Calculating Jacobian coefficients of primitive constraints with respect to Euler parameters

Yong Liu, Hai-Chuan Song, Jun-Hai Yong

ORIGINAL ARTICLE

Calculating Jacobian coefficients of primitive constraints with respect to Euler parameters

Yong Liu · Hai-Chuan Song · Jun-Hai Yong

Abstract It is a fundamental problem to calculate Jacobian coefficients of constraint equations in assembly constraint solving because most approaches to solving an assembly constraint system will finally resort to a numerical iterative method that requires the first-order derivatives of the constraint equations. The most-used method of deriving the Jacobian coefficients is to use *virtual rotation* which is originally presented to derive the equations of motion of constrained mechanical systems. However, when Euler parameters are adopted as the state variables to represent the transformation matrix, using the virtual rotation will yield erroneous formulae of Jacobian coefficients. The reason is that Euler parameters are incompatible with virtual rotation. In this paper, correct formulae of Jacobian coefficients of geometric constraints with respect to Euler parameters are presented in both Cartesian coordinates and relative generalized coordinates. Experimental results show that our proposed formulae make Newton–Raphson iterative method converge faster and more stable.

Y. Liu (✉) · H.-C. Song · J.-H. Yong
School of Software, Tsinghua University,
Beijing 100084, People's Republic of China
e-mail: tyronelau@gmail.com

Y. Liu · H.-C. Song
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, People's Republic of China

Y. Liu · H.-C. Song · J.-H. Yong
Key Laboratory for Information System Security,
Ministry of Education, Beijing 100084,
People's Republic of China

Y. Liu · H.-C. Song · J.-H. Yong
Tsinghua National Laboratory for Information Science
and Technology, Beijing 100084, People's Republic of China

Keywords Assembly constraint · Primitive constraint · Jacobian coefficient · Virtual rotation · Variational method

1 Introduction

Assembly modeling plays an important role in product design activities because most products are composed of a number of parts. Instead of designating the position and orientation of each part, a designer usually specifies geometric constraints or kinematic constraints between parts, which involves solving those constraints to obtain the postures of parts [1–3]. Numerous approaches to solving 2D or 3D geometric constraint problems have been proposed in recent decades. The main idea of them is to split a large geometric constraint problem into smaller and easier ones using graph-based decomposing algorithm, rule-based strategy, algebraic theory, or numerical methods. Except for a small set of subproblems that can be solved analytically [4, 5], most of them will resort to numerical methods, e.g., a Newton-type iterative method or mathematical optimization method [6–11].

To use a numerical method, the geometric constraints are converted into a set of nonlinear equations first. Those nonlinear equations are called primitive constraints or basic constraints. The selections of state variables can be classified into two categories: Cartesian coordinates and generalized coordinates. The Cartesian coordinate formulation yields a maximal set of highly sparse equations, but the form of the equations is independent on the topological structure of assembly systems. On the other hand, recursive generalized coordinate formulation can dramatically reduce the number of constraint equations and variables, but the form of the equations varies with the topological structure of assembly systems. In both cases, to compute the

Jacobian coefficients of constraint equations, a variational-vector calculus approach is widely adopted in literature, which is originally presented for deriving the equations of motions in dynamics [12–16]. The approach introduces a concept called *virtual rotation*, with which, it is convenient to derive the Jacobian coefficients of primitive constraints with respect to the state variables. However, this concept should not be applied when Euler parameters are taken as the state variables, because Euler parameters break its prerequisites. Otherwise, taking the concept of the virtual rotation will yield wrong formulae of Jacobian coefficients, as presented in literature [6, 9]. In this paper, the prerequisites of virtual rotation are discussed. The correct Jacobian coefficients in Cartesian coordinates are derived using variational-vector calculus method. Moreover, a recursive formulation of Jacobian coefficients of constraint equations in relative generalized coordinates is also derived, which can be applied to any type of relative generalized coordinates, compared with the restricted formulation presented in [6].

The rest of this paper is organized as follows. In Section 2, we first generalize primitive constraints in literature to a formal representation. The reason of the wrong formulae presented in literature is discussed. Then the correct formulae of Jacobian coefficients with respect to Euler parameters in both Cartesian coordinates and recursive generalized coordinates are derived. In Section 3, four examples are provided to illustrate the correctness and efficiency of presented formulae. Finally, conclusions are made in Section 4.

2 Problem statement

In an assembly system, the posture of a rigid body i, could be identified by $\mathbf{r}_i$ and $\mathbf{A}_i$, where $\mathbf{r}_i$ is the origin and $\mathbf{A}_i$ is the transformation matrix from the local reference frame to the global reference frame. The values in the global reference frame, of a vector $\mathbf{v}_i$ and point $\mathbf{P}_i$ fixed on body i can be obtained from the following transformations

$$\begin{aligned}\mathbf{v}_i &= \mathbf{A}_i\mathbf{v}_i'\\ \mathbf{P}_i &= \mathbf{r}_i + \mathbf{v}_i^P = \mathbf{r}_i + \mathbf{A}_i\mathbf{v}_i^{P'},\end{aligned}$$

where the symbols with a superscript $'$ denote the values of the vector or point in the local reference frame. There exists an orthogonal condition

$$\mathbf{A}_i\mathbf{A}_i^T = \mathbf{I} \tag{1}$$

and then the variation $\delta\mathbf{A}_i$ satisfies

$$\delta\mathbf{A}_i\mathbf{A}_i^T = -\mathbf{A}_i\delta\mathbf{A}_i^T \tag{2}$$

A tilde ($\sim$) operator on a vector $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$, forms a skew symmetric matrix $\tilde{\mathbf{v}} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$. From Eq. 2, the *virtual rotation* $\delta\boldsymbol{\pi}_i$ is defined by $\delta\tilde{\boldsymbol{\pi}}_i = \delta\mathbf{A}_i\mathbf{A}_i^T$ and $\delta\boldsymbol{\pi}_i' = \mathbf{A}_i^T\delta\boldsymbol{\pi}_i$.

First, we generalize primitive constraints introduced in [15] to the following formal definition.

Definition 1 A primitive constraint is a scalar function $f = f(\mathbf{r}_{i_1}, \mathbf{r}_{i_2}, \cdots, \mathbf{r}_{i_M}, \mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \cdots, \mathbf{v}_{j_N})$, where $\mathbf{r}_{i_k}$ denotes the origin of body i_k, and $\mathbf{v}_{j_k}$ be any vector fixed on body j_k.

In this paper, we use this formal definition to illustrate the derivations of Jacobian coefficients of primitive constraints, instead of a concrete constraint. The derivations of Jacobian coefficients with respect to both Cartesian coordinates and relative generalized coordinates are presented.

2.1 Cartesian coordinates and Euler parameters

A spatial mechanism, or assembly, constructed using Cartesian coordinates and Euler parameters, is defined in terms of the state variables $\mathbf{q} = \begin{bmatrix}\mathbf{r}_1 & \mathbf{p}_1 & \cdots & \mathbf{r}_n & \mathbf{p}_n\end{bmatrix}^T$, where $\mathbf{p}_i = \begin{bmatrix}e_{0i} & e_{1i} & e_{2i} & e_{3i}\end{bmatrix}^T$ denotes Euler parameters of body i and satisfies the normalization condition $\mathbf{p}_i^T\mathbf{p}_i = 1$. Let $\mathbf{e}_i = \begin{bmatrix}e_{1i} & e_{2i} & e_{3i}\end{bmatrix}^T$, $\mathbf{E}_i = \begin{bmatrix}-\mathbf{e}_i, \tilde{\mathbf{e}}_i + e_{0i}\mathbf{I}\end{bmatrix}$, and $\mathbf{G}_i = \begin{bmatrix}-\mathbf{e}_i, -\tilde{\mathbf{e}}_i + e_{0i}\mathbf{I}\end{bmatrix}$. $\mathbf{A}_i$ can be expressed as $\mathbf{A}_i = \mathbf{E}_i\mathbf{G}_i^T$. In [15], $\delta\boldsymbol{\pi}_i'$ is derived as

$$\delta\boldsymbol{\pi}_i' = 2\mathbf{G}_i\delta\mathbf{p}_i \tag{3}$$

Using the virtual rotation, Peng et al. [9] derived the variation of a vector $\mathbf{v}_i$ as

$$\delta\mathbf{v}_i = \delta\mathbf{A}_i\mathbf{v}_i' = \mathbf{A}_i\delta\tilde{\boldsymbol{\pi}}_i'\mathbf{v}_i' = -\mathbf{A}_i\tilde{\mathbf{v}}_i'\delta\boldsymbol{\pi}_i' = -2\mathbf{A}_i\tilde{\mathbf{v}}_i'\mathbf{G}_i\delta\mathbf{p}_i. \tag{4}$$

Then the variation of the function f is derived as

$$\begin{aligned}\delta f &= \sum_k \frac{\partial f}{\partial \mathbf{r}_{i_k}}\delta\mathbf{r}_{i_k} + \sum_k \frac{\partial f}{\partial \mathbf{v}_{j_k}}\delta\mathbf{v}_{j_k}\\ &= \sum_k \frac{\partial f}{\partial \mathbf{r}_{i_k}}\delta\mathbf{r}_{i_k} - 2\sum_k \frac{\partial f}{\partial \mathbf{v}_{j_k}}\mathbf{A}_{j_k}\tilde{\mathbf{v}}_{j_k}'\mathbf{G}_{j_k}\delta\mathbf{p}_{j_k}\end{aligned} \tag{5}$$

from which the Jacobian coefficients of f are obtained as

$$f_{\mathbf{r}_{i_k}} = \frac{\partial f}{\partial \mathbf{r}_{i_k}}, \quad k = 1, \cdots, M \tag{6a}$$

$$f_{\mathbf{p}_{j_k}} = -2\frac{\partial f}{\partial \mathbf{v}_{j_k}}\mathbf{A}_{j_k}\tilde{\mathbf{v}}_{j_k}'\mathbf{G}_{j_k}, \quad k = 1, \cdots, N, \tag{6b}$$

where the partial derivatives $\frac{\partial f}{\partial \mathbf{r}_{i_k}}$ and $\frac{\partial f}{\partial \mathbf{v}_{j_k}}$ are only determined by the form of f itself and easy to compute.

As is pointed out by Yen et al. [17], however, Eq. 6b is not the true derivative of f with respect to Euler parameters $\mathbf{p}_{jk}$, because the definition of the *virtual rotation* implies the establishment of Eqs. 1 and 2, which requires $\mathbf{p}_i^T \mathbf{p}_i = 1$ and $\mathbf{p}_i^T \delta\mathbf{p}_i = 0$. It results in a contradiction. Yen et al. presented a formula of the variation of a vector $\mathbf{v}_i$ to compute the correct Jacobian coefficients of primitive constraints, but his formula is complicated. In this paper, using the variational-vector calculus method, we derive this variation in a more concise form as follows.

$$
\begin{aligned}
\delta\mathbf{v}_i &= \delta\left(\mathbf{A}_i\mathbf{v}_i'\right) = \delta\left(\mathbf{E}_i\mathbf{G}_i^T\mathbf{v}_i'\right) \\
&= \delta\left(\left(\mathbf{e}_i\mathbf{e}_i^T + \tilde{\mathbf{e}}_i\tilde{\mathbf{e}}_i + 2\mathbf{e}_{0i}\tilde{\mathbf{e}}_i + \mathbf{e}_{0i}^2\mathbf{I}\right)\mathbf{v}_i'\right) \\
&= \delta\left(\mathbf{e}_i\mathbf{e}_i^T\mathbf{v}_i'\right) + \delta\left(\tilde{\mathbf{e}}_i\tilde{\mathbf{e}}_i\mathbf{v}_i'\right) + \delta\left(2e_{0i}\tilde{\mathbf{e}}_i\mathbf{v}_i'\right) + \delta\left(e_{0i}^2\mathbf{v}_i'\right) \\
&= \delta\mathbf{e}_i\mathbf{e}_i^T\mathbf{v}_i' + \mathbf{e}_i\delta\mathbf{e}_i^T\mathbf{v}_i' + \delta\tilde{\mathbf{e}}_i\tilde{\mathbf{e}}_i\mathbf{v}_i' + \tilde{\mathbf{e}}_i\delta\tilde{\mathbf{e}}_i\mathbf{v}_i' + 2\delta e_{0i}\tilde{\mathbf{e}}_i\mathbf{v}_i' \\
&\quad + 2e_{0i}\delta\tilde{\mathbf{e}}_i\mathbf{v}_i' + 2e_{0i}\mathbf{v}_i'\delta e_{0i} \\
&= \mathbf{e}_i^T\mathbf{v}_i'\delta\mathbf{e}_i + \mathbf{e}_i\mathbf{v}_i'^T\delta\mathbf{e}_i - \widetilde{\tilde{\mathbf{e}}_i\mathbf{v}_i'}\delta\mathbf{e}_i - \tilde{\mathbf{e}}_i\tilde{\mathbf{v}}_i'\delta\mathbf{e}_i + 2\tilde{\mathbf{e}}_i\mathbf{v}_i'\delta e_{0i} \\
&\quad - 2e_{0i}\tilde{\mathbf{v}}_i'\delta\mathbf{e}_i + 2e_{0i}\mathbf{v}_i'\delta e_{0i} \\
&= 2\left(\tilde{\mathbf{e}}_i + e_{0i}\mathbf{I}\right)\mathbf{v}_i'\delta e_{0i} \\
&\quad + \left(\mathbf{e}_i^T\mathbf{v}_i'\mathbf{I} + \mathbf{e}_i\mathbf{v}_i'^T - \widetilde{\tilde{\mathbf{e}}_i\mathbf{v}_i'} - \tilde{\mathbf{e}}_i\tilde{\mathbf{v}}_i' - 2e_{0i}\tilde{\mathbf{v}}_i'\right)\delta\mathbf{e}_i \\
&= \mathbf{K}\left(\mathbf{v}_i', \mathbf{p}_i\right)\delta\mathbf{p}_i, \qquad (7)
\end{aligned}
$$

where

$$
\mathbf{K}(\mathbf{u}, \mathbf{p}) = \left[2(\tilde{\mathbf{e}} + e_0\mathbf{I})\mathbf{u} \quad \mathbf{e}^T\mathbf{u}\mathbf{I} + \mathbf{e}\mathbf{u}^T - \tilde{\mathbf{e}}\tilde{\mathbf{u}} - \widetilde{\tilde{\mathbf{e}}\mathbf{u}} - 2e_0\tilde{\mathbf{u}}\right].
$$

Substituting Eq. 7 into Eq. 5 yields

$$
f_{\mathbf{p}_{jk}} = \frac{\partial f}{\partial \mathbf{v}_{jk}}\mathbf{K}\left(\mathbf{v}_{jk}', \mathbf{p}_{jk}\right), \quad k = 1, \cdots, N. \qquad (8)
$$

2.2 Relative generalized coordinates

Using relative generalized coordinates, $\mathbf{r}_i$ and $\mathbf{A}_i$ can be represented recursively instead. The relation of a pair of coupled bodies, bodies i and j, is depicted in Fig. 1. Vectors that locate joint attachment points in bodies i and j are denoted by $\mathbf{s}_{ij}$ and $\mathbf{s}_{ji}$, respectively. Orthogonal matrices $\mathbf{C}_{ij}$, $\mathbf{C}_{ji}$, and $\mathbf{A}_{ij}''$ are transformations from the joint definition frames ($''$) to the body frames($'$) on bodies i and j and from the joint definition frame on body j to the joint definition frame on body i, respectively. From Fig. 1, we have the following relations:

$$
\mathbf{r}_j = \mathbf{r}_i + \mathbf{s}_{ij} + \mathbf{d}_{ij} - \mathbf{s}_{ji} \qquad (9a)
$$

$$
\mathbf{A}_j = \mathbf{A}_i\mathbf{C}_{ij}\mathbf{A}_{ij}''\mathbf{C}_{ji}^T \qquad (9b)
$$

and

$$
\mathbf{d}_{ij} = \mathbf{A}_i\mathbf{C}_{ij}\mathbf{d}_{ij}'' \qquad (10a)
$$

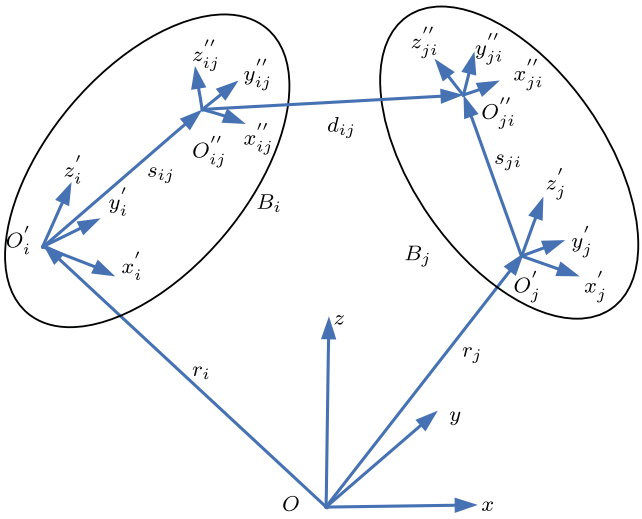

Fig. 1 The representation of relative motions of two bodies

$$
\mathbf{s}_{ij} = \mathbf{A}_i\mathbf{s}_{ij}' \qquad (10b)
$$

$$
\mathbf{s}_{ji} = \mathbf{A}_j\mathbf{s}_{ji}' \qquad (10c)
$$

where $\mathbf{s}_{ij}'$ and $\mathbf{s}_{ji}'$ are fixed vectors on each body frame separately. In the equations above, $\mathbf{A}_{ij}''$ and $\mathbf{d}_{ij}''$ are only the functions of relative generalized coordinates $\mathbf{q}_{ij}$ between bodies i and j.

A mechanical system can be represented by a graph $G = G(V, E)$, in which each node of V represents a rigid body, and each edge $e \in E$ represents the joint between two bodies. Both an open-loop and closed-loop mechanical systems can be abstracted as a topological tree structure with additional geometric constraints. Consider any body i. There exists a unique path $0 \to \cdots \to i-1 \to i$ from base body 0 to body i. Using Eqs. 9 and 10, $\mathbf{r}_i$ and $\mathbf{A}_i$ can be represented recursively by the relative generalized coordinates $\mathbf{q}_{0,1}, \mathbf{q}_{1,2}, \cdots, \mathbf{q}_{i-1,i}$ [13].

Let $\mathbf{q}$ denote the collection of all relative generalized coordinates. To obtain the Jacobian coefficients of f, the variations $\delta\mathbf{r}_i$ and $\delta\mathbf{v}_i$ should be derived first. Let $\delta\mathbf{r}_i$ and $\delta\mathbf{v}_i$ have the following forms

$$
\delta\mathbf{r}_i = \mathbf{J}_i^r\delta\mathbf{q} \qquad (11a)
$$

$$
\delta\mathbf{v}_i = \mathbf{J}_i^v\delta\mathbf{q} \qquad (11b)
$$

The recursive formulae of $\mathbf{J}_i^r$ and $\mathbf{J}_i^v$ were derived using the virtual rotation in [6, 11]. However, similarly, when a free rotational joint is involved and Euler parameters are chosen as the relative generalized coordinates, the virtual rotation should not be applied. Otherwise, using the virtual rotation will yield erroneous formulae. In this paper, we derive the recursive formulae directly instead. First, two auxiliary coefficient matrices are introduced.

Definition 2 Let $\mathbf{u}$ be any 3D vector. The auxiliary coefficient matrices $\mathbf{J}_{i-1,i}(\mathbf{u})$ and $\mathbf{J}_i(\mathbf{u})$ satisfy $\mathbf{J}_{i-1,i}(\mathbf{u})\delta\mathbf{q}_{i-1,i} = \delta\mathbf{A}''_{i-1,i}\mathbf{u}$ and $\mathbf{J}_i(\mathbf{u})\delta\mathbf{q} = \delta\mathbf{A}_i\mathbf{u}$ separately.

From the definition, $\mathbf{J}_{i-1,i}$ is only determined by relative generalized coordinates $\mathbf{q}_{i-1,i}$ and can be deduced from the joint definition. For example, if $\mathbf{q}_{i-1,i}$ is the Euler parameters, we can obtain its formula from Eq. 7.

$$\mathbf{J}_{i-1,i}(\mathbf{u})\delta\mathbf{q}_{i-1,i} = \delta\mathbf{A}''_{i-1,i}\mathbf{u} = \mathbf{K}(\mathbf{u}, \mathbf{q}_{i-1,i})\delta\mathbf{q}_{i-1,i} \tag{12}$$

The recursive formula of $\mathbf{J}_i(\mathbf{u})$ is derived as follows

$$\begin{aligned}
\mathbf{J}_i(\mathbf{u})\delta\mathbf{q} &= \delta\mathbf{A}_i\mathbf{u} \\
&= \delta\left(\mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\right)\mathbf{u} \\
&= \delta\mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\mathbf{u} \\
&\quad + \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\delta\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\mathbf{u} \\
&= \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\mathbf{u}\right)\delta\mathbf{q} \\
&\quad + \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}J_{i-1,i}\left(\mathbf{C}^T_{i,i-1}\mathbf{u}\right)\delta\mathbf{q}_{i-1,i} \\
&= \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\mathbf{u}\right)\delta\mathbf{q} \\
&\quad + \left[\mathbf{0}\ \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}J_{i-1,i}\left(\mathbf{C}^T_{i,i-1}\mathbf{u}\right)\ \mathbf{0}\right]\delta\mathbf{q}
\end{aligned}$$

from which we obtain

$$\begin{aligned}
\mathbf{J}_i(\mathbf{u}) &= \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{A}''_{i-1,i}\mathbf{C}^T_{i,i-1}\mathbf{u}\right) \\
&\quad + \left[\mathbf{0}\ \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}J_{i-1,i}\left(\mathbf{C}^T_{i,i-1}\mathbf{u}\right)\ \mathbf{0}\right].
\end{aligned} \tag{13}$$

The variation $\delta\mathbf{v}_i$ can be obtained as

$$\delta\mathbf{v}_i = \delta\mathbf{A}_i\mathbf{v}'_i = \mathbf{J}_i\left(\mathbf{v}'_i\right)\delta\mathbf{q} \tag{14}$$

from which we obtain

$$\mathbf{J}^v_i = \mathbf{J}_i\left(\mathbf{v}'_i\right). \tag{15}$$

The variation $\mathbf{r}_i$ is derived as follows

$$\begin{aligned}
\delta\mathbf{r}_i &= \delta\left(\mathbf{r}_{i-1} + \mathbf{s}_{i-1,i} + \mathbf{d}_{i-1,i} - \mathbf{s}_{i,i-1}\right) \\
&= \delta\mathbf{r}_{i-1} + \delta\mathbf{A}_{i-1}\mathbf{s}'_{i-1,i} + \delta\left(\mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\mathbf{d}''_{i-1,i}\right) \\
&\quad - \delta\mathbf{A}_i\mathbf{s}'_{i,i-1} \\
&= \delta\mathbf{r}_{i-1} + \mathbf{J}_i\left(\mathbf{s}'_{i-1,i}\right)\delta\mathbf{q} + \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{d}''_{i-1,i}\right)\delta\mathbf{q} \\
&\quad + \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\delta\mathbf{d}''_{i-1,i} - \mathbf{J}_i\left(\mathbf{s}'_{i,i-1}\right)\delta\mathbf{q} \\
&= \delta\mathbf{r}_{i-1} + \mathbf{J}_i\left(\mathbf{s}'_{i-1,i}\right)\delta\mathbf{q} + \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{d}''_{i-1,i}\right)\delta\mathbf{q} \\
&\quad + \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\mathbf{D}_{i-1,i}\delta\mathbf{q}_{i-1,i} - \mathbf{J}_i\left(\mathbf{s}'_{i,i-1}\right)\delta\mathbf{q}
\end{aligned} \tag{16}$$

from which we obtain

$$\begin{aligned}
\mathbf{J}^r_i &= \mathbf{J}^r_{i-1} + \mathbf{J}_i\left(\mathbf{s}'_{i-1,i}\right) + \mathbf{J}_{i-1}\left(\mathbf{C}_{i-1,i}\mathbf{d}''_{i-1,i}\right) \\
&\quad + \left[\mathbf{0}\ \mathbf{A}_{i-1}\mathbf{C}_{i-1,i}\mathbf{D}_{i-1,i}\ \mathbf{0}\right] - \mathbf{J}_i\left(\mathbf{s}'_{i,i-1}\right),
\end{aligned}$$

where $\mathbf{D}_{i-1,i}$ is the Jacobian coefficient of $\mathbf{d}''_{i-1,i}$ with respect to local relative generalized coordinates $\mathbf{q}_{i-1,i}$. Substituting Eqs. 14 and 16 into Eq. 5 yields

$$\begin{aligned}
\delta f &= \sum_k \frac{\partial f}{\partial \mathbf{r}_{i_k}}\delta\mathbf{r}_{i_k} + \sum_k \frac{\partial f}{\partial \mathbf{v}_{j_k}}\delta\mathbf{v}_{j_k} \\
&= \sum_k \frac{\partial f}{\partial \mathbf{r}_{i_k}}\mathbf{J}^r_{i_k}\delta\mathbf{q} + \sum_k \frac{\partial f}{\partial \mathbf{v}_{j_k}}\mathbf{J}_i\left(\mathbf{v}'_{j_k}\right)\delta\mathbf{q}
\end{aligned}$$

from which, the Jacobian matrix of f is obtained as

$$f_{\mathbf{q}} = \sum_k \frac{\partial f}{\partial \mathbf{r}_{i_k}}\mathbf{J}^r_{i_k} + \sum_k \frac{\partial f}{\partial \mathbf{v}_{j_k}}\mathbf{J}_i\left(\mathbf{v}'_{j_k}\right). \tag{17}$$

2.3 Jacobian coefficients of primitive constraints

Based on Haug's work [15], some primitive geometric constraints were presented to build the geometric constraint library [11]. Most important primitive constraints are *dot-1* constraint, *dot-2* constraint, *angle* constraint, and *distant* constraint.

$$\Phi^{d1}(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i^T\mathbf{v}_j = 0 \tag{18a}$$

$$\Phi^{d2}(\mathbf{v}_i, \mathbf{d}_{ij}) = \mathbf{v}_i^T\mathbf{d}_{ij} = \mathbf{v}_i^T\mathbf{d}_{ij} = 0 \tag{18b}$$

$$\Phi^{ang}(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i^T\mathbf{v}_j - \cos\alpha_0 = 0 \tag{18c}$$

$$\Phi^{dist}(\mathbf{d}_{ij}) = \mathbf{d}_{ij}^T\mathbf{d}_{ij} - d_0^2 = 0, \tag{18d}$$

where

$$\mathbf{d}_{ij} = \mathbf{Q}_j - \mathbf{P}_i = \mathbf{r}_j + \mathbf{A}_j\mathbf{s}_j^{Q'} - \mathbf{r}_i - \mathbf{A}_i\mathbf{s}_i^{P'}.$$

The Jacobian coefficients of these constraints are derived as follows. For Cartesian coordinates, those are

$$\Phi^{d1}_{\mathbf{r}_i} = \Phi^{d1}_{\mathbf{r}_j} = 0 \tag{19a}$$

$$\Phi^{d1}_{\mathbf{p}_i} = \mathbf{v}_j^T\mathbf{K}\left(\mathbf{v}'_i, \mathbf{p}_i\right) \tag{19b}$$

$$\Phi^{d1}_{\mathbf{p}_j} = \mathbf{v}_i^T\mathbf{K}\left(\mathbf{v}'_j, \mathbf{p}_j\right) \tag{19c}$$

$$\Phi^{d2}_{\mathbf{r}_i} = -\mathbf{v}_i^T \tag{19d}$$

$$\Phi^{d2}_{\mathbf{r}_j} = \mathbf{v}_i^T \tag{19e}$$

$$\Phi^{d2}_{\mathbf{p}_i} = \mathbf{d}_{ij}^T\mathbf{K}\left(\mathbf{v}'_i, \mathbf{p}_i\right) - \mathbf{v}_i^T\mathbf{K}\left(\mathbf{s}_i^{p'}, \mathbf{p}_i\right) \tag{19f}$$

$$\Phi^{d2}_{\mathbf{p}_j} = \mathbf{v}_i^T\mathbf{K}\left(\mathbf{s}_j^{p'}, \mathbf{p}_j\right) \tag{19g}$$

$$\Phi^{dist}_{\mathbf{r}_i} = -2\mathbf{d}_{ij}^T \tag{19h}$$

$$\Phi^{dist}_{\mathbf{r}_j} = 2\mathbf{d}^T_{ij} \tag{19i}$$

$$\Phi^{dist}_{\mathbf{p}_i} = -2\mathbf{d}^T_{ij}\mathbf{K}\left(\mathbf{s}^{p'}_i, \mathbf{p}_i\right) \tag{19j}$$

$$\Phi^{dist}_{\mathbf{p}_j} = 2\mathbf{d}^T_{ij}\mathbf{K}\left(\mathbf{s}^{p'}_j, \mathbf{p}_j\right), \tag{19k}$$

and for relative generalized coordinates,

$$\Phi^{d1}_{\mathbf{q}} = \mathbf{v}^T_j \mathbf{J}_i\left(\mathbf{v}'_i\right) + \mathbf{v}^T_i \mathbf{J}_j\left(\mathbf{v}'_j\right) \tag{20a}$$

$$\Phi^{d2}_{\mathbf{q}} = \mathbf{d}^T_{ij}\mathbf{J}_i\left(\mathbf{v}'_i\right) + \mathbf{v}^T_i\left(\mathbf{J}^r_j + \mathbf{J}_j\left(\mathbf{s}^{p'}_j\right) - \mathbf{J}^r_i - \mathbf{J}_i\left(\mathbf{s}^{p'}_i\right)\right) \tag{20b}$$

$$\Phi^{dist}_{\mathbf{q}} = 2\mathbf{d}^T_{ij}\left(\mathbf{J}^r_j + \mathbf{J}_j\left(\mathbf{s}^{p'}_j\right) - \mathbf{J}^r_i - \mathbf{J}_i\left(\mathbf{s}^{p'}_i\right)\right) \tag{20c}$$

In the next section, experimental results will be presented to show that the formulae presented in this paper are more feasible than those proposed in [9].

3 Experimental results

In this section, we use the Newton–Raphson iteration method listed in Algorithm 1 to evaluate the effectiveness of the presented formulae. Four examples are used to illustrate the advantages of our proposed formulae over those presented by Peng et al. [9]. Without loss of generality, all the examples are two-body constraint systems with fixed origins so that each of them only has three rotational degrees of freedom left. First, we use Example 1 to demonstrate that the iteration using Eq. 6b breaks the assumption of orthogonality of the transformation matrix, and then use these four examples to illustrate that our formulae are more applicable.

Example 1 Let origins of bodies 1 and 2 be coincident. There are three dot-2 constraints ($\Phi^{d2}(\mathbf{v}_i, \mathbf{P}_i, \mathbf{Q}_i)$, $i = 1 \cdots 3$) between the bodies. This constraint system is well constrained.

With the parameters given in Table 1, this constraint system has real solutions. The unknowns of the constraint system is the $\mathbf{q} = \mathbf{p}_2 = \begin{bmatrix} e_0 & e_1 & e_2 & e_3 \end{bmatrix}^T$. The equation set of the constraint system is represented by

$$\Phi(\mathbf{q}) = \begin{cases} \mathbf{q}^T\mathbf{q} - 1 \\ \mathbf{v}^T_i\left(\mathbf{Q}_i - \mathbf{P}_i\right), i = 1 \cdots 3 \end{cases}$$

Algorithm 1: Newton iteration procedure

input : The initial estimate $\mathbf{q}_0$
the set of equations Φ
output: The solved status

```
q ← q0
n ← 0
while n < N and ‖Φ(q)‖ < ε do
    J ← Φq(q)
    if J is invertible then
        q ← q − J^{-1}Φ(q)
    end
    else
        /* Use Penrose-Moore inverse
        instead */
        q ← q − J^+Φ(q)
    end
    n ← n + 1
end
if n == N then
    return Failed
end
return Success
```

of which the Jacobian matrix $\Phi_{\mathbf{q}}$ is derived from Eqs. 19f and 19g as

$$\Phi_{\mathbf{q}}(\mathbf{q}) = \begin{bmatrix} 2\mathbf{q}^T \\ \left(\mathbf{Q}_1 - \mathbf{P}_1\right)^T K\left(\mathbf{v}'_1, \mathbf{q}\right) - \mathbf{v}^T_1 K\left(\mathbf{P}'_1, \mathbf{q}\right) \\ \left(\mathbf{Q}_2 - \mathbf{P}_2\right)^T K\left(\mathbf{v}'_2, \mathbf{q}\right) - \mathbf{v}^T_2 K\left(\mathbf{P}'_2, \mathbf{q}\right) \\ \left(\mathbf{Q}_3 - \mathbf{P}_3\right)^T K\left(\mathbf{v}'_3, \mathbf{q}\right) - \mathbf{v}^T_3 K\left(\mathbf{P}'_3, \mathbf{q}\right) \end{bmatrix} \tag{21}$$

while Eq. 6b gives

$$\hat{\Phi}_{\mathbf{q}}(\mathbf{q}) = \begin{bmatrix} 2\mathbf{q}^T \\ 2\left(\mathbf{v}'^T_1\tilde{\mathbf{P}}'_1 - \left(\mathbf{Q}_1 - \mathbf{P}_1\right)^T A\tilde{\mathbf{v}}'_1\right)G \\ 2\left(\mathbf{v}'^T_2\tilde{\mathbf{P}}'_2 - \left(\mathbf{Q}_2 - \mathbf{P}_2\right)^T A\tilde{\mathbf{v}}'_2\right)G \\ 2\left(\mathbf{v}'^T_3\tilde{\mathbf{P}}'_3 - \left(\mathbf{Q}_3 - \mathbf{P}_3\right)^T A\tilde{\mathbf{v}}'_3\right)G \end{bmatrix}. \tag{22}$$

Given an initial estimate $\mathbf{q}_0 = \begin{bmatrix} 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}^T$, which satisfies the normalization constraint, we can get

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{A}\mathbf{A}^T = \mathbf{I},$$

Substituting $\mathbf{A}$ into Eq. 22 yields

$$\hat{\Phi}_{\mathbf{q}_0} = \begin{bmatrix} 1.0000 & -1.0000 & -1.0000 & 1.0000 \\ 6.9282 & 1.1547 & 0 & -5.7735 \\ -1.7889 & -2.6833 & -3.5777 & -4.4721 \\ 2.8284 & 1.4142 & -2.8284 & -4.2426 \end{bmatrix}.$$

Table 1 The parameters used in examples

Example	Parameters
1	$\mathbf{v}'_1 = \begin{bmatrix} 1 \\ \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}, \mathbf{v}'_2 = \begin{bmatrix} 0 \\ 2 \\ \frac{\sqrt{5}}{1} \\ \frac{1}{\sqrt{5}} \end{bmatrix}, \mathbf{v}'_3 = \begin{bmatrix} 1 \\ \frac{\sqrt{2}}{1} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}, \mathbf{P}'_1 = \begin{bmatrix} 4 \\ 4 \\ -2 \end{bmatrix}, \mathbf{P}'_2 = \begin{bmatrix} 10 \\ 0 \\ 7 \end{bmatrix}, \mathbf{P}'_3 = \begin{bmatrix} 0 \\ 3 \\ 10 \end{bmatrix}, \mathbf{Q}'_1 = \begin{bmatrix} 0 \\ 5 \\ 1 \end{bmatrix}, \mathbf{Q}'_2 = \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix}, \mathbf{Q}'_3 = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$
2	$\mathbf{P}'_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{P}'_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{P}'_3 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{Q}'_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{Q}'_2 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \mathbf{Q}'_3 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, d_1 = 1, d_2 = \sqrt{6}, d_3 = \sqrt{5}$
3	$\mathbf{u}'_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{u}'_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{v}'_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}'_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{u}'_3 = \begin{bmatrix} 1 \\ \frac{\sqrt{3}}{1} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix}, \mathbf{P}'_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \mathbf{Q}'_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$
4	$\mathbf{u}'_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{u}'_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{v}'_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}'_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{P}'_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \mathbf{Q}'_1 = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \alpha_1 = \frac{\pi}{6}, \alpha_2 = \frac{\pi}{3}, d_1 = 2$

From Algorithm 1, we can get the next iteration point $\mathbf{q}_1 = \begin{bmatrix} 0.3750 & -0.6250 & -0.8750 & 0.1250 \end{bmatrix}^T$, and the transformation matrix $\mathbf{A} = \begin{bmatrix} -0.2500 & 1.0000 & -0.8125 \\ 1.1875 & 0.5000 & 0.2500 \\ 0.5000 & -0.6875 & -1.0000 \end{bmatrix}$ which breaks Eq. 1. Continuing this procedure, we will find that the iteration fails to converge. On the other hand, if we adopt Eq. 21, the iteration converges in about 10 times given a tolerance 10^{-14}.

To study the convergency generally, many initial estimates are generated to test whether Algorithm 1 converges, and if it does, in how many iterations. Our tests are

Table 2 The simulation results

Experiment	Estimate region	Jacobian formulae	Convergent cases	Average iterations
Exam. 1, body 2 movable	*hyper box*	Proposed	10,000	9.43
		Peng's	9,919	23.38
	hyper sphere	Proposed	10,000	10.65
		Peng's	9,824	27.19
Exam. 1, body 1 movable	*hyper box*	Proposed	3,439	10.98
		Peng's	30	12.77
	hyper sphere	Proposed	2,666	10.07
		Peng's	238	10.83
Exam. 2	*hyper box*	Proposed	8,120	8.92
		Peng's	651	58.74
	hyper sphere	Proposed	6,446	7.42
		Peng's	1,252	62.76
Exam. 3	*hyper box*	Proposed	1,592	8.88
		Peng's	1,681	13.65
	hyper sphere	Proposed	1,311	8.49
		Peng's	1,531	13.63
Exam. 4	*hyper box*	Proposed	8,250	8.42
		Peng's	7,748	10.94
	hyper sphere	Proposed	8,547	8.43
		Peng's	8,082	10.99

performed as follows. For each example, we generate 10, 000 initial estimates uniformly at random from an estimate region to test whether Algorithm 1 converges, using our and compared formulae separately. We count the convergent cases and calculate the average iterations needed for each successful case. The two estimate regions to concern are

1. $\{(e_0, e_1, e_2, e_3) | \, |e_i| \leq 1\}$ (denoted as *hyper box*)
2. $\{\mathbf{q}_0 | \mathbf{q}_0^T \mathbf{q}_0 = 1\}$ (denoted as *hyper sphere*)

The difference between these two regions is that the latter is the set where the initial estimate satisfies the normalization condition.

Other used examples are listed as follows:

Example 2 The constraint system consists of three distant constraints $(\Phi^{dist}(\mathbf{P}_i, \mathbf{Q}_i, d_i), i = 1 \cdots 3)$.

Example 3 The constraint system consists of two dot-1 constraints $(\Phi^{d1}(\mathbf{u}_i, \mathbf{v}_i), \; i = 1, 2)$ and one dot-2 constraint $(\Phi^{d2}(\mathbf{u}_3, \mathbf{P}_1, \mathbf{Q}_1))$.

Example 4 The constraint system consists of two angle constraints $(\Phi^{ang}(\mathbf{u}_i, \mathbf{v}_i, \alpha_i), \; i = 1, 2)$ and one distant constraint $(\Phi^{dist}(\mathbf{P}_1, \mathbf{Q}_1, d_1))$.

The parameters used in the examples are given in Table 1. The simulation results are listed in Table 2. The results show that the iteration methods adopting our presented formulae as Jacobian coefficients are more stable. As depicted in Table 2, there are more convergent cases in general, using proposed formulae. Moreover, our proposed formulae will make the iteration method converge faster in fewer iterations. The experimental results demonstrate the feasibility of the method presented in this paper.

4 Conclusion

In this paper, the erroneous derivation of Jacobian coefficients of primitive constraint equations with respect to Euler parameters presented in literature is discussed, and correct formulae are derived using variational-vector calculus method. A recursive formulation of Jacobian coefficients of primitive constraints in relative generalized coordinates is also proposed. Compared with the formulae presented in literature, this formulation can be applied to all types of joint variables. Experimental results are presented to illustrate the correctness and computational efficiency of the formulae presented in this paper.

Acknowledgments The research was supported by Chinese 973 Program (2010CB328001) and Chinese 863 Program (2012AA040902). The first author was supported by the NSFC(61035002, 61173077). The second author was supported by the NSFC (61063029). The last author was supported by the NSFC (61272235).

References

1. Kim SH, Lee K (1989) An assembly modelling system for dynamic and kinematic analysis. Comput Aided Des 21:2–12
2. Anantha R, Kramer GA, Crawford RH (1996) Assembly modeling by geometric constraint satisfaction. Comput Aided Des 28:707–722
3. Kim JS, Kim KS, Lee JY, Jeong JH (2005) Generation of assembly models from kinematic constraints. Int J Adv Manuf Technol 26(1–2):131–137
4. Kramer GA (1992) A geometric constraint engine. Artif Intell 58:327–360
5. Kim J, Kim K, Choi K, Lee JY (2000) Solving 3D geometric constraints for assembly modelling. Int J Adv Manuf Technol 16:843–849
6. Zou H, Abdel-Malek K, Wang J (1996) Computer-aided design using the method of cut-joint kinematic constraints. Comput Aided Des 28:795–806
7. Li YT, Hu SM, Sun JG (2002) A constructive approach to solving 3-D geometric constraint systems using dependence analysis. Comput Aided Des 34:97–108
8. Kim JS, Kim KS, Lee JY, Jung HB (2004) Solving 3D geometric constraints for closed-loop assemblies. Int J Adv Manuf Technol 23:755–761
9. Peng XB, Lee KW, Chen LP (2006) A geometric constraint solver for 3-D assembly modeling. Int J Adv Manuf Technol 28:561–570
10. Shi ZL, Chen LP (2007) An angular constraints solving approach for assembly modeling based on spherical geometry. Int J Adv Manuf Technol 32:366–377
11. Xia HJ, Wang BX, Chen LP, Huang ZD (2008) 3D geometric constraint solving using the method of kinematic analysis. Int J Adv Manuf Technol 35:711–722
12. Haug EJ, McCullough MK (1986) A variational-vector calculus approach to machine dynamics. J Mech Transm Autom Des 108(1):25–30
13. Bae DS, Haug EJ (1987) A recursive formulation for constrained mechanical system dynamics: Part I. Open loop systems. Mech Struct Mach 15:359–382
14. Bae DS, Haug EJ (1987) A recursive formulation for constrained mechanical system dynamics: Part II. Closed loop systems. Mech Struct Mach 15:481–506
15. Haug EJ (1989) Computer aided kinematics and dynamics of mechanical systems: basic method. Allyn and Bacon, Boston
16. Bae DS, Han JM, Yoo HH (1999) A generalized recursive formulation for constrained mechanical system dynamics. Mech Struct Mach 27(3):293–315
17. Yen J, Chou CC (1993) Automatic generation and numerical integration of differential-algebraic equations of multibody dynamics. Comput Methods Appl Mech Engrg 104(3):317–331