applied sciences

MDPI

Article

Application of the Q-Less QR Factorization to Resolve Sparse Linear Over-Constraints

Jeong-Rae Cho [1], Keunhee Cho [1], Hyejin Yoon [1], Dong Sop Rhee [2] and Jin Ho Lee [3,*]

[1] Department of Structural Engineering Research, Korea Institute of Civil Engineering and Building Technology (KICT), Goyang 10223, Republic of Korea; chojr@kict.re.kr (J.-R.C.); kcho@kict.re.kr (K.C.); hiyoon@kict.re.kr (H.Y.)
[2] Department of Hydro Science and Engineering Research, Korea Institute of Civil Engineering and Building Technology (KICT), Goyang 10223, Republic of Korea; dsrhee@kict.re.kr
[3] Department of Ocean Engineering, Pukyong National University, Busan 48513, Republic of Korea
* Correspondence: jholee0218@pknu.ac.kr

Abstract

Over-constraint issues frequently arise in complex models with multiple constraints, such as those used in multibody systems and contact analyses. When over-constraints exist, commonly used constraint-handling methods, such as the transformation method, the Lagrange multiplier method, and the penalty method, can produce small or zero pivot values in the system matrix. This may result in non-convergence, slow convergence, or incorrect solutions. Furthermore, in the transformation method, in which the slave degrees of freedom (DOFs) corresponding to each constraint equation are eliminated, most finite element software restricts the reuse of slave DOFs that have already been referenced in other constraint equations. This makes an already complex modeling process even more challenging. This paper first summarizes a systematic algorithm proposed by Cho for detecting and resolving over-constraints in linear constraint equations, highlighting its efficiency in less complex scenarios but also its limitations for large-scale models due to excessive computation time. To address these limitations, this paper introduces a new algorithm that leverages the sparsity of the constraint matrix and applies sparse QR factorization without generating a dense Q matrix. The effectiveness of this approach in handling complex, large-scale finite element (FE) models is demonstrated, confirming its applicability to complex engineering problems.

Keywords: over-constraint; slave DOFs; Gaussian elimination; Q-less QR factorization; sparse constraint matrix; finite element analysis

check for updates

Academic Editor: Jean-Jacques Sinou

Received: 12 November 2025
Revised: 5 December 2025
Accepted: 9 December 2025
Published: 11 December 2025

Citation: Cho, J.-R.; Cho, K.; Yoon, H.; Rhee, D.S.; Lee, J.H. Application of the Q-Less QR Factorization to Resolve Sparse Linear Over-Constraints. *Appl. Sci.* **2025**, *15*, 13059. https://doi.org/10.3390/app152413059

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over-constraint problems frequently occur in complex finite element (FE) models that use numerous constraints, contact analyses, and multibody systems subjected to constraints. The presence of over-constraints can cause difficulties in obtaining a stable solution, as solvers may encounter small or zero pivots regardless of the constraint-handling method—transformation method, Lagrange multiplier method, or penalty method. A typical over-constraint problem is illustrated in Figure 1. Three constraint equations are required at nodes D, F, I, and L for them to move together, but four constraints are generated owing to the superposition of four line-to-line constraints. Hence, one redundant over-constraint must be removed. In addition, the constraint between nodes C and H created by the line-to-line condition conflicts with the specified boundary condition at those nodes. To resolve this

issue, constraints should not be imposed as line-to-line relationships; instead, each degree of freedom (DOF) should be constrained individually to prevent over-constraints [1–4]. This process is tedious and prone to error.

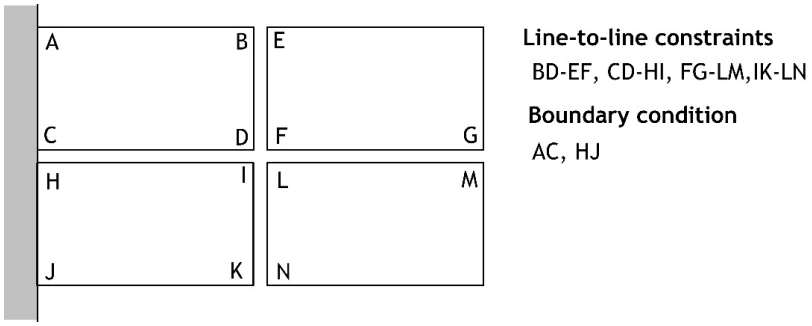


Figure 1. Typical over-constraint problem.

Particular attention should be given to handling constraints using the transformation method when addressing over-constraint problems. Figure 2 depicts a simple three-dimensional frame structure in which the DOFs on the first floor are constrained in the X-direction. Most FE software restricts the reuse of slave DOFs that have already been assigned in another constraint equation. Consequently, in programs that automatically assign the first DOF appearing in each constraint equation as the slave DOF, the user input shown in Figure 2 results in an error. In such cases, the constraints that can be processed by the program are $u_1 - u_4 = 0$, $u_2 - u_4 = 0$, and $u_3 - u_4 = 0$. Care must be taken when selecting slave DOFs while imposing constraints [1–4].

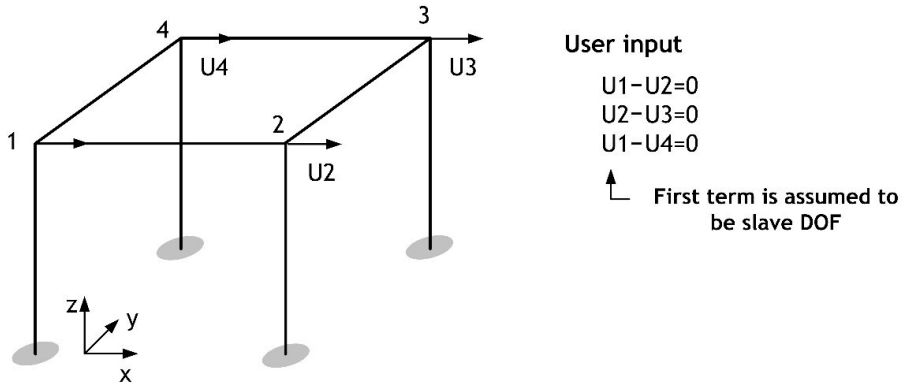


Figure 2. Input-inducing errors in the transformation method.

Cho [5] proposed a systematic algorithm in his Ph.D. dissertation for detecting and resolving over-constraints that frequently occur in FE models with linear constraint equations. The algorithm is based on Gaussian elimination with column pivoting and can automatically select slave DOFs in the transformation method and reconstruct appropriate constraint equations. This algorithm performs well when the constraints are not overly complex. However, in large-scale models with highly coupled or complex constraints, it requires excessive computation time, making it necessary to consider the sparsity of the constraint matrix. ANSYS introduced the capability to automatically compute slave DOFs and detect over-constrained DOFs through the OVCHECK command, beginning with version 18.2 [6]. This command provides two options for detecting over-constraints: a default topological approach and an algebraic approach. The topological approach is faster but less robust, whereas the algebraic approach is more accurate but computationally demanding. Slave DOFs are determined using factorization with full pivoting applied to the constraint matrix. If a pivot is too small, the corresponding constraint equation

is regarded as redundant and removed from the set of equations to be satisfied. When redundant constraints are detected, the program outputs information about their removal. This procedure is similar to that proposed by Cho [5].

This paper first reviews the systematic algorithm proposed in [5] and then presents a new algorithm that accounts for the sparsity of the constraint matrix in large-scale FE models with complex constraints. The proposed method employs sparse QR factorization without generating a dense Q matrix and has proven effective for complex, large-scale problems. A key contribution of this study is that the proposed Q-less sparse QR factorization is formulated specifically for an under-determined constraint system, where explicitly forming the orthogonal matrix Q would destroy sparsity and render large-scale analysis impractical; the Q-less formulation therefore provides a necessary and scalable mechanism for stable rank detection and redundancy elimination.

2. Systematic Approach for Detecting and Resolving Over-Constraints

This section summarizes the algorithm proposed by Cho [5] for detecting and resolving over-constraints in linear static systems subjected to linear constraint conditions. The fundamental equations are expressed as

$$\mathbf{Ku} = \mathbf{f}, \tag{1}$$

with

$$\mathbf{Cu} - \mathbf{b} = \mathbf{0}, \tag{2}$$

where $\mathbf{K}$ is an $n \times n$ stiffness matrix, $\mathbf{u}$ is an $n \times 1$ vector of DOFs, and $\mathbf{f}$ is the corresponding $n \times 1$ force vector. $\mathbf{C}$ is an $n_s \times n$ matrix of coefficients, and $\mathbf{b}$ is an $n_s \times 1$ vector of constants. n denotes the number of DOFs, and n_s is the number of constraint equations. Generally, the constraint equation can be formulated as a vector function $\mathbf{g}(\mathbf{u}) = \mathbf{0}$. However, the constraints commonly encountered in finite element modeling are typically expressed as linear relationships among the DOFs. The algorithm proposed in [5] is therefore limited to linear constraint conditions of the form (2). For nonlinear or dynamic systems, since solving the problem involves equations in the form of (1), the algorithm presented in this section can still be applied, provided that the constraint conditions remain linear.

2.1. Solution Methods for Systems of Equations with Constraints

Representative methods for handling constraints include the transformation method, the Lagrange multiplier method, and the penalty method. These methods are briefly reviewed below. The transformation method classifies the degrees of freedom (DOFs) into slave DOFs corresponding to each constraint equation and master DOFs for the remaining variables.

This method handles the constraints by eliminating the slave DOFs through the transformation of the entire system of equations. Equations (1) and (2) can be rewritten as follows by distinguishing the slave–master DOFs:

$$\mathbf{Ku} = \mathbf{f}, \tag{3}$$

subject to

$$\mathbf{Cu} - \mathbf{b} = \mathbf{0}. \tag{4}$$

In Equation (3), the dimensions of the submatrices $\mathbf{K}_{mm}$, $\mathbf{K}_{ms}$, $\mathbf{K}_{sm}$, $\mathbf{K}_{ss}$ are, respectively, $n_m \times n_m$, $n_m \times n_s$, $n_s \times n_m$, and $n_s \times n_s$. The vectors $\mathbf{u}_m$, $\mathbf{u}_s$, $\mathbf{f}_m$, and $\mathbf{f}_s$ have dimensions $n_m \times 1$, $n_s \times 1$, $n_m \times 1$, and $n_s \times 1$, respectively. In Equation (4), the dimensions of $\mathbf{C}_s$ and $\mathbf{C}_m$ are $n_s \times n_s$ and $n_s \times n_m$. Here, n_s denotes the number of slave DOFs (or the number of

constraints), and n_m is the number of master DOFs; thus, the total number of DOFs is given by $n = n_s + n_m$. If the inverse matrix of $\mathbf{C}_s$ exists, Equation (4) can be arranged as follows:

$$\mathbf{u}_s = \mathbf{T}_{sm}\mathbf{u}_m + \mathbf{b}_s, \tag{5}$$

where $\mathbf{b}_s = \mathbf{C}_s^{-1}\mathbf{b}$ with dimension $n_s \times 1$, and $\mathbf{T}_{sm} = -\mathbf{C}_s^{-1}\mathbf{C}_m$ with dimension $n_s \times n_m$. In order to implement transformation, the whole DOFs $\mathbf{u}$ should be expressed as follows in terms of the master DOFs:

$$\mathbf{u} = \mathbf{T}\mathbf{u}_m + \mathbf{b}_0 \text{ or } \begin{bmatrix}\mathbf{u}_m \\ \mathbf{u}_s\end{bmatrix} = \begin{bmatrix}\mathbf{I} \\ \mathbf{T}_{sm}\end{bmatrix}\mathbf{u}_m + \begin{bmatrix}\mathbf{0} \\ \mathbf{b}_s\end{bmatrix} \tag{6}$$

Using the transformation in Equation (6), a condensed system in which the slave DOFs have been removed from the whole system of equations can be constituted. Applying the transformation of Equation (6) in Equation (3), the following condensed system can be obtained [7]:

$$\mathbf{K}_{mm}^{*}\mathbf{u}_m = \mathbf{f}_m^{*}, \tag{7}$$

where

$$\mathbf{K}_{mm}^{*} = \mathbf{T}^T\mathbf{K}\mathbf{T} = \mathbf{K}_{mm} + \mathbf{K}_{ms}\mathbf{T}_{sm} + \mathbf{T}_{sm}^T\mathbf{K}_{cm} + \mathbf{T}_{sm}^T\mathbf{K}_{ss}\mathbf{T}_{sm} \tag{8}$$

$$\mathbf{f}_m^{*} = \mathbf{T}^T(\mathbf{P} - \mathbf{K}\mathbf{b}_0) = \mathbf{f}_m + \mathbf{T}_{sm}^T\mathbf{b}_s - \mathbf{K}_{ms}\mathbf{b}_s - \mathbf{T}_{sm}^T\mathbf{K}_{ss}\mathbf{b}_s \tag{9}$$

Even though the transformation method reduces the dimension of the system matrix, it requires explicit partitioning of slave and master DOFs and involves complex implementation in program coding [1,2]. Because of this partitioning issue, most programs adopt user input in the form of Equation (5) and prohibit the reuse of already assigned slave DOFs in other constraint equations [1,2]. To simplify implementation, a homogeneous linear constraint (Q = 0) is often assumed. In this case, the last two terms on the right-hand side of Equation (9) vanish, and condensation of the force vector can be performed more easily. When nonhomogeneous constraints exist, dummy DOFs are introduced to handle them [1,2].

The Lagrange multiplier and penalty methods offer simpler implementation than the transformation method and are also capable of handling nonlinear constraints. For a linear static problem, the governing equations of these two methods can be formulated as follows [7]:

$$\begin{bmatrix}\mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{u} \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\mathbf{f} \\ \mathbf{b}\end{bmatrix} \tag{10}$$

$$\left[\mathbf{K} + \mathbf{C}^T\boldsymbol{\alpha}\mathbf{C}\right]\mathbf{u} = \mathbf{f} - \mathbf{C}^T\boldsymbol{\alpha}\mathbf{b} \tag{11}$$

where $\boldsymbol{\lambda}$ in Equation (10) is the Lagrange multiplier vector of dimension of $n_s \times 1$, $\boldsymbol{\alpha} = [\alpha_n]$ in Equation (11) is a diagonal matrix of penalty parameters of dimension $n_s \times n_s$. In the Lagrange multiplier method, n_s additional variables (Lagrange multipliers) are introduced, which increase the size of the system matrix and may compromise positive definiteness. Nevertheless, this method is straightforward to implement and provides exact enforcement of the constraints. Conversely, the penalty method preserves the original matrix size and is easy to implement, but it yields an approximate solution depending on the magnitude of the penalty parameters.

Although classical techniques such as the transformation method, the Lagrange multiplier method, and the penalty method are well established, their limitations become more pronounced in large-scale sparse finite element models. The transformation method can introduce substantial fill-in, destroying matrix sparsity; the Lagrange multiplier method

produces an indefinite saddle-point system whose conditioning deteriorates with an increasing number of constraints; and penalty methods become sensitive to the choice of penalty parameters in high-dimensional settings.

A few studies have examined redundant constraints in more specialized areas. For example, redundant or rank-deficient constraint matrices have been analyzed in multibody system dynamics [8,9]. However, these works are problem-specific and do not provide general numerical algorithms for automatically detecting and eliminating redundant constraints in large, sparse finite element systems. To the best of the authors' knowledge, systematic methods that resolve over-constraints through sparsity-preserving factorizations such as Q-less sparse QR have not been reported, underscoring the need for the present study.

2.2. Gauss Elimination-Based Algorithm

From a mathematical point of view, over-constraints frequently encountered when handling constraint equations indicate that the matrix **C** in Equation (2) is rank-deficient. In other words, the given constraints are not linearly independent. Over-constraints can be classified as either consistent or inconsistent. Consistent over-constraints can be eliminated from the constraint equations, whereas inconsistent over-constraints represent errors in the constraint formulation [1,2]. For example, assume the following constraint equations:

$$u_1 - u_2 + 2u_3 = 1 \tag{12}$$

$$u_1 - 2u_2 - u_3 = 0 \tag{13}$$

$$2u_1 + u_2 + u_3 = 1 \tag{14}$$

In the constraint Equations (12)–(14), the rank of the coefficient matrix **C** is 2, indicating that the matrix is rank-deficient. Equation (14) can be expressed as a linear combination of Equations (12) and (13); thus, it represents a consistent over-constraint that can be eliminated. However, if the right-hand side of Equation (14) is changed to zero, the constraint becomes an inconsistent over-constraint, since no values of u_1, u_2, and u_3 can simultaneously satisfy all three equations.

Figure 3a shows the results obtained after applying Gaussian elimination to the augmented matrix $\mathbf{A} = \begin{bmatrix} \mathbf{C} & -\mathbf{b} \end{bmatrix}$, constructed from Equations (12)–(14). Since all the terms in the last row are zero, this row represents a consistent over-constraint that can be removed. Consequently, the constraints given in Equations (12)–(14) can be reduced to the following two independent constraint equations:

$$u_1 - u_2 + 2u_3 = 1 \tag{15}$$

$$3u_2 - 3u_3 = -1 \tag{16}$$

Figure 3b shows the same procedure applied to the case where the right-hand side of Equation (14) is zero. In this case, the constraint equations produce an inconsistent over-constraint, since no values of u_1, u_2, and u_3 satisfy the last row $0 \cdot u_1 + 0 \cdot u_2 + 0 \cdot u_3 = -1$.

If the constraints in Equations (12)–(14) are handled using the transformation method, a slave–master DOF relationship of the form of Equation (4) can be derived by performing back substitution on the upper-triangular matrix obtained after eliminating the over-constraint. Figure 4 illustrates this procedure. The constraints of Equations (12)–(14) in the final matrix form are transformed into the following equations, for which the transformation method can be easily applied.

$$u_1 = -u_3 + \frac{2}{3} \tag{17}$$

$$u_2 = u_3 - \frac{1}{3} \tag{18}$$

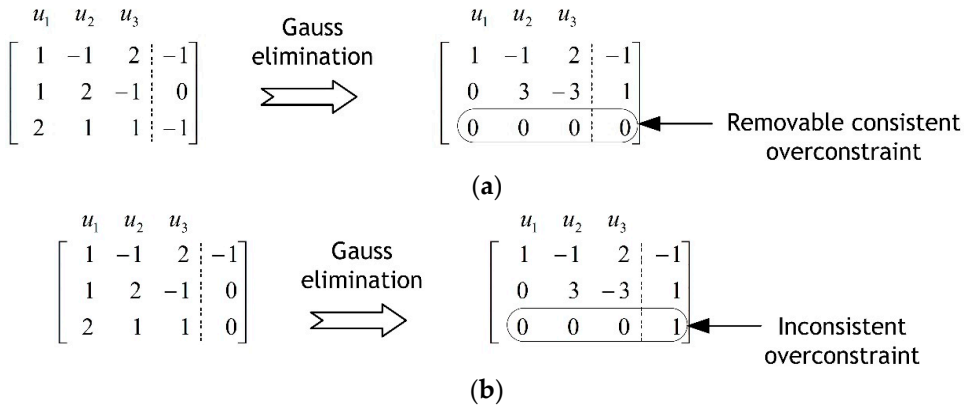


Figure 3. Gauss elimination of constraint equations: (**a**) consistent over-constraint; (**b**) inconsistent over-constraint.

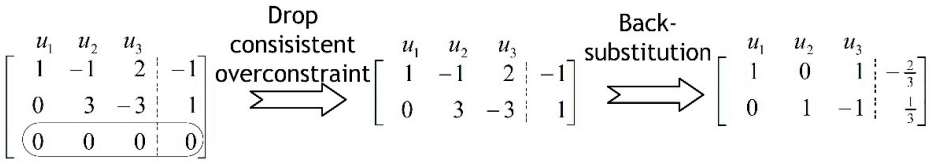


Figure 4. Back substitution of constraint equations.

For the general application, both row and column pivoting should be performed simultaneously, since a row pivot may become zero during Gaussian elimination. Moreover, because the constraint matrix **C** consists only of the DOFs involved in the constraint equations, rather than the entire set of DOFs, the computational efficiency is significantly improved. The approach described above enables the over-constraint problem and the selection of slave DOFs in the transformation method to be handled efficiently. However, a slight modification of the algorithm is required when boundary conditions—also referred to as single-point constraints—are considered.

The first issue arises when a boundary condition explicitly involves a prescribed DOF in the constraint equation. In general, nonzero boundary conditions appear during nonlinear analyses performed using displacement-control or arc-length methods [10–12]. Since the prescribed value of the corresponding DOF varies at each increment, such a boundary condition is regarded as a load condition rather than a single-point constraint. This issue is resolved by placing the DOF with the prescribed boundary condition in a subordinate column when constructing the augmented matrix $\mathbf{A} = \begin{bmatrix} \mathbf{C} & -\mathbf{b} \end{bmatrix}$, and by omitting column pivoting for that DOF. Figure 5 illustrates the application of the proposed algorithm for the case where a prescribed boundary condition is imposed on u_2. It can be seen that u_2 is placed in a subordinate column before the algorithm is executed.

The second issue related to boundary conditions occurs when the constraints are imposed solely on DOFs that are already subjected to boundary conditions. In this case, the situation can be recognized as an over-constraint problem. Constraint equations exhibiting such conditions can be handled during the over-constraint detection stage, prior to Gaussian elimination, by placing the corresponding DOFs in subordinate columns when constructing the matrix **A**.

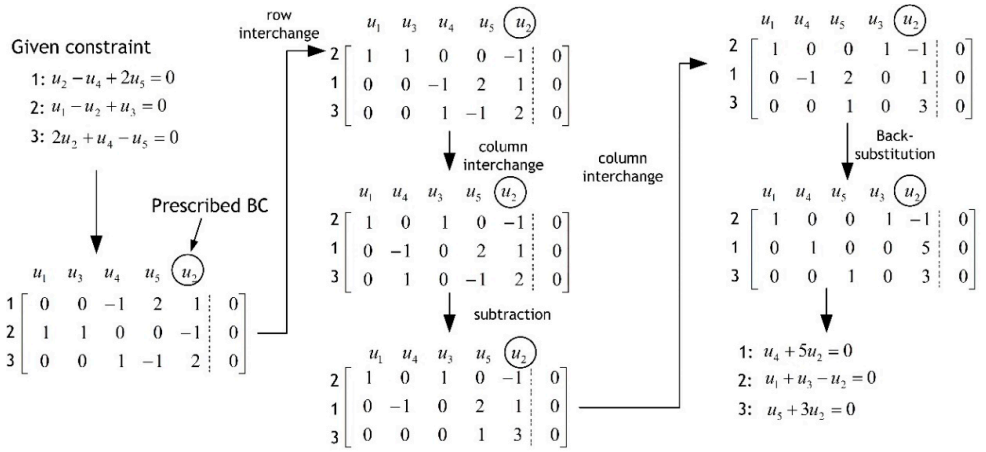


Figure 5. Constraint related to DOF imposed with the boundary condition.

The third issue concerning boundary conditions arises from the fact that new boundary conditions can be generated during the execution of the algorithm. In this paper, such boundary conditions are referred to as "generated boundary conditions," to distinguish them from prescribed ones. For example, consider a DOF u_4 that is subjected to a boundary condition. If a constraint equation $u_1 - u_4 = 0$ is imposed, DOF u_1 implicitly becomes subjected to the boundary condition. If an external load is explicitly applied to u_1, this constitutes a modeling error; otherwise, it simply represents a normal constraint in which the reaction load corresponding to the support at u_1 is computed. Figure 6 illustrates a more complex case in which the generated boundary conditions are handled. As shown, boundary conditions are generated for u_3 and u_5, meaning that the values of these DOFs are implicitly determined to satisfy the given constraints. Because loads cannot be applied to DOFs that are subjected to boundary conditions, such cases result in inconsistent boundary conditions when external loads are present at these DOFs.

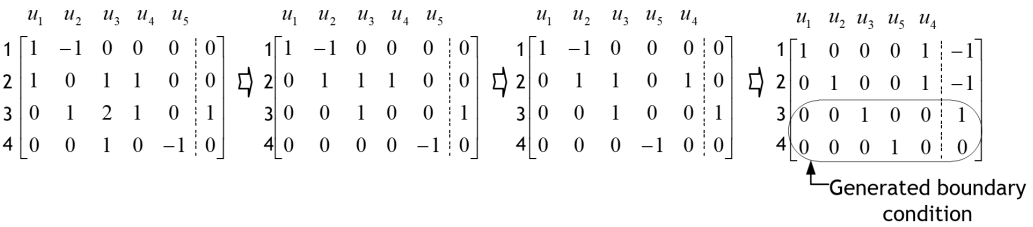


Figure 6. Generated boundary condition.

Since generated boundary conditions may exist, their presence should be checked until the back-substitution stage, even when applying the Lagrange multiplier or penalty method.

Figure 7 summarizes the algorithm proposed in this study. Step 1 constructs the augmented matrix $\mathbf{A} = \begin{bmatrix} \mathbf{C} & -\mathbf{b} \end{bmatrix}$, where only the DOFs involved in the constraint equations are included. Step 2 corresponds to the preprocessing of prescribed boundary conditions. Step 3 performs triangularization using Gaussian elimination. Step 4 identifies over-constraints in the rows that appear after the zero-pivot row detected in Step 3. Step 5 performs back substitution. Step 6 checks for generated boundary conditions. If a combination of the transformation, Lagrange multiplier, and penalty methods is adopted, row indices can be assigned to selectively apply each method to the matrix obtained at the final stage.

Given $\mathbf{Cu} = \mathbf{b}$

Step 1. Construct augmented matrix A=[C−b] using only the DOFs used in the constraint equations.

Step 2. Preprocessing for boundary conditions
- Arrange DOFs subject to boundary conditions at subordinate columns.
- Arrange constraint equations constituted only with DOFs imposed with boundary conditions at subordinate rows.

Step 3. Gauss elimination with row and column pivoting up to zero pivot
- Column pivoting not applicable to DOFs subject to boundary conditions.

Step 4. In case of zero pivot, overconstraint handling for the subsequent rows. (assume non-zero pivot for m row)
- For inconsistent overconstraint, constraint error → STOP
- For consistent overconstraint, remove the corresponding row.

Step 5. Perform backsubstitution for the partial matrix.

Step 6. Check the existence of generated boundary condition.
- In case of DOF with given external load, error → STOP

Figure 7. Gauss elimination-based algorithm for automatic detection and resolution of over-constraints and automatic selection of slave DOFs.

The algorithm presented in Figure 7 should be implemented during the modeling preprocessing stage, prior to the analysis. In cases where constraints are introduced during the analysis, such as in contact problems, the algorithm should be executed before the assembly and solution of the system matrix. Because the size of the constraint matrix **C** in the algorithm is much smaller than that of the global system matrix, the additional computational cost introduced by the algorithm is negligible.

2.3. Illustrative Example

The proposed algorithm is applied to a typical over-constraint problem illustrated in Figure 8. The problem involves 16 constraint equations arising from four line-to-line constraints. Among the DOFs included in these constraint equations, u_1 and u_{213} are subjected to zero boundary conditions. Figure 9 shows the augmented matrix $\mathbf{A} = \begin{bmatrix} \mathbf{C} & -\mathbf{b} \end{bmatrix}$ after Step 2 of the proposed algorithm. The results obtained by implementing the algorithm are summarized in Figure 10. Over-constraints are observed in rows 15 and 16, corresponding to the 16th and first constraint equations, respectively, and these are consistent over-constraints.

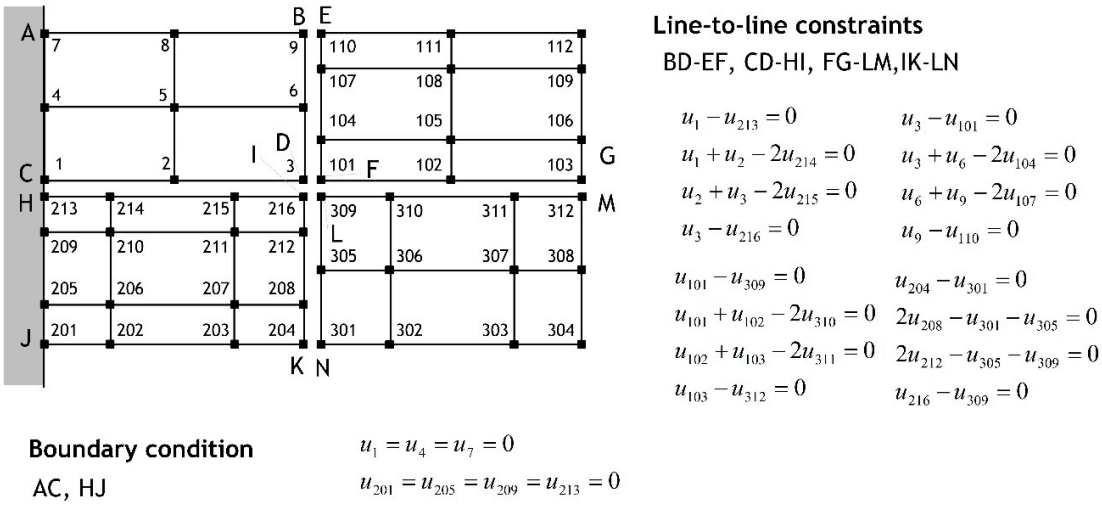


Figure 8. Illustrative example: Gauss elimination-based algorithm.

DOF numbers

	2	3	6	9	101	102	103	104	107	110	204	208	212	214	215	216	301	305	309	310	311	312	1	213	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0	0	0	0	0	0	1	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0
5	0	1	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	1	0	0	0	0	−2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	1	0	0	0	0	−2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	0
10	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0	0
11	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0
12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	−1	−1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	−1	−1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	−1	0	0	0	0	0	0

Figure 9. Augmented matrix after Step 2: Gauss elimination-based algorithm.

Slave DOFs

DOF numbers

	2	3	6	9	101	102	103	104	214	216	204	208	212	309	215	110	301	305	107	310	311	312	1	213	
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0	2	−2	1	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	2	−1	0	0	0
6	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	−1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	2	−1	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−2	1	0	0	0
11	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	−1	0	−1	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	$\frac{1}{2}$	0	0	−1	1	1	$-\frac{1}{2}$	$-\frac{1}{2}$	0	0
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	−2	−1	$\frac{1}{2}$	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	−1	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	$-\frac{1}{2}$	$-\frac{1}{2}$	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	$-\frac{1}{2}$	0	−1	1	$-\frac{1}{2}$	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	−2	2	−1	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	−1	0

Figure 10. Augmented matrix after completion: Gauss elimination-based algorithm.

3. Algorithm for Large-Scale Models with Complex Constraints

The Gaussian elimination-based algorithm presented in Section 2 can be effectively applied when the constraints are not overly complex. However, in large-scale models with numerous and intricate constraint conditions, the computational cost becomes significant, requiring consideration of the sparsity of the constraint matrix. To address this issue, two enhanced approaches are proposed. First, an improved method that applies the Gaussian elimination-based algorithm to the constraint matrix divided into independent block units is introduced. Second, a new algorithm employing sparse QR factorization is proposed.

3.1. Block-Wise Gauss Elimination-Based Algorithm

A simple enhancement of the Gaussian elimination-based algorithm for large-scale problems can be achieved by partitioning the constraint matrix into independent block units. As shown in Figure 11, the original constraint equation (i.e., the augmented matrix) can be divided into smaller matrix blocks based on connectivity. In this example, there are 24 degrees of freedom and nine constraints, which are divided into five independent blocks.

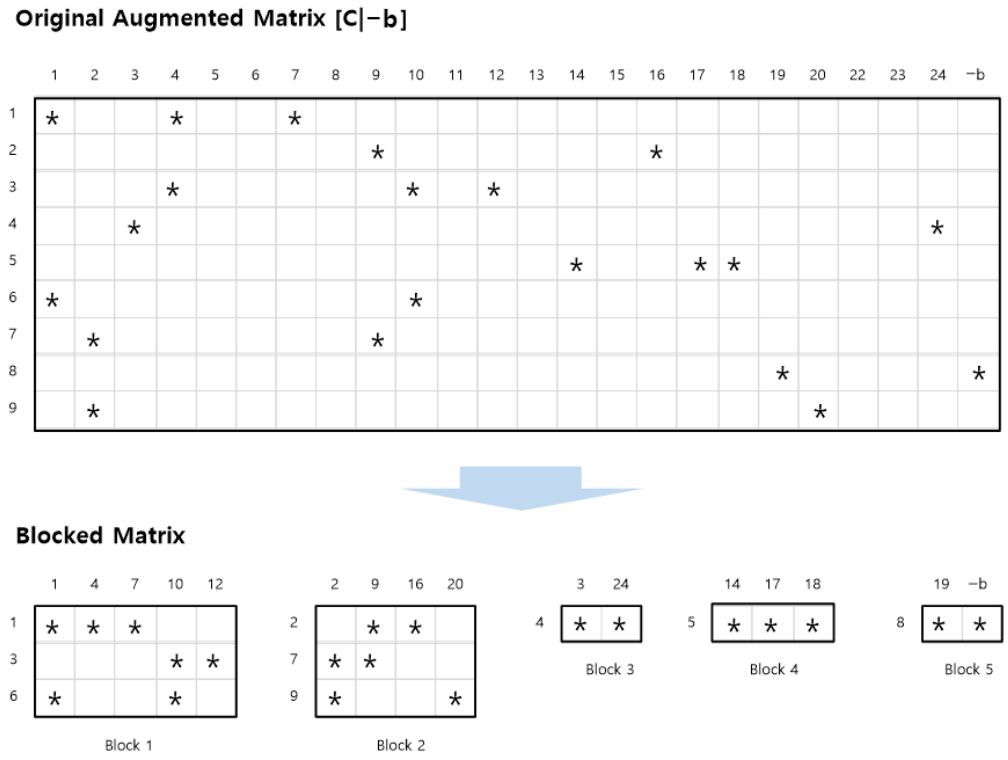


Figure 11. Independent matrix block for block-wise Gauss elimination-based algorithm. Asterisks (*) indicate nonzero entries in the augmented matrix.

This approach allows the Gaussian elimination-based algorithm to be applied to each block separately. Each submatrix block is treated as a dense matrix, enabling efficient computation even when the overall constraint matrix is large, provided that the constraint relationships within each block are not excessively complex. However, determining an efficient method for partitioning the matrix into sub-blocks remains a challenge, and if the size of a single sub-block becomes too large, computational efficiency is significantly reduced.

3.2. Q-Less Sparse QR Factorization Algorithm

Instead of using the Gaussian elimination method, a sparse QR factorization approach is proposed. QR factorization is an essential matrix decomposition technique in linear

algebra, widely used in solving linear systems, least-squares problems, and singular value decomposition (SVD) [13]. For a matrix **A**, the column-pivoted QR factorization is written as

$$\mathbf{AP} = \mathbf{QR}, \tag{19}$$

where **P** is a permutation matrix, **Q** is an orthogonal matrix, and **R** is an upper triangular matrix. QR factorization can be applied to non-square matrices, including the determined constraint matrices that arise in FE constraint elimination. Figure 12 illustrates the conceptual forms of QR factorization for full-rank and rank-deficient under-determined systems.

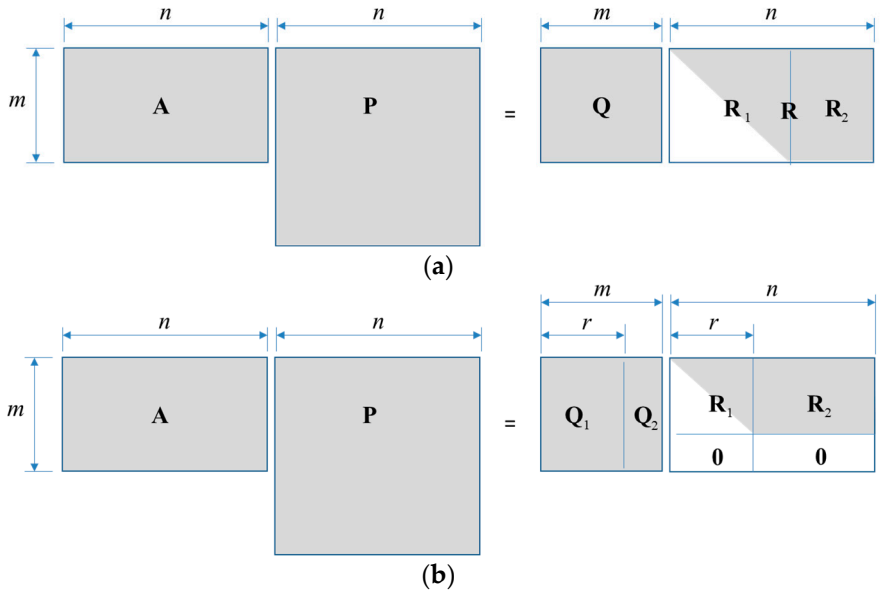


Figure 12. Schematic drawing of QR factorization for an under-determined system: (**a**) full-rank matrix; (**b**) rank-deficient matrix. Matrix blocks A, P, Q, Q_1, Q_2, R_1, and R_2 correspond to those defined in the text; shading is used only for visual grouping.

Although the proposed approach does not follow the Q-less QR factorization technique used for overdetermined systems, the need to avoid explicitly forming the Q matrix is conceptually related to the motivation discussed in [14]. Accordingly, the constraint Equation (3) can be rewritten as follows:

$$\begin{bmatrix}\mathbf{C}_f & \mathbf{C}_p\end{bmatrix}\begin{bmatrix}\mathbf{u}_f \\ \mathbf{u}_p\end{bmatrix} = \mathbf{b}, \text{ or equivalently, } \mathbf{C}_f\mathbf{u}_f + \mathbf{C}_p\mathbf{u}_p = \mathbf{b} \tag{20}$$

Here, the subscript f denotes free (or forced) DOFs, and p represents prescribed DOFs. The vector $\mathbf{u}_f$ can be partitioned into slave DOFs $\mathbf{u}_s$ and non-slave free DOFs $\mathbf{u}_q$. Applying QR factorization to $\mathbf{C}_f$,

$$\mathbf{C}_f\mathbf{P} = \mathbf{QR} \tag{21}$$

Using the orthogonality of **Q**, the transformed constraint equation becomes

$$\mathbf{R}\mathbf{P}^T\mathbf{u}_f + \mathbf{Q}^T\mathbf{C}_p\mathbf{u}_p = \mathbf{Q}^T\mathbf{b} \tag{22}$$

If $\mathbf{C}_f$ is full rank (i.e., no over-constraints exist), **R** is partitioned as $\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\end{bmatrix}$, and the constraint equation reduces to

$$\begin{bmatrix}\mathbf{I} & \mathbf{R}_1^{-1}\mathbf{R}_2\end{bmatrix}\mathbf{P}^T\mathbf{u}_f + \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{C}_p\mathbf{u}_p = \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{b} \tag{23}$$

This has the same form as Equation (4), meaning no redundant constraints exist. If $\mathbf{C}_f$ is rank-deficient (i.e., there exist over-constraints), the QR factorization becomes

$$\mathbf{C}_f\mathbf{P} = \begin{bmatrix}\mathbf{Q_1} & \mathbf{Q_2}\end{bmatrix}\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 \\ \mathbf{0} & \mathbf{0}\end{bmatrix} \tag{24}$$

and the rows corresponding to the zero block identify redundant or inconsistent constraints. For consistent redundancy, the transformed terms involving $\mathbf{Q}_2^T\mathbf{C}_p$ and $\mathbf{Q}_2^T b$ vanish, leading to

$$\begin{bmatrix}\mathbf{I} & \mathbf{R}_1^{-1}\mathbf{R}_2\end{bmatrix}\mathbf{P}^T\mathbf{u}_f + \mathbf{R}_1^{-1}\mathbf{Q}_1^T\mathbf{C}_p\mathbf{u}_p = \mathbf{R}_1^{-1}\mathbf{Q}_1^T\mathbf{b} \tag{25}$$

The quantities

$$\mathbf{C}_B = \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{C}_p \text{ and } \mathbf{b}_B = \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{b} \tag{26}$$

are obtained directly through the QR solver's "solve" operation, without explicitly forming $\mathbf{Q}$. The detection of inconsistent constraints is performed by checking whether the rows in

$$\mathbf{B} = \mathbf{C}_f\mathbf{R}_1^{-1}\mathbf{Q}_1^T\mathbf{C}_p \tag{27}$$

$$\mathbf{r} = \mathbf{C}_\mathrm{f}\mathbf{R}_1^{-1}\mathbf{Q}_1^\mathrm{T}\mathbf{b} \tag{28}$$

vanish after rank reduction.

For the QR factorization of dense matrices, solvers provided by LAPACK can be employed [15]. In contrast, developing efficient QR factorization solvers for sparse matrices remains a challenging problem in computational linear algebra and computer science. Publicly available sparse QR solvers include those implemented in Intel oneAPI MKL [16], SuiteSparseQR [17], and Eigen's SparseQR [18]. Among these, the sparse QR solver in Intel oneAPI MKL is limited to overdetermined systems, making it unsuitable for the present study. In this study, Eigen3's SparseQR solver was adopted. It employs the same algorithm as the multi-threaded, parallel SuiteSparseQR solver but is easier to use, albeit without parallel-processing support. For these reasons, Eigen's SparseQR solver was selected. Figure 13 presents a step-by-step summary of the proposed Q-less sparse QR factorization algorithm. To improve readability and emphasize practical applicability, the detailed algebraic derivation has been condensed accordingly.

The novelty of the proposed approach is that it applies a Q-less sparse QR factorization to an under-determined constraint system. Conventional sparse QR methods are developed for over-determined least-squares problems, where the orthogonal matrix Q is omitted primarily to reduce memory usage. In contrast, the present constraint-elimination problem is inherently under-determined, and forming Q would immediately result in a dense matrix, eliminating sparsity and greatly increasing computational cost. Therefore, the Q-less formulation is not only an efficiency consideration but a necessity for preserving sparsity while enabling stable rank detection and the identification of redundant constraints.

Both the block-wise Gauss elimination-based algorithm and the Q-less sparse QR factorization algorithm require a pivot tolerance to determine numerical rank deficiency, but their sensitivity to small pivots differs substantially. In the block-wise Gauss elimination-based algorithm, the largest entry in each row is selected as the pivot, and rows with $|\text{pivot}| \leq 10^{-5}$ are treated as numerically dependent and moved to the bottom of the block. However, small but nonzero pivots directly affect the elimination process because the multipliers A(i,k)/A(k,k) grow large when A(k,k) is close to zero. This may amplify roundoff errors, increase fill-in, or even lead to numerical breakdown.

Given $\mathbf{Cu} = \mathbf{b}$

1. Partition DOFs: $\mathbf{C}_f\mathbf{u}_f + \mathbf{C}_p\mathbf{u}_p = \mathbf{b}$ [Eq. 20]
2. Sparse QR factorization with column pivoting: $\mathbf{C}_f\mathbf{P} = \mathbf{QR}$ [Eq. 21]
   - $\mathbf{Q}$ is NOT formed (Q-less)
3. Compute $\mathbf{Q}^T\mathbf{C}_p$ and $\mathbf{Q}^T\mathbf{b}$ applying $\mathbf{Q}^T$ implicitly using QR solver [used in Eqs. 22-26] (implicitly via solver.solve())
4. Rank analysis and constraint classification

   If full rank → no overconstraints [Eq. 23]

   If rank-deficient:

   $$\mathbf{C_f P} = [\mathbf{Q_1} \quad \mathbf{Q_2}]\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2 \\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$ [Eq. 24]

   - Rows associated with $\mathbf{Q_2}$ identify redundent/inconsistent constraints
   - Check consistency:

     $\mathbf{Q}_2^T\mathbf{C}_p = \mathbf{0}$ and $\mathbf{Q}_2^T\mathbf{b} = \mathbf{0}$ → redundant

     Otherwise → Inconsistent
5. Construct reduced constraint system : $\mathbf{C}_B = \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{C}_p$ and $\mathbf{b}_B = \mathbf{R}_1^{-1}\mathbf{Q}^T\mathbf{b}$ [Eq. 26]

Figure 13. Q-less sparse QR factorization algorithm.

In contrast, the Q-less sparse QR factorization algorithm is based on orthogonal transformations whose numerical stability is independent of pivot scaling. The pivot threshold affects only the rank-decision logic. Eigen's default threshold $\tau_{default} = 20(m+n)\|A\|_2\varepsilon$ can fall below 10^{-10} for matrices of size 10^4–10^5, which is insufficient to detect near-dependent rows. For this reason, we adopt $\tau = 10^{-5}$, which provides a practical compromise between accurate rank detection and numerical stability for the constraint matrices used in this study.

3.3. Comparison of Algorithms in Large-Scale Analysis Models

This section compares the performance of the block-wise Gauss elimination-based algorithm and the Q-less sparse QR factorization algorithm using a large-scale fluid–structure–soil interaction (FSSI) system. The computational model consists of near-field and far-field soil regions represented by solid elements, a storage tank represented by shell elements, and an internal fluid domain formulated using acoustic elements. Constraint equations arise at several coupling interfaces, including those between the near-field and far-field regions, between the tank wall and the near-field soil, and between the tank wall and the fluid domain. At the fluid–structure interface, although the fluid domain is described in terms of pressure degrees of freedom, an equivalent normal displacement degree of freedom is introduced on the fluid side to ensure kinematic compatibility with the structural wall. These multiple coupling interfaces generate a large number of interdependent constraint equations, resulting in a highly complex constraint system. Additional details of the FSSI formulation, including both conformal and non-conformal coupling schemes, are provided in Appendix A, and the present analysis adopts the non-conformal approach.

To systematically evaluate the scalability and efficiency of the proposed Q-less sparse QR factorization method, a mesh-refinement study was conducted using the FSSI model. The geometric configuration of the near-field, far-field, tank, and fluid domains was kept constant, while the mesh refinement level was scaled from 50% to 110%. Here, 100% corresponds to the reference mesh used in the original FSSI analysis, whereas 50–90% represent coarser discretizations and 110% denotes an additional finer mesh used only for performance assessment. As the mesh is refined, the number of degrees of freedom (DOFs) and constraint equations increases substantially, leading to a progressively larger and more

redundant constraint matrix. Figure 14a,b illustrate the two representative mesh densities—50% (coarse) and 100% (reference fine mesh)—highlighting how mesh refinement affects the size of the coupled system. For each mesh configuration, both the block-wise Gauss elimination method and the Q-less sparse QR factorization method were executed three times, and the average computation times are summarized in Table 1. All computations were performed on a workstation equipped with an Intel(R) Core(TM) i9-14900K CPU @ 3.20 GHz and 128 GB RAM.

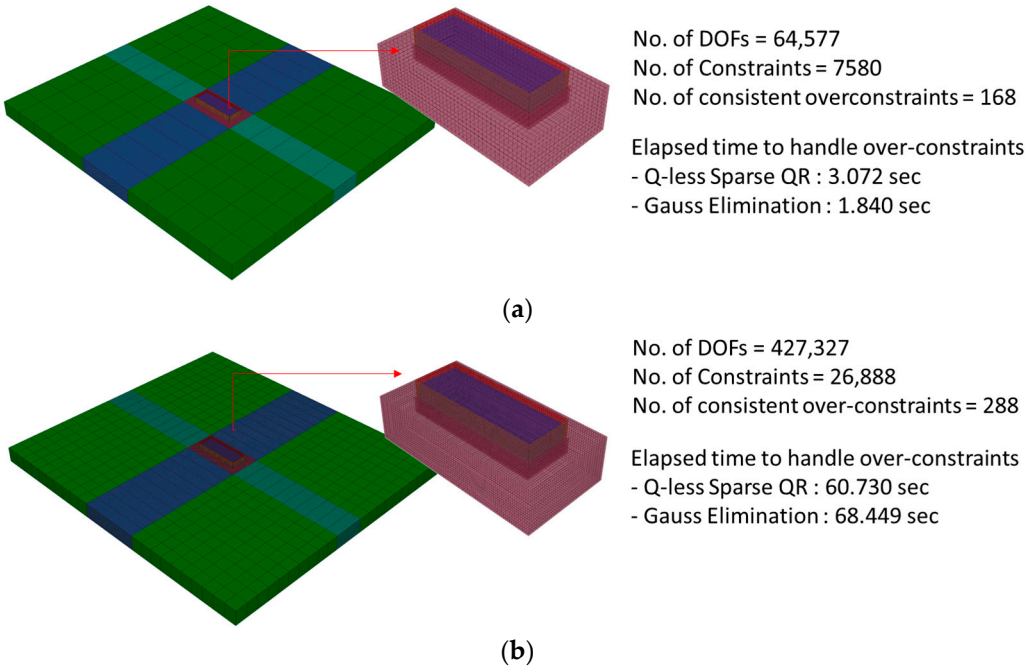


Figure 14. FSSI analysis model used in the mesh-refinement study: (**a**) 50% mesh (coarse) and (**b**) 100% mesh (fine).

Table 1. Summary of DOFs, numbers of constraints, consistent over-constraints, and elapsed computation times for the block-wise Gauss elimination and Q-less sparse QR methods, evaluated at mesh densities ranging from 50% (coarse) to 110% (fine).

		← Coarse		Mesh Refinement Level				Fine →
		50%	**60%**	**70%**	**80%**	**90%**	**100%**	**110%**
No. of DOFs		64,577	101,765	159,240	229,460	320,295	427,327	556,979
No. of given constraints		7580	10,300	13,808	17,408	22,360	26,888	31,832
No. of consistent over-constraints		168	192	216	240	276	288	300
No. of constraints after handling		7412	10,108	13,592	17,168	22,084	26,600	31,532
Elapsed time (s) of Q-less sparse QR	Trial 1	3.061	5.929	11.510	20.999	39.294	60.456	99.192
	Trial 2	3.127	5.939	11.475	21.185	38.614	60.309	96.681
	Trial 3	3.029	5.992	11.442	20.908	38.267	61.426	96.195
	Average	3.072	5.953	11.475	21.031	38.725	60.730	97.356
Elapsed time (s) of block-wise Gauss Elimination-Based	Trial 1	1.807	4.894	10.239	20.795	42.539	67.737	126.666
	Trial 2	1.895	4.926	10.099	20.799	42.525	68.974	128.127
	Trial 3	1.818	4.853	10.149	20.954	43.309	68.636	128.739
	Average	1.840	4.891	10.162	20.849	42.791	68.449	127.844

Across the different mesh densities, the relative performance of the two algorithms varies systematically. For coarse meshes (50–60%), the block-wise Gauss elimination

method is faster because the constraint matrix is smaller and fill-in remains limited. As the mesh becomes moderately refined (70–80%), the performance gap between the two methods narrows, and the computation times become comparable. For fine meshes (90–110%), however, the Q-less sparse QR factorization exhibits a clear advantage. As both the matrix size and the number of redundant constraints grow, SparseQR achieves significantly shorter computation times, particularly at the 100% and 110% mesh refinement levels. These trends are visualized in Figure 15, which plots the elapsed computation time against the number of constraints for each mesh level. The block-wise Gauss elimination method shows near-linear scaling in the coarse-mesh regime but deteriorates rapidly once the constraint matrix becomes larger and more redundant. In contrast, the Q-less sparse QR factorization maintains favorable scaling behavior throughout the refinement range, demonstrating markedly better performance for large systems. These results indicate that while Gauss elimination may be efficient for small or moderately sized constraint matrices, the Q-less sparse QR algorithm provides superior scalability and numerical robustness for the large, highly redundant constraint systems encountered in realistic FSSI problems.

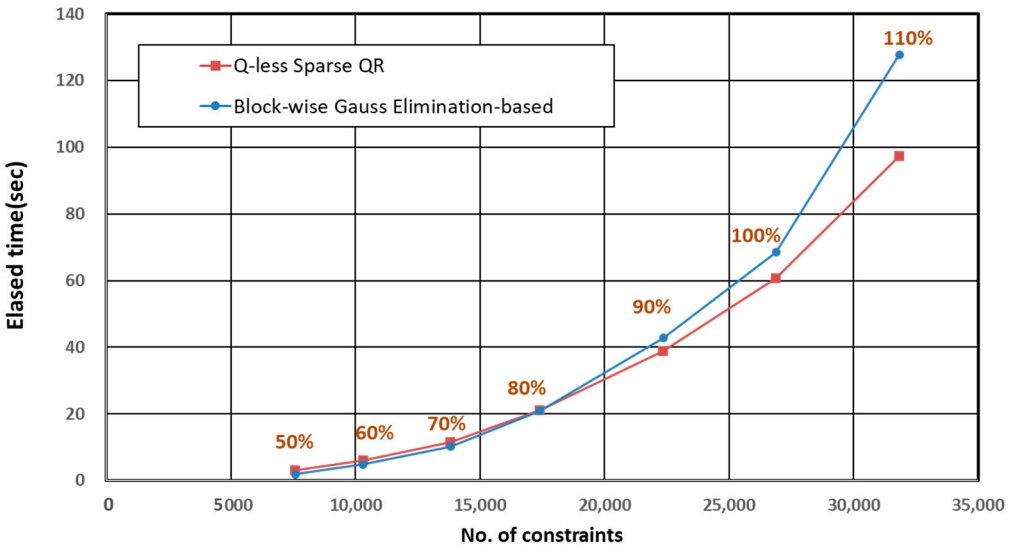


Figure 15. Computation time versus number of constraints for the block-wise Gauss elimination and Q-less sparse QR algorithms under mesh refinement (50–110%).

Parallel processing was not employed in this study, as no Gauss elimination solver for non-square matrices currently supports parallel computation. However, since SuiteSparseQR [17] provides a parallel implementation of sparse QR factorization, applying it to the proposed Q-less sparse QR factorization algorithm is expected to yield further improvements in computational efficiency.

It should be emphasized that this remark is intended only as a potential direction for future enhancement. Parallel Gauss elimination for non-square constraint matrices is not available, and incorporating the parallel version of SuiteSparseQR into the present solver framework would require substantial restructuring and recomputation of the entire FSSI example. Therefore, no parallel-performance test is included in this study, and the discussion of parallelization is presented as a possible extension rather than an experimentally validated result.

It is worth noting that commercial finite element solvers such as Abaqus and ANSYS do not provide an explicit numerical algorithm for resolving redundant or inconsistent constraint equations. These solvers typically issue warning messages when an over-constraint is detected and expect users to modify the model manually. Therefore, a direct algorithmic

comparison with commercial codes is not feasible, because no reference numerical solution is available for such systems.

It should be noted that an inconsistent constraint system does not possess a solution, and therefore, the role of the proposed method in such cases is limited to detecting inconsistency rather than producing a reduced, solvable constraint set. Because physically meaningful engineering models such as the FSSI system necessarily employ compatible coupling conditions, inconsistent constraint configurations do not arise in practice. For this reason, the demonstration of inconsistent cases is restricted to theoretical examples, whereas the FSSI example illustrates the behavior of the algorithm under large-scale but consistent over-constraints.

4. Conclusions

In this study, we addressed the challenge of over-constraints in finite element models with complex constraint relationships by proposing a Q-less sparse QR factorization algorithm. Traditional methods—such as the transformation method, Lagrange multiplier method, and penalty method—often suffer from small or zero pivot values, which can lead to non-convergence or slow convergence of solutions. Furthermore, the reuse of slave DOFs across multiple constraint equations complicates the modeling process.

The algorithm previously proposed by Cho [5] was reviewed, highlighting its effectiveness for relatively simple systems but also its computational limitations when applied to large-scale models.

To overcome these limitations, we developed a new algorithm that exploits the sparsity of the constraint matrix and performs sparse QR factorization without explicitly forming the dense **Q** matrix. This approach significantly enhances computational efficiency and reduces memory usage.

The proposed Q-less sparse QR factorization algorithm reformulates the constraint equation to separate forced and prescribed DOFs. By applying QR factorization to the constraint matrix and leveraging the orthogonality property of **Q**, the algorithm effectively resolves over-constraint conditions without generating **Q** explicitly. It also distinguishes between consistent and inconsistent over-constraints, thereby ensuring accurate and stable solutions.

Comparative analyses between the block-wise Gauss elimination-based and Q-less sparse QR factorization algorithms demonstrated that the latter performs more efficiently in large-scale problems involving numerous and complex constraint relationships. When the number of constraints and over-constraint conditions increases, the proposed algorithm outperforms the block-wise Gauss elimination-based approach in both computation time and overall efficiency. Although parallel processing was not employed in this study, integrating a parallel implementation such as SuiteSparseQR [15] is expected to further accelerate computation and enhance scalability.

In conclusion, the proposed Q-less sparse QR factorization algorithm provides a robust and efficient framework for handling over-constraints in finite element models, particularly for large-scale and highly constrained systems. Future research will focus on incorporating parallel computing capabilities and extending the algorithm to multiphysics and nonlinear analysis domains.

Author Contributions: Conceptualization, J.-R.C. and J.H.L.; methodology, J.-R.C. and K.C.; software, J.-R.C.; validation, J.-R.C., K.C., and H.Y.; formal analysis, J.-R.C.; investigation, J.-R.C. and H.Y.; resources, J.-R.C. and D.S.R.; data curation, H.Y.; writing—original draft preparation, J.-R.C.; writing—review and editing, D.S.R. and J.H.L.; visualization, J.-R.C.; supervision, J.H.L.; project administration, D.S.R.; funding acquisition, D.S.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work is financially supported by the Korea Ministry of Climate, Energy and Environment (MCEE) as "Technology development project to optimize planning, operation, and maintenance of urban flood control facilities" (RS-2024-00397821).

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Acknowledgments: The authors would like to thank the Korea Institute of Civil Engineering and Building Technology (KICT) for providing computational resources and technical support. During the preparation of this manuscript, the authors used ChatGPT-5 (OpenAI, model release November 2025) to improve the language and readability. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of this publication.

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

Abbreviations

The following abbreviations are used in this manuscript:

DOF	Degree of Freedom
FE	Finite Element
FEA	Finite Element Analysis
FSSI	Fluid–Structure–Soil Interaction
FSI	Fluid–Structure Interaction
CPU	Central Processing Unit
MKL	Math Kernel Library
QR	Orthogonal–Triangular (QR) Factorization
LU	Lower–Upper (LU) Factorization
SVD	Singular Value Decomposition
CSR	Compressed Sparse Row
KICT	Korea Institute of Civil Engineering and Building Technology
MOE	Ministry of Environment (Republic of Korea)

Appendix A. Formulation of the FSSI Problem

The formulation of the fluid–structure–soil interaction (FSSI) problem can be categorized into two coupling schemes: conformal and non-conformal. Section A.1 describes the conformal coupling formulation, where nodal compatibility is directly enforced at the interface, while Section A.2 presents the non-conformal coupling formulation, which introduces constraint equations to ensure displacement and traction continuity between non-matching meshes.

Appendix A.1. Conformal Coupling Formulation

The soil–structure interaction (SSI) analysis is formulated by applying the domain reduction method (DRM) proposed by [19] as follows:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega}+\mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega}+\mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega}+\mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \end{bmatrix} = \begin{bmatrix} \mathbf{P}_i \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_b^{\text{eff}} \\ \mathbf{f}_e^{\text{eff}} \end{bmatrix} \tag{A1}$$

Here, $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ denote the mass, damping, and stiffness matrices, respectively; $\mathbf{P}_i$ is the external load vector; and $\mathbf{f}^{\text{eff}}$ is the effective load induced by ground motion. The subscripts i, b, and e refer to the interior of the near-field, the boundary, and the interior of

the far-field, respectively. The superscripts Ω and Ω^+ represent the near-field and far-field domains. $\mathbf{u}_i$ and $\mathbf{u}_b$ denote the absolute displacements in the near-field, while we represents the relative motion with respect to the free-field displacement in the far-field.

The governing equations of the fluid–structure interaction (FSI) system are expressed as follows when the fluid domain exists inside the near-field, and no external forces act within the fluid domain as expressed in [7]:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{0} \\ \rho\mathbf{S} & \mathbf{M}_p + \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & -\mathbf{S}^T \\ \mathbf{0} & \mathbf{K}_p \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_i \\ \mathbf{0} \end{bmatrix} \tag{A2}$$

where $\mathbf{p}$ is the pressure degree of freedom.

By applying the interface equilibrium condition $\mathbf{P}_i + \mathbf{F}_i = 0$, which ensures that the interaction forces at the coupling boundary are equal and opposite, the governing equation for the coupled FSSI system can be expressed as follows:

$$\begin{aligned} &\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega} + \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} & \mathbf{0} \\ \rho\mathbf{S} & \mathbf{0} & \mathbf{0} & \mathbf{M}_p + \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \\ \ddot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega} + \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \\ \dot{\mathbf{p}} \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} & -\mathbf{S}^T \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega} + \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_p \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_b^{\text{eff}} \\ \mathbf{f}_e^{\text{eff}} \\ \mathbf{0} \end{bmatrix} \end{aligned} \tag{A3}$$

Appendix A.2. Non-Conformal Coupling Formulation

In practical applications, modeling a complex FSSI system using a conformal mesh is often difficult. In such cases, a nonconformal mesh can be used. The FSSI system corresponding to Equation (A1) can be expressed for an unaligned mesh as follows, subject to the constraint condition defined in Equation (A5):

$$\begin{aligned} &\begin{bmatrix} \mathbf{M}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \end{bmatrix} + \begin{bmatrix} \mathbf{C}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{K}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \end{bmatrix} = \begin{bmatrix} \mathbf{P}_i \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_b^{\text{eff}} \\ \mathbf{f}_e^{\text{eff}} \end{bmatrix} \end{aligned} \tag{A4}$$

where the compatibility between the near-field and far-field domains is enforced by the following constraint equation:

$$\mathbf{C}^s \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \end{bmatrix} = \mathbf{0} \tag{A5}$$

Here, $\mathbf{C}^s$ represents the constraint matrix that enforces compatibility between the near-field and far-field domains, which are modeled independently.

In the FSI system, the same concept can be applied as shown in [20] as follows:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \rho\mathbf{S} & \mathbf{M}_p + \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_n \\ \ddot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_n \\ \dot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{S}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_p \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_n \\ \mathbf{p} \end{bmatrix} = - \begin{bmatrix} \mathbf{F}_i \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{A6}$$

Here, $\mathbf{u}_n$ denotes the normal displacement of the structure defined at the fluid–structure interface, which satisfies the following constraint condition.

$$\mathbf{C}^w \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_n \end{bmatrix} = \mathbf{0} \tag{A7}$$

Here, $\mathbf{C}^w$ represents the constraint matrix that enforces compatibility between the structural displacement and the equivalent normal displacement defined on the fluid side at the fluid–structure interface. This constraint, in conjunction with $\mathbf{S}$, ensures consistent coupling between the solid and fluid domains by linking the structure's boundary motion to the corresponding normal motion of the acoustic fluid field.

By applying the condition $\mathbf{P}_i + \mathbf{F}_i = 0$, the governing equation for the coupled FSSI system can be derived accordingly.

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega} + \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \rho\mathbf{S} & \mathbf{M}_p + \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \\ \ddot{\mathbf{u}}^n \\ \ddot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega} + \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \\ \dot{\mathbf{u}}^n \\ \dot{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega} + \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{S}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_p \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \\ \mathbf{u}^n \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_b^{\text{eff}} \\ \mathbf{f}_e^{\text{eff}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{A8}$$

References

1. Abaqus, Inc. *Abaqus Analysis User's Manual*, Version 6.7; Abaqus, Inc.: Providence, RI, USA, 2007.
2. Boulbes, R.J. *Troubleshooting Finite-Element Modeling with Abaqus*; Wiley: Hoboken, NJ, USA, 2020.
3. ANSYS, Inc. *Release 11.0 Documentation for ANSYS—Multibody Analysis Guide*; ANSYS, Inc.: Canonsburg, PA, USA, 2007.
4. ADINA R&D, Inc. *Theory and Modeling Guide, ADINA System 8.3*; ADINA R&D, Inc.: Watertown, MA, USA, 2005.
5. Cho, J.-R. Object-Oriented Finite Element Framework Using Hybrid Programming. Ph.D. Dissertation, Seoul National University, Seoul, Republic of Korea, 2009.
6. ANSYS, Inc. ANSYS Release 18.2. Available online: https://www.ansys.com (accessed on 18 January 2018).
7. Cook, R.D.; Malkus, D.S.; Plesha, M.E.; Witt, R.J. *Concepts and Applications of Finite Element Analysis*, 4th ed.; Wiley: New York, NY, USA, 2001.
8. Pękal, M.; Frączek, J. Comparison of selected formulations for multibody system dynamics with redundant constraints. *Arch. Mech. Eng.* **2016**, *63*, 93–110. [CrossRef]
9. Pękal, M.; Wojtyra, M. Constraint-matrix-based method for reaction and driving forces uniqueness analysis in overconstrained or overactuated multibody systems. *Mech. Mach. Theory* **2023**, *188*, 105368. [CrossRef]
10. Belytschko, T.; Liu, W.K.; Moran, B. *Nonlinear Finite Elements for Continua and Structures*; Wiley: Chichester, UK, 2000.
11. Crisfield, M.A. *Non-Linear Finite Element Analysis of Solids and Structures*; Wiley: Chichester, UK, 1996; Volume 1.
12. Riks, E. An incremental approach to the solution of snapping and buckling problems. *Int. J. Solids Struct.* **1979**, *15*, 529–551. [CrossRef]
13. Davis, T.A. *Direct Methods for Sparse Linear Systems*; SIAM: Philadelphia, PA, USA, 2006.
14. Cornell Computer Science. Sparse Least Squares and Q-Less QR. Available online: https://www.cs.cornell.edu/ bindel/class/cs4220-s16/lec/2016-02-29-notes.pdf (accessed on 12 November 2025).
15. Anderson, E.; Bai, Z.; Bischof, C.; Blackford, L.S.; Demmel, J.; Dongarra, J.; Croz, J.D.; Greenbaum, A.; Hammarling, S.; McKenney, A.; et al. *LAPACK Users' Guide*, 3rd ed.; SIAM: Philadelphia, PA, USA, 1999.
16. Intel Corporation. Intel Math Kernel Library. Available online: https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onemkl.html(accessed on 12 November 2025).
17. Davis, T.A. SuiteSparseQR: Multifunctional multithreaded rank-revealing sparse QR factorization. *ACM Trans. Math. Softw.* **2011**, *38*, 8. [CrossRef]

18. Guennebaud, G.; Jacob, B.; Jolivet, R.; Falcou, J.; Gosmann, J.; Keck, M.; Iglberger, K.; Fistinig, M.; Thom, A.; Lelarge, M.; et al. Eigen v3. Available online: http://eigen.tuxfamily.org (accessed on 12 November 2025).
19. Bielak, J.; Loukakis, K.; Hisada, Y.; Yoshimura, C. Domain reduction method for three-dimensional earthquake modeling in localized regions, Part I: Theory. *Bull. Seismol. Soc. Am.* **2003**, *93*, 817–824. [CrossRef]
20. Cho, J.-R.; Lee, J.-H.; Cho, K.; Yoon, H. Acoustic interface element on nonconformal finite element mesh for fluid–structure interaction problem. *EESK J. Earthq. Eng.* **2023**, *27*, 163–170. (In Korean) [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.