \\ C_3 & 0 & 0 & 0 & G_3 & E_3G_{11} & 0 & -C_3G_9 & 0 \\ E_2 & -C_2 & 0 & G_2 & 0 & C_2G_7 & 0 & 0 & 0 \\ 0 & E_1 & G_1 & 0 & 0 & 0 & 0 & 0 & C_1G_5 \\ 0 & 0 & G_1 & 0 & 0 & 0 & 0 & 0 & D_4 \\ C_6 & 0 & 0 & 0 & G_8F_2 & 0 & 0 & 0 & 0 \\ 0 & C_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} F_1 & 0 & -F_1 & 0 & D_5 & 0 & 0 & 0 \\ 0 & D_2 & F_1 & 0 & 0 & D_5 & 0 & 0 \\ 0 & F_2 & 0 & D_3 & 0 & 0 & D_6 & 0 \\ D_1 & 0 & 0 & D_3 & 0 & 0 & 0 & D_4 \end{bmatrix} + \operatorname{rank}\begin{bmatrix} E_2 & C_2 \\ C_3 & 0 \\ 0 & E_1 \\ C_6 & 0 \\ 0 & C_5 \end{bmatrix}. \tag{3.20}$$

Under Conditions (i)–(iii), the general solution to the system given in (2.4) *is stated as*

$$X = C_4^{\dagger}G_4 + L_{C_4}G_5D_4^{\dagger} + L_{C_4}X_1R_{D_4}, Y = C_5^{\dagger}G_6 + L_{C_5}G_7D_5^{\dagger} + L_{C_5}Y_1R_{D_5}, \tag{3.21}$$

$$X_1 = P_1^{\dagger}R_1Q_1^{\dagger} - P_1^{\dagger}P_2M_1^{\dagger}R_1Q_1^{\dagger} - P_1^{\dagger}S_1P_2^{\dagger}R_1N_1^{\dagger}Q_2Q_1^{\dagger} - P_1^{\dagger}S_1P_2^{\dagger}R_1N_1^{\dagger}Q_2Q_1^{\dagger} - P_1^{\dagger}S_1T_1R_{N_1}Q_2Q_1^{\dagger} + L_{P_1}T_2 + T_3R_{Q_1},$$

$$Y_1 = M_1^{\dagger}R_1Q_2^{\dagger} + S_1^{\dagger}S_1P_2^{\dagger}R_1N_1^{\dagger} + L_{M_1}L_{S_1}T_4 + L_{M_1}T_1R_{N_1} + T_6R_{Q_2},$$

$$Y_1 = P_3^{\dagger}R_2Q_3^{\dagger} - P_3^{\dagger}P_4M_2^{\dagger}R_2Q_3^{\dagger} - P_3^{\dagger}S_2P_4^{\dagger}R_2N_2^{\dagger}Q_4Q_3^{\dagger} - P_3^{\dagger}S_2P_4^{\dagger}R_2N_2^{\dagger}Q_4Q_3^{\dagger} - P_3^{\dagger}S_2U_1R_{N_2}Q_4Q_3^{\dagger} + L_{P_3}U_2 + U_3R_{Q_3},$$

$$Z = C_6^{\dagger}G_8 + L_{C_6}G_9D_6^{\dagger} + L_{C_6}Z_1R_{D_6}, \tag{3.22}$$

$$Z_1 = M_2^{\dagger}R_2Q_4^{\dagger} + S_2^{\dagger}S_2P_4^{\dagger}R_2N_2^{\dagger} + L_{M_2}L_{S_2}U_4 + L_{M_2}U_1R_{N_2} + U_6R_{Q_4},$$

$$W = C_7^{\dagger}G_{10} + L_{C_7}G_{11}D_7^{\dagger} + L_{C_7}W_1R_{D_7}, \tag{3.23}$$

$$W_1 = M_3^{\dagger}R_3Q_6^{\dagger} + S_3^{\dagger}S_3P_6^{\dagger}R_3N_3^{\dagger} + L_{M_3}L_{S_3}V_4 + L_{M_3}V_1R_{N_3} + V_6R_{Q_6},$$

where

$$\begin{bmatrix} U_2T_4 \end{bmatrix}^{\top} = A_{11}^{\dagger}(E_{11} - C_{11}T_1D_{11} - F_{11}U_1G_{11}) - T_7B_{11} + L_{A_{11}}T_8, \tag{3.24}$$

$$\begin{bmatrix} U_3 & T_6 \end{bmatrix} = R_{A_{11}}(E_{11} - C_{11}T_1D_{11} - F_{11}U_1G_{11})B_{11}^{\dagger} + A_{11}U_7 + U_8R_{B_{11}},$$

with

$$T_1 = A_{33}^{\dagger}E_{33}B_{33}^{\dagger} - A_{33}^{\dagger}C_{33}M_{11}^{\dagger}E_{33}B_{33}^{\dagger} - A_{33}^{\dagger}S_{11}C_{33}^{\dagger}E_{33}N_{11}^{\dagger}D_{33}B_{33}^{\dagger} - A_{33}^{\dagger}S_{11}V_7R_{N_{11}}D_{33}B_{33}^{\dagger} + L_{A_{33}}V_8 + V_9R_{B_{33}},$$
$$U_1 = M_{11}^{\dagger}E_{33}D_{33}^{\dagger} + S_{11}^{\dagger}S_{11}C_{33}^{\dagger}E_{33}N_{11}^{\dagger} + L_{M_{11}}L_{S_{11}}W_2 + L_{M_{11}}V_7R_{N_{11}} + W_4R_{D_{33}}, \tag{3.25}$$

$$\begin{bmatrix} V_2 \\ U_4 \end{bmatrix} = A_{22}^{\dagger}(E_{22} - C_{22}U_1D_{22} - F_{22}V_1G_{22}) - T_9B_{22} + L_{A_{22}}T_{10}, \tag{3.26}$$

$$\begin{bmatrix} V_3 & U_6 \end{bmatrix} = R_{A_{22}}(E_{22} - C_{22}U_1D_{22} - F_{22}V_1G_{22})B_{22}^{\dagger} + A_{22}U_9 + U_{10}R_{B_{22}},$$
$$U_1 = A_{44}^{\dagger}E_{44}B_{44}^{\dagger} - A_{44}^{\dagger}C_{44}M_{22}^{\dagger}E_{44}B_{44}^{\dagger} - A_{44}^{\dagger}S_{22}C_{44}^{\dagger}E_{44}N_{22}^{\dagger}D_{44}B_{44}^{\dagger} - A_{44}^{\dagger}S_{22}W_6R_{N_{22}}D_{44}B_{44}^{\dagger} + L_{A_{44}}V_9 + V_{10}R_{B_{44}}, \tag{3.27}$$

$$V_1 = M_{22}^{\dagger}E_{44}D_{44}^{\dagger} + S_{22}^{\dagger}S_{22}C_{44}^{\dagger}E_{44}N_{22}^{\dagger} + L_{M_{22}}L_{S_{22}}W_5 + L_{M_{22}}W_6R_{N_{22}} + W_7R_{D_{44}}, \tag{3.28}$$

$$\begin{bmatrix} V_9 \\ W_2 \end{bmatrix} = A_{55}^{\dagger}(E_{55} - C_{55}V_7D_{55} - F_{55}W_6G_{55}) - T_{11}B_{55} + L_{A_{55}}T_{12}, \tag{3.29}$$

$$\begin{bmatrix} V_{10} & W_4 \end{bmatrix} = R_{A_{55}}(E_{55} - C_{55}V_7D_{55} - F_{55}W_6G_{55})B_{55}^{\dagger} + A_{55}U_{11} + U_{12}R_{B_{55}},$$
$$V_7 = A_{66}^{\dagger}E_{66}B_{66}^{\dagger} - A_{66}^{\dagger}C_{66}M_{33}^{\dagger}E_{66}B_{66}^{\dagger} - A_{66}^{\dagger}S_{33}C_{66}^{\dagger}E_{66}N_{33}^{\dagger}D_{66}B_{66}^{\dagger} - A_{66}^{\dagger}S_{33}W_9R_{N_{33}}D_{66}B_{66}^{\dagger} + L_{A_{66}}V_{11} + V_{12}R_{B_{66}},$$
$$W_6 = M_{33}^{\dagger}E_{66}D_{66}^{\dagger} + S_{33}^{\dagger}S_{33}C_{66}^{\dagger}E_{66}N_{33}^{\dagger} + L_{M_{33}}L_{S_{33}}W_8 + L_{M_{33}}W_9R_{N_{33}} + W_{10}R_{D_{66}}, \tag{3.30}$$

and $T_2,\ T_3,\ T_7, \ldots, T_{12},\ U_7, \ldots, U_{12},\ V_4,\ V_6,\ V_8,\ V_{11},\ V_{12},\ W_5,\ W_7, \ldots, W_{10}$ *are arbitrary matrices of appropriate dimensions over* $\mathbb{H}$*. These matrices act as free parameters in the general solution, meaning that any specific values assigned to them yield particular solutions of the system.*

Remark 3.1. *It is worth noting that by setting certain matrices to zero for the expressions stated in in* (2.4)*, we can derive simplified systems providing valuable insights into specific cases. For example, by nullifying particular coefficients or coupling terms in the subsystems presented in* (A.1)–(A.3)*, we obtain reduced systems of equations that correspond to simpler quaternionic matrix problems, such as uncoupled Sylvester-type equations or conditions involving reduced rank. In these cases, the necessary and sufficient conditions for the existence of solutions, as well as the general form of these solutions, can be directly derived by applying Theorem 3.1. This theorem addresses the full system and provides a versatile framework for handling reduced cases, enabling us to deduce the results for simplified systems such as those defined in* (A.1)–(A.3)*. Thus, this approach demonstrates the flexibility of Theorem 3.1, offering a unified methodology for solving both the general and reduced forms of quaternionic matrix equations.*

4. Algorithm for solving the quaternionic matrix system

4.1. Step-by-step calculation of X_1 and X

To compute X, we must first calculate X_1, which serves as an essential intermediate step. On the basis of the definitions and results in Theorem 3.1, particularly the equation given in (3.21), we follow the steps below, ensuring that all matrices have either been previously computed or explicitly specified.

We begin by calculating the intermediate matrices $\widehat{X}_{01}$ and $\widehat{Y}_{01}$, which are defined in (3.21) as follows: (i) $\widehat{X}_{01}$ depends on the given matrices C_4, G_4, G_5, and D_4; and (ii) $\widehat{Y}_{01}$ depends on the matrices C_5, G_6, G_7, and D_5, which are also given.

Next, we obtain the matrices P_1 and Q_1, as defined in Theorem 3.1, as (i) P_1 depends on the given matrix C_1 and L_{C_4}, which is a function of the given matrix C_4 and already used in Step 1; and (ii) Q_1 depends on R_{D_4} and D_1, where R_{D_4} is derived from the given matrix D_4, and D_1 is also given. Thus, all the necessary matrices for P_1 and Q_1 have either been given or calculated in the previous steps.

Building on these results, we compute the residual matrix R_1 defined in Theorem 3.1. The dependencies for R_1 are: (i) G_1, which is known; (ii) $\widehat{X}_{01}$, which was calculated in Step 1; (iii) C_1 and D_1, both of which are given; (iv) $\widehat{Y}_{01}$, also obtained in Step 1; and (v) E_1 and F_1, both of which are given. At this point, all the matrices required to compute R_1 have either been given or calculated.

With R_1 already obtained, we now generate M_1, S_1, P_2, Q_2, and N_1, as defined in Theorem 3.1, as (i) P_2 depends on the given matrix E_1 and L_{C_5}, which involves C_5 (already used in Step 1); (ii) Q_2 is calculated using the residuals R_{D_5} and F_1, both of which are given; (iii) M_1 depends on R_{P_1} and P_2, where P_1 was obtained in Step 2; (iv) S_1 depends on P_2 (which was obtained above) and M_1, which we are currently computing; and (v) N_1 depends on Q_2 (which was generated above) and other given matrices as defined in Theorem 3.1. Thus, the matrices required for M_1, S_1, P_2, Q_2, and N_1 have either been given or calculated in previous steps.

Now that we have reached R_1, M_1, S_1, P_2, Q_2, and N_1, we can compute X_1 using the expression stated in (3.21) from: (i) P_1 (calculated in Step 2); (ii) R_1 (reached in Step 3); (iii) Q_1 (generated in Step 2); (iv) P_2 and M_1 (determined in Step 4); (v) S_1 (implemented in Step 4); and (vi) N_1 (presented in Step 4). At this stage, all matrices necessary to compute X_1 have either been given or calculated.

Once X_1 is known, the matrix X is computed using the expression stated in (3.21). Specifically, X depends on: (i) The given matrices C_4, G_4, G_5, and D_4; and (ii) X_1, obtained in Step 5. Thus, all the necessary components to compute X are now available, concluding the calculation.

4.2. Step-by-step computation of Y_1 and Y

The calculation of Y_1 and Y follows similarly to the process outlined for X_1 and X. Since some matrices have already been obtained in previous steps, we build on those results rather than starting from scratch. The solution is based on the expressions defined in (3.21), and we proceed by outlining the necessary calculations.

The matrix $\widehat{Y}_{01}$, defined in (3.21), was generated in Step 1 during the process of obtaining X_1. Recall that $\widehat{Y}_{01}$ depends on the given matrices C_5, G_6, G_7, and D_5. Since this matrix has already been generated, we can proceed to the next step without recalculating $\widehat{Y}_{01}$.

Now, we need to compute P_3 and Q_3, as defined in Theorem 3.1, as follows: (i) P_3 depends on the given matrices C_2 and L_{C_5}, which were involved in the calculation of $\widehat{Y}_{01}$ in Step 1; and (ii) Q_3 depends on R_{D_5}, which is obtained using D_5, and D_2, which are given. Thus, all the matrices required for computing P_3 and Q_3 are either given or have been computed previously.

Before proceeding to obtain the residual matrix R_2, we first need to compute the intermediate matrix $\widehat{Z}_{01}$, as defined in Theorem 3.1. $\widehat{Z}_{01}$ depends on the given matrices C_6, G_8, G_9, and D_6. This matrix is crucial for the subsequent calculation of R_2, so we obtain it now.

Once $\widehat{Z}_{01}$ is obtained in Step 9, we proceed to compute the residual matrix R_2, which is defined in Theorem 3.1. The matrix R_2 depends on: (i) G_2, which is given; (ii) $\widehat{Y}_{01}$, which was calculated earlier in Step 7; (iii) C_2 and D_2, both of which are given; and (iv) $\widehat{Z}_{01}$, which was just generated in Step 9. With all the necessary components implemented or given, we can now obtain the matrix R_2.

We now compute the matrices M_2, S_2, P_4, Q_4, and N_2, as defined in Theorem 3.1, assuming that: (i) P_4 depends on the given matrices E_2 and L_{C_6}, which involve the given matrix C_6 already used in Step 9; (ii) Q_4 is computed using the residuals R_{D_6} and D_2, both of which are given; (iii) M_2 depends on R_{P_3} and P_4, where P_3 was reached in Step 7; (iv) S_2 depends on P_4 (implemented above) and M_2, which is being calculated; and (v) N_2, which depends on Q_4 (obtained above and defined in Theorem 3.1). Thus, the matrices required for M_2, S_2, P_4, Q_4, and N_2 are given or were computed in previous steps.

Now that we have the necessary components, we can compute Y_1 using the expression stated in (3.21). Specifically, Y_1 depends on: (i) M_2, calculated in Step 11; (ii) R_2, obtained in Step 10; (iii) Q_3, generated in Step 11; (iv) S_2, reached in Step 11; and (v) N_2, implemented in Step 11. Since all the required components have been calculated or given, we can now obtain Y_1.

Once Y_1 is known, we can calculate Y using the formulation established in (3.21). The matrix Y depends on: (i) The given matrices C_5, G_6, G_7, and D_5; and (ii) Y_1, obtained in Step 12. With all components available, we can now compute Y, completing this step.

4.3. Discussion of the two forms of Y_1

In the system described in Theorem 3.1, two distinct expressions for the matrix Y_1 arise, as shown in the equations given in (A.4) and (A.5). These two forms are not contradictory but rather serve different purposes depending on the structure of the subsystems to which they correspond.

The first formula for Y_1 emerges from solving the subsystem stated in (A.1), which involves the matrices X and Y in the context of the equation $C_1XD_1 + E_1YF_1 = G_1$. This expression for Y_1 is linked to the first part of the problem, where the unknowns X and Y are coupled via the given matrices C_1, D_1, E_1, F_1, and the residual matrix R_1. The role of Y_1 here is to help decouple X and Y so that X may be obtained more directly from the linear relations.

The second formula for Y_1 appears in the solution to the subsystem presented in (A.2), which couples Y and Z via the equation $C_2YD_2 + E_2ZF_2 = G_2$. Here, Y_1 depends not only on the matrices related to Y but also on those related to Z. The introduction of Z_1 in the system requires Y_1 to be recomputed with respect to the coupled relationship between Y and Z, thus giving rise to a second expression for Y_1. This coupling introduces additional complexity, since the matrix Y_1 must now account for the constraints imposed by both Y and Z.

Although there are two distinct forms of Y_1, they must ultimately be consistent with each other for the solution of the system to be valid. This consistency is enforced through the comparison of the equations formulated in (A.4) and (A.5). As shown in the proof, equating the two forms of Y_1 leads to the equation given in (A.8), which imposes additional conditions on the free parameters involved in the general solution. These conditions ensure that the system remains solvable and that the two expressions for Y_1 do not introduce contradictions.

The existence of two forms for Y_1 reflects the underlying structure of the system, where different subsystems are coupled through shared variables. The two expressions highlight the modularity of the solution process, as each subsystem can be solved individually, yet the solutions must ultimately align to ensure consistency across the entire system. This is particularly important in quaternionic matrix systems, where non-commutativity introduces additional challenges in ensuring that all expressions are compatible. In practice, the second expression for Y_1 acts as a crucial consistency check. It ensures that the solutions derived from the subsystems do not conflict with one another, and it gives a way to validate the overall solution by ensuring that all conditions of the system are satisfied.

4.4. Step-by-step calculation of Z_1 and Z

The process for computing Z_1 and Z mirrors the methodology used in the earlier steps for X_1, Y_1, and other variables. Here, we carefully outline the key steps required, ensuring that all the necessary matrices are either pre-calculated or given. The computation is based on the equation given in (3.22), and the following steps detail the necessary operations.

Before proceeding further, we utilize the intermediate matrix $\widehat{Z}_{01}$, which was previously computed in Step 9. This matrix plays a crucial role in the calculation of Z_1, and its dependencies are based on the given matrices C_6, G_8, G_9, and D_6. Since $\widehat{Z}_{01}$ has already been obtained, we can directly apply it in subsequent steps without recalculating.

We now proceed to compute the matrices P_5 and Q_5, as described in Theorem 3.1. These matrices are essential for calculating Z_1 and depend on the following components: (i) P_5 is derived from the given matrix C_3 and the previously obtained matrix L_{C_6}, which involves C_6 a matrix that already was used in Step 14 for $Z01$; and (ii) Q_5 is determined by the residual R_{D_6}, which must be obtained using D_6, along with the given matrix D_3. Therefore, all matrices required for computing P_5 and Q_5 are either given or can be derived from previously calculated quantities.

Next, we compute the intermediate matrix $\widehat{W}_{01}$ defined in (3.23). This matrix depends on the given matrices C_7, G_{10}, G_{11}, and D_7. Since it is an essential component for subsequent steps, we compute it at this stage before proceeding further. With $\widehat{W}_{01}$ now available from Step 16, we compute the residual matrix R_3, as described in Theorem 3.1. The dependencies for R_3 are as follows: (i) G_3, which is given; (ii) $\widehat{Z}_{01}$, calculated in Step 14; (iii) C_3 and D_3, both of which are given; and (iv) $\widehat{W}_{01}$, obtained in Step 16. Since all necessary components have been either computed or provided, we can now obtain R_3.

After calculating R_3, we compute the matrices M_3, S_3, P_6, Q_5, and N_3, as outlined in Theorem 3.1, as follows: (i) P_6 depends on the given matrices E_3 and L_{C_7}, which involves the previously obtained C_7 (Step 15); (ii) Q_5 is obtained using the given residuals R_{D_6} and D_3; (iii) M_3 depends on R_{P_5} and P_6, where P_5 was obtained in Step 14; (iv) S_3 depends on P_6 and $M3$, which is currently being calculated; and (v) N_3, which depends on Q_5 (implemented above and defined in Theorem 3.1). Thus, the matrices required for M_3, S_3, P_6, Q_5, and N_3 are either given or have been calculated in previous steps.

At this point, as R_3, M_3, S_3, P_6, and Q_5 were calculated, we can compute Z_1 using the expression stated in (3.22). The computation of Z_1 relies on the following components: (i) M_2, obtained in Step 11; (ii) R_2, computed in Step 9; (iii) Q_4, derived from the given matrices; and (iv) S_2, obtained earlier in Step 11. Therefore, all components required for computing Z_1 have either been obtained in earlier steps or are given. Once Z_1 is known, we can compute Z using the equation defined in (3.22). Specifically, Z depends on: (i) The given matrices C_6, G_8, G_9, and D_6; as well as (ii) the previously computed Z_1 from Step 19. Since all components have been either calculated or provided, we can now compute Z, thereby concluding the detailed computation of this step.

4.5. Step-by-step calculation of W_1 and W

To obtain W_1 and W, we follow a similar structure to that used for X_1, Y_1, and Z_1. On the basis of the definitions and results in (3.1), particularly the equation given in (3.23), we proceed with the steps below, ensuring that all matrices have either been previously generated or are explicitly given.

We start by computing the intermediate matrix $\widehat{W}_{01}$, as defined in (3.23). This matrix depends on the given matrices C_7, G_{10}, G_{11}, and D_7. The matrix $\widehat{W}_{01}$ is crucial for the subsequent computations of the residuals and the final solution.

Next, we compute P_6 and Q_6, which are necessary for the residual calculations as follows: (i) P_6 depends on the given matrix E_3 and the operator L_{C_7}, which is derived from the matrix C_7 (also given in Step 21); and (ii) Q_6 depends on R_{D_7} and F_3, where R_{D_7} needs to be computed using the given matrix D_7, and F_3 is also given. With these matrices available, we can proceed to compute the residual matrix.

We now calculate the residual matrix R_4, which is defined by the relation stated in Theorem 3.1. The matrix R_4 depends on: (i) G_4, which is given; (ii) $\widehat{W}_{01}$, obtained in Step 16; and (iii) C_4 and D_4, both of which are given. At this stage, all components required for R_4 have been either computed or provided, enabling us to determine the residual.

With the residual R_4 already obtained, we now calculate the matrices M_4, S_4, P_7, Q_6, and N_4, as outlined in Theorem 3.1 as follows: (i) P_7 depends on the given matrix E_4 and L_{C_8}, which involves the given matrix C_8; (ii) Q_6 is obtained from the residual R_{D_7} and F_3, both of which are given; (iii) M_4 depends on R_{P_6} and P_7, where P_6 was generated in Step 22; (iv) S_4 depends on P_7 (implemented above) and M_4, which is currently being computed; and (v) N_4 depends on Q_6 (computed above and defined in Theorem 3.1). Thus, the matrices required for M_4, S_4, P_7, Q_6, and N_4 are either given or have been calculated in previous steps.

Now that M_4, S_4, P_7, Q_6, and R_4 have been calculated, we can proceed with the computation of W_1, using the relation stated in (3.23). Specifically, W_1 depends on: (i) M_4 (generated in Step 24); (ii) R_4 (implemented in Step 23); (iii) Q_6 (reached in Step 22); (iv) S_4 (calculated in Step 24); and (v) N_4 (reached in Step 24). With all components available, W_1 can now be computed. Therefore, we generate the matrix W, which is defined in (3.23). The matrix W depends on (i) given matrices C_7, G_{10}, G_{11}, and D_7; and (ii) W_1, which was obtained in Step 25. With all necessary components available, we may obtain the matrix W, concluding the calculation process.

4.6. Discussion of the two forms of Z_1

As presented in Theorem 3.1, two distinct expressions for Z_1 arise, similar to the case of Y_1 due to different structural constraints within the system. These two forms of Z_1, given in (A.6) and (A.7), correspond to the different subsystems within the overall system, each governed by distinct structural constraints.

The first expression for Z_1 arises from solving subsystem (A.2), which involves coupling between Y and Z through the equation $C_2YD_2 + E_2ZF_2 = G_2$. In this subsystem, Z_1 is primarily determined by the matrices related to Y and Z. Specifically, this form of Z_1 facilitates the decoupling of Y and Z, allowing the remaining linear relationships to be solved in a straightforward manner. The first formula serves the purpose of resolving the coupling between Y and Z.

The second expression for Z_1 arises from the subsystem presented in (A.3), which couples Z and W through the equation $C_3ZD_3 + E_3WF_3 = G_3$. Here, Z_1 depends not only on the matrices associated with Z but also on those associated with W. The introduction of W_1 into the system requires Z_1 to account for the coupling between Z and W, thereby leading to the second expression. This adds additional complexity, as Z_1 and W_1 become interdependent and must satisfy mutual consistency conditions.

Although these two expressions for Z_1 are derived from different subsystems, they must be consistent to ensure the solvability of the overall system. By equating the two forms of Z_1, as given in (A.6) and (A.7), we obtain the constraint expressed in (A.9), which imposes conditions on the free parameters to guarantee consistency between the two solutions. These constraints serve as a consistency criterion for the global solution.

The existence of two distinct expressions for Z_1 highlights the modular architecture of the system and reveals the coupling between its subsystems. While the first expression is derived from the interaction between Z and Y, the second originates from the coupling between Z and W. This interdependence emphasizes the necessity of solving each subsystem independently while ensuring the alignment of the results across them. In quaternionic matrix systems such as the one considered here, maintaining consistency among the interrelated components is particularly crucial due to the non-commutative nature of quaternions. The second expression for Z_1 plays a key role in preserving the coherence of the overall solution, as it verifies the compatibility of the equations derived from both subsystems. Ensuring the alignment of these two forms confirms that the system behaves as expected, avoiding contradictions or internal inconsistencies.

5. Numerical results and implementation

5.1. Algorithm for solving the quaternionic matrix system

Algorithm 1 presents the step-by-step procedure for solving the quaternionic matrix system described in Theorem 3.1. A visual overview of this process is provided in Figure 1.

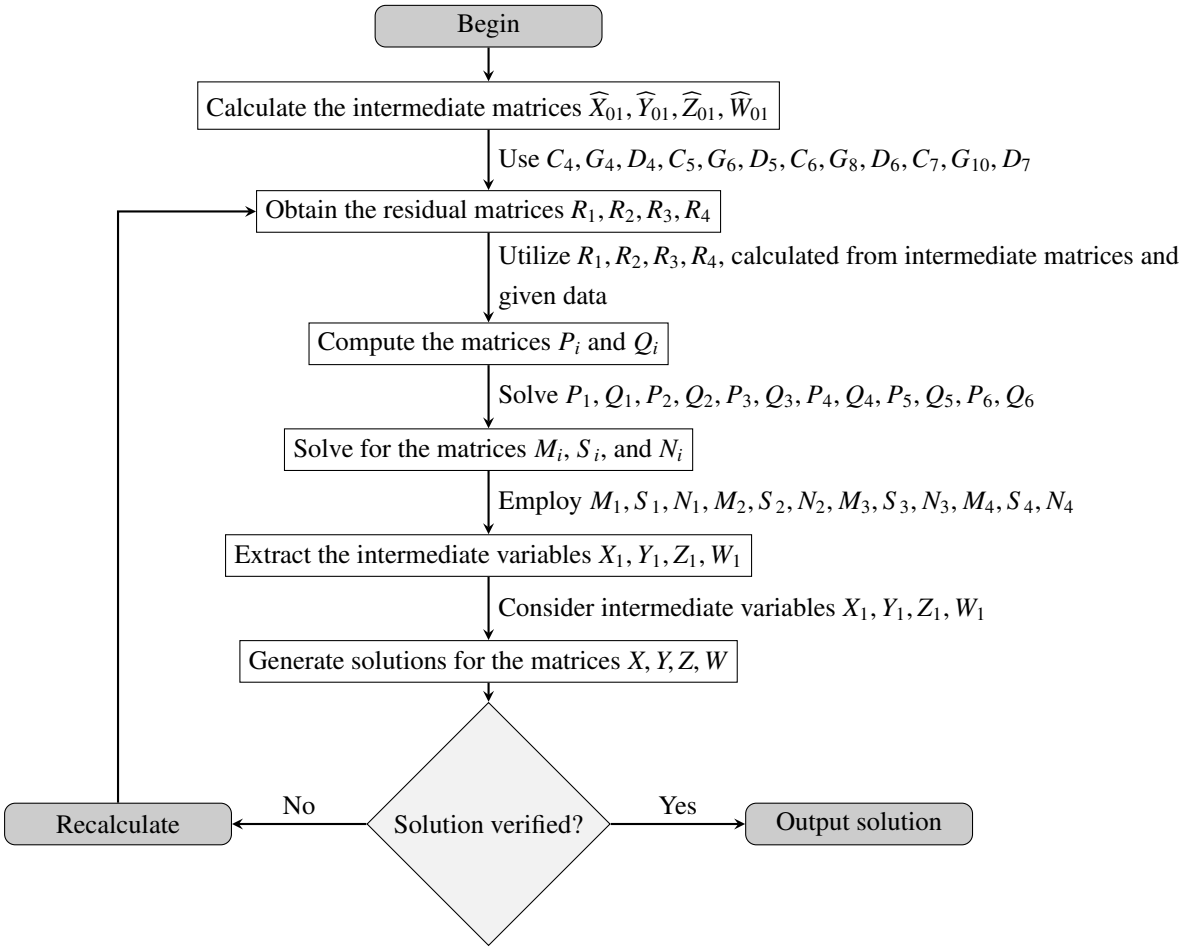

Figure 1. Flowchart of the quaternionic matrix system solver algorithm.

Algorithm 1 Quaternionic matrix system solver.

1: **Input:** Known matrices $C_1, C_2, C_3, C_4, C_5, C_6, C_7$, $D_1, D_2, D_3, D_4, D_5, D_6, D_7$, E_1, E_2, E_3, F_1, F_2, F_3, and $G_1, \ldots, G_{11}$.

2: **Calculate the intermediate matrices:** First, obtain the intermediate matrices $\widehat{X}_{01}$, $\widehat{Y}_{01}$, $\widehat{Z}_{01}$, and $\widehat{W}_{01}$, via following the steps:

- $\widehat{X}_{01}$, obtained in Step 1, as defined in (3.21).
- $\widehat{Y}_{01}$, computed in Step 1, shared with $\widehat{X}_{01}$ stated in (3.21).
- $\widehat{Z}_{01}$, generated in Step 14, from the expression presented in (3.22).
- $\widehat{W}_{01}$, reached in Step 16, using the formulation given in (3.23).

3: **Obtain the residual matrices:** The residual matrices R_1, R_2, R_3, and R_4 are reached as follows:

- R_1, calculated in Step 3, from Theorem 3.1.
- R_2, computed in Step 10.
- R_3, generated in Step 17.
- R_4, obtained in Step 23.

4: **Compute the matrices P_i and Q_i:** Using the residual matrices, obtain the matrices P_i and Q_i as follows:

- P_1 and Q_1, calculated in Step 2.
- P_2 and Q_2, obtained in Step 4.
- P_3 and Q_3, generated in Step 8.
- P_4 and Q_4, reached in Step 11.
- P_5 and Q_5, extracted in Step 15.
- P_6 and Q_6, implemented in Step 22.

5: **Solve the matrices M_i, S_i, and N_i:** Using the previously obtained matrices P_i and Q_i, solve for the following matrices:

- M_1, S_1, and N_1, calculated in Step 4.
- M_2, S_2, and N_2, computed in Step 11.
- M_3, S_3, and N_3, generated in Step 18.
- M_4, S_4, and N_4, obtained in Step 24.

6: **Extract the intermediate variables X_1, Y_1, Z_1, and W_1:** Once the residual matrices and the intermediate P_i, Q_i, M_i, and S_i matrices are obtained, solve for the following intermediate variables:

- X_1, calculated in Step 5.
- Y_1, computed in Step 12.
- Z_1, generated in Step 19.
- W_1, obtained in Step 25.

7: **Generate the solution for X, Y, Z, and W:** Using the intermediate variables X_1, Y_1, Z_1, and W_1, obtain the values of X, Y, Z, and W as follows:

- X, calculated in Step 6.
- Y, computed in Step 13.
- Z, obtained in Step 20.
- W, reached in Step 26.

8: **Verify the solution:** To ensure consistency, apply the consistency check using Lemma 3.3 to verify that the computed solution satisfies all the conditions of the system.

9: **Output:** Provide the solution matrices X, Y, Z, and W, which solve the quaternionic matrix system.

5.2. Example of the quaternionic matrix system solver

As anticipated in Subsection 1.2, we now present a comprehensive numerical example involving the robotic manipulator problem to illustrate the application of our algorithm, as described in Subsection 5.1. Specifically, we instantiate the general system given in (2.4) of Subsection 2.3 by stating the corresponding quaternionic matrices given by

$$C_1, C_2, C_3, C_4, C_5, C_6, C_7, D_1, D_2, D_3, D_4, D_5, D_6, D_7, E_1, E_2, E_3, F_1, F_2, F_3, G_1, \ldots, G_{11}.$$

All matrices lie in the quaternionic space $\mathbb{H}$, with the dimensions selected to ensure that all the required matrix products and sums are well defined. Their entries are specified in accordance with Example 5.1.

Example 5.1.

$$C_1 = \begin{bmatrix} 2+\mathrm{j} & 3+2\mathrm{k} \\ 17+4\mathrm{k} & 11\mathrm{i}+6\mathrm{j} \end{bmatrix}, C_2 = \begin{bmatrix} 2+7\mathrm{k} & 3+2\mathrm{i}+5\mathrm{j} & 2+7\mathrm{j} \\ 2\mathrm{i}+9\mathrm{k} & 3+2\mathrm{j} & 5+7\mathrm{k} \end{bmatrix}, C_3 = \begin{bmatrix} \mathrm{i}+2\mathrm{j} & 2+3\mathrm{k} & 7+2\mathrm{j} \\ 11+3\mathrm{j} & 3+7\mathrm{k} & 3+2\mathrm{k} \end{bmatrix},$$

$$C_4 = \begin{bmatrix} 2+\mathrm{j} & \mathrm{i}+3\mathrm{k} \\ 2+\mathrm{j} & 5+2\mathrm{k} \\ 5+\mathrm{j} & 7+\mathrm{k} \end{bmatrix}, C_5 = \begin{bmatrix} 2 & \mathrm{i}+\mathrm{j} & \mathrm{i}+\mathrm{k} \\ \mathrm{i}+2\mathrm{k} & 7+7\mathrm{j} & \mathrm{j}+5+7\mathrm{k} \end{bmatrix}, C_6 = \begin{bmatrix} 2+\mathrm{k} & 1+\mathrm{j} & \mathrm{i}+\mathrm{j}+\mathrm{k} \\ \mathrm{i}+2\mathrm{j} & \mathrm{k}+6 & 7+\mathrm{j} \\ 3+\mathrm{i}+\mathrm{j} & 2+\mathrm{j} & \mathrm{i}+\mathrm{k} \end{bmatrix},$$

$$C_7 = \begin{bmatrix} \mathrm{i}+2\mathrm{j} & \mathrm{k}+\mathrm{j}+8 & 12+7\mathrm{j} \\ 2+\mathrm{k} & 7+5\mathrm{j} & 8+3\mathrm{k} \\ \mathrm{i}+\mathrm{j} & 2+5\mathrm{k} & 7+10\mathrm{j} \end{bmatrix}, D_1 = \begin{bmatrix} \mathrm{i}+12\mathrm{j} & 3\mathrm{k}+7 & 8+2\mathrm{k} \\ 16+20\mathrm{j} & 23\mathrm{k}+10\mathrm{i} & 5\mathrm{j}+7 \\ 6\mathrm{i}+2\mathrm{j} & 13\mathrm{k}+6 & 8+2\mathrm{k} \end{bmatrix},$$

$$D_2 = \begin{bmatrix} 2+3\mathrm{j} & 4+7\mathrm{k} & 3+\mathrm{j} \\ 2\mathrm{i}+3\mathrm{j} & 13\mathrm{j}+7 & 8+2\mathrm{k} \\ 3\mathrm{j}+21 & 2\mathrm{i}+3\mathrm{k} & 7+2\mathrm{k} \end{bmatrix}, D_3 = \begin{bmatrix} 3+5\mathrm{j} & 7+2\mathrm{k} & 3+9\mathrm{k} \\ 3+2\mathrm{j} & 5+3\mathrm{j} & 6\mathrm{i}+2\mathrm{k} \end{bmatrix}, D_4 = \begin{bmatrix} 2+\mathrm{j} & 5+\mathrm{k} & \mathrm{i}+\mathrm{j} \\ 2 & 7\mathrm{j} & 8\mathrm{k} \\ \mathrm{i}+\mathrm{k} & \mathrm{j}+\mathrm{k} & \mathrm{k} \end{bmatrix},$$

$$D_5 = \begin{bmatrix} 2+\mathrm{j} & \mathrm{i}+\mathrm{k} & \mathrm{j}+\mathrm{k} \\ 15 & 7+\mathrm{k} & 8\mathrm{j} \\ 13+2\mathrm{j} & 8\mathrm{k}+6 & 12+\mathrm{j} \end{bmatrix}, D_6 = \begin{bmatrix} 2\mathrm{i}+3\mathrm{j} & 5\mathrm{i}+7\mathrm{k} \\ 20+7\mathrm{j} & 15+\mathrm{j}+12\mathrm{k} \end{bmatrix}, D_7 = \begin{bmatrix} 7+2\mathrm{k} & 8+2\mathrm{j} & 7+2\mathrm{k} \\ 16+2\mathrm{j} & 20\mathrm{i}+7\mathrm{k} & 17+18\mathrm{j} \\ 12+2\mathrm{j} & 7+3\mathrm{j} & 11+2\mathrm{k} \end{bmatrix},$$

$$E_1 = \begin{bmatrix} 2+7\mathrm{j} & 3+8\mathrm{k} & 2+7\mathrm{i}+3\mathrm{k} \\ 12+3\mathrm{j} & 7\mathrm{j}+5 & 8+9\mathrm{j}+2\mathrm{k} \end{bmatrix}, E_2 = \begin{bmatrix} 2+3\mathrm{j} & 4\mathrm{k}+8 & 7+13\mathrm{j} \\ 7+5\mathrm{i}+2\mathrm{j} & 3+2\mathrm{j} & 9+3\mathrm{k} \end{bmatrix},$$

$$E_3 = \begin{bmatrix} 7+3\mathrm{j} & 2+5\mathrm{k} & 7+10\mathrm{j} \\ 3+5\mathrm{j} & 2\mathrm{i}+3\mathrm{k} & 2+7\mathrm{k} \end{bmatrix}, F_1 = \begin{bmatrix} \mathrm{i}+\mathrm{j} & \mathrm{j}+\mathrm{k}+2 & 3+2\mathrm{k} \\ 2+7\mathrm{k} & \mathrm{i}+7\mathrm{j} & 2\mathrm{j}+5 \\ \mathrm{i}+2\mathrm{j} & 7+\mathrm{k} & 3\mathrm{j}+10 \end{bmatrix}, F_2 = \begin{bmatrix} \mathrm{i}+\mathrm{j} & \mathrm{j}+5 & 2\mathrm{k}+3 \\ 2\mathrm{k}+7 & 3\mathrm{j}+5\mathrm{i} & 7\mathrm{j}+2 \end{bmatrix},$$

$$F_3 = \begin{bmatrix} 2+2\mathrm{k} & 3+5\mathrm{j} & 7\mathrm{i}+3\mathrm{k} \\ 2+3\mathrm{j} & 7+8\mathrm{k} & 3+2\mathrm{k} \\ 3+5\mathrm{j} & 8+3\mathrm{k} & 20+6\mathrm{k} \end{bmatrix}, G_1 = \begin{bmatrix} \mathrm{i} & \mathrm{i}-\mathrm{j} & 2 \\ \mathrm{k} & -1 & \mathrm{j} \end{bmatrix}, G_2 = \begin{bmatrix} \mathrm{j} & 1 & \mathrm{k} \\ \mathrm{i}+\mathrm{k} & 2 & 1 \end{bmatrix}, G_3 = \begin{bmatrix} \mathrm{i}+1 & 2 & \mathrm{j}-1 \\ 3 & \mathrm{k} & \mathrm{i} \end{bmatrix},$$

$$G_4 = \begin{bmatrix} \mathrm{i} & \mathrm{j}-\mathrm{k} & 2 \\ \mathrm{j} & \mathrm{i}-\mathrm{k} & \mathrm{j}+\mathrm{k} \\ 1 & \mathrm{i}+\mathrm{j} & 2\mathrm{k} \end{bmatrix}, G_5 = \begin{bmatrix} \mathrm{i} & \mathrm{j} & \mathrm{i}+\mathrm{k} \\ 2 & \mathrm{k} & \mathrm{i} \end{bmatrix}, G_6 = \begin{bmatrix} 1 & \mathrm{i} & \mathrm{k} \\ \mathrm{j} & \mathrm{k} & 3 \end{bmatrix}, G_7 = \begin{bmatrix} \mathrm{k} & \mathrm{i}-\mathrm{j} & 3 \\ 1 & 2 & \mathrm{i}+\mathrm{j}+\mathrm{k} \\ \mathrm{i} & 1 & \mathrm{k} \end{bmatrix},$$

$$G_8 = \begin{bmatrix} \mathrm{i} & \mathrm{j} \\ \mathrm{k} & \mathrm{i}-\mathrm{j} \\ \mathrm{i} & \mathrm{k}+2 \end{bmatrix}, G_9 = \begin{bmatrix} 1 & \mathrm{k} \\ \mathrm{i} & \mathrm{j} \\ \mathrm{j} & 2 \end{bmatrix}, G_{10} = \begin{bmatrix} 1 & 2 & \mathrm{i}+\mathrm{j} \\ \mathrm{j} & \mathrm{k} & 2 \\ \mathrm{i} & \mathrm{j} & \mathrm{i}+\mathrm{k} \end{bmatrix}, G_{11} = \begin{bmatrix} \mathrm{i}+1 & 2 & 1 \\ \mathrm{k} & \mathrm{i} & \mathrm{j} \\ \mathrm{i} & \mathrm{j} & \mathrm{k} \end{bmatrix}.$$

By applying the solver described in Subsection 5.1, we obtain particular solutions for the unknown matrices, presented as

$$X = \begin{bmatrix} 1+\mathrm{i}+\mathrm{j} & 3+\mathrm{j}+2\mathrm{k} & 7+5\mathrm{j} \\ 7+\mathrm{i}+\mathrm{j}+\mathrm{k} & 2\mathrm{i}+5\mathrm{k} & 1+3\mathrm{j}+5\mathrm{k} \end{bmatrix}, \quad Y = \begin{bmatrix} \mathrm{i}+\mathrm{k} & 2\mathrm{j}+3\mathrm{k}+4 & 12+2\mathrm{k} \\ 5 & \mathrm{j}+\mathrm{k} & 2+3\mathrm{j}+4\mathrm{k} \\ 3\mathrm{i}+\mathrm{k} & 12+\mathrm{j} & 3+2\mathrm{k} \end{bmatrix}, \tag{5.1}$$

$$Z = \begin{bmatrix} 1+\mathrm{k} & 2+\mathrm{j}+3\mathrm{k} \\ 3+5\mathrm{j} & \mathrm{i}+2\mathrm{k} \\ \mathrm{i}+3\mathrm{j} & 2+3\mathrm{j}+4\mathrm{k} \end{bmatrix}, \quad W = \begin{bmatrix} 2+3\mathrm{j} & \mathrm{j}+3\mathrm{k}+4 & \mathrm{i}+3\mathrm{j}+7\mathrm{k} \\ 12+\mathrm{j} & \mathrm{i}+2\mathrm{j} & 2\mathrm{k}+3\mathrm{j} \\ 1+\mathrm{j}+\mathrm{k} & 2 & 7+4\mathrm{k} \end{bmatrix}. \tag{5.2}$$

This example shows the applicability of our quaternionic matrix solver. Each solution matrix (X, Y, Z, W), as presented in (5.1) and (5.2), conforms to the problem structure and satisfies the original equations, highlighting the robustness of the method for real-world quaternionic scenarios.

5.3. Numerical implementation and validation

Next, we describe the numerical procedures used to perform the quaternionic matrix operations required to solve the system of equations stated in (2.4). Our primary focus is on computing the Moore-Penrose inverse and the associated left and right projection matrices, which are essential for determining the solution matrices X, Y, Z, and W given in (5.1) and (5.2).

Due to the noncommutative nature of quaternions, standard linear algebra operations cannot be applied directly. To address this, we adopt a systematic methodology consisting of the following stages:

(i) Conversion to real matrix form. Each quaternionic matrix is transformed into an equivalent real matrix representation via the standard isomorphism $\mathbb{H} \simeq \mathbb{R}^{4\times4}$. For instance, a single quaternion corresponds to a 4×4 real block. As a result, a quaternionic matrix of size $m \times n$ becomes a real matrix of size $4m \times 4n$.

(ii) Computation of the Moore-Penrose inverse. Once the quaternionic matrix is converted to real form, we employ numerical linear algebra routines specifically, the singular value decomposition (SVD)-based `pinv` function from the `numpy` library in `Python` to compute the pseudoinverse. This approach utilizes robust numerical methods while circumventing the algebraic complications arising from noncommutativity.

(iii) Reconversion to quaternionic form. After obtaining the pseudoinverse in real form, the $4m \times 4n$ matrix is partitioned into 4×4 blocks. Each block is then mapped back to a single quaternion, thus reconstructing the Moore-Penrose inverse in quaternionic form, denoted $C^\dagger$.

(iv) Verification of the pseudoinverse. To assess the numerical accuracy, we evaluate the product $CC^\dagger$. Ideally, this product should approximate the identity matrix in the corresponding dimension. In practice, we verify that $\|CC^\dagger - I\|_F$ is in the order of 10^{-6} in double precision, ensuring a high degree of numerical fidelity.

(v) Computation of the left and right projections. In the presence of rank deficiencies or structural constraints, projection operators are applied to restrict the solution space accordingly. For a quaternionic matrix $C \in \mathbb{H}^{m\times n}$ with the pseudoinverse $C^\dagger \in \mathbb{H}^{n\times m}$, the left and right projection matrices are defined as

$$L_C = I - C^\dagger C, \quad R_C = CC^\dagger - I,$$

where I denotes the identity matrix of appropriate size. Applying L_C and R_C ensures that the computed solutions X, Y, Z, and W remain compatible with the rank profile of C.

The hardware and software environment used for the numerical experiments are detailed as follows. All computations were conducted on a workstation equipped with an AMD Ryzen 9700X CPU (8 cores, 16 threads) and 64 GB of RAM, running `Python 3.9`. We employed the `numpy` library for standard linear algebra operations and `numpy-quaternion` for handling quaternionic data. Since quaternionic matrix computations are not natively supported in `numpy`, we implemented custom routines for the following tasks:

- Quaternionic matrix multiplication.
- Conjugation and transposition.
- Block-based conversion between quaternionic and real forms.
- Integration of projections and pseudoinverse computations.

All floating-point operations, including those involving the SVD-based pseudoinverse, were performed in double precision. Detailed code snippets demonstrating these operations are provided in AppendixB.

To ensure numerical stability and reliability, we performed the following validation checks:

- Pseudoinverse verification. For a test matrix C, we computed both $CC^{\dagger}$ and $C^{\dagger}C$, verifying that they approximate the identity matrix within a tolerance of 10^{-6} in the Frobenius norm.
- Solution consistency. The computed solutions (X, Y, Z, W) were substituted into the original system in Eq (2.4) to confirm that the resulting expressions matched the known right-hand sides within a small numerical error.
- Precision monitoring. During computation, intermediate values were monitored to prevent the accumulation of floating-point errors. This step is especially critical due to the order-sensitive nature of quaternionic multiplication.

This validation confirms that our methods achieve the desired levels of accuracy and robustness. Consequently, the solution matrices (X, Y, Z, W) produced by the algorithm are sufficiently precise for practical applications in quaternionic matrix problems, such as those illustrated in Subsection 5.2.

5.4. Practical application: Quaternion-based orientation control in robotics

Recall the simplified robotic manipulator scenario from Subsection 1.2, where the objective is to determine the intermediate joint rotations in a two-joint system such that the end effector reaches a desired orientation. We now present a concrete numerical example to illustrate this case.

We compute specific quaternions for the rotations stated as

$$\begin{aligned} q_{A1} &= q_{A2} = \cos(22.5^\circ) + \sin(22.5^\circ)\mathrm{j} \approx 0.924 + 0.383\mathrm{j}, \\ q_{C1} &= q_{C2} = \cos(45^\circ) + \sin(45^\circ)\mathrm{i} \approx 0.707 + 0.707\mathrm{i}, \\ q_{E1} &= \cos(60^\circ) + \sin(60^\circ)\mathrm{n}_1 \approx 0.5 + 0.5(\mathrm{i} + \mathrm{j} + \mathrm{k}), \\ q_{E2} &= \cos(75^\circ) + \sin(75^\circ)\mathrm{n}_2 \approx 0.259 + 0.557\mathrm{i} - 0.557\mathrm{j} + 0.557\mathrm{k}, \end{aligned}$$

where $\mathrm{n}_1 = (1/\sqrt{3})(\mathrm{i} + \mathrm{j} + \mathrm{k})$ and $\mathrm{n}_2 = (1/\sqrt{3})(\mathrm{i} - \mathrm{j} + \mathrm{k})$.

Substituting these values into the matrices A, C, and E in the equation given in (1.1), we obtain

$$A = \begin{bmatrix} 0.924 + 0.383\mathrm{j} & 0 \\ 0 & 0.924 + 0.383\mathrm{j} \end{bmatrix}, C = \begin{bmatrix} 0.707 + 0.707\mathrm{i} & 0 \\ 0 & 0.707 + 0.707\mathrm{i} \end{bmatrix},$$

$$E = \begin{bmatrix} 0.5 + 0.5(\mathrm{i} + \mathrm{j} + \mathrm{k}) & 0 \\ 0 & 0.259 + 0.557\mathrm{i} - 0.557\mathrm{j} + 0.557\mathrm{k} \end{bmatrix}.$$

We apply the solver (Algorithm 1) to find X and Y. First, we convert A, C, and E into real matrix form via `quaternion_matrix_to_real_matrix` (see Appendix B) as follows:

```
# Convert A, C, and E to real matrices
A_real = quaternion_matrix_to_real_matrix(A)
C_real = quaternion_matrix_to_real_matrix(C)
E_real = quaternion_matrix_to_real_matrix(E)
```

Step 1. Convert the quaternionic matrices to real matrices.

Next, we form an augmented real matrix M_{real} by horizontally stacking A_{real} and C_{real}:

```
# Construct augmented real matrix
M_real = np.hstack((A_real, C_real))
```

Step 2. Construct the augmented real matrix.

Then, we vectorize E_{real} into E_{vec} and solve the linear system as

$$M_{\text{real}} \begin{pmatrix} \text{vec}(X_{\text{real}}) \\ \text{vec}(Y_{\text{real}}) \end{pmatrix} = E_{\text{vec}},$$

using the Moore-Penrose inverse as follows:

```
# Compute pseudoinverse and solve
M_pinv = np.linalg.pinv(M_real)
Z_real = M_pinv @ E_real_vec
```

Step 3. Solve the linear system using pseudoinverse.

We extract and reshape the solution parts $\text{vec}(X_{\text{real}})$ and $\text{vec}(Y_{\text{real}})$ as follows:

```
    # Number of elements for X and Y
    num_elements_X = A_real.size
    num_elements_Y = C_real.size

    # Partition Z_real
    Z_real_X = Z_real[:num_elements_X]
    Z_real_Y = Z_real[num_elements_X:]

    # Reshape to match X and Y dimensions
    X_real = Z_real_X.reshape(A_real.shape)
    Y_real = Z_real_Y.reshape(C_real.shape)
```

Step 4. Partition and reshape solution vectors.

Then, we convert these real matrices back to quaternionic form as follows:

```
X = real_matrix_to_quaternion_matrix(X_real, A.shape[0], A.shape[1])
Y = real_matrix_to_quaternion_matrix(Y_real, C.shape[0], C.shape[1])
```

Step 5. Convert the real matrices back to quaternionic matrices.

The resulting quaternionic matrices X and Y, representing the incremental rotations at each joint, are decomposed in Table 2 into scalar and imaginary parts.

Table 2. Computed quaternionic matrices X and Y. The scalar component is a and the imaginary components are with respect to i, j, and k.

Matrix	Element	Scalar (a)	i	j	k
X	X_{11}	0.653	0.271	0.271	0.653
	X_{12}	0	0	0	0
	X_{21}	0	0	0	0
	X_{22}	0.924	0	0.383	0
Y	Y_{11}	−0.271	0.653	0.653	0.271
	Y_{12}	0	0	0	0
	Y_{21}	0	0	0	0
	Y_{22}	−0.383	0	0.924	0

Substituting X and Y into the equation $AX + CY = E$ results in a Frobenius-norm error of approximately 10^{-6}, confirming high numerical accuracy. Additionally, we introduced slight perturbations to the entries of A, C, and E. The solver consistently produced accurate solutions, highlighting its robustness to small variations in the input data.

In this context, the matrix X represents the primary rotational adjustment for the first joint, while Y compensates for any residual misalignment at the second joint. Together, they ensure that the end effector achieves the target orientation. This level of accuracy and numerical stability demonstrates the suitability of the proposed method for real-world robotic applications that demand precise orientation control, such as those encountered in aerospace systems and advanced manufacturing environments.

6. Conclusions

In this article, we presented a comprehensive framework for solving a class of two-sided Sylvester-type quaternionic matrix equations, extending classical results to the noncommutative setting of $\mathbb{H}$. By leveraging rank conditions, matrix inverses, and suitable projection operators, we derived necessary and sufficient solvability criteria and proposed an efficient algorithm capable of handling both full-rank and rank-deficient cases.

Numerical examples demonstrated the stability and accuracy of our method, with residual norms in the order of 10^{-6}, including a case study of quaternion-based orientation control for robotic manipulators which highlights the practical relevance of the proposed method. While quaternionic operations are inherently more computationally demanding than their real or complex counterparts, performance can be improved through parallelization, graphics processing unit-accelerated computation, and fast matrix multiplication algorithms [21].

Future research directions include extending the proposed framework to broader algebraic systems, such as octonions or Clifford algebras, and exploring regularization techniques for inconsistent systems [33]. We also provide open-source `Python` implementations of our solver to facilitate reproducibility and adoption, encouraging further exploration in big data contexts [22], optimization-based methods [34], and precision-demanding robotic applications [35–37].

Moreover, the integration of fast algorithms and modern hardware capabilities, such as Intel AMX, which extends the x86 instruction set to accelerate matrix operations for artificial intelligence [38] and machine learning workloads [39], is increasingly vital for scaling to large or real-time quaternionic systems.

Overall, the results presented here offer both a robust theoretical foundation and practical tools for addressing quaternionic matrix problems across a range of scientific and engineering domains.

Author contributions

Abdur Rehman: Conceptualization, formal analysis, investigation, methodology, validation, writing-original draft; Cecilia Castro: Conceptualization, formal analysis, investigation, methodology, validation, writing-review and editing; Víctor Leiva: Conceptualization, formal analysis, investigation, methodology, validation, writing-review and editing; Carlos Martin-Barreiro: Conceptualization, formal analysis, investigation, methodology, writing-review and editing; Muhammad Zia Ur Rahman: Formal analysis, methodology, writing-original draft. All authors have read and agreed to the published version of the article.

Use of generative-AI tools declaration

The authors declare they have not used artificial intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors thank the anonymous reviewers for their insightful comments and suggestions, which improved the presentation of this article. Cecilia Castro gratefully acknowledges partial financial support from the Centro de Matemática da Universidade do Minho (CMAT/UM), through UID/00013.

Conflict of interest

The authors declare that they have no competing interests, whether financial or non-financial, directly or indirectly related to the submitted work.

References

1. W. R. Hamilton, *On quaternions; or on a new system of imaginaries in algebra*, Philosophical Magazine, **25** (1844), 489–495. https://doi.org/10.1080/14786444408645047
2. S. L. Adler, *Quaternionic quantum mechanics and quantum fields*, New York: Oxford University Press, 1995. https://doi.org/10.1093/oso/9780195066432.001.0001
3. C. C. Took, D. P. Mandic, Augmented second-order statistics of quaternion random signals, *Signal Proce.*, **91** (2011), 214–224. https://doi.org/ 10.1016/j.sigpro.2010.06.024
4. S. D. Leo, G. Scolarici, Right eigenvalue equation in quaternionic quantum mechanics, *J. Phys. A: Math. Gen.*, **33** (2000), 2971–2995. https://doi.org/10.1088/0305-4470/33/15/306
5. F. Z. Zhang, Quaternions and matrices of quaternions, *Linear Algebra Appl.*, **251** (1997), 21–57. https://doi.org/ 10.1016/0024-3795(95)00543-9

6. V. L. Syrmos, F. L. Lewis, Coupled and constrained Sylvester equations in system design, *Circ. Syst. Signal Pr.*, **13** (1994), 663–694. https://doi.org/10.1007/BF02523122
7. Q. W. Wang, Z. H. He, Systems of coupled generalized Sylvester matrix equations, *Automatica*, **50** (2014), 2840–2844. https://doi.org/10.1016/j.automatica.2014.10.033
8. Z. H. He, Q. W. Wang, The η-biHermitian solution to a system of real quaternion matrix equations, *Linear Multilinear A.*, **62** (2013), 1509–1528. https://doi.org/10.1080/03081087.2013.839667
9. Z. H. He, Q. W. Wang, Y. Zhang, The complete equivalence canonical form of four matrices over an arbitrary division ring, *Linear Multilinear A.*, **66** (2018), 74–95. https://doi.org/10.1080/03081087.2017.1284740
10. A. Barraud, S. Lesecq, N. Christov, From sensitivity analysis to random floating point arithmetics-application to Sylvester equations, In: *Numerical analysis and its applications, Springer, Berlin*, 2000, 35–41. https://doi.org/10.1007/3-540-45262-1_5
11. H. K. Wimmer, Consistency of a pair of generalized Sylvester equations, *IEEE Transact. Automat. Contr.*, **39** (1994), 1014–1016. https://doi.org/10.1109/9.284883
12. H. L. Wang, N. C. Wu, Y. F. Nie, Two accelerated gradient-based iteration methods for solving the Sylvester matrix equation $AX + XB = C$, *AIMS Math.*, **9** (2024), 34734–34752. https://doi.org/10.3934/math.20241654
13. S. B. Aoun, N. Derbel, H. Jerbi, T. E. Simos, S. D. Mourtas, V. N. Katsikis, A quaternion Sylvester equation solver through noise-resilient zeroing neural networks with application to control the SFM chaotic system, *AIMS Math.*, **8** (2023), 27376–27395. https://doi.org/ 10.3934/math.20231401
14. A. Rehman, Q. W. Wang, Z. H. He, Solution to a system of real quaternion matrix equations involving η-Hermicity, *Appl. Math. Comput.*, **265** (2015), 945–957. https://doi.org/10.1016/j.amc.2015.05.104
15. A. Rehman, Q. W. Wang, I. Ali, M. Akram, M. O. Ahmad, A constraint system of generalized Sylvester quaternion matrix equations, *Adv. Appl. Clifford Algebras*, **27** (2017), 3183–3196. https://doi.org/10.1007/s00006-017-0803-1
16. Z. H. He, Some new results on a system of Sylvester-type quaternion matrix equations, *Linear Multilinear A.*, **69** (2021), 3069–3091. https://doi.org/10.1080/03081087.2019.1704213
17. A. Rehman, I. Kyrchei, M. Z. U. Rahman, V. Leiva, C. Castro, Solvability and algorithm for Sylvester-type quaternion matrix equations with potential applications, *AIMS Math.*, **9** (2024), 19967–19996. https://doi.org/10.3934/math.2024974
18. A. Rehman, M. Z. U. Rahman, A. Ghaffar, C. Martin-Barreiro, C. Castro, V. Leiva, et al., Systems of quaternionic linear matrix equations: solution, computation, algorithm, and applications, *AIMS Math.*, **9** (2024), 26371–26402. https://doi.org/10.3934/math.20241284
19. A. Rehman, I. Kyrchei, Solving and algorithm to system of quaternion Sylvester-type matrix equations with $*$-Hermicity, *Adv. Appl. Clifford Algebras*, **32** (2022), 49. https://doi.org/10.1007/s00006-022-01222-2
20. J. Qin, Q. W. Wang, Solving a system of two-sided Sylvester-like quaternion tensor equations, *Comp. Appl. Math.*, **42** (2023), 232. https://doi.org/10.1007/s40314-023-02349-z
21. J. S. Respondek, *Fast matrix multiplication with applications*, Switzerland: Springer, 2025. https://doi.org/10.1007/978-3-031-76930-6

22. R. G. Aykroyd, V. Leiva, F. Ruggeri, Recent developments of control charts, identification of big data sources and future trends of current research, *Technol. Forecast. Soc. Change*, **144** (2019), 221–232. https://doi.org/10.1016/j.techfore.2019.01.005

23. D. Calvetti, L. Reichel, Application of ADI iterative methods to the restoration of noisy images, *SIAM J. Matrix Anal. Appl.*, **17** (1996), 165–186. https://doi.org/10.1137/S0895479894273687

24. D. Rosen, Some results on homogeneous matrix equations, *SIAM J. Matrix Anal. Appl.*, **14** (1993), 137–145. https://doi.org/10.1137/0614013

25. Z. H. He, The general solution to a system of coupled Sylvester-type quaternion tensor equations involving η-Hermicity, *Bull. Iran. Math. Soc.*, **45** (2019), 1407–1430. https://doi.org/10.1007/s41980-019-00205-7

26. Z. H. He, Q. W. Wang, Y. Zhang, A system of quaternary coupled Sylvester-type real quaternion matrix equations, *Automatica*, **87** (2018), 25–31. https://doi.org/10.1016/j.automatica.2017.09.008

27. H. Liping, Z. Qingguang, The matrix equation $AXB + CYD = E$ over a simple artinian ring, *Linear Multilinear A.*, **38** (1995), 225–232. https://doi.org/10.1080/03081089508818358

28. Q. W. Wang, Z. H. He, Y. Zhang, Constrained two-sided coupled Sylvester-type quaternion matrix equations, *Automatica*, **101** (2019), 207–213. https://doi.org/10.1016/j.automatica.2018.12.001

29. Q. W. Wang, A. Rehman, Z. H. He, Y. Zhang, Constraint generalized Sylvester matrix equations, *Automatica*, **69** (2016), 60–64. https://doi.org/10.1016/j.automatica.2016.02.024

30. G. Marsaglia, G. P. H. Styan, Equalities and inequalities for ranks of matrices, *Linear Multilinear A.*, **2** (1974), 269–292. https://doi.org/10.1080/03081087408817070

31. Q. W. Wang, Z. C. Wu, C. Y. Lin, Extremal ranks of a quaternion matrix expression subject to consistent systems of quaternion matrix equations with applications, *Appl. Math. Comput.*, **182** (2006), 1755–1764. https://doi.org/10.1016/j.amc.2006.06.012

32. Z. H. He, Q. W. Wang, The general solutions to some systems of matrix equations, *Linear Multilinear A.*, **63** (2014), 2017–2032. https://doi.org/10.1080/03081087.2014.896361

33. J. A. Díaz-García, V. Leiva-Sánchez, M. Galea, Singular elliptic distribution: Density and applications, *Commun. Stat. Theor. M.*, **31** (2002), 665–681. https://doi.org/10.1081/STA-120003646

34. J. A. Ramírez-Figueroa, C. Martin-Barreiro, A. B. Nieto, V. Leiva, M. P. Galindo-Villardón, A new principal component analysis by particle swarm optimization with an environmental application for data science, *Stoch. Environ. Res. Risk Assess.*, **35** (2021), 1969–1984. https://doi.org/10.1007/s00477-020-01961-3

35. A. Ghaffar, M. Z. U. Rahman, V. Leiva, C. Martin-Barreiro, X. Cabezas, C. Castro, Efficiency, optimality, and selection in a rigid actuation system with matching capabilities for an assistive robotic exoskeleton, *Eng. Sci. Technol. Inter. J.*, **51** (2024), 101613. https://doi.org/10.1016/j.jestch.2023.101613

36. A. Ghaffar, M. Z. U. Rahman, V. Leiva, C. Castro, Optimized design and analysis of cable-based parallel manipulators for enhanced subsea operations, *Ocean Eng.*, **297** (2024), 117012. https://doi.org/10.1016/j.oceaneng.2024.117012

37. A. Ghaffar, M. Z. U. Rahman, V. Leiva, C. Castro, C. Martin-Barreiro, Multi-factor optimization and failure-tolerant design of cable-driven parallel manipulators in deep-sea robotics, *IEEE Access*, **13** (2025), 79280–79290. https://doi.org/10.1109/ACCESS.2025.3561041

38. V. Leiva, C. Castro, Artificial intelligence and blockchain in clinical trials: Enhancing data governance efficiency, integrity, and transparency, *Bioanalysis*, **17** (2025), 161–176. https://doi.org/10.1080/17576180.2025.2452774

39. W. Alkady, K. ElBahnasy, V. Leiva, W. Gad, Classifying COVID-19 based on amino acids encoding with machine learning algorithms, *Chemometr. Intell. Lab. Syst.*, **224** (2022), 104535. https://doi.org/10.1016/j.chemolab.2022.104535

Appendixes

A. Proof of Theorem 1

Proof. (i) ⇔ (ii): The equations in the system given in (2.4) can be classified into subsystems for better understanding and solution. The system naturally decomposes into three subsystems, each focusing on particular cases. The first subsystem is given by

$$C_4X = G_4, XD_4 = G_5, C_5Y = G_6, YD_5 = G_7, C_1XD_1 + E_1YF_1 = G_1, \tag{A.1}$$

which focuses on the unknowns X and Y, involving both linear equations and a coupled equation. The linear equations correspond to matrix multiplications, while the coupled equation introduces complexity by linking X and Y. The second subsystem is defined as

$$C_5Y = G_6, YD_5 = G_7, C_6Z = G_8, ZD_6 = G_9, C_2YD_2 + E_2ZF_2 = G_2, \tag{A.2}$$

shifting the focus to the unknowns Y and Z, as well as maintaining the structure of linear and coupled equations. The interdependence between Y and Z is captured by the coupled equation. The third subsystem is given by

$$C_6Z = G_8, ZD_6 = G_9, C_7W = G_{10}, WD_7 = G_{11}, C_3ZD_3 + E_3WF_3 = G_3, \tag{A.3}$$

which involves Z and W, following a similar structure as the previous subsystems. The coupling between Z and W in the equation introduces the same complexity as in the earlier subsystems. These classifications simplify the problem into manageable parts, allowing for systematic analysis and solution. Each subsystem corresponds to specific unknowns, while the coupled equations ensure that the interdependence of these variables is maintained throughout the solution process.

By Lemmas 3.2 and 3.3, the equations stated in (A.1) and (A.3) have a solution if and only if $R_{C_4}G_4 = 0$, $R_{C_5}G_6 = 0$, $R_{C_6}G_8 = 0$, $R_{C_7}G_{10} = 0, D_4L_{G_5} = 0$, $G_7L_{D_5} = 0$, $D_6L_{G_9} = 0$, $G_{11}L_{D_7} = 0, C_4G_5 = G_4D_4$, $C_5G_7 = G_6D_5$, $C_6G_9 = G_8D_6$, $C_7G_{11} = G_{10}D_7, R_{P_1}R_1L_{Q_2} = 0$, $R_{P_2}R_1L_{Q_1} = 0$, $R_{M_1}R_{P_1}R_1 = 0$, $R_1L_{Q_1}L_{N_1} = 0, R_{P_3}R_2L_{Q_4} = 0$, $R_{P_4}R_2L_{Q_3} = 0$, $R_{M_2}R_{P_3}R_2 = 0$, $R_2L_{Q_3}L_{N_2} = 0, R_{P_5}R_3L_{Q_6} = 0$, $R_{P_6}R_3L_{Q_5} = 0$, $R_{M_3}R_{P_5}R_3 = 0$, and $R_3L_{Q_5}L_{N_3} = 0$. In this case, the general solution to the equation stated in (A.1) and (A.3) can be described as

$$X = C_4^{\dagger}G_4 + L_{C_4}G_5D_4^{\dagger} + L_{C_4}X_1R_{D_4}, Y = C_5^{\dagger}G_6 + L_{C_5}G_7D_5^{\dagger} + L_{C_5}Y_1R_{D_5},$$
$$X_1 = P_1^{\dagger}R_1Q_1^{\dagger} - P_1^{\dagger}P_2M_1^{\dagger}R_1Q_1^{\dagger} - P_1^{\dagger}S_1P_2^{\dagger}R_1N_1^{\dagger}Q_2Q_1^{\dagger} - P_1^{\dagger}S_1T_1R_{N_1}Q_2Q_1^{\dagger} + L_{P_1}T_2 + T_3R_{Q_1},$$

$$Y_1 = M_1^{\dagger}R_1Q_2^{\dagger} + S_1^{\dagger}S_1P_2^{\dagger}R_1N_1^{\dagger} + L_{M_1}L_{S_1}T_4 + L_{M_1}T_1R_{N_1} + T_6R_{Q_2}, \tag{A.4}$$

$$Y_1 = P_3^{\dagger}R_2Q_3^{\dagger} - P_3^{\dagger}P_4M_2^{\dagger}R_2Q_3^{\dagger} - P_3^{\dagger}S_2P_4^{\dagger}R_2N_2^{\dagger}Q_4Q_3^{\dagger} - P_3^{\dagger}S_2U_1R_{N_2}Q_4Q_3^{\dagger} + L_{P_3}U_2 + U_3R_{Q_3}, \tag{A.5}$$

$$Z = C_6^{\dagger}G_8 + L_{C_6}G_9D_6^{\dagger} + L_{C_6}Z_1R_{D_6},$$

$$Z_1 = M_2^{\dagger}R_2Q_4^{\dagger} + S_2^{\dagger}S_2P_4^{\dagger}R_2N_2^{\dagger} + L_{M_2}L_{S_2}U_4 + L_{M_2}U_1R_{N_2} + U_6R_{Q_4}, \tag{A.6}$$

$$W = C_7^{\dagger}G_{10} + L_{C_7}G_{11}D_7^{\dagger} + L_{C_7}W_1R_{D_7},$$

$$Z_1 = P_5^{\dagger}R_3Q_5^{\dagger} - P_5^{\dagger}P_6M_3^{\dagger}R_3Q_5^{\dagger} - P_5^{\dagger}S_3P_6^{\dagger}R_3N_3^{\dagger}Q_6Q_5^{\dagger} - P_5^{\dagger}S_3V_1R_{N_3}Q_6Q_4^{\dagger} + L_{P_5}V_2 + V_3R_{Q_5}, \tag{A.7}$$

$$W_1 = M_3^{\dagger}R_3Q_6^{\dagger} + S_3^{\dagger}S_3P_6^{\dagger}R_3N_3^{\dagger} + L_{M_3}L_{S_3}V_4 + L_{M_3}V_1R_{N_3} + V_6R_{Q_6},$$

where $T_1, \ldots, T_4, T_6, U_1, \ldots, U_4, U_6, V_1, \ldots, V_4, V_6$ are arbitrary matrices of appropriate dimensions over $\mathbb{H}$, representing free parameters in the general solution. The appearance of two distinct formulas for Y_1 resembles the expressions stated in (A.4) and (A.5), whereas for Z_1, it resembles the formulations given in (A.6) and (A.7), reflecting their roles in different parts of the system. Specifically, the first formula for Y_1 is derived from the direct solution of the subsystem presented in (A.1) involving Y, while the second formula arises when Y_1 is coupled with Z in subsystem established in (A.2). Similarly, the two formulas for Z_1 emerge from solving different stages of the system. The first formula relates Z_1 to the subsystem given in (A.2) involving Z, and the second formula incorporates the coupling between Z and W in the subsystem stated in (A.3). These multiple representations facilitate the modular decomposition of the system while ensuring consistency between the solutions for the interconnected variables.

By equating the two expressions for Y_1 from the expressions stated in (A.4) and (A.5) and simplifying, we get

$$A_{11}\begin{bmatrix} U_2 \\ T_4 \end{bmatrix} + \begin{bmatrix} U_3 & T_6 \end{bmatrix} B_{11} + C_{11}T_1D_{11} + F_{11}U_1G_{11} = E_{11}. \tag{A.8}$$

According to Lemma 3.3, the equation given in (A.8) is consistent if and only if the conditions stated in (3.6) are satisfied. In such a case, its general solution can be expressed as in (3.24) and (3.25). Similarly, by comparing the equations presented in (A.6) and (A.7), and then making the appropriate adjustments, we obtain

$$A_{22}\begin{bmatrix} V_2 \\ U_4 \end{bmatrix} + \begin{bmatrix} V_3 & U_6 \end{bmatrix} B_{22} + C_{22}U_1D_{22} + F_{22}V_1G_{22} = E_{22}. \tag{A.9}$$

Once again, by Lemma 3.3, the equation stated in (A.9) is consistent if and only if the conditions presented in (3.7) are satisfied. In this case, its general solution is given in (3.26)–(3.28). Likewise, by comparing the expressions established in (3.25) and (3.27), and making the necessary modifications, we obtain

$$A_{55}\begin{bmatrix} V_9 \\ W_2 \end{bmatrix} + \begin{bmatrix} V_{10} & W_4 \end{bmatrix} B_{55} + C_{55}V_7D_{55} + F_{55}W_6G_{55} = E_{55}. \tag{A.10}$$

By Lemma 3.3, the equation stated in (A.10) is solvable if and only if the conditions presented in (3.8) are satisfied, and its general solution can be expressed as in (3.29) and (3.30).

(ii) $\Leftrightarrow$ (iii): It is straightforward to show that the expressions stated in (3.3) and (3.4) are equivalent to those given in (3.9) and (3.10), and the first two rank equalities formulated in (3.11), using

Lemma 3.1. We demonstrate that the first equation stated in (3.5) is equivalent to the last equation presented in (3.11). From Lemma 3.1, we have

$$R_{P_1}R_1L_{Q_2} = 0 \Leftrightarrow \text{rank}\begin{bmatrix} R_1 & P_1 \\ Q_2 & 0 \end{bmatrix} = \text{rank}(P_1) + \text{rank}(Q_2)$$

$$\Leftrightarrow \text{rank}\begin{bmatrix} R_1 & P_1 \\ Q_2 & 0 \end{bmatrix} = \text{rank}(P_1) + \text{rank}(Q_2)$$

$$\Leftrightarrow \text{rank}\begin{bmatrix} G_1 - C_1X_{01}D_1 - E_1Y_{01}F_1 & C_1 & 0 \\ F_1 & 0 & D_5 \\ 0 & C_4 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} C_1 \\ C_4 \end{bmatrix} + \text{rank}[D_4 \quad F_1]$$

$$\Leftrightarrow \text{rank}\begin{bmatrix} G_1 & C_1 & E_1G_7 \\ F_1 & 0 & D_4 \\ G_4D_1 & C_4 & 0 \end{bmatrix} = \text{rank}\begin{bmatrix} C_1 \\ C_4 \end{bmatrix} + \text{rank}[D_4 \quad F_1],$$

which is equivalent to the last equation stated in (3.11). The other equalities can be proven similarly.

(i) ⇔ (iii): Next, we show that the first term $R_{A_{33}}E_{33}L_{D_{33}} = 0$ stated in (3.6) is equivalent to the expression given in (3.12). To do this, we begin with

$$\text{rank}(R_{A_{33}}E_{33}L_{D_{33}}) = \text{rank}\begin{bmatrix} E_{33} & A_{33} \\ D_{33} & 0 \end{bmatrix} - \text{rank}(A_{33}) - \text{rank}(D_{33})$$

$$= \text{rank}\begin{bmatrix} R_{A_{11}}E_{11}L_{B_{11}} & R_{A_{11}}C_{11} \\ G_{11}L_{B_{11}} & 0 \end{bmatrix} - \text{rank}(R_{A_{11}}C_{11}) - \text{rank}(G_{11}L_{B_{11}})$$

$$= \text{rank}\begin{bmatrix} E_{11} & C_{11} & A_{11} \\ G_{11} & 0 & 0 \\ B_{11} & 0 & 0 \end{bmatrix} - \text{rank}\begin{bmatrix} A_{11} & C_{11} \end{bmatrix} - \text{rank}\begin{bmatrix} G_{11} \\ B_{11} \end{bmatrix}$$

$$= \text{rank}\begin{bmatrix} Y_{02} - Y_{01} & L_{M_1} & -L_{P_3} & L_{M_1}L_{S_1} \\ R_{N_2}Q_4Q_3^{\dagger} & 0 & 0 & 0 \\ -R_{Q_3} & 0 & 0 & 0 \\ R_{Q_2} & 0 & 0 & 0 \end{bmatrix} - \text{rank}\begin{bmatrix} -L_{P_3} & L_{M_1}L_{S_1} & L_{M_1} \end{bmatrix} - \text{rank}\begin{bmatrix} R_{N_2}Q_4Q_3^{\dagger} \\ -R_{Q_3} \\ R_{Q_2} \end{bmatrix}$$

$$= \text{rank}\begin{bmatrix} Y_{02} - Y_{01} & I & -I & 0 & 0 & 0 \\ Q_4Q_3^{\dagger} & 0 & 0 & N_2 & 0 & 0 \\ -I & 0 & 0 & 0 & Q_3 & 0 \\ I & 0 & 0 & 0 & 0 & Q_2 \\ 0 & M_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_3 & 0 & 0 & 0 \end{bmatrix} - \text{rank}\begin{bmatrix} -I & I \\ P_3 & 0 \\ 0 & M_1 \end{bmatrix} - \text{rank}\begin{bmatrix} Q_4Q_3^{\dagger} & N_2 & 0 & 0 \\ -I & 0 & Q_3 & 0 \\ I & 0 & 0 & Q_2 \end{bmatrix}.$$

By using $P_2Y_{01}Q_2 = R_1 - P_1X_{01}Q_1, P_3Y_{02}Q_3 = R_2 - P_4Z_{01}Q_4, R_1 = G_1 - C_1X_{01}D_1 - E_1Y_{01}F_1, R_2 = G_2 - C_2Y_{02}D_2 - E_2Z_{01}F_2$, and making some simplifications, we get

$$\operatorname{rank}\begin{bmatrix} 0 & F_2 & 0 & 0 & D_6 & 0 \\ 0 & D_2 & F_1 & 0 & 0 & D_5 \\ E_1 & 0 & -G_1 & C_1 & 0 & 0 \\ C_2 & G_2 & 0 & 0 & E_2G_9 & C_2G_7 \\ C_5 & 0 & -G_6F_1 & 0 & 0 & 0 \\ 0 & 0 & -G_4D_1 & C_4 & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} C_2 & 0 \\ E_1 & C_1 \\ C_5 & 0 \\ 0 & C_4 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} F_2 & 0 & D_6 & 0 \\ D_2 & F_1 & 0 & D_5 \end{bmatrix}. \tag{A.11}$$

Now, the expression stated in (A.11) is equal to zero, which gives the result presented (3.12). Similarly, the formulations established in (3.13) and (3.14) can be shown.

We next demonstrate that the condition $R_{A_{44}}E_{44}L_{D_{44}} = 0$ given in (3.7) is equivalent to the equation stated in (3.15). To establish this equivalence, we begin by considering

$$\begin{aligned}
&\operatorname{rank}(R_{A_{44}}E_{44}L_{D_{44}}) = \operatorname{rank}\begin{bmatrix} E_{44} & A_{44} \\ D_{44} & 0 \end{bmatrix} - \operatorname{rank}(A_{44}) - \operatorname{rank}(D_{44}) \\
&= \operatorname{rank}\begin{bmatrix} R_{A_{22}}E_{22}L_{B_{22}} & R_{A_{22}}C_{22} \\ G_{22}L_{B_{22}} & 0 \end{bmatrix} - \operatorname{rank}(R_{A_{22}}C_{22}) - \operatorname{rank}(G_{22}L_{B_{22}}) \\
&= \operatorname{rank}\begin{bmatrix} E_{22} & C_{22} & A_{22} \\ G_{22} & 0 & 0 \\ B_{22} & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} A_{22} & C_{22} \end{bmatrix} - \operatorname{rank}\begin{bmatrix} G_{22} \\ B_{22} \end{bmatrix} \\
&= \operatorname{rank}\begin{bmatrix} Z_{02} - Z_{01} & L_{M_2} & -L_{P_5} & L_{M_2}L_{S_2} \\ R_{N_3}Q_6Q_5^{\dagger} & 0 & 0 & 0 \\ -R_{Q_5} & 0 & 0 & 0 \\ R_{Q_4} & 0 & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} -L_{P_5} & L_{M_2}L_{S_2} & L_{M_2} \end{bmatrix} - \operatorname{rank}\begin{bmatrix} R_{N_2}Q_6Q_5^{\dagger} \\ -R_{Q_5} \\ R_{Q_4} \end{bmatrix} \\
&= \operatorname{rank}\begin{bmatrix} Z_{02} - Z_{01} & I & -I & 0 & 0 & 0 \\ Q_6Q_5^{\dagger} & 0 & 0 & N_3 & 0 & 0 \\ -I & 0 & 0 & 0 & Q_5 & 0 \\ I & 0 & 0 & 0 & 0 & Q_4 \\ 0 & M_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_3 & 0 & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} -I & I \\ P_5 & 0 \\ 0 & M_2 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} Q_6Q_5^{\dagger} & N_3 & 0 & 0 \\ -I & 0 & Q_5 & 0 \\ I & 0 & 0 & Q_4 \end{bmatrix}.
\end{aligned}$$

By applying the relations $P_5Z_{02}Q_5 = R_3 - P_6W_{01}Q_6, P_3Y_{02}Q_3 = R_2 - P_4Z_{01}Q_4, R_3 = -G_3 + C_3Z_{02}D_3 - E_3W_{01}F_3, R_2 = -G_2 + C_2Y_{02}D_2 + E_2Z_{01}F_2$, and performing the necessary simplifications, we obtain

$$\operatorname{rank}\begin{bmatrix} 0 & F_3 & 0 & 0 & D_7 & 0 \\ 0 & D_3 & -F_2 & 0 & 0 & D_6 \\ E_2 & 0 & -G_2 & C_2 & 0 & 0 \\ C_3 & -G_3 & 0 & 0 & -E_3G_{11} & -C_3G_9 \\ C_6 & 0 & -G_8F_2 & 0 & 0 & 0 \\ 0 & 0 & -G_6D_2 & C_5 & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} C_3 & 0 \\ E_2 & C_2 \\ C_6 & 0 \\ 0 & C_5 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} F_3 & 0 & D_7 & 0 \\ D_3 & F_2 & 0 & D_6 \end{bmatrix}. \tag{A.12}$$

Now, by setting the expression stated in (A.12) equal to zero, we derive the result presented in (3.15). In a similar manner, the expressions stated in (3.16) through the equation given in (3.17) can be established. Next, we demonstrate that the first term $R_{A_{44}}E_{44}L_{D_{44}} = 0$ from the formula stated in (3.8) is equivalent to those given in (3.18). For this purpose, we begin with

$$\operatorname{rank}(R_{A_{66}}E_{66}L_{D_{66}}) = \operatorname{rank}\begin{bmatrix} E_{66} & A_{66} \\ D_{66} & 0 \end{bmatrix} - \operatorname{rank}(A_{66}) - \operatorname{rank}(D_{66}) =$$

$$\operatorname{rank}\begin{bmatrix} R_{A_{55}}E_{55}L_{B_{55}} & R_{A_{55}}C_{55} \\ G_{55}L_{B_{55}} & 0 \end{bmatrix} - \operatorname{rank}(R_{A_{55}}C_{55}) - \operatorname{rank}(G_{55}L_{B_{55}}) =$$

$$\operatorname{rank}\begin{bmatrix} E_{55} & C_{55} & A_{55} \\ G_{55} & 0 & 0 \\ B_{55} & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} A_{55} & C_{55} \end{bmatrix} - \operatorname{rank}\begin{bmatrix} G_{55} \\ B_{55} \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} U_{02}-U_{01} & L_{M_{11}} & -L_{A_{44}} & L_{M_{11}}L_{S_{11}} \\ R_{N_{22}}D_{44}B_{44}^{\dagger} & 0 & 0 & 0 \\ -R_{B_{44}} & 0 & 0 & 0 \\ R_{D_{33}} & 0 & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} -L_{A_{44}} & L_{M_{11}}L_{S_{11}} & L_{M_{11}} \end{bmatrix}$$

$$- \operatorname{rank}\begin{bmatrix} R_{N_{22}}D_{44}B_{44}^{\dagger} \\ -R_{B_{44}} \\ R_{D_{33}} \end{bmatrix} = \operatorname{rank}\begin{bmatrix} Z_{02}-Z_{01} & I & -I & 0 & 0 & 0 \\ D_{44}B_{44}^{\dagger} & 0 & 0 & N_{22} & 0 & 0 \\ -I & 0 & 0 & 0 & B_{44} & 0 \\ I & 0 & 0 & 0 & 0 & D_{33} \\ 0 & M_{11} & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{44} & 0 & 0 & 0 \end{bmatrix}$$

$$- \operatorname{rank}\begin{bmatrix} -I & I \\ A_{44} & 0 \\ 0 & M_{11} \end{bmatrix} - \operatorname{rank}\begin{bmatrix} D_{44}B_{44}^{\dagger} & N_{22} & 0 & 0 \\ -I & 0 & B_{44} & 0 \\ I & 0 & 0 & D_{33} \end{bmatrix} =$$

$$\operatorname{rank}\begin{bmatrix} 0 & D_{44} & 0 & 0 \\ 0 & B_{44} & D_{33} & 0 \\ C_{33} & 0 & -E_{33} & A_{33} \\ A_{44} & E_{44} & 0 & 0 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} A_{44} & 0 \\ C_{33} & A_{33} \end{bmatrix} - \operatorname{rank}\begin{bmatrix} D_{44} & 0 \\ B_{44} & D_{33} \end{bmatrix}.$$

By using $R_1 = G_1 - C_1X_{01}D_1 - E_1Y_{01}F_1, R_3 = G_3 - C_3Z_{02}D_3 - E_3W_{01}F_3, R_2 = G_2 - C_2Y_{02}D_2 - E_2Z_{01}F_2$, and making some simplifications, we get

$$\operatorname{rank}\begin{bmatrix} 0 & 0 & F_3 & 0 & 0 & 0 & 0 & D_7 & 0 & 0 & 0 \\ 0 & 0 & 0 & -F_2 & F_2 & 0 & 0 & 0 & D_6 & 0 & 0 \\ 0 & 0 & D_3 & F_2 & 0 & 0 & 0 & 0 & 0 & D_6 & 0 \\ 0 & 0 & 0 & 0 & D_2 & 0 & 0 & 0 & 0 & 0 & D_5 \\ C_2 & E_2 & 0 & 0 & G_2 & 0 & 0 & 0 & E_2G_9 & E_2G_9 & C_2G_7 \\ E_1 & 0 & 0 & 0 & 0 & -G_1 & C_1 & 0 & 0 & 0 & 0 \\ C_5 & 0 & 0 & 0 & 0 & G_6F_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & C_6 & G_8D_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{14}D_1 & C_4 & 0 & 0 & 0 & 0 \end{bmatrix} - \tag{A.13}$$

$$\operatorname{rank}\begin{bmatrix} P_5 & 0 & 0 \\ P_4 & P_3 & 0 \\ 0 & P_2 & P_1 \end{bmatrix} - \operatorname{rank}\begin{bmatrix} Q_6 & 0 & 0 & 0 \\ 0 & -Q_4 & Q_4 & 0 \\ Q_5 & Q_4 & 0 & 0 \\ 0 & 0 & Q_3 & Q_2 \end{bmatrix}.$$

By setting the equation given in (A.13) equal to zero, we recover the formulation presented in (3.18), thereby confirming the equivalence between the two expressions. Following the same line of reasoning, the relations established in (3.19) and (3.20) can be similarly derived through appropriate matrix manipulations and rank-based arguments. Hence, the full system is resolved, and the proof is complete. □

B. Quaternionic matrix operations

B.1. Quaternion to real matrix conversion

The function `quaternion_to_4x4` converts a single quaternion q into its equivalent 4×4 real matrix representation. Given $q = q_w + q_x\mathrm{i} + q_y\mathrm{j} + q_z\mathrm{k}$, the corresponding 4×4 real matrix is stated as

$$Q = \begin{pmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{pmatrix}.$$

```
def quaternion_to_4x4(q):
 w, x, y, z = q.w, q.x, q.y, q.z
 return np.array([
 [w, -x, -y, -z],
 [x, w, -z, y],
 [y, z, w, -x],
 [z, -y, x, w]
 ])
```

B.2. Quaternionic matrix to real matrix conversion

The function `quaternion_matrix_to_real_matrix` transforms a quaternionic matrix Q of size $m \times n$ into its real matrix equivalent of size $4m \times 4n$ by applying `quaternion_to_4x4` to each element as follows:

```
def quaternion_to_4x4(q):
 w, x, y, z = q.w, q.x, q.y, q.z
 return np.array([
 [w, -x, -y, -z],
 [x, w, -z, y],
 [y, z, w, -x],
 [z, -y, x, w]
 ])
```

B.3. Real matrix to quaternionic matrix conversion

The function `real_matrix_to_quaternion_matrix` converts a real matrix (previously generated by the inverse process) back into a quaternionic matrix by extracting each 4×4 block and converting it to a quaternion as follows:

```
def quaternion_matrix_to_real_matrix(Q):
 m, n = Q.shape
 real_matrix = np.zeros((4*m, 4*n))
 for i in range(m):
 for j in range(n):
 Q_block = quaternion_to_4x4(Q[i, j])
 real_matrix[4*i:4*(i+1), 4*j:4*(j+1)] = Q_block
 return real_matrix
```

B.4. Quaternionic matrix multiplication

The function `multiply_quaternion_matrices` implements matrix multiplication for two quaternionic matrices A and B, each entry being a quaternion as follows:

```
def real_matrix_to_quaternion_matrix(real_matrix, m, n):
 Q = np.empty((m, n), dtype=np.quaternion)
 for i in range(m):
 for j in range(n):
 block = real_matrix[4*i:4*(i+1), 4*j:4*(j+1)]
 q = real_to_quaternion(block)
 Q[i, j] = q
 return Q
def real_to_quaternion(block):
 "Converts a 4x4 real block back into a quaternion"
 return np.quaternion(block[0, 0], block[1, 0], block[2, 0], block[3, 0])
```

B.5. Conjugate transpose of a quaternionic matrix

The function `conjugate_transpose_quaternion` computes the conjugate transpose of a quaternionic matrix (analogous to the Hermitian transpose in the complex case) as follows:

```
def conjugate_transpose_quaternion(matrix):
 return np.conjugate(matrix.T)
```

B.6. Pseudoinverse of a quaternionic matrix

To compute the Moore-Penrose inverse of a quaternionic matrix C, we first convert C into its real matrix representation. Next, we compute the pseudoinverse using standard real linear algebra techniques, such as an SVD-based method. Then, the resulting real pseudoinverse is transformed back into a quaternionic matrix.

```
def quaternion_pseudoinverse(C):
 # Convert quaternionic matrix to real matrix
 C_real = quaternion_matrix_to_real_matrix(C)
 # Compute the pseudoinverse of the real matrix
 C_real_pinv = np.linalg.pinv(C_real)
 # Convert the real pseudoinverse back to quaternionic form
 m, n = C.shape
 C_pinv = real_matrix_to_quaternion_matrix(C_real_pinv, n, m)
 return C_pinv
```

B.7. Left and right projections

When rank deficiencies appear, the left and right projections help isolate the subspaces. For a quaternionic matrix C with the pseudoinverse $C^{\dagger}$, define $L_C = I - C^{\dagger}C$ and $R_C = CC^{\dagger} - I$, where I is the identity matrix of appropriate dimension.

```
def left_projection(A, A_dagger):
 projection = multiply_quaternion_matrices(A_dagger, A)
 I = np.eye(projection.shape[0], dtype=np.quaternion)
 return I - projection
def right_projection(A, A_dagger):
 projection = multiply_quaternion_matrices(A, A_dagger)
 I = np.eye(projection.shape[0], dtype=np.quaternion)
 return projection - I
```

B.8. Example usage

An illustrative example that computes the pseudoinverse and the associated projections of a quaternionic matrix C_4 is presented as follows:

```
# Install the required package for quaternion operations:
# pip install numpy-quaternion
import numpy as np
import quaternion
# Define the quaternionic matrix C_4
C_4 = np.array([
 [np.quaternion(2, 0, 1, 0), np.quaternion(0, 1, 0, 3)],
 [np.quaternion(2, 0, 1, 0), np.quaternion(5, 0, 0, 2)],
 [np.quaternion(5, 0, 1, 0), np.quaternion(7, 0, 0, 1)]
])
# Compute the pseudoinverse of C_4
C_4_dagger = quaternion_pseudoinverse(C_4)
print("Quaternionic pseudoinverse of C_4:")
print(C_4_dagger)
```

```
# Verify the pseudoinverse by computing the product C_4 * C_4_dagger
product = multiply_quaternion_matrices(C_4, C_4_dagger)
print("Product of C_4 * C_4_dagger (should be close to identity):")
print(product)
# Compute the left and right projections
L_C_4 = left_projection(C_4, C_4_dagger)
print("Left projection L_{C_4}:")
print(L_C_4)
R_C_4 = right_projection(C_4, C_4_dagger)
print("Right projection R_{C_4}:")
print(R_C_4)
```

B.9. Implementation considerations

When implementing the proposed methods, it is important to consider the following factors to ensure both accuracy and robustness:

- Software dependencies. It verifies that `numpy` and `numpy-quaternion` are correctly installed and configured. These libraries are essential for performing numerical routines and handling quaternionic data types, respectively.
- Quaternionic data representation. It uses the `np.quaternion` data type for representing quaternionic arrays. Proper handling of this type is critical to ensure consistent operations throughout quaternionic matrix computations.
- Matrix dimensionality. It ensures that all matrices have compatible dimensions throughout the computation process. Carefully checking the size of rows and columns prevents dimension mismatch errors and ensures valid matrix operations.
- Numerical precision and stability. Floating-point arithmetic may introduce small numerical errors. It is advisable to use a tolerance threshold when verifying properties such as $CC^{\dagger}C \approx C$ to accommodate minor deviations due to limited precision.

AIMS Press