AIMS Mathematics, 8(9): 21501–21530.
DOI: 10.3934/math.20231096
Received: 03 January 2023
Revised: 14 June 2023
Accepted: 19 June 2023
Published: 06 July 2023

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

Research article

A framework for establishing constraint Jacobian matrices of planar rigid-flexible-multibody systems

Lina Zhang, Xiaoting Rui, Jianshu Zhang*, Guoping Wang, Junjie Gu and Xizhe Zhang

Institute of Launch Dynamics, Nanjing University of Science and Technology, Jiangsu, China

* **Correspondence:** Email: jszhang@njust.edu.cn; Tel: +8602584315901.

Abstract: Constraint violation correction is an important research topic in solving multibody system dynamics. For a multibody system dynamics method which derives acceleration equations in a recursive manner and avoids overall dynamics equations, a fast and accurate solution to the violation problem is paramount. The direct correction method is favored due to its simplicity, high accuracy and low computational cost. This method directly supplements the constraint equations and performs corrections, making it an effective solution for addressing violation problems. However, calculating the significant Jacobian matrices for this method using dynamics equations can be challenging, especially for complex multibody systems. This paper presents a programmatic framework for deriving Jacobian matrices of planar rigid-flexible-multibody systems in a simple semi-analytic form along two paths separated by a secondary joint. The approach is verified by comparing constraint violation errors with and without the constraint violation correction in numerical examples. Moreover, the proposed method's computational speed is compared with that of the direct differential solution, verifying its efficiency. The straightforward, highly programmable and universal approach provides a new idea for programming large-scale dynamics software and extends the application of dynamics methods focused on deriving acceleration equations.

Keywords: constraint violation correction; Jacobian matrix; semi-analysis; direct differentiation method; numerical integration
Mathematics Subject Classification: 65D17, 65D30

1. Introduction

Multibody system dynamics (MSD) guides practical engineering applications by studying the movement of systems with multiple linked elements. Several main approaches for MSD have been developed since the 1970s [1–8]. The absolute coordinate methods [9], known for their programmatic nature and ability to handle both tree and non-tree systems, are commonly utilized. The approaches employ global coordinates to describe the motion of the system and obtain a general form for the dynamical equations, written as

$$\begin{cases} \boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{\Phi}_q^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{f} \\ \boldsymbol{\Phi}(\boldsymbol{q}, t) = \boldsymbol{0} \end{cases}, \tag{1}$$

where $\boldsymbol{q}$ denotes the generalized coordinates, which consist of the position and orientation of each rigid body-fixed coordinate system in the global inertial coordinate system, as well as the position and orientation of the flexible body floating coordinate system in the global inertial coordinate system and the flexible deformation generalized coordinates. $\ddot{\boldsymbol{q}}$, the second time derivative of $\boldsymbol{q}$, denotes the generalized accelerations. $\boldsymbol{M}$ represents a generalized mass matrix. $\boldsymbol{f}$ represents generalized forces on the system. $\boldsymbol{\lambda}$ consists of Lagrange multipliers. $\boldsymbol{\Phi}$ describes the constraint equations caused by the connection relation of all joint elements in the system. $\boldsymbol{\Phi}_q$ denotes the partial derivative of the constraint equations with respect to generalized coordinates.

However, when dealing with a complex multi-degree-of-freedom system, the solution of the dynamics can be quite challenging to derive. To overcome this issue, one technique utilized in structural dynamics is the static condensation method, which was extended by Alessandro Cammarata and colleagues to mechanisms and structures with internal joints [10]. Alternatively, one can use relative coordinates to establish the dynamics equations. The relative coordinate methods, developed by Robertson and Wittenburg [3,6], reduce the number of unknown variables and are used in tree structures with elegant types. Among the relative coordinate methods, highly programmatic and efficient approaches are coming to the fore, including Featherstone's algorithm [11,12], the reduced multibody system transfer matrix method (RMSTMM) [13,14], the Brandl method [15,16], etc. The approaches utilize hinge coordinates as generalized coordinates and avoid global dynamics equations with a system inertia matrix. They can fully describe the configuration of tree systems [17,18] in the form of ordinary differential equations, written as

$$\ddot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t), \tag{2}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the generalized coordinates, velocities and accelerations, which have analogous meanings to those given in Eq (1) but with much lower dimensions. The focus of these methods is to recursively derive generalized accelerations $\ddot{\boldsymbol{q}}$ for a mechanical multibody system by utilizing its state vectors $(\boldsymbol{q}, \dot{\boldsymbol{q}})$. However, in the context of closed-loop systems, the hinge coordinates are no longer independent [19]. To overcome this limitation, a set of differential algebraic equations (DAEs) is established by adding a set of algebraic equations that represent the constraint of cut-joints [4,7], as follows

$$\begin{cases} \ddot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) \\ \boldsymbol{c}(\boldsymbol{q}) = \boldsymbol{0} \end{cases}, \tag{3}$$

where $\boldsymbol{c}$ is assembled by the constraint equations of all cut joints. Despite the approaches' effectiveness, numerical solutions are prone to divergence, as indicated by the red line in Figure 1.

Ensuring the accuracy of solutions for DAEs is known to be numerically challenging. To address this issue, various methods have been developed and broadly classified into three main categories: (i) constraint stabilization approaches, (ii) coordinate partitioning methods and (iii) direct correction methods [20,21]. Constraint stabilization approaches involve developing and modifying the Baumgarte method, which introduces the control feedback theory. However, one problem with the Baumgarte stabilization method is the ambiguity of the stabilization parameters [22]. To address this issue, Lin and Huang [23] proposed a method that used the Runge-Kutta algorithm to determine the stabilization parameters. Other approaches, such as those by Zhang [24] and Flores [25], assisted with stabilization coefficient determination by combining it with Taylor expansion, but the methods lack catholicity.

Another approach is the coordinate partitioning method, which involves dividing the generalized coordinates into independent and dependent sets. Numerical integration is then performed for the independent generalized coordinates, while the constraint equations are solved for the dependent generalized coordinates [26]. However, this method is numerically inefficient, as it necessitates frequent changes to the set of independent coordinates, limiting its effectiveness [27]. Decomposing the Jacobi matrix, currently available as triangular decomposition [28], singular value decomposition [29] or orthogonal trigonometric decomposition [30], is the technique used to accomplish this method.

The direct correction method, which is popular in correcting violation errors, introduces position and velocity corrections based on direct integration [31]. Over the years, a variety of direct correction techniques have been developed [20,32–35]. Yoon et al. [36] established a direct correction method that directly corrects the values of state variables, leading to a better fit with the constraint equations. However, this method is limited to the position level. Yu and Chen [37] corrected violation errors by implementing constraints on both position and velocity levels. The effectiveness of this approach was demonstrated through a simple case, which was compared to the standard formulation and the Baumgarte method. The direct correction method offers a more accurate solution compared to stabilization methods and is less computationally costly than coordinate partitioning methods [38]. In dealing with constraint violations arising from the DAEs, a fast and accurate solution to violation errors is crucial. The direct correction method is favored due to its simplicity, high accuracy and low computational cost.

For dynamics methods involving global dynamics equations, the analytic form of the constraint Jacobian matrix can be obtained explicitly [39]. However, for methods avoiding global dynamics equations and using hinge coordinates, the value of the constraint Jacobian matrix becomes uncertain dealing with constraint violation problems in closed-loop systems. To address this issue, the direct differentiation method can be used in computational programming by taking partial derivatives of each quality encountered along the path rather than solving the Jacobian matrix in its specific form. As the complexity of the closed-loop system increases, the constraint Jacobian matrix of the system in the direct correction method becomes more complex. Nevertheless, this approach is computationally intensive and time-consuming when dealing with large and complex systems, which limits the

advantages of the direct correction method.

In this paper, semi-analytic Jacobian matrices of the direct correction method are derived to solve the divergence of algorithms. The correction method is first introduced in Section 2. A further attempt is to build a general framework for constraint Jacobian matrices. The form of constraint equations in the framework is shown in Section 3. The correlated variables affected by generalized coordinates in the constraint equations are encapsulated in the class of a secondary joint (cut-joint in a closed-loop system [40]). In Section 4, the variables of a secondary joint can be described by primary joints (other joints except for the secondary joint in the closed-loop system [40]) or flexible bodies. Among the correlation variables described, the ones affected by generalized coordinates are encapsulated in the corresponding class of the primary joint or the flexible body. The Jacobian matrices between different classes are programmatically derived in Section 5. The quantities of the secondary joint are cycled in the closed loop, sweeping through the classes of the primary joint or the flexible body. The solved Jacobian matrix is then substituted back into the constraint equations. Section 6 gives the pseudo-code for solving Jacobian matrices by direct differentiation. The corrected generalized coordinates and velocities are obtained using the Newton-Raphson formula in Section 7. Section 8 offers numerical examples validating the program. Finally, the concluding remarks are given in Section 9.

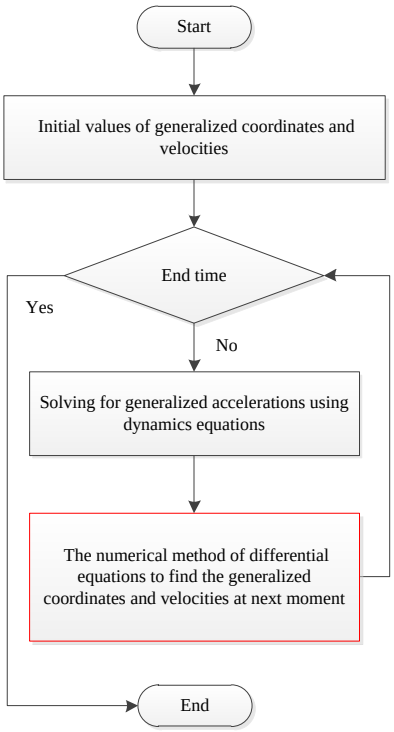

Figure 1. Numerical solution flow chart.

2. Constraint violation correction approach

The generalized coordinates of the closed-loop system are not independent, causing constraint violations. Corrected generalized coordinates and velocities can be obtained by the Newton-Raphson method. The Newton-Raphson formula of the constraint equation on position level can be written as

$$\begin{aligned} \boldsymbol{0} &= \boldsymbol{c}\left(\boldsymbol{q}+\Delta\boldsymbol{q}\right) \\ &\approx \boldsymbol{c}\left(\boldsymbol{q}\right)+\boldsymbol{c}_{\boldsymbol{q}}\left(\boldsymbol{q}\right)\Delta\boldsymbol{q}, \end{aligned} \tag{4}$$

where $\boldsymbol{c}$ denotes constraints on position level and $\boldsymbol{c}_{\boldsymbol{q}}\left(\boldsymbol{q}\right)$ is the constraint Jacobian matrix of the constraint equations with respect to the generalized coordinates. Here, the generalized coordinates include the modal coordinates and the coordinates of the primary joint elements.

Upon differentiating the aforementioned equation, the correction equation for the generalized velocity is obtained, wherein the well-known relation $\boldsymbol{c}_{\boldsymbol{q}} = \dot{\boldsymbol{c}}_{\boldsymbol{q}}$ exists. The correction formula can be expressed as

$$\begin{aligned} \boldsymbol{0} &= \dot{\boldsymbol{c}}\left(\boldsymbol{q},\dot{\boldsymbol{q}}+\Delta\dot{\boldsymbol{q}}\right) \\ &\approx \dot{\boldsymbol{c}}\left(\boldsymbol{q},\dot{\boldsymbol{q}}\right)+\dot{\boldsymbol{c}}_{\boldsymbol{q}}\Delta\dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{c}}\left(\boldsymbol{q},\dot{\boldsymbol{q}}\right)+\boldsymbol{c}_{\boldsymbol{q}}\Delta\dot{\boldsymbol{q}}. \end{aligned} \tag{5}$$

If the system has m constraint equations and n generalized coordinates, the constraint Jacobian matrix is given as

$$\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial c_1}{\partial q_1} & \frac{\partial c_1}{\partial q_2} & \cdots & \frac{\partial c_1}{\partial q_n} \\ \frac{\partial c_2}{\partial q_1} & \frac{\partial c_2}{\partial q_2} & \cdots & \frac{\partial c_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial c_m}{\partial q_1} & \frac{\partial c_m}{\partial q_2} & \cdots & \frac{\partial c_m}{\partial q_n} \end{bmatrix}. \tag{6}$$

3. Form of constraint equations of a secondary joint

The constraint equations of a secondary joint on both position and velocity levels can be established. The description of secondary joints, as shown in Figure 2, is based on the works of Roberson [40]. A span system is generated by cutting the joint which is called the secondary joint. The joints still in the span system are called primary joints. A joint connects two bodies of the system. One of the two bodies is called the base body and the other one is the moving body of the joint. The points B and M are fixed on the secondary joint, connecting the base body and the moving body, respectively.

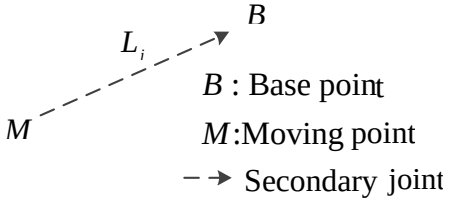

Figure 2. The sketch of a secondary joint.

For example, a planar pin joint in the j th closed loop labeled L_j is cut. The relative displacement of the pin joint moving in the plane is restricted. The constraint equation on position level could be expressed as

$$\boldsymbol{c}_{L_j} = \left[\boldsymbol{r}_{\mathrm{OM}} - \boldsymbol{r}_{\mathrm{OB}}\right]_{L_j} = \boldsymbol{0}, \tag{7}$$

and the constraint equation on velocity level could be expressed as

$$\dot{\boldsymbol{c}}_{L_j} = \left[\dot{\boldsymbol{r}}_{\mathrm{OM}} - \dot{\boldsymbol{r}}_{\mathrm{OB}}\right]_{L_j} = \boldsymbol{0}, \tag{8}$$

where $\boldsymbol{r}_{\mathrm{OB}}$ denotes the position of the base point w.r.t. the inertial frame. $\boldsymbol{r}_{\mathrm{OM}}$ denotes the position of the moving point. $\dot{\boldsymbol{r}}_{\mathrm{OB}}\left(\dot{\boldsymbol{r}}_{\mathrm{OM}}\right)$ is the absolute velocity of the base (moving) point. Note that if the vector in the plane is $\boldsymbol{r} = [x \quad y]^{\mathrm{T}}$, then its cross product matrix $\tilde{\boldsymbol{r}} = \left[-y \quad x\right]$ is defined. Required variables $\boldsymbol{r}_{\mathrm{OM}}, \boldsymbol{r}_{\mathrm{OB}}, \dot{\boldsymbol{r}}_{\mathrm{OM}}, \dot{\boldsymbol{r}}_{\mathrm{OB}}$ in Eqs (7) and (8) will be encapsulated in the class of the secondary joint, as shown in Figure 3. Function CalculateD0ConstraintEq() denotes constraints on position level. Analogically, the function CalculateD1ConstraintEq() gathers the process of constraints on velocity level. The Jacobian matrix, also known as the matrix of partial derivatives, is an essential tool for making accurate corrections in the system. The process of solving the constraint Jacobian matrix is put into the function CalculateConstraintJacobian. The process of solving the constraint Jacobian matrix is shown in the following.

SecondaryHingeElement
+R_om: Vector3d +dR_om: Vector3d +R_ob: Vector3d +dR_ob: Vector3d
+CalculateD0ConstraintEq() +CalculateD1ConstraintEq() +CalculateConstraintJacobian()

Figure 3. UML diagram of a secondary joint.

4. Intermediate variables of partial derivatives of constraints with respect to state vectors

The constraint equations established are ultimately affected since the position of the secondary joint is affected by state vectors of a system. The effect of each state vector in the system on the constraint equations will be considered here. For the pin secondary joint mentioned above, focusing on a specific column $\partial \boldsymbol{c}/\partial q_i \left(i = 1 \cdots n\right)$ in Eq (6), there is

$$\frac{\partial \boldsymbol{c}}{\partial q_i} = \frac{\partial \boldsymbol{r}_{\mathrm{OM}}}{\partial q_i} - \frac{\partial \boldsymbol{r}_{\mathrm{OB}}}{\partial q_i}. \tag{9}$$

For convenience, two separate paths are discussed here. If element i is associated with the moving point of the pin secondary joint, variables $\boldsymbol{r}_{\mathrm{OB}}$ are not related to generalized coordinate q_i. The partial derivative of $\boldsymbol{r}_{\mathrm{OM}}$ with respect to q_i in this path is $\boldsymbol{r}_{\mathrm{OM},q}$. According to Eq (9), the partial derivatives of the constraint equation on position level with respect to the generalized coordinates in this path can be written as

$$\boldsymbol{c}_{L_j,q_i} = \left[\boldsymbol{r}_{\mathrm{OM},q}\right]. \tag{10}$$

Similarly, on the other path, there is

$$\boldsymbol{c}_{L_j,q_i} = \left[\boldsymbol{r}_{\mathrm{OB},q}\right]. \tag{11}$$

Intermediate variables $\boldsymbol{r}_{\mathrm{OM},q_i}$, $\boldsymbol{r}_{\mathrm{OB},q_i}$ are collected in group I in a unified form. The generalized coordinates of a multi-rigid-flexible body system include the joint coordinate and the modal coordinates of flexible bodies. These are discussed below.

4.1. Jacobian intermediate for a primary joint

For elucidating the effect of generalized coordinates on constraint equations, subscript p can be replaced with base B or moving M point of the secondary joint. A primary joint moving in a plane connects to bodies with a root end described as R and a tip end described as T. $\boldsymbol{r}$ and $\boldsymbol{A}$ are chosen to represent the position level quantities considering the different types of primary joints. As shown in Figures 4 and 5, the kinematic quantities of the p node can be expressed as

$$\begin{cases} \boldsymbol{r}_{\mathrm{OP}} = \boldsymbol{r}_{\mathrm{OR}} + \boldsymbol{r}_{\mathrm{RT}} + \boldsymbol{r}_{\mathrm{TP}} = \boldsymbol{r}_{\mathrm{OR}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{l}_{\mathrm{RT}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT}}\boldsymbol{l}_{\mathrm{TP}} \\ \boldsymbol{A}_{\mathrm{OP}} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT}}\boldsymbol{A}_{\mathrm{TP}} \end{cases}, \tag{12}$$

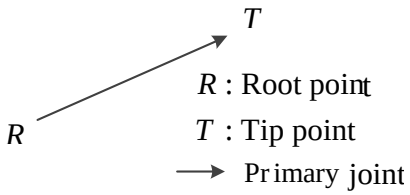

Figure 4. The sketch of a primary joint moving in a plane.

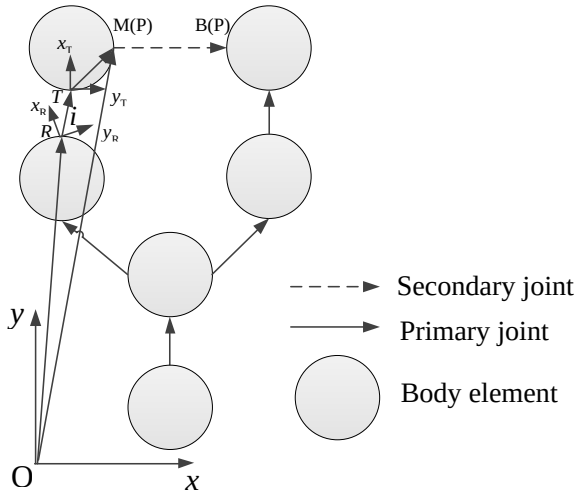

Figure 5. The sketch of the geometrical relationship.

where subscript O denotes the inertial coordinate system. Subscripts R and T denote the body-fixed coordinate system with origin R and T, respectively. $\boldsymbol{r}_{\mathrm{OP}}$ denotes the position of the P point w.r.t. the inertial frame. $\boldsymbol{l}_{\mathrm{RT}}$ describes position vector from the R point to the T point described in the body-fixed coordinate system with the R point. Cosine matrix $\boldsymbol{A}_{\mathrm{OP}}$ for the coordinate transformation from the body-fixed coordinate system with the P point to the inertial frame reads as

$$\boldsymbol{A}_{\mathrm{OP}} = \begin{bmatrix} \cos\theta_{\mathrm{OP}} & -\sin\theta_{\mathrm{OP}} \\ \sin\theta_{\mathrm{OP}} & \cos\theta_{\mathrm{OP}} \end{bmatrix}. \tag{13}$$

Generalized coordinates q_i in the equation only impact the parameters of the element, namely, $\boldsymbol{l}_{\mathrm{RT}}$ and $\boldsymbol{A}_{\mathrm{RT}}$. Therefore, variables in Eq (12) varying with generalized coordinate $\boldsymbol{q}_{H_i}$ can be expressed as

$$\begin{cases} \boldsymbol{r}_{\mathrm{OP},q} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{l}_{\mathrm{RT},q} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT},q}\boldsymbol{l}_{\mathrm{TP}}, \\ \boldsymbol{A}_{\mathrm{OP},q} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT},q}\boldsymbol{A}_{\mathrm{TP}}, \end{cases} \tag{14}$$

where the partial derivatives with respect to the generalized coordinates are indicated as subscripts, denoted as $\partial \boldsymbol{r}_{\mathrm{OM}} / \partial q_i = \boldsymbol{r}_{\mathrm{OM},q}$.

Variables $\boldsymbol{l}_{\mathrm{RT},q}$, $\boldsymbol{A}_{\mathrm{RT},q}$ changing with the generalized coordinate of a primary joint are encapsulated in group II of the primary joint class. The unified modeling language (UML) diagram [41] of a primary joint is provided in Figure 6. For convenience, the above derivation is loaded in the function PreprocessConstraintJacobian of the primary joint.

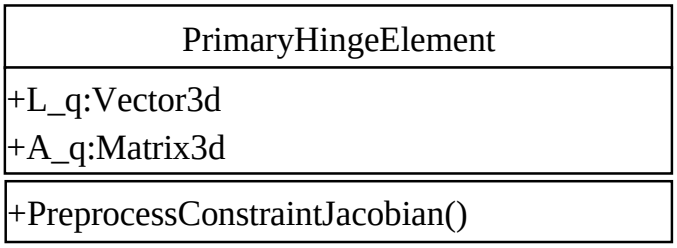

Figure 6. UML diagram of primary joint.

A smooth pin joint moving in a plane is taken as an example to illustrate the main process in detail, as shown in Figure 6. Pin joint i rotates around the z-axis of the body-fixed coordinate system with the T point, which has only one generalized coordinate. The generalized coordinate can be described as

$$\boldsymbol{q}_{H_i} = [\theta_z]. \tag{15}$$

Variables in group II can be described here. The pin joint is rotated around the z-axis by an angle θ_{RT}, and it has no relative slip along the z-axis, $\boldsymbol{l}_{\mathrm{RT}} = \boldsymbol{0}$. $\boldsymbol{A}_{\mathrm{RT}}$ is a direction cosine matrix corresponding to a simple rotation around the z-axis in the body-fixed frame. The corresponding $\boldsymbol{A}_{\mathrm{RT}}$, $\boldsymbol{l}_{\mathrm{RT}}$ can be described as

$$\begin{cases} \boldsymbol{l}_{\mathrm{RT}} = \boldsymbol{0}, \\ \boldsymbol{A}_{\mathrm{RT}} = \boldsymbol{A}_{\mathrm{z}}(\theta_{\mathrm{z}}), \end{cases} \tag{16}$$

Variables $\boldsymbol{l}_{\mathrm{RT}}$, $\boldsymbol{A}_{\mathrm{RT}}$ with respect to generalized coordinates can be read as

$$\begin{cases} \dfrac{\partial \boldsymbol{l}_{\mathrm{RT}}}{\partial \theta_{\mathrm{z}}} = \boldsymbol{0}, \\ \dfrac{\partial \boldsymbol{A}_{\mathrm{RT}}}{\partial \theta_{\mathrm{z}}} = \boldsymbol{A}_{\mathrm{z}}(\theta_{\mathrm{z}})\boldsymbol{D} = \boldsymbol{A}_{\mathrm{RT}}\boldsymbol{D}, \end{cases} \tag{17}$$

The above equations are substituted into Eq (14). The kinematic quantities influenced by the generalized coordinates become

$$\begin{cases} \dfrac{\partial \boldsymbol{r}_{\mathrm{OP}}}{\partial \theta_{\mathrm{z}}} = \tilde{\boldsymbol{r}}_{\mathrm{TP}}, \\ \dfrac{\partial \boldsymbol{A}_{\mathrm{OP}}}{\partial \theta_{\mathrm{z}}} = \boldsymbol{A}_{\mathrm{OT}}\boldsymbol{D}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{A}_{\mathrm{OP}} = \boldsymbol{D}\boldsymbol{A}_{\mathrm{OP}}, \end{cases} \tag{18}$$

where $\boldsymbol{r}_{\mathrm{TB}} = \boldsymbol{A}_{\mathrm{OT}}\boldsymbol{l}_{\mathrm{TB}}$. Meanwhile, the relationship $\boldsymbol{A}_{\mathrm{OT}}\boldsymbol{D}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}} = \boldsymbol{D}$ can be proven.

For further insight into partial derivatives of constraints with respect to state vectors of a primary joint, Eq (18) is elucidated. Only intermediate variables in group II are used for subsequent assembly in the framework for the system Jacobian matrix.

4.2. Jacobian intermediate for a flexible body

The tip and root ends, denoted by R and T, describe the points on a flexible body connected to joints, as shown in Figure 7. Under the assumption of small deformation, dynamics equations of flexible bodies are described by the widely used floating coordinate system. The motion of the body is regarded as a superposition of the large range motion of the floating coordinate system and the small elastic deformation [42–45].

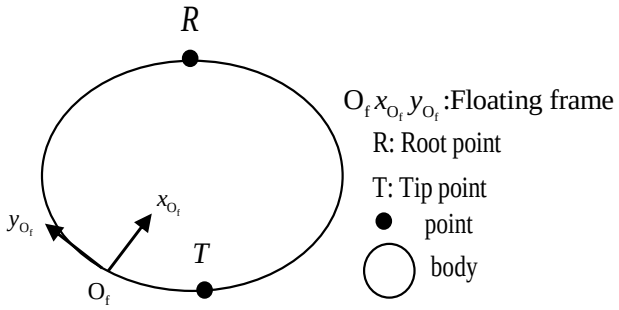

Figure 7. The sketch of a flexible body.

Using the kinematic quantities of a flexible body i, the kinematic quantities of node p for position level can be described as

$$\begin{cases} \boldsymbol{r}_{\mathrm{OP}} = \boldsymbol{r}_{\mathrm{OR}} + \boldsymbol{r}_{\mathrm{RO_f}} + \boldsymbol{r}_{\mathrm{O_fT}} + \boldsymbol{r}_{\mathrm{TP}} = \boldsymbol{r}_{\mathrm{OR}} - \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T}} + \boldsymbol{A}_{\mathrm{OT}}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TP}}, \\ \boldsymbol{A}_{\mathrm{OP}} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{A}_{\mathrm{TP}}, \end{cases} \tag{19}$$

where subscript f denotes a floating coordinate system. $\boldsymbol{\rho}_{\mathrm{R}}\left(\boldsymbol{\rho}_{\mathrm{T}}\right)$ denotes a deformed displacement vector from the origin of floating frame O_{f} to the $\mathrm{R}\left(\mathrm{T}\right)$ end described in the floating frame. Variables $\boldsymbol{r}_{\mathrm{OR}}, \boldsymbol{A}_{\mathrm{OR}}, \theta_{\mathrm{OR}}$ do not vary with the modal coordinates because they are transferred from the element associated to the flexible body, similar to the primary joint. $\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TP}}$ described in the body-fixed frame are independent of generalized coordinates and velocities, respectively. Only quantities $\boldsymbol{A}_{\mathrm{RO_f}}, \boldsymbol{\rho}_{\mathrm{R}}, \boldsymbol{\rho}_{\mathrm{T}}$ and $\boldsymbol{A}_{\mathrm{O_fT}}$ change with modal coordinates. Kinematic quantities for position level affected by the modal coordinates become

$$\begin{cases} \boldsymbol{r}_{\mathrm{OP},\boldsymbol{q}_{\mathrm{f}}} = -\boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{\rho}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R},\boldsymbol{q}_{\mathrm{f}}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{\rho}_{\mathrm{T}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T},\boldsymbol{q}_{\mathrm{f}}} \\ \qquad + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TP}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{A}_{\mathrm{O_fT},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TP}} \\ \boldsymbol{A}_{\mathrm{OP},\boldsymbol{q}_{\mathrm{f}}} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{A}_{\mathrm{TP}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{A}_{\mathrm{O_fT},\boldsymbol{q}_{\mathrm{f}}}\boldsymbol{A}_{\mathrm{TP}}. \end{cases} \tag{20}$$

Observing the above equation, intermediate variables $\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{\rho}_{\mathrm{R},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{A}_{\mathrm{O_fT},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{\rho}_{\mathrm{T},\boldsymbol{q}_\mathrm{f}}$ are selected in group II. The quantities $\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{\rho}_{\mathrm{R},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{\rho}_{\mathrm{T},\boldsymbol{q}_\mathrm{f}}$, $\boldsymbol{A}_{\mathrm{O_fT},\boldsymbol{q}_\mathrm{f}}$ vary with the modal coordinates. The others vary with the generalized velocities. The UML diagram of a flexible body is given in Figure 8. The above derivation is loaded in the function PreprocessConstraintJacobian.

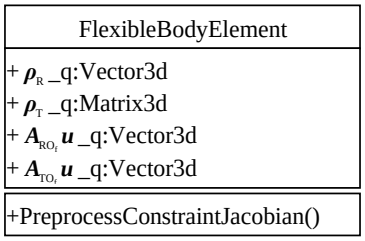

Figure 8. UML diagram of flexible body.

For flexible body i, the dynamics equations are derived based on the floating frame of reference. Component modal synthesis methods are applied to reduce the order of the dynamics equations of the flexible bodies [42–45]. The modal coordinates of the flexible body are denoted as

$$\boldsymbol{q}_{F_i} = \left[\boldsymbol{q}_\mathrm{f}\right]. \tag{21}$$

The quantities of the flexible body will be presented here, as shown in Figure 9. Radius vectors $\boldsymbol{\rho}_\mathrm{R}$, $\boldsymbol{\rho}_\mathrm{T}$ and angular deformation vector θ_{L,N_k} of the N_k node could be described as

$$\begin{cases} \boldsymbol{\rho}_\mathrm{R} = \boldsymbol{u}_{\mathrm{O_fR}} + \boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}}\boldsymbol{q}_\mathrm{f} \\ \boldsymbol{\rho}_\mathrm{T} = \boldsymbol{u}_{\mathrm{O_fT}} + \boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}}\boldsymbol{q}_\mathrm{f} \\ \theta_{L,\mathrm{N}_k} = \boldsymbol{\Phi}_{\theta,\mathrm{N}_k}\boldsymbol{q}_\mathrm{f} \end{cases} \tag{22}$$

where θ_{L,N_k} describes the attitude of the flexible body moving in the plane. $\boldsymbol{u}_{\mathrm{O_fR}}\left(\boldsymbol{u}_{\mathrm{O_fT}}\right)$ represents the original coordinate of node $\mathrm{R}\left(\mathrm{T}\right)$ in the undeformed configuration. The modal analysis and synthesis are employed to discretize deformation displacement $\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}}\boldsymbol{q}_\mathrm{f}\left(\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}}\boldsymbol{q}_\mathrm{f}\right)$. $\boldsymbol{\Phi}$ is a shape function, i.e., modal function matrix in modal analysis. Subscript $u\left(\theta\right)$ denotes deformation in the displacement (angle). The directional cosine matrix of the flexible body from the body-fixed frame $o_{\mathrm{N}_k}x_{\mathrm{N}_k}y_{\mathrm{N}_k}$ to the floating frame is represented as $\boldsymbol{A}_{\mathrm{O_fN}_k}$. Directional cosine matrix $\boldsymbol{A}_{\mathrm{O_fN}_k}$ can be expressed as

$$\boldsymbol{A}_{\mathrm{O_fN}_k} = \begin{bmatrix} \cos\theta_{L,\mathrm{N}_k} & -\sin\theta_{L,\mathrm{N}_k} \\ \sin\theta_{L,\mathrm{N}_k} & \cos\theta_{L,\mathrm{N}_k} \end{bmatrix}. \tag{23}$$

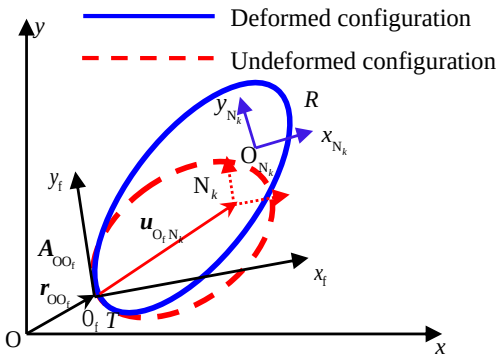

Figure 9. Deformation description of a flexible body.

Directional cosine matrix $\boldsymbol{A}_{\mathrm{O_f N_k}}$ cannot be reduced using the same method as that in a primary joint. However, the appearance of the directional cosine matrix $\boldsymbol{A}_{\mathrm{O_f N_k}}$ in the equation is inevitably multiplied by a variable independent of the modal coordinates. A mathematical relationship has been discovered, which is

$$\begin{aligned}\frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial \boldsymbol{q}_{\mathrm{f}}} &= \frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial \theta_{L,\mathrm{N_k}}}\frac{\partial \theta_{L,\mathrm{N_k}}}{\partial \boldsymbol{q}_{\mathrm{f}}} \\ &= \boldsymbol{D}\boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{N_k}}.\end{aligned} \tag{24}$$

The proof process is as follows. Since $\boldsymbol{u}$ is an unrelated variable, $\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}/\partial t$ can be obtained by the direct derivative rule as follows:

$$\frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial t} = \boldsymbol{D}\boldsymbol{A}_{\mathrm{O_f N_k}}\dot{\theta}_{L,\mathrm{N_k}}\boldsymbol{u}\cdot \tag{25}$$

The chain rule can be used for the same variable $\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}/\partial t$, giving

$$\frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial t} = \frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial \theta_{L,\mathrm{N_k}}}\frac{\partial \theta_{L,\mathrm{N_k}}}{\partial t} = \frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial \theta_{L,\mathrm{N_k}}}\dot{\theta}_{L,\mathrm{N_k}}. \tag{26}$$

Combining Eqs (25) and (26) expressed in different forms, there is

$$\frac{\partial \boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}}{\partial \theta_{L,\mathrm{N_k}}} = \boldsymbol{D}\boldsymbol{A}_{\mathrm{O_f N_k}}\boldsymbol{u}. \tag{27}$$

Substituting Eq (27) into the original formula, the relationships between relevant variables $\boldsymbol{A}_{\mathrm{RO_f},q}$, $\boldsymbol{A}_{\mathrm{O_f T},q}$, $\boldsymbol{u}$ and $\boldsymbol{q}_{\mathrm{f}}$ are

$$\begin{aligned}\frac{\partial \boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{u}}{\partial \boldsymbol{q}_{\mathrm{f}}} &= \frac{\partial \boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{u}}{\partial \theta_{L,\mathrm{O_fT}}}\frac{\partial \theta_{L,\mathrm{O_fT}}}{\partial \boldsymbol{q}_{\mathrm{f}}} \\ &= \boldsymbol{D}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{u}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}},\end{aligned} \tag{28}$$

$$\begin{aligned}\frac{\partial \boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{u}}{\partial \boldsymbol{q}_{\mathrm{f}}} &= \frac{\partial \boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{u}}{\partial \theta_{L.\mathrm{O_fR}}}\frac{\partial \theta_{L.\mathrm{O_fR}}}{\partial \boldsymbol{q}_{\mathrm{f}}} \\ &= -\boldsymbol{D}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{u}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}}.\end{aligned} \tag{29}$$

The intermediate variables in group II could be expressed as

$$\begin{cases}\dfrac{\partial \boldsymbol{\rho}_{\mathrm{R}}}{\partial \boldsymbol{q}_{\mathrm{f}}} = \boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}}, \\ \dfrac{\partial \boldsymbol{\rho}_{\mathrm{T}}}{\partial \boldsymbol{q}_{\mathrm{f}}} = \boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}}, \\ \dfrac{\partial \boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{u}}{\partial \boldsymbol{q}_{\mathrm{f}}} = -\boldsymbol{D}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{u}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}}, \\ \dfrac{\partial \boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{u}}{\partial \boldsymbol{q}_{\mathrm{f}}} = \boldsymbol{D}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{u}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}}.\end{cases} \tag{30}$$

Substituting Eq (30) into Eq (20), variables $\boldsymbol{r}_{\mathrm{OP},q}$, $\boldsymbol{A}_{\mathrm{OP},q}$ affected by state vectors of a flexible body can be read as

$$\begin{cases}\dfrac{\partial \boldsymbol{r}_{\mathrm{OP}}}{\partial \boldsymbol{q}_{\mathrm{f}}} = \boldsymbol{D}\left(\boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{T}} - \boldsymbol{r}_{\mathrm{TP}}\right)\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}} + \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}} + \boldsymbol{D}\boldsymbol{r}_{\mathrm{TP}}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}}, \\ \dfrac{\partial \boldsymbol{A}_{\mathrm{OP}}}{\partial \boldsymbol{q}_{\mathrm{f}}} = -\boldsymbol{A}_{\mathrm{OR}}\boldsymbol{D}\boldsymbol{A}_{\mathrm{RP}}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}} + \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{D}\boldsymbol{A}_{\mathrm{O_fP}}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}}.\end{cases} \tag{31}$$

The existence of Eq (31) makes it possible to see the partial derivatives of the kinematic quantities clearly and also facilitates the composition of the constraint Jacobian matrix in the following. Nonetheless, this step does not appear in the actual programmatic implementation but only facilitates understanding of the solution concepts.

5. Semi-analytic solution for constraint Jacobian matrix

For a system with arbitrary topologies, the Jacobian matrix is a given temporary matrix changed with time. The calculation of the Jacobian matrix becomes very complicated, so a framework is established to simplify computation:

5.1. Composition of the system Jacobian matrix

A programmatic derivation for the constraint Jacobian matrices is described using intermediate variables in groups I and II. It is known that the constraint Jacobian matrix for correcting the

generalized coordinates is identical to correcting generalized velocities. A closed-loop subsystem L_j in which the secondary pin joint j is associated with a primary pin joint i is taken as an example to explain in detail. The process of assembling Jacobian matrix $\boldsymbol{c}_{L_j,q_{H_i}}$ for generalized coordinate q_{H_i} is first demonstrated. Assume that pin joint i is associated with the base point of the secondary joint. Intermediate variables $\boldsymbol{r}_{\mathrm{OB},q}$ in group I can be expressed by variables in group II as

$$\boldsymbol{r}_{\mathrm{OB},q} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{l}_{\mathrm{RT},q} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT},q}\boldsymbol{l}_{\mathrm{TB}}. \tag{32}$$

Intermediate variables under the effect of the primary joint become

$$\frac{\partial \boldsymbol{r}_{\mathrm{OB}}}{\partial q_{H_i}} = \boldsymbol{A}_{\mathrm{OR}}\frac{\partial \boldsymbol{A}_{\mathrm{RT}}}{\partial \theta_z}\boldsymbol{l}_{\mathrm{TB}}, \tag{33}$$

Constraint Jacobian matrix $\boldsymbol{c}_{L_j,q_{H_i}}$ could be expressed as

$$\boldsymbol{c}_{L_j,q_{H_i}} = \left[-\boldsymbol{A}_{\mathrm{OR}}\left(\partial \boldsymbol{A}_{\mathrm{RT}}/\partial\theta_z\right)\boldsymbol{l}_{\mathrm{TB}}\right]_{L_j}, \tag{34}$$

Submitting Eq (17) into the above equation, we have

$$\boldsymbol{c}_{L_j,q_{H_i}} = -\tilde{\boldsymbol{r}}_{\mathrm{TB}}. \tag{35}$$

In the other path, pin joint i is associated with the moving point. The position of the moving point can be expressed as

$$\boldsymbol{r}_{\mathrm{OM},q} = \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{l}_{\mathrm{RT},q} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RT},q}\boldsymbol{l}_{\mathrm{TM}}. \tag{36}$$

Constraint Jacobian matrix $\boldsymbol{c}_{L_j,q_{H_i}}$ in this path could be expressed as

$$\boldsymbol{c}_{L_j,q_{H_i}} = \left[\boldsymbol{A}_{\mathrm{OR}}\left(\partial \boldsymbol{A}_{\mathrm{RT},q}/\partial\theta_z\right)\boldsymbol{l}_{\mathrm{TM}}\right]_{L_j} = \left[\tilde{\boldsymbol{r}}_{\mathrm{TM}}\right]_{L_j}. \tag{37}$$

Observing Eqs (35) and (37), the Jacobian matrix of the hinge i can be solved by calling the intermediate variables of the corresponding hinge element. Putting the calculated Jacobian matrix into the corresponding position, the Jacobian matrix for the closed-loop subsystem L_j is obtained. If the closed-loop subsystem L_j is without any flexible bodies, the corresponding Jacobian matrix can be written as

$$\boldsymbol{c}_{L_j,\boldsymbol{q}} = \left[\boldsymbol{c}_{L_j,q_{H_1}} \quad \cdots \quad \boldsymbol{c}_{L_j,q_{H_n}}\right]. \tag{38}$$

The assembly process can be applied for flexible bodies. Jacobian matrix $\boldsymbol{c}_{L_j,q_{F_i}}$ is derived as follows, if flexible body element i is associated with the base point of the secondary joint. According to Eq (20), kinematic quantities of the flexible body element are used to express intermediate quantities $\boldsymbol{r}_{\mathrm{OB},\boldsymbol{q}_{\mathrm{f}}}$. Intermediate quantities can be read as

$$\begin{aligned}\boldsymbol{r}_{\mathrm{OB},\boldsymbol{q}_\mathrm{f}} =& -\boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_\mathrm{f}}\boldsymbol{\rho}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R},\boldsymbol{q}_\mathrm{f}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_\mathrm{f}}\boldsymbol{\rho}_{\mathrm{T}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T},\boldsymbol{q}_\mathrm{f}} \\ &+\boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f},\boldsymbol{q}_\mathrm{f}}\boldsymbol{A}_{\mathrm{O_fT}}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TB}} + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{A}_{\mathrm{O_fT},\boldsymbol{q}_\mathrm{f}}\boldsymbol{A}_{\mathrm{OT}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TB}}\end{aligned}. \tag{39}$$

According to Eq (30), the partial derivative of $\boldsymbol{r}_{\mathrm{OB}}$ with respect to the modal coordinates can be written as

$$\begin{aligned}\frac{\partial \boldsymbol{r}_{\mathrm{OB}}}{\partial \boldsymbol{q}_\mathrm{f}} =& -\boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) + \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) \\ &+\boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) + \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO}}\boldsymbol{r}_{\mathrm{TB}}/\partial\boldsymbol{q}_\mathrm{f}\right) + \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\left(\partial\boldsymbol{A}_{\mathrm{OO_f}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TB}}/\partial\boldsymbol{q}_\mathrm{f}\right).\end{aligned} \tag{40}$$

According to the constraint equation of the secondary pin joint shown in Eq (7), constraint Jacobian matrix $\boldsymbol{c}_{L_j,q_{\mathrm{F_i}}}$ reads as

$$\begin{aligned}\boldsymbol{c}_{L_j,q_{\mathrm{F_i}}} =& \begin{bmatrix}\boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) + \boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) \\ -\boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO}}\boldsymbol{r}_{\mathrm{TB}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\left(\partial\boldsymbol{A}_{\mathrm{OO_f}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TB}}/\partial\boldsymbol{q}_\mathrm{f}\right)\end{bmatrix} \\ =& \left[\boldsymbol{D}\left(\boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{T}} - \boldsymbol{r}_{\mathrm{TB}}\right)\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}} + \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}} + \boldsymbol{D}\boldsymbol{r}_{\mathrm{TB}}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}}\right].\end{aligned} \tag{41}$$

Similarly, if flexible body i is associated with the moving point of the secondary joint, point P will be replaced with M in Eq (31). The partial derivative of the constraint equation on position level with respect to modal coordinates $q_{\mathrm{F_i}}$ can be written as

$$\begin{aligned}\boldsymbol{c}_{L_j,\boldsymbol{q}_{\mathrm{F_i}}} =& \begin{bmatrix}\boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) + \boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{R}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO_f}}\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) \\ -\boldsymbol{A}_{\mathrm{OO_f}}\left(\partial\boldsymbol{\rho}_{\mathrm{T}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\left(\partial\boldsymbol{A}_{\mathrm{RO}}\boldsymbol{r}_{\mathrm{TB}}/\partial\boldsymbol{q}_\mathrm{f}\right) - \boldsymbol{A}_{\mathrm{OR}}\boldsymbol{A}_{\mathrm{RO_f}}\left(\partial\boldsymbol{A}_{\mathrm{OO_f}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{TM}}/\partial\boldsymbol{q}_\mathrm{f}\right)\end{bmatrix} \\ =& \left[\boldsymbol{D}\left(\boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\rho}_{\mathrm{T}} - \boldsymbol{r}_{\mathrm{TB}}\right)\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{R}} - \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{R}} + \boldsymbol{A}_{\mathrm{OO_f}}\boldsymbol{\Phi}_{\boldsymbol{u},\mathrm{T}} + \boldsymbol{D}\boldsymbol{r}_{\mathrm{TM}}\boldsymbol{\Phi}_{\boldsymbol{\theta},\mathrm{T}}\right].\end{aligned} \tag{42}$$

The Jacobian matrix of the flexible body i can be solved by calling the intermediate variables of a flexible body. For the closed-loop subsystem L_j, the corresponding Jacobian matrix consists of that for each element, written as

$$\boldsymbol{c}_{L_j,\boldsymbol{q}} = \left[\boldsymbol{c}_{L_j,q_{H_1}} \quad \cdots \quad \boldsymbol{c}_{L_j,q_{H_n}} \,\middle|\, \boldsymbol{c}_{L_j,q_{\mathrm{F_1}}} \quad \cdots \quad \boldsymbol{c}_{L_j,q_{\mathrm{F}_n}}\right]. \tag{43}$$

The Jacobian matrix for all primary joints is described as $\boldsymbol{c}_{L_j,\boldsymbol{q}_H}$, and the Jacobian matrix for flexible bodies is described as $\boldsymbol{c}_{L_j,\boldsymbol{q}_F}$. The Jacobian matrix $\boldsymbol{c}_{L_j,q}$ can be composed as

$$\boldsymbol{c}_{L_j,\boldsymbol{q}} = [\boldsymbol{c}_{L_j,\boldsymbol{q}_H} \quad \boldsymbol{c}_{L_j,\boldsymbol{q}_F}], \tag{44}$$

where Jacobian matrix $\boldsymbol{c}_{L_j,\boldsymbol{q}_H}$ is composed of the Jacobian matrix for each primary joint swept through the path, i.e., $\boldsymbol{c}_{L_j,\boldsymbol{q}_H} = \left[\boldsymbol{c}_{L_j,\boldsymbol{q}_{H_1}} \quad \boldsymbol{c}_{L_j,\boldsymbol{q}_{H_2}} \quad \cdots \quad \boldsymbol{c}_{L_j,\boldsymbol{q}_{H_n}}\right]$. Jacobian matrix $\boldsymbol{c}_{L_j,\boldsymbol{q}_F}$ is composed of the Jacobian matrix for each flexible body swept through the path, i.e., $\boldsymbol{c}_{L_j,\boldsymbol{q}_F} = \left[\boldsymbol{c}_{L_j,\boldsymbol{q}_{\mathrm{F_1}}} \quad \boldsymbol{c}_{L_j,\boldsymbol{q}_{\mathrm{F_2}}} \quad \cdots \quad \boldsymbol{c}_{L_j,\boldsymbol{q}_{\mathrm{F}_n}}\right]$.

According to the topological structure of a multibody system, Jacobian matrix $\boldsymbol{c}_q$ can be obtained by assembling the Jacobian matrices of all sub-closed loops in the system, denoted as

$$\boldsymbol{c}_q = \begin{bmatrix} \boldsymbol{c}_{L_1,q} \\ \boldsymbol{c}_{L_2,q} \\ \vdots \\ \boldsymbol{c}_{L_n,q} \end{bmatrix}. \tag{45}$$

Then, the system Jacobian matrix can be expressed as

$$\boldsymbol{c}_{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{c}_{L_1,\boldsymbol{q}} \\ \boldsymbol{c}_{L_2,\boldsymbol{q}} \\ \vdots \\ \boldsymbol{c}_{L_n,\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_{L_1,\boldsymbol{q}_H} & \boldsymbol{c}_{L_1,\boldsymbol{q}_F} \\ \boldsymbol{c}_{L_2,\boldsymbol{q}_H} & \boldsymbol{c}_{L_2,\boldsymbol{q}_F} \\ \vdots & \vdots \\ \boldsymbol{c}_{L_n,\boldsymbol{q}_H} & \boldsymbol{c}_{L_n,\boldsymbol{q}_F} \end{bmatrix}. \tag{46}$$

The specific programming procedure for the system Jacobian matrix can be described below.

5.2. Assembling the system Jacobian matrix

The universalized composition process of the system Jacobian matrix based on the intermediate variables in groups I and II can be implemented by computer, as shown in Algorithm 1. The Jacobian matrix for the j th secondary joint is solved by sweeping through the primary joint elements and flexible body elements along the path from the base point of the j th secondary joint element to the root element. In this process, the PreprocessConstraintJacobian() function in the classes of the primary joint element and the flexible body element is called to solve the intermediate variables in group I. The same process takes place on the other path. Corresponding constraint Jacobian matrices $\boldsymbol{c}_{L_j,\boldsymbol{q}_{H_i}}$ and $\boldsymbol{c}_{L_j,\boldsymbol{q}_{F_i}}$ are obtained. According to the location of the generalized coordinates, constraint Jacobian matrices $\boldsymbol{c}_{L_j,\boldsymbol{q}_{H_i}}$ and $\boldsymbol{c}_{L_j,\boldsymbol{q}_{F_i}}$ are composed to obtain constraint Jacobian matrix $\boldsymbol{c}_{L_j,q}$ of the j th secondary joint. System Jacobian matrix $\boldsymbol{c}_q$ is obtained ultimately by assembling constrained Jacobian matrices $\boldsymbol{c}_{L_j,q}\left(j=1\ldots n\right)$ according to Eq (46). The pseudo-code of the computational process is illustrated in Algorithm 1.

In the programmatic derivation, multiple calculations of the intermediate variables in group II are avoided. The required variables can be obtained in the first calculation and called continuously in the subsequent calculations.

Algorithm 1. Computation of system Jacobian matrix

Function *assemblingsystemJacobianmatrix* (*i*:**int**, *ω*:**double**, *s*:**int**)
{Return Jacobian matrix $\boldsymbol{C}_q$ of the system}
{ $\boldsymbol{C}_{L,q}$ is a Jacobian matrix of the secondary joint element.}
{ *SecondaryJointElement* is an indicator of the secondary joint element.}
{ *PrimaryJointElement* is an indicator of the primary joint element.}
{ *FlexibleBodyElement* is an indicator of the flexible body element.}
for *k*=1 **to** the number of secondary joint elements **do**
  for *i* **from** element connected to the BasePoint of secondary joint *k* **to** the root element in closed-loop *k* **do**
    if element *i* is a primary joint element **then**
      SecondaryJointElement[*k*]. $\boldsymbol{C}_{L,q}$ [*i*] ← *SecondaryJointElement*[*k*].*Calculate*
      ConstraintJacobian() ← *PrimaryJointElement*[*i*].*PreprocessConstraintJacobian()*
    end
    if element *i* is a flexible body element **then**
      SecondaryJointElement[*k*]. $\boldsymbol{C}_{L,q}$ [*i*] ← *SecondaryJointElement*[*k*]. *Calculate*
      ConstraintJacobian() ← *FlexibleBodyElement*[*i*].*PreprocessConstraintJacobian()*
    end
  end
  for *j* **from** element connected to the MovingPoint of secondary joint *k* **to** the root element in closed-loop *k* **do**
    if element *j* is a primary joint element **then**
      SecondaryJointElement[*k*]. $\boldsymbol{C}_{L,q}$ [*j*] ← *SecondaryJointElement*[*k*].*Calculate*
      ConstraintJacobian() ← *PrimaryJointElement*[*j*].*PreprocessConstraintJacobian()*
    end
    if element *j* is a flexible body element **then**
      SecondaryJointElement[*k*]. $\boldsymbol{C}_{L,q}$ [*j*] ← *SecondaryJointElement*[*k*]. *Calculate*
      ConstraintJacobian() ← *FlexibleBodyElement*[*j*].*PreprocessConstraintJacobian()*
    end
  end
  $\boldsymbol{C}_q$ [k]←*SecondaryJointElement*[*k*]. $\boldsymbol{C}_{L,q}$
return $\boldsymbol{C}_q$

6. Direct differentiation for Jacobian matrix

The direct differentiation method obtains the Jacobian matrix by differentiating the constraint equations with respect to state variables in accordance with the chain rule of differentiation, which obtains an accurate calculation result.

The Jacobian matrix can be calculated row-wise. The same row is the partial derivatives of the constraints with respect to the same generalized coordinate. The partial derivatives in the same row are passed along the transfer path from the system root end to the position of the secondary joint recursively.

The pin joint is still used as an example with the constraint equation shown in Eq (7). In this case, the partial derivative of r with respect to the generalized coordinates $\boldsymbol{q}_j$ is put in the corresponding row. To obtain the solution for $\partial \boldsymbol{r}_{\mathrm{OM}}/\partial q_i$, the direct differentiation method is applied. The parameters $\partial \boldsymbol{r}_{\mathrm{OT}_G}/\partial q_i$ and $\partial A_{\mathrm{OT}_G}/\partial q_i$ of the root element of the system can be solved using initial conditions. There exists a relationship where the parameter of the point connected to the root element is equal to the

parameter of the root point of the element j to which it is connected. With this information, the quantities $\partial \boldsymbol{r}_{\mathrm{OR}_j} / \partial q_i$ and $\partial A_{\mathrm{OR}_j} / \partial q_i$ of the element j can be determined. The kinematic relations of the element j can be used to express $\partial \boldsymbol{r}_{\mathrm{OT}_j} / \partial q_i$, $\partial A_{\mathrm{OT}_j} / \partial q_i$ for the tip end, which can be written as

$$\boldsymbol{r}_{\mathrm{OT}} = \boldsymbol{r}_{\mathrm{OR}} + \boldsymbol{A}_{\mathrm{OR}} \boldsymbol{l}_{\mathrm{RT}} \tag{47}$$

$$\boldsymbol{A}_{\mathrm{OT}} = \boldsymbol{A}_{\mathrm{OR}} \boldsymbol{A}_{\mathrm{RT}} \tag{48}$$

The process is iterated until the values of $\partial \boldsymbol{r}_{\mathrm{OT}} / \partial q_i$ and $\partial A_{\mathrm{OT}} / \partial q_i$ for the tip end connected element to point M are obtained. The geometry of the transfer is such that the parameter of the M point is equivalent to the parameter of the tip end, leading to the determination of $\partial \boldsymbol{r}_{\mathrm{OM}} / \partial q_i$. This process is then repeated cyclically to obtain $\partial \boldsymbol{r}_{\mathrm{OM}} / \partial q_i \left(i = 1 \cdots a\right)$ for different generalized coordinates. The solution for $\partial \boldsymbol{r}_{\mathrm{OB}} / \partial q_i \left(i = a \cdots n\right)$ is obtained in a similar fashion as that of $\partial \boldsymbol{r}_{\mathrm{OM}} / \partial q_i \left(i = 1 \cdots a\right)$.

By analogy, the partial derivatives are calculated separately for all the generalized coordinates in the system. The system Jacobian matrix is obtained, where the partial derivatives are placed in the appropriate rows. The direct differentiation method for Jacobian matrices uses a for loop to achieve recursion, and the algorithm is shown below.

Algorithm 2. Computation of system Jacobian matrix.

Function *assemblingsystemJacobianmatrix* (*i*:**int**, *ω*:**double**, *s*:**int**)
{Return Jacobian matrix $\boldsymbol{C}_q$ of the system}
{ $\boldsymbol{C}_{L,q}$ is a Jacobian matrix of the secondary joint element.}
{ *SecondaryJointElement* is an indicator of the secondary joint element.}
{ *PrimaryJointElement* is an indicator of the primary joint element.}
{ *FlexibleBodyElement* is an indicator of the flexible body element.}
{ *BodyElement* is an indicator of the body element.}
{ *Index1 (Index2)* is an indicator of the generalized coordinate.}
for *i* **from** *0* **to** the number of generalized coordinates **do**
  Index1 += the number of generalized coordinate of the element[*i*]
  Ground. $\partial r/\partial q_i$ .setzero().
  Ground. $\partial A/\partial q_i$ .setzero().
  for *j* **from** *0* **to** the number of body elements **do**
    Index2 += the number of generalized coordinate of the element[*j*]
    if Index2 = Index1 **then**
      PrimaryJointElement[*j*].*PreprocessConstraintJacobian()*
      FlexibleBodyElement[*j*].*PreprocessConstraintJacobian()*
    end
    PrimaryJointElement[*j*] $\partial r/\partial q_i\left(\partial A/\partial q_i\right)$ ←Ground. $\partial r/\partial q_i\left(\partial A/\partial q_i\right)$.
    BodyElement[*j*] $\partial r/\partial q_i\left(\partial A/\partial q_i\right)$ ←*PrimaryJointElement*[*j*]. $\partial r/\partial q_i\left(\partial A/\partial q_i\right)$.
  end
  for *k*=1 **to** the number of secondary joint elements **do**
    SecondaryJointElement[*k*]. $\boldsymbol{C}_{L,q}$ ←*SecondaryJointElement*[*k*]. $\partial r/\partial q_i\left(\partial A/\partial q_i\right)$
  end
end
for *k*=1 **to** the number of secondary joint elements **do**
  $\boldsymbol{C}_q$ [k]←*SecondaryJointElement*[*k*]. $\boldsymbol{C}_{L,q}$
end
return $\boldsymbol{C}_q$

7. Constraint violation correction approach

As illustrated in Figure 10, the complete correction procedure is described as follows:

(i) Uncorrected generalized coordinates and velocities are obtained during the numerical solution. Then, constraint equations on both position and velocity levels are established in the class of secondary joint elements. The established constraint equations of each secondary joint are combined into the system constraint equations.

(ii) If any values in the system constraint equations are greater than the tolerance error 10^{-10}, the system will be penalized to correct the generalized coordinates and velocities. Algorithm 1 or Algorithm 2 is used to assemble the system Jacobian matrix. Using Eq (4), the corrected generalized coordinates can be solved. The value of the constraint equation is calculated again. If any value of the constraint violation errors is still greater than the tolerance error, continue with the correction. Otherwise, the correction is stopped.

(iii) The system constraint equations on velocity level are obtained by assembling the constraint equations on velocity level of each secondary joint. If the constraint violation errors on velocity level

are greater than the tolerance error, the generalized velocity of the system is corrected using Eq (5). Then, the value of the constraint equation is calculated again. If the constraint violation errors are less than the tolerance error, the corrected generalized coordinates and velocity are obtained.

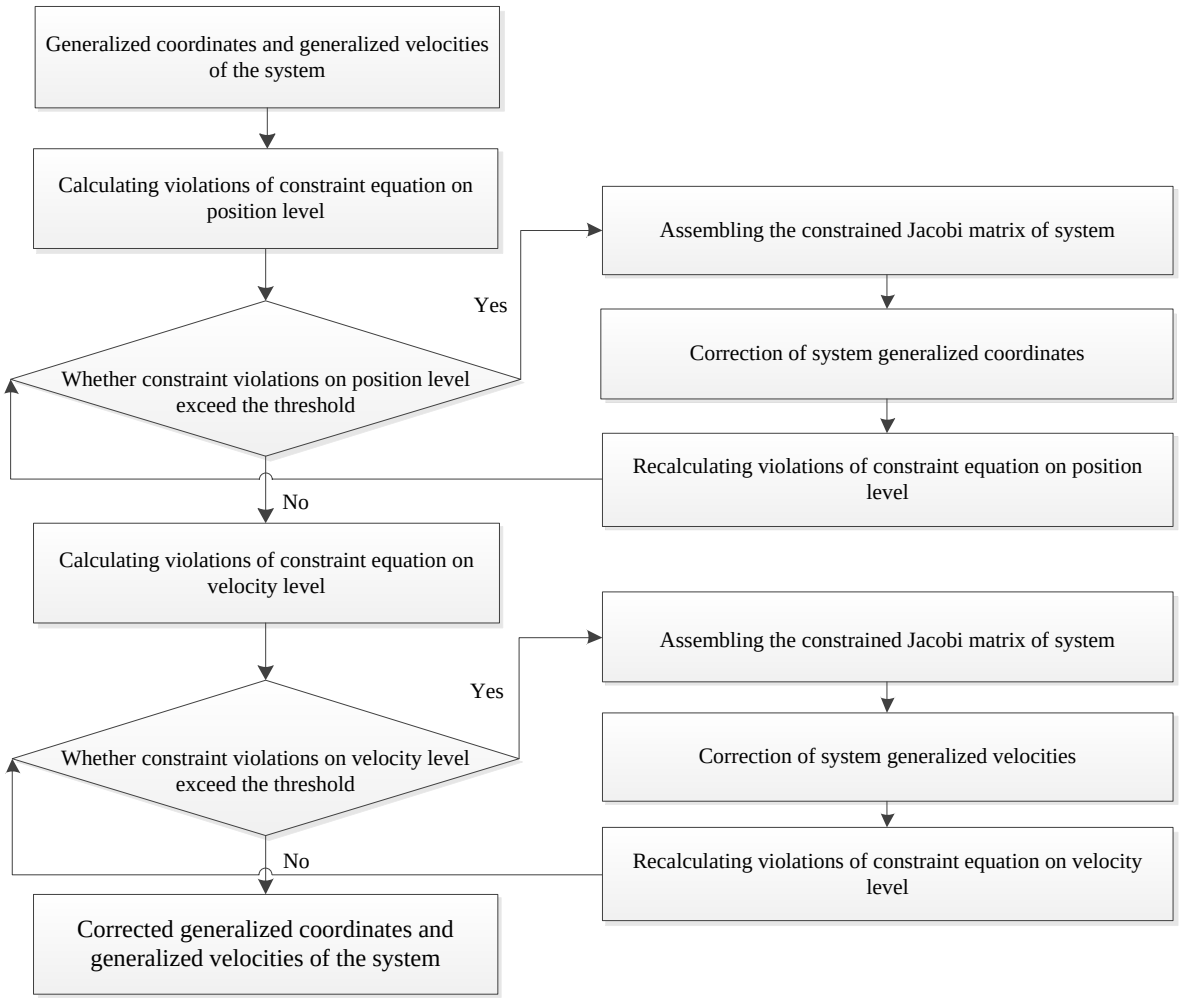

Figure 10. Flow chart of the generalized coordinates and velocity correction.

8. Application example using the approach

The simplicity of engineering models, such as the four-link mechanism, can have a wide range of practical applications, such as frames and rotating dampers [46,47]. Therefore, the ability to simulate these models quickly and accurately is crucial for engineering. A four-link mechanism is used to demonstrate the validity of the method. The system is depicted in Figure 11 and is subject only to gravity. All rigid links are assumed to be identical in length (1 meter) and mass (1 kg), and the secondary joint C is removed to create a spanning system. Element 3 is a flexible beam with two modes. Dynamic calculations were performed using the RMSTMM method, and initial conditions were set at $q_1 = q_3 = \pi/3$, $q_2 = -\pi/3$, $q_4 = q_4 = 0$. A fourth-order Runge-Kutta algorithm with a very small step size (0.01 s) was employed for the numerical solution.

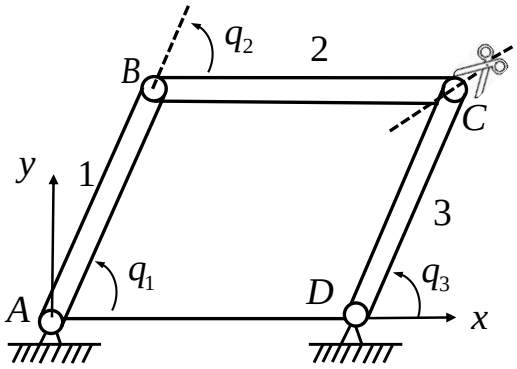

Figure 11. Multibody model of a planar four bar mechanism.

Table 1. Computational time of dynamics without stabilization, proposed method and direct differentiation method for the system.

Simulation time (s)	CPU time (s)		
	Without stabilization	Proposed approach	Direct differentiation
50	19.036	19.766	24.875
100	38.017	38.531	44.547
200	79.593	80.985	87.516

Figure 12 compares the constraint violations on x-direction of secondary joint C with and without the correction. The comparison confirms the correctness of the correction and the derived Jacobian matrix. Figure 13 shows violations of secondary joint C on x-direction by different methods. The results demonstrate that both the direct differential method and the proposed approach are successful in eliminating constraint violations. However, as anticipated, the proposed method is faster in terms of computational speed, as demonstrated by the comparison in Table 1 for a simple four-link mechanism.

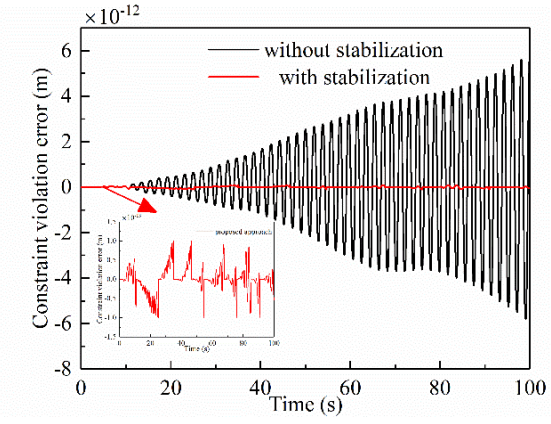

Figure 12. Violation of secondary joint on x-direction.

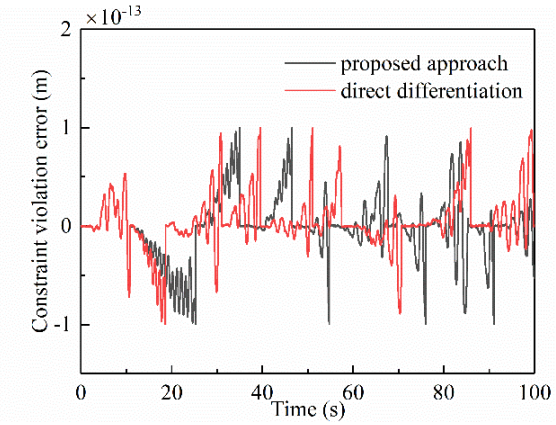

Figure 13. Violation of secondary joint on x-direction by different methods.

In contrast to the previous example which had only a single closed loop, the upcoming example features an increased number of elements, closed loops and flexible body elements as depicted in Figures 14 and 15. The body elements 5, 11 and 21 are flexible and have two order modes, and the other elements are still rigid. The system is moved by gravity only. The dynamics of the system is simulated using the RMSTMM.

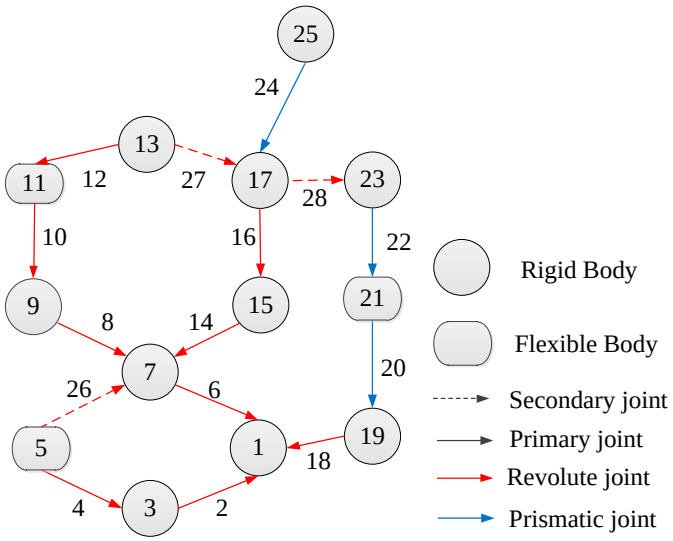

Figure 14. Topology figures of multi-rigid-flexible body systems.

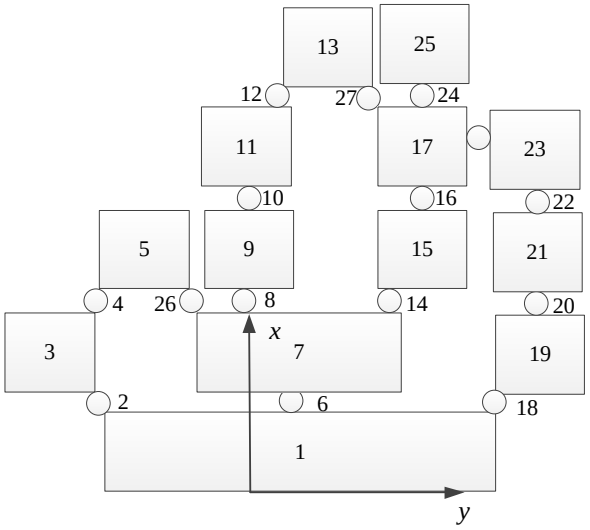

Figure 15. Physical figures of multi-rigid-flexible body systems.

Table 2. Computational time of dynamics without stabilization, proposed method and direct differentiation method for the system.

Simulation time (s)	CPU time (s)		
	Without stabilization	Proposed approach	Direct differentiation
50	73.125	73.469	88.593
100	146.969	148.625	177.89
200	296.516	311.297	371.703

The result shows a divergent trend without constraints, as shown in Figure 16. The violations obtained by the Direct differentiation methods and proposed approach are shown in Figure 17, and both effectively keep the violations within the tolerance error. As shown in table 2, as the system calculation time increases, so does the CPU time used by the system. What is clear is that the longer the system computation time is, the longer the CPU time used by the direct differentiation method compared to the proposed method.

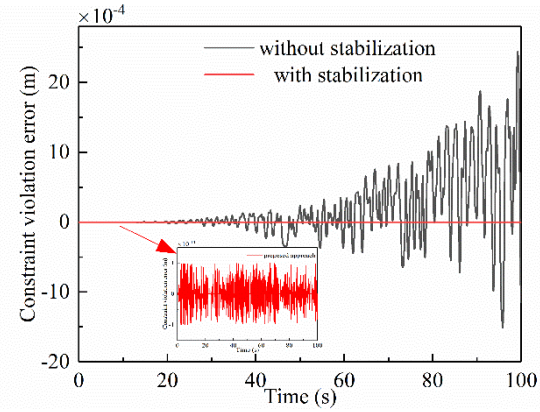

Figure 16. Violation of secondary joint 27 on x-direction.

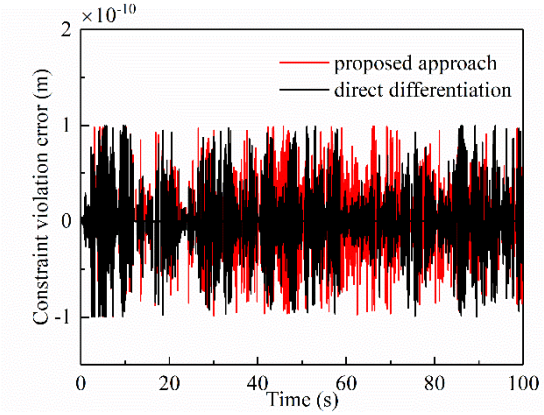

Figure 17. Violation of secondary joint 27 on x-direction by different methods.

The third numerical example is shown in Figures 18 and 19 to verify the effectiveness of the programmatic stabilization solution. The example with rigid and flexible bodies is labeled with thick solid arrows and circles. Joint 19 is a secondary joint affiliated with the corresponding spanning tree. For primary joint, 1, 3, 5, 7, 9 are pin joints, and 11, 13, 15, 17 are prismatic joints. The others are rigid body elements labeled 2, 4, 8, 10, 12, 14, 16 and 18. The quantities of the body elements are the same as in the previous example but in the shape of rods.

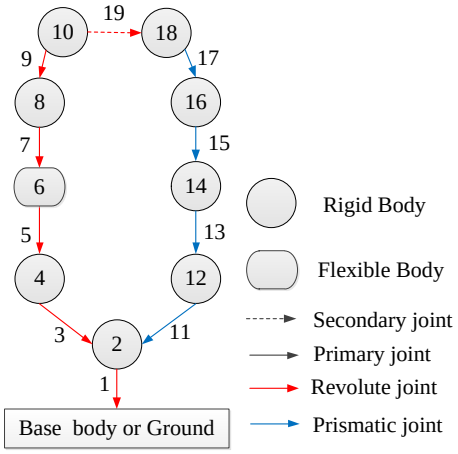

Figure 18. Topology figures of multi-rigid-flexible body systems with single closed-loop.

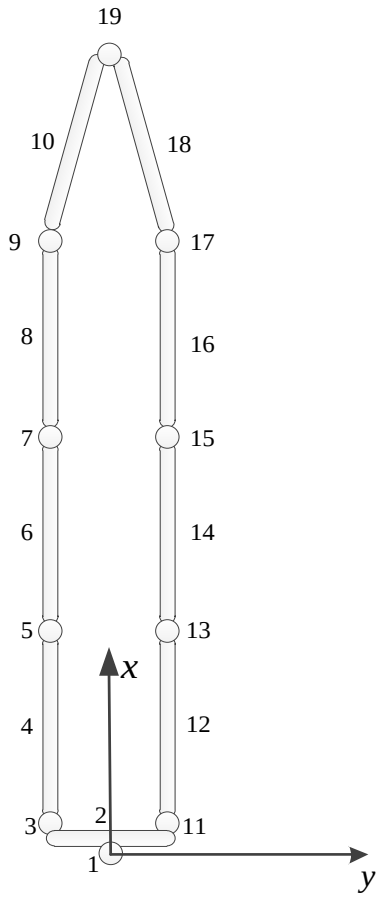

Figure 19. Topology figures of multi-rigid-flexible body systems with single closed-loop.

Table 3. Computational time of dynamics without stabilization, proposed method and direct differentiation method for the system.

Simulation time (s)	CPU time (s)		
	Without stabilization	Proposed approach	Direct differentiation
50	34.922	35.641	42.968
100	75	78.235	96.391
200	144.75	151.297	181.25

The example uses the RMSTMM for dynamics analysis, too. If the correction is not added, the violation error on the x-direction of secondary joint 19 shows a clear divergence trend in Figure 20. The direct correction method is imposed while solving the Jacobian matrix using the direct differentiation method and the proposed method, respectively. The variation of the violation error is shown in Figure 21. Both methods are workable for correcting violation errors. An attempt is made below to compare the computation speed.

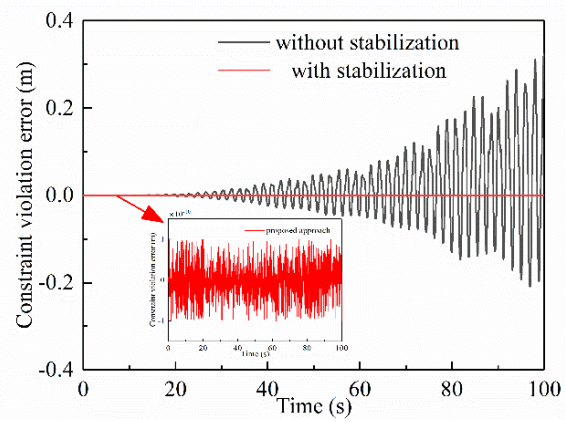

Figure 20. Violation of secondary joint 19 on x-direction.

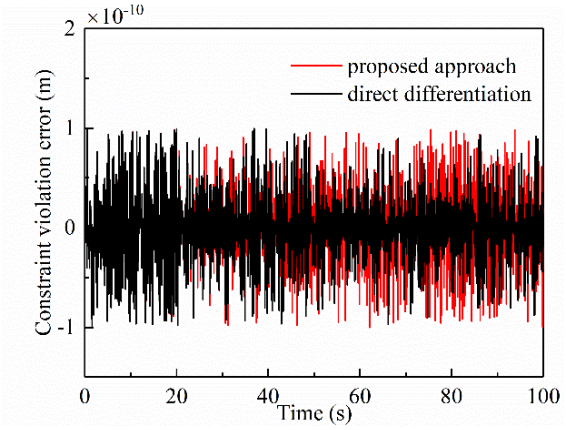

Figure 21. Violation of secondary joint 19 on x-direction by different methods.

It can be found that the proposed approach is faster. The longer the system simulation time is, the longer the CPU time used by the direct differentiation method compared to the proposed method.

For computational speed assessment, the proposed approach and the direct differentiation are compared for a different total number of bodies in Figure 22. The abscissa represents the count of connected bodies within a single closed-loop system. The specific configuration of this system is illustrated in Figure 19. It is worth noting that the number of rods in the system shown in Figure 19 is no longer limited to 7. Instead, it includes various possibilities, such as 100, 200, 300 and 500 rods, respectively.

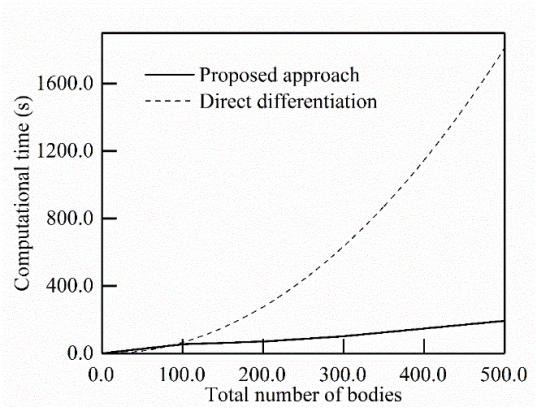

Figure 22. Computational time of proposed approach compared to that of direct differentiation method for loop system.

Upon comparing the two methods in Figure 22, it becomes evident that the proposed strategy outperforms the direct differentiation method in terms of computational time. The results indicate that the proposed strategy is far more efficient than the direct differentiation method.

9. Conclusions

This work proposes a framework for programmatical deriving of a system Jacobin matrix, which has significant advantages for correcting violations in the efficient nonlinear dynamics method focused on acceleration level. Simultaneously, Jacobian matrices in the direct correction method on both position and velocity levels are derived simply semi-analytically. The class for the system Jacobian matrix is designed, leading to a better-structured framework. Intermediate variables in the class of a secondary joint described by variables in the class of primary joints or flexible body elements can be used to assemble the Jacobian matrix. Next, the Jacobian matrix is assembled according to the topological structure of the system. The Newton-Raphson iteration method is used to obtain the corrected generalized coordinates and velocities. The proposed method is sufficiently accurate and faster than the widely used direct differentiation method. Numerical examples are used to verify the proposed approach's effectiveness by comparing the constraint violation errors of multi-rigid-flexible body systems with different methods. Due to their simplicity, the proposed method is well suited for rapid, draft-style modeling of multibody systems.

Use of AI tools declaration

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No: 11902158), the Natural Science Foundation of Jiangsu Province (Grant No: BK20190438) and the National Natural Science Foundation of China (Grant No: 92266201).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. F. M. Amirouche, *Computational methods in multibody dynamics*, Englewood Cliffs, NJ: Prentice-Hall, 1992.
2. J. G. De Jalon, E. Bayo, *Kinematic and dynamic simulation of multibody systems: The real-time challenge*, Springer, 2012. https://doi.org/10.1007/978-1-4612-2600-0
3. W. Jens, *Dynamics of systems of rigid bodies*, Berlin: Springer, 2013. https://doi.org/10.1007/978-3-322-90942-8
4. Y. Liu, Z. Pan, X. Ge, *Dynamics of multibody systems*, Beijing: Higher Education Press, 2014.
5. P. E. Nikravesh, *Computer-aided analysis of mechanical systems*, Upper Saddle River: Prentice-Hall, 1988.
6. R. E. Roberson, R. Schwertassek, *Dynamics of multibody systems*, Berlin: Springer, 2012. https://doi.org/10.1007/978-3-642-86464-3
7. A. A. Shabana, *Dynamics of multibody systems*, New York: Cambridge University Press, 2020. https://doi.org/10.1017/9781108757553
8. S. Werner, *Multibody systems handbook*, Berlin: Springer, 1990. https://doi.org/10.1007/978-3-642-50995-7
9. W. M. Silver, On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators, *Int. J. Rob. Res.*, **1** (1982), 60–70. https://doi.org/10.1177/027836498200100204
10. A. Cammarata, R. Sinatra, P. D. Maddìo, Static condensation method for the reduced dynamic modeling of mechanisms and structures, *Arch. Appl. Mech.*, **89** (2019), 2033–2051. https://doi.org/10.1007/s00419-019-01560-x
11. R. Featherstone, *Rigid body dynamics algorithms*, Springer, 2014. https://doi.org/10.1007/978-1-4899-7560-7
12. F. I. T. Petrescu, Advanced dynamics processes applied to an articulated robot, *Processes*, **10** (2022), 640. https://doi.org/10.3390/pr10040640
13. X. Rui, J. Zhang, X. Wang, B. Rong, B. He, Z. Jin, Multibody system transfer matrix method: The past, the present, and the future, *Int. J. Mech. Syst. Dyn.*, **2** (2022), 3–26. https://doi.org/10.1002/msd2.12037
14. R. Xue, D. Bestle, Reduced multibody system transfer matrix method using decoupled hinge equations, *Int. J. Mech. Syst. Dyn.*, **1** (2021), 12. https://doi.org/10.1002/msd2.12026
15. H. Brandl, R. Johanni, M. Otter, A very efficient algorithm for the ssimulation of robots and similar multibody systems without inversion of the mass matrix, *IFAC Proc.*, **19** (1986), 95–100. https://doi.org/10.1016/S1474-6670(17)59460-4
16. A. Cammarata, R. Sinatra, P. D. Maddio, Interface reduction in flexible multibody systems using the floating frame of reference formulation, *J. Sound Vib.*, **523** (2022). https://doi.org/10.1016/j.jsv.2021.116720
17. K. S. Anderson, *Recursive derivation of explicit equations of motion for efficient dynamic/control simulation of large multibody systems*, Stanford University, 1990.

18. A. Jain, G. Rodriguez, Recursive flexible multibody system dynamics using spatial operators, *J. Guid. Control Dyn.*, **15** (1992), 1453–1466. https://doi.org/10.2514/3.11409
19. H. Brandl, An algorithm for the simulation of multibody systems with kinematic loops, *Proc.7th World Congr. Theory Mach. Mech.*, 1987.
20. F. Marques, A. P. Souto, P. Flores, On the constraints violation in forward dynamics of multibody systems, *Multibody Syst. Dyn.*, **39** (2017), 385–419. https://doi.org/10.1007/s11044-016-9530-y
21. P. E. Nikravesh, *Some methods for dynamic analysis of constrained mechanical systems: a survey*, Berlin: Springer, 1984. https://doi.org/10.1007/978-3-642-52465-3_14
22. J. Baumgarte, Stabilization of constraints and integrals of motion in dynamical systems, *Comput. Methods Appl. Mech. Eng.*, **1** (1972), 1–16. https://doi.org/10.1016/0045-7825(72)90018-7
23. S. T. Lin, J. N. Huang, Stabilization of baumgarte's method using the Runge-Kutta approach, *J. Mech. Design*, **124** (2002). https://doi.org/10.1115/1.1519277
24. P. Zhang, A Stabilization of constraints in the numerical solution of Euler-Lagrange equation, *Chin. J. Eng. Math.*, **20** (2003), 13–18. https://doi.org/10.3969/j.issn.1005-3085.2003.04.003
25. P. Flores, M. Machado, E. Seabra, T. Miguel, A parametric study on the baumgarte stabilization method for forward dynamics of constrained multibody systems, *J. Comput. Nonlinear Dyn.*, **6** (2009), 011019. https://doi.org/10.1115/1.4002338
26. K. T. Wehage, R. A. Wehage, B. Ravani, Generalized coordinate partitioning for complex mechanisms based on kinematic substructuring, *Mech. Mach. Theory*, **92** (2015), 464–483. https://doi.org/10.1016/j.mechmachtheory.2015.06.006
27. P. Fisette, B. Vaneghem, Numerical integration of multibody system dynamic equations using the coordinate partitioning method in an implicit Newmark scheme, *Comput. Methods Appl. Mech. Eng.*, **135** (1996), 85–105. https://doi.org/10.1016/0045-7825(95)00926-4
28. E. J. Haug, J. Yen, Generalized coordinate partitioning methods for numerical integration of differential-algebraic equations of dynamics, In: *Real-time integration methods for mechanical system simulation*, Springer, 1991. https://doi.org/10.1007/978-3-642-76159-1_5
29. R. Singh, P. Likins, Singular value decomposition for constrained dynamical systems, *J. Appl. Mech.*, **52** (1985), 943–948. https://doi.org/10.1115/1.3169173
30. S. Kim, M. Vanderploeg, QR decomposition for state space representation of constrained mechanical dynamic systems, *J. Mech. Design*, **108** (1986), 183–188. https://doi.org/10.1115/1.3260800
31. Q. Yu, J. Hong, A new violation correction method for constraint multibody systems, *Chin. J. Theor. Appl.*, **30** (1998), 300–306. https://doi.org/10.6052/0459-1879-1998-3-1995-130
32. G. Lyu, R. Liu, Errors control of constraint violation in dynamical simulation for constrained mechanical systems, *J. Comput. Nonlinear Dyn.*, **14** (2019). https://doi.org/10.1115/1.4042493
33. X. Xu, J. Luo, Z. Wu, Extending the modified inertia representation to constrained rigid multibody systems, *J. Appl. Mech.*, **87** (2020), 011010. https://doi.org/10.1115/1.4045001
34. J. Zhang, D. Liu, Y. Liu, A constraint violation suppressing formulation for spatial multibody dynamics with singular mass matrix, *Multibody Syst Dyn.*, **36** (2016), 87–110. https://doi.org/10.1007/s11044-015-9458-7
35. L. Zhang, X. Rui, J. Zhang, J. Gu, H. Zheng, T. Li, Study on transfer matrix method for the planar multibody system with closed-loops, *J. Comput. Nonlinear Dyn.*, **16** (2021). https://doi.org/10.1115/1.4052433

36. S. Yoon, R. M. Howe, D. T. Greenwood, Geometric elimination of constraint violations in numerical simulation of lagrangian equations, *J. Mech. Design*, **116** (1994), 1058–1064. https://doi.org/10.1115/1.2919487
37. Y. Q, I. M. Chen, A direct violation correction method in numerical simulation of constrained multibody systems, *Comput. Mech.*, **26** (2000), 52–57. https://doi.org/10.1007/s004660000149
38. J. Hong, *Computational multibody system dynamics*, Beijing: Higher Education Press, 1999.
39. D. Negrut, A. Dyer, *Adams/solver primer*, MSC Software Ann Arbor, 2004.
40. J. Wittenburg, Dynamics of mulitibody systems-a brief review, *Space Humanity*, 1989, 89–92. https://doi.org/10.1016/B978-0-08-037877-0.50015-6
41. R. James, *The unified modeling language reference manual,* Addison-Wesley Professional, 2006.
42. F. Liu, J. Zhang, Q. Hu, A modified constraint force algorithm for flexible multibody dynamics with loop constraints, *Nonlinear Dyn.*, **90** (2017), 1885–1906. https://doi.org/10.1007/s11071-017-3770-0
43. H. Lu, X. Rui, Y. Ding, Y. Chang, Y. Chen, J. Ding, X. Zhang, A hybrid numerical method for vibration analysis of linear multibody systems with flexible components, *Appl. Math. Model.*, **101** (2022), 748–771. https://doi.org/10.1016/j.apm.2021.09.015
44. Y. Lu, Z. Chang, Y. Lu, Y. Wang, Development and kinematics/statics analysis of rigid-flexible-soft hybrid finger mechanism with standard force sensor, *Robot. Comput. Integr. Manuf.*, **67** (2021), 101978. https://doi.org/10.1016/j.rcim.2020.101978
45. J. Zhang, X. Rui, F. Liu, Q. Zhou, L. Gu, Substructuring technique for dynamics analysis of flexible beams with large deformation, *J. Shanghai Jiaotong Univ.*, **22** (2017), 562–569. https://doi.org/10.1007/s12204-017-1875-8
46. A. E. Nabawy, A. A. Abdelrahman, W. S. Abdalla, A. M. Abdelhaleem, S. S. Alieldin, Analysis of the dynamic behavior of the double wishbone suspension system, *Int. J. Appl. Mech.*, **11** (2019), 1950044. https://doi.org/10.1142/S1758825119500443
47. B. Zhang, Z. Li, Mathematical modeling and nonlinear analysis of stiffness of double wishbone independent suspension, *J. Mech. Sci. Technol.*, **35** (2021), 5351–5357. https://doi.org/10.1007/s12206-021-1107-x

AIMS Press