Geometric Integration on Euclidean Group With Application to Articulated Multibody Systems

Jonghoon Park, *Member, IEEE,* and Wan-Kyun Chung, *Member, IEEE*

***Abstract*—Numerical integration methods based on the Lie group theoretic geometrical approach are applied to articulated multibody systems with rigid body displacements, belonging to the special Euclidean group $SE(3)$, as a part of generalized coordinates. Three Lie group integrators, the Crouch–Grossman method, commutator-free method, and Munthe–Kaas method, are formulated for the equations of motion of articulated multibody systems. The proposed methods provide singularity-free integration, unlike the Euler-angle method, while approximated solutions always evolve on the underlying manifold structure, unlike the quaternion method. In implementing the methods, the exact closed-form expression of the differential of the exponential map and its inverse on $SE(3)$ are formulated in order to save computations for its approximation up to finite terms. Numerical simulation results validate and compare the methods by checking energy and momentum conservation at every integrated system state.**

***Index Terms*—Differential equations on Lie groups, multibody dynamics, numerical integration, special Euclidean group.**

I. Introduction

ARTICULATED multibody systems play an important role in many aspects of robotics. The significance became magnified recently as many new platforms employing articulated multibody configurations emerge, such as humanoids, vehicles with manipulators, and the like. In the study of these systems, the capability of dynamic simulation is an indispensable requirement.

In general, dynamic simulation consists of two parts: forward dynamics and integration. Forward dynamics computes the derivative of state variables including system accelerations, given force command. There are many efficient recursive algorithms available for forward dynamics computation, for both open-loop and closed-loop systems (see [1]–[4] and references therein). To complete dynamic simulation, the computed state derivatives should be integrated to predict the ensuing motion. Integration, or equivalently, solving the differential equations, cannot be done analytically unless the system, as well as the force command, are particularly simple. Numerical integration hence should be adopted. For a multibody system with *independent* generalized coordinates, integration is a straightforward application of many available numerical algorithms solving ordinary differential equations (ODEs) [5], e.g., the Runge–Kutta (RK) method and its variants. The independence of the generalized coordinates is a very important premise which guarantees successful approximation of a numerically integrated solution, since under the assumption, the underlying state space is considered to be $\mathbb{R}^n$, the n-dimensional real vector space. It is the case with most open-loop grounded systems, as the generalized coordinates consist of independent joint coordinates expressing each joint motion.

Many articulated multibody systems include the rigid-body displacements of some bodies within the generalized coordinates. A most natural mathematical way to embody rigid body displacements is to represent them as an element of the *special Euclidean group* $SE(3)$ [6]. The special Euclidean group $SE(3)$ is the semidirect product [7] of $\mathbb{R}^3$ and the *special orthogonal group* $SO(3)$ consisting of orthogonal matrices having unit determinant, that is, rotation matrices. An element belonging to $SE(3)$ is represented by the homogeneous transformation matrix. The difficulty arises as the group is not $\mathbb{R}^6$. Consequently, the differential equations evolving on $SE(3)$ cannot be properly integrated by conventional RK methods developed for $\mathbb{R}^n$.

Parametrization of rotation matrices using a set of independent three or more parameters has been applied to circumvent this difficulty. Two most common reparametrization methods are Euler angle representation and unit quaternion representation [6]. The Euler angle representation, adopting only three independent parameters, suffers from artificial and *nonintrinsic* singularity, which leads to numerical and analytical difficulty. The unit quaternions approach leads to singularity-free parametrization of rotation matrices, but at the cost of one additional parameter, hence, in total, four. Numerical integration should deal with one algebraic equation expressing normalization constraint. This algebraic constraint is resolved at every step by projecting the approximated solution onto the manifold consisting of unit quaternions. For the unit quaternion method, this is usually done by normalizing the approximated quaternion. One can find details of implementing rotation matrices reparametrization within the framework of dynamic simulation in [8].

Recently, many geometric integration methods have been developed [9]. Underlying manifold structure can be preserved by integrating differential equations on the manifold [10], [11]. For good summary of this approach, one can consult [12] and [13]. Alternatively, one can enforce the constraints explicitly by imposing the constraints within a time-stepping framework [14], [15] or within the framework of differential algebraic equations [16]. In particular, free rotational dynamics of a rigid body attracted great interest due to its property of conserving energy

Manuscript received July 8, 2004; revised November 10, 2004. This paper was recommended for publication by Associate Editor K. Lynch and Editor I. Walker upon evaluation of the reviewers' comments. This work was supported in part by the Korea Research Foundation under Grant KRF-2003-003-D00015. This paper was presented in part at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004.

The authors are with the Department of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang 790-784, Republic of Korea (e-mail: coolcat@postech.ac.kr; wkchung@postech.ac.kr).

Digital Object Identifier 10.1109/TRO.2005.852253

and momentum, as well as symplectic structure [17]–[20]. To impose a time-symmetry property on Lie-group integrators, implicit schemes have been developed [21]. Implicit numerical integration algorithms require a numerical solution to some algebraic equations while integrating. For nonlinear equations defined on a Lie group, a quadratically convergent Newton iteration scheme has also been developed [22].

In this paper, we propose Lie group theoretic geometric methods[1] for numerical integration of the equations of motion of articulated multibody systems with rigid body displacements as a part of generalized coordinates. The key insight lies in the recognition that the equations of motion result in differential equations on a Lie group. Then, numerical integration methods are applied to preserve the group structure while integrating the equation. Highlighting the advantage of the methods in advance, they lead to singularity-free integration, like the quaternions method. Furthermore, the algorithms do not involve any algebraic constraints, unlike the quaternion method. The cost is the additional computations required for matrix multiplications associated with exponential matrices. This is the cost from the viewpoint of numerical computation. On the other hand, there is a gain from the analytical viewpoint. The method does handle the equation of motion as it is, without any artificial reparametrization, or transformation, of them.

The paper is structured as follows. First, a brief summary of the special Euclidean group, $SE(3)$, is provided in Section II. The structure of its Lie algebra $\mathfrak{se}(3)$ and a novel closed-form expression of the associated exponential map are given. Further, the closed-form formula of the differential of the exponential map and its inverse are provided. Then, differential equations on a matrix Lie group are introduced briefly, and three important classes of *explicit* Lie-group integrators, that is the Crouch–Grossman (CG) method, the commutator-free (CF) method, and the Munthe-Kaas (MK) method, are briefly summarized, focusing on their algorithms in Section III. They are the bases of the proposed methods. The equations of motion of a single rigid body are examined in Section IV, and we formulate the numerical integration methods for the resulting differential equation on a Lie group. Then, the algorithms are applied to general articulated multibody systems in the next section. Numerical simulation results employing an articulated multibody system are summarized in Section VI to validate the properness of the proposed method and to compare performance of individual integration methods.

II. Special Euclidean Group $SE(3)$

The set of every rigid transformation on $\mathbb{R}^3$, denoted by T such that, for every $p \in \mathbb{R}^3$, $T(p) = Rp + r$ for $R \in SO(3)$ and $r \in \mathbb{R}^3$, forms the special Euclidean group $SE(3)$ [6]. One can identify every element in $SE(3)$, say $T \in SE(3)$, as a 4×4 homogeneous transformation matrix of the form

$$T = \begin{bmatrix} R & r \\ 0 & 1 \end{bmatrix}. \tag{1}$$

[1]Robotics is one of the many research areas which benefited from Lie group theory. Recent application of Lie group theory to robotics includes [23] and [24].

It can be verified that the group $SE(3)$ is indeed a Lie group of six dimensions with the usual matrix multiplication. A *Lie group* is a group G which is a differentiable manifold, and for which the product $G \times G \to G$ is a differentiable mapping. We consider *matrix Lie groups*, that is, Lie groups which are subgroups of $GL(n)$, the group of invertible $n \times n$ matrices with the usual matrix product as the group operation. As mentioned, the special Euclidean group $SE(3)$ is the semidirect product [7] of $\mathbb{R}^3$ and the *special orthogonal group* $SO(3)$. Rotation matrices form the group $SO(3)$ defined by

$$SO(3) = \{R \in \mathbb{R}^{3\times 3} | RR^T = I, \det R = 1\} \tag{2}$$

where det is the matrix determinant, while displacement vectors form $\mathbb{R}^3$. The matrix denoted by I stands for the identity matrix of suitable dimension.

A. Lie Algebra $\mathfrak{se}(3)$

Define the tangent space $\mathfrak{g} = T_I G$ at the identity I of a matrix Lie group G. This forms the *Lie algebra* of the Lie group G. The Lie algebra of $SE(3)$, denoted by $\mathfrak{se}(3)$, is isomorphic to $\mathbb{R}^6$, that is $\mathfrak{se}(3)$ consists of the vectors $V = (v^T, \omega^T)^T \in \mathbb{R}^6$ for $v, \omega \in \mathbb{R}^3$. Every element of $\mathfrak{se}(3)$, say V, can be identified with a 4 × 4 matrix, denoted by $\lceil V \rceil$, of the form

$$\lceil V \rceil = \begin{bmatrix} \lceil \omega \rceil & v \\ 0 & 0 \end{bmatrix} \tag{3}$$

where $\lceil \omega \rceil$ is the 3×3 skew-symmetric matrix representation of $\mathfrak{so}(3)$, the Lie algebra of the special orthogonal group $SO(3)$. For $\omega = (\omega_1, \omega_2, \omega_3)^T \in \mathbb{R}^3$, we have[2]

$$\lceil \omega \rceil = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{4}$$

We adopt the convention that $V \in \mathfrak{se}(3)$ is the left trivialization[3] of the tangent vector at $T \in SE(3)$, i.e.,

$$\lceil V \rceil = T^{-1}\dot{T}. \tag{5}$$

The Lie bracket for $\mathfrak{se}(3) \simeq \mathbb{R}^6$, denoted by $[\cdot,\cdot] : \mathfrak{se}(3) \times \mathfrak{se}(3) \to \mathfrak{se}(3)$ is the usual matrix commutator defined by $[\lceil V_1 \rceil, \lceil V_2 \rceil] = \lceil V_1 \rceil \lceil V_2 \rceil - \lceil V_2 \rceil \lceil V_1 \rceil$. Making use of the adjoint operator defined by

$$\mathrm{ad}_V = \begin{bmatrix} \lceil \omega \rceil & \lceil v \rceil \\ 0 & \lceil \omega \rceil \end{bmatrix} \tag{6}$$

for $V = (v^T, \omega^T)^T$, it is readily verified that

$$[\lceil V_1 \rceil, \lceil V_2 \rceil] = \lceil \mathrm{ad}_{V_1} V_2 \rceil$$

since

$$\mathrm{ad}_{V_1} V_2 = \begin{bmatrix} \omega_1 \times v_2 - \omega_2 \times v_1 \\ \omega_1 \times \omega_2 \end{bmatrix}.$$

[2]Note that the operator $\lceil \cdot \rceil$ is overloaded. When it applies to a vector $\omega \in \mathfrak{so}(3)$, it yields the matrix representation of $\mathfrak{so}(3)$, as in (4). When it applies to a twist $V \in \mathfrak{se}(3)$, this yields the matrix representation of $\mathfrak{se}(3)$, as in (3).

[3]In terms of the terminology of [6], the left trivialized form of the tangent vector corresponds to the body twist, while the right trivialized form to the spatial twist of a rigid body.

Coordinate transformation of $V \in \mathfrak{se}(3)$ is done by the adjoint transformation induced by $T \in SE(3)$ by the following relationship:

$$\lceil \mathrm{Ad}_T V \rceil = T \lceil V \rceil T^{-1}. \tag{7}$$

It is easy to verify that

$$\mathrm{Ad}_T = \begin{bmatrix} R & \lceil r \rceil R \\ 0 & R \end{bmatrix} \tag{8}$$

for T given in (1). Then, the derivative of Ad_T is related to ad_V by

$$\mathrm{ad}_V = \mathrm{Ad}_T^{-1} \frac{d}{dt} \mathrm{Ad}_T \tag{9}$$

where V is given by (5).

B. Exponential Map and Its Differential

For a general matrix Lie group G and its Lie algebra $\mathfrak{g}$, the exponential map, $\exp : \mathfrak{g} \to G$, is defined as the matrix exponential. For $SE(3)$, the exponential map of $V \in \mathfrak{se}(3)$ is defined as the matrix exponential of $\lceil V \rceil$, i.e., $\exp(V) = \mathrm{expm}(\lceil V \rceil) \in SE(3)$, where $\mathrm{expm}(A) = \sum_{j=0}^{\infty} (1/j!) A^j$. Upon series evaluation, it is analytically given, for $\omega = 0$

$$\exp(V) = \begin{bmatrix} I & v \\ 0 & 1 \end{bmatrix} \tag{10a}$$

and for nonzero ω

$$\exp(V) = \begin{bmatrix} \Omega & \frac{1}{\|\omega\|^2} \left((I - \Omega)(\omega \times v) + \omega \omega^T v \right) \\ 0 & 1 \end{bmatrix} \tag{10b}$$

where $\Omega = \exp(\omega) \triangleq \mathrm{expm}(\lceil \omega \rceil) \in SO(3)$ is defined by

$$\exp(\omega) = I + \frac{\sin(\|\omega\|)}{\|\omega\|} \lceil \omega \rceil + \frac{\sin^2\left(\frac{\|\omega\|}{2}\right)}{\frac{\|\omega\|^2}{2}} \lceil \omega \rceil^2. \tag{11}$$

The closed-form expression (10) is known as Rodrigues' formula [6]. For efficient computation of its differential and the product inverse given below, we express the formula (11) as

$$\exp(\omega) = I + \alpha \lceil \omega \rceil + \frac{1}{2} \beta \lceil \omega \rceil^2 \tag{12}$$

where

$$\alpha = sc \quad \text{and} \quad \beta = s^2 \tag{13}$$

for

$$s = \frac{\sin\left(\frac{\|\omega\|}{2}\right)}{\frac{\|\omega\|}{2}}, \quad \text{and} \quad c = \cos\left(\frac{\|\omega\|}{2}\right). \tag{14}$$

For later use, let us define γ as

$$\gamma = \frac{c}{s}. \tag{15}$$

The differential of the exponential map is defined as the tangent of the exponential map, that is, as a function $\mathrm{dexp} : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, such that

$$\frac{d}{dt} \exp(A(t)) = \mathrm{dexp}_{A(t)} \left(\dot{A}(t) \right) \exp(A(t)).$$

It is expressed as[4]

$$\begin{aligned} \mathrm{dexp}_A(C) &= C + \frac{1}{2!}[A, C] + \frac{1}{3!}[A, [A, C]] + \cdots \\ &= \sum_{j=0}^{\infty} \frac{1}{(j+1)!} \mathrm{ad}_A^j C. \end{aligned} \tag{16}$$

Its inverse is given by

$$\begin{aligned} \mathrm{dexp}_A^{-1}(C) &= C - \frac{1}{2}[A, C] + \frac{1}{12}[A, [A, C]] + \cdots \\ &= \sum_{j=0}^{\infty} \frac{B_j}{j!} \mathrm{ad}_A^j C \end{aligned} \tag{17}$$

where B_j is the jth Bernoulli number for $j = 0, 1, 2, \ldots$ [12].

The following lemma shows that the differential of exponential map and its inverse on $\mathfrak{se}(3)$ can be expressed using closed-form formulas, just as the exponential map on $\mathfrak{se}(3)$. It is known that on $\mathfrak{so}(3)$, the analytic expression[5] has been obtained [12] by

$$\mathrm{dexp}_\omega = I + \frac{\sin^2\left(\frac{\|\omega\|}{2}\right)}{\frac{\|\omega\|^2}{2}} \lceil \omega \rceil + \frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3} \lceil \omega \rceil^2$$

$$\mathrm{dexp}_\omega^{-1} = I - \frac{1}{2} \lceil \omega \rceil - \frac{\|\omega\| \cot\left(\frac{\|\omega\|}{2}\right) - 2}{2\|\omega\|^2} \lceil \omega \rceil^2.$$

When $\omega = 0$, $\mathrm{d}\exp_\omega = \mathrm{d}\exp_\omega^{-1} = I$. It looks a little bit more involved to compute the differential and its inverse of the exponential map on $SO(3)$, mainly due to computation of an additional trigonometric function. However, one can see that they can be easily computed by the following formula:

$$\mathrm{dexp}_\omega = I + \frac{1}{2} \beta \lceil \omega \rceil + \frac{1}{\|\omega\|^2} (1 - \alpha) \lceil \omega \rceil^2 \tag{18}$$

$$\mathrm{dexp}_\omega^{-1} = I - \frac{1}{2} \lceil \omega \rceil + \frac{1}{\|\omega\|^2} (1 - \gamma) \lceil \omega \rceil^2 \tag{19}$$

where α, β, and γ are computed previously by (13) and (15).

In the same spirit, we obtained the following analytic expressions of the differential of exponential map and its inverse on $\mathfrak{se}(3)$. It is a simple matter to verify the derived expressions using some symbolic expression-manipulation packages.

[4]Slight abuse of notation has been allowed to express (16) and (17). Precisely expressing (16), employing the representation of a Lie algebra using $\mathbb{R}^n$, would be

$$\begin{aligned} \lceil \mathrm{dexp}_A(C) \rceil &= \lceil C \rceil + \frac{1}{2!} [\lceil A \rceil, \lceil C \rceil] \\ &\quad + \frac{1}{3!} [\lceil A \rceil, [\lceil A \rceil, \lceil C \rceil]] + \cdots \\ &= \sum_{j=0}^{\infty} \frac{1}{(j+1)!} \lceil \mathrm{ad}_A^j C \rceil. \end{aligned}$$

Similar expression applies to (17).

[5]The closed-form expression has another advantage besides the exact evaluation. It should be noted that dexp_ω is only singular for $\sin(\|\omega\|/2) = 0$, i.e., $\|\omega\| = \pm 2k\pi$ for $k = 1, 2, \ldots$. To the contrary, the series expression of $\mathrm{dexp}_\omega^{-1}$, given by (17), does not converge only if $|\omega| \geq 2\pi$.

Lemma 1: For a given $V = (v^T, \omega^T)^T \in \mathfrak{se}(3)$ with nonzero ω, the differential of exponential map is represented as a 6×6 matrix

$$\mathrm{dexp}_V = \begin{bmatrix} \mathrm{dexp}_\omega & C + \frac{1}{2}\lceil v \rceil \\ 0 & \mathrm{dexp}_\omega \end{bmatrix} \tag{20}$$

where

$$C = -\frac{1-\beta}{2}\lceil v \rceil + \frac{1-\alpha}{\|\omega\|^2}\lceil v, w \rceil + \frac{\alpha-\beta}{\|\omega\|^2}(\omega^T v)\lceil \omega \rceil + \frac{1}{\|\omega\|^2}\left(\frac{1}{2}\beta - \frac{3}{\|\omega\|^2}(1-\alpha)\right)(\omega^T v)\lceil \omega \rceil^2 \tag{21}$$

where

$$\lceil v, w \rceil = \lceil v \rceil \lceil \omega \rceil + \lceil \omega \rceil \lceil v \rceil. \tag{22}$$

Its inverse exists if $\|\omega\| \neq 2k\pi$ for $k = 1, 2, \ldots$, and is expressed as

$$\mathrm{dexp}_V^{-1} = \begin{bmatrix} \mathrm{dexp}_\omega^{-1} & D - \frac{1}{2}\lceil v \rceil \\ 0 & \mathrm{dexp}_\omega^{-1} \end{bmatrix} \tag{23}$$

where

$$D = \frac{1-\gamma}{\|\omega\|^2}\lceil v, w \rceil + \frac{\frac{1}{\beta} + \gamma - 2}{\|\omega\|^4}(\omega^T v)\lceil \omega \rceil^2. \tag{24}$$

If $\omega = 0$, $C = 0$ and $D = 0$.

When $\sin(\|\omega\|/2) = 0$ and $\omega \neq 0$, dexp_V becomes singular. In other words, only then, dexp_V^{-1} does not exist.

The computational complexity in computing the exponential map and the additional operations to compute the inverse of the differential of the exponential map on $SE(3)$ are analyzed in Appendix I-A. Summarizing, one computation of $\exp(V)$ requires 26 scalar additions, 38 scalar multiplications, two scalar divisions as well as one square-root operation, and two trigonometric function evaluations. The additional computations necessary to compute $\mathrm{dexp}_\omega^{-1}$ is summarized to be 33 scalar additions, the same number of multiplications, and three scalar divisions. Note that this is almost same complexity required for one matrix-vector multiplication of dimension 6×6.

III. Numerical Integration on Lie Groups

Let us recall the explicit RK methods [5]. Suppose that the differential equations $\dot{x} = f(x(t), t)$ evolving on $x \in \mathbb{R}^n$ are given. Given the state value x_k at time $t = t_k$, the solution to the equation at time $t_{k+1} = t_k + h$, denoted by $x(t_{k+1})$, is approximated by x_{k+1} computed as

$$x_{k+1} = x_k + h\sum_{i=1}^{s} b_i \mathcal{K}^{(i)} \tag{25a}$$

$$\mathcal{K}^{(i)} = f\left(x^{(i)}, t_k + c_i h\right) \tag{25b}$$

$$x^{(i)} = x_k + h\sum_{j=1}^{i-1} a_{ij}\mathcal{K}^{(j)}. \tag{25c}$$

The algorithm is called the s-stage explicit RK method. The algorithm is explicit in that for computing the value of $x^{(i)}$, the values of $\mathcal{K}^{(j)}$ for $j \geq i$ are not required. In general, an s-stage RK method can be compactly represented by a Butcher tableau [5]

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

The accuracy of the approximation is controlled by the notion of order. An RK method has order p if the Taylor series for the exact solution $x(t_{k+1})$ and for the approximated solution x_{k+1} coincide up to (and including) the term h^p, i.e., $\|x(t_{k+1}) - x_{k+1}\| \leq \alpha h^{p+1}$ for $\alpha \geq 0$ [5]. The order is determined by the constants a_{ij}, b_i, and c_j. A third-order and fourth-order explicit RK method can be implemented in three and four stages with the coefficient tableaux given by Table III(a) and (b), respectively.

A. Differential Equations on Lie Group

Differential equations on a matrix Lie group G are written as [9], [25]

$$\dot{Y} = A(t, Y)Y, \quad Y(0) = Y_0 \in G \tag{26}$$

where $A(t, Y) \in \mathfrak{g}$, the Lie algebra of G, for all $t \geq 0$ and $Y \in G$. As the tangent space at $Y \in G$ has the form

$$T_Y G = \{AY | A \in \mathfrak{g}\} \tag{27}$$

the solution of (26) satisfies $Y(t) \in G$ for all t.

Caution should be exercised in approximating, or numerically integrating, a solution to the differential equation on a matrix Lie group, as the conventional RK methods have the shortcoming that the approximated solution Y_{k+1} does not belong to the Lie group any more, even when $Y_k \in G$. For example, consider a differential equation on $SO(3)$ of the form (26) with $A(t, Y) \in \mathfrak{so}(3)$. It can be verified that $g(Y) = \det Y$ is an invariant of the differential equation, that is, $g'(Y)AY = 0$, where $g'(Y)$ denotes the partial derivative with respect to Y. However, no RK method can conserve $\det Y = \det Y_0 = 1$. In fact, it was shown that no RK method can conserve polynomial invariants of degree n for $n \geq 3$ [9, Th. IV.3.3]. As the determinant is polynomial of degree 3, it cannot be conserved by RK methods. Consequently, though $Y(t_k) = Y_k \in SO(3)$, an approximation Y_{k+1} at $t_{k+1} = t_k + h$ for $h > 0$ by any RK method does not belong to $SO(3)$.

B. Lie Group Method: Crouch–Grossman Method

For $\mathbb{R}^n$ the update operation by RK methods (25a) corresponds to the exponential map of the associated Lie algebra, which is also $\mathbb{R}^n$. Hence, the updated states evolve on $\mathbb{R}^n$. If the group is not $\mathbb{R}^n$ but a Lie group, one can conjecture that the update should be made by the exponential map of its Lie algebra. This is the very idea of Crouch and Grossman [10]. The algorithm below is referred to as the CG method hereinafter. For $i = 1, 2, \ldots, s$

$$Y^{(i)} = \exp\left(h a_{i,i-1} K^{(i-1)}\right) \cdots \exp\left(h a_{i1} K^{(1)}\right) Y_k \tag{28a}$$

$$K^{(i)} = A\left(t_k + h c_i, Y^{(i)}\right) \tag{28b}$$

$$Y_{k+1} = \exp\left(h b_s K^{(s)}\right) \cdots \exp\left(h b_1 K^{(1)}\right) Y_k. \tag{28c}$$

TABLE I
TABLEAUX OF CG METHOD OF ORDER 3 AND 4. (a) CG3 [10]. (b) CG4 [27]

0				
3/4	3/24			
17/24	119/216	17/108		
	13/51	−2/3	24/17	

(a)

0					
$\frac{1458}{1783}$	$\frac{1458}{1783}$				
$\frac{743}{1925}$	$\frac{1039}{3247}$	$\frac{97}{1470}$			
$\frac{368}{1135}$	$\frac{997}{1082}$	$\frac{1167}{2335}$	$-\frac{475}{433}$		
$\frac{406}{463}$	$\frac{173}{487}$	$\frac{751}{3141}$	$\frac{547}{393}$	$-\frac{680}{613}$	
	$\frac{407}{2969}$	$-\frac{135}{7349}$	$\frac{543}{734}$	$-\frac{267}{1400}$	$\frac{696}{2095}$

(b)

TABLE II
TABLEAUX OF CF METHOD OF ORDER 3 AND 4. (a) CF3 [28]. (b) CF4 [28]

0			
1/3	1/3		
2/3	0	2/3	
	1/3	0	0
	−1/12	0	3/4

(a)

0				
1/2	1/2			
1/2	0	1/2		
1	1/2	0	0	
	−1/2	0	1	
	1/4	1/6	1/6	−1/12
	−1/12	1/6	1/6	1/4

(b)

The definition given by (26) asserts that $ahA(t_k + hc_i, Y^{(i)})$ belongs to the associated Lie algebra. By the property of the exponential map and by the construction of the update operation, the CG methods always give rise to an approximation Y_{k+1}, which lies exactly on the manifold defined by the Lie group.

The accuracy of the CG methods is also indicated by the order condition [9]. Explicit computation of the coefficients up to the fourth-order condition [26] has been made, and we employed the tableaux presented in Table I. It is worth noting that the fourth-order CG method cannot be implemented in four stages, but five stages are required.

C. Commutator-Free Lie Group Method

The fourth-order CG method requires 15 evaluations of the exponential map, and 15 homogeneous transformation matrix multiplications during one step. For a general matrix Lie group, saving computation of exponential maps is desired, as it is very time-consuming.[6] One method to circumvent this is the MK method, involving commutator evaluation, which will be described in the next subsection. The other one is the CF Lie group method [28].

The CF method is implemented as follows. For $i = 1, 2, \ldots, s$

$$Y^{(i)} = \exp\left(\Sigma_j h\alpha_{i,j}^{[\kappa]} K^{(j)}\right) \cdots \exp\left(\Sigma_j h\alpha_{i,j}^{[1]} K^{(j)}\right) Y_k \quad (29a)$$

$$K^{(i)} = A\left(t_k + hc_i, Y^{(i)}\right) \quad (29b)$$

$$Y_{k+1} = \exp\left(\Sigma_j h\beta_j^{[\kappa]} K^{(j)}\right) \cdots \exp\left(\Sigma_j h\beta_j^{[1]} K^{(j)}\right) Y_k. \quad (29c)$$

In the algorithm described above, the parameter κ counts the number of exponentials for each stage. It is allowed for κ to have a different value for each stage. It is determined so as to minimize the total number of exponentials. If it is s and $\alpha_{ij}^{[\kappa]} = \delta_{j\kappa} a_{i\kappa}$, the algorithm reduces to the CG method.

Saving of computation of exponentials is possible by reusing the previously computed exponentials. The following example will clarify this. The third-order CF method is implemented by the tableau given in Table II(a), where the first row of the b section contains $\beta^{[1]}$ and the second row $\beta^{[2]}$. As $\alpha_{2,j}^{[1]} = \beta_j^{[1]}$ for all j, one can save one computation of exponential. In particular, the tableau leads to

$$Y^{(1)} = Y_k,\ K^{(1)} = A\left(t_k, Y^{(1)}\right)$$

$$Y^{(2)} = \exp\left(\left(\frac{h}{3}\right)K^{(1)}\right)Y_k,\ K^{(2)} = A\left(t_k + \left(\frac{h}{3}\right), Y^{(2)}\right)$$

$$Y^{(3)} = \exp\left(\left(\frac{2h}{3}\right)K^{(2)}\right)Y_k,\ K^{(3)} = A\left(t_k + \left(\frac{2h}{3}\right), Y^{(3)}\right)$$

$$Y_{k+1} = \exp\left(-\left(\frac{h}{12}\right)K^{(1)} + \left(\frac{3h}{4}\right)K^{(3)}\right)\exp\left(\left(\frac{h}{3}\right)K^{(1)}\right)Y_k.$$

Note that as the last two terms in computing Y_{k+1} are same as $Y^{(2)}$, one can save the computation by

$$Y_{k+1} = \exp\left(-\left(\frac{h}{12}\right)K^{(1)} + \left(\frac{3h}{4}\right)K^{(3)}\right)Y^{(2)}.$$

Only three evaluations of the exponential map and three `hmmult` are required to implement the third-order method, while the third-order CG method requires six exponential maps and six multiplications.

One can also implement the fourth-order CF method using the tableau given in Table II(b). Note that the α_4 section has two rows, and the first row is the same as the row of α_2, which saves one computation. According to the tableau, we have the following CF4 method requiring five evaluations of exponential maps and five multiplications:

$$Y^{(1)} = Y_k,\ K^{(1)} = A\left(t_k, Y^{(1)}\right)$$

$$Y^{(2)} = \exp\left(\frac{h}{2}K^{(1)}\right)Y_k,\ K^{(2)} = A\left(t_k + \frac{h}{2}, Y^{(2)}\right)$$

$$Y^{(3)} = \exp\left(\frac{h}{2}K^{(2)}\right)Y_k,\ K^{(3)} = A\left(t_k + \frac{h}{2}, Y^{(3)}\right)$$

$$Y^{(4)} = \exp\left(-\frac{h}{2}K^{(1)} + K^{(3)}\right)Y^{(2)},\ K^{(4)} = A\left(t_k + h, Y^{(4)}\right)$$

$$Y_{k+1} = \exp\left(-\frac{h}{12}K^{(1)} + \frac{h}{6}K^{(2)} + \frac{h}{6}K^{(3)} + \frac{h}{4}K^{(4)}\right) \cdot \exp\left(\frac{h}{4}K^{(1)} + \frac{h}{6}K^{(2)} + \frac{h}{6}K^{(3)} - \frac{h}{12}K^{(4)}\right)Y_k.$$

[6] It was reported that when a Lie group in question is realized as a group of $n \times n$ matrices, the cost of the exponential is typically of size $25n^3$ flops, and the commutator may cost $4n^3$ flops for large n [28].

TABLE III
TABLEAUX OF RK AND MK METHOD OF ORDER 3 AND 4.
(a) RK3 AND MK3. (b) RK4 AND MK4

0				
1/3	1/3			
2/3	0	2/3		
	1/4	0	3/4	

(a)

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	2/6	2/6	1/6

(b)

D. Lie Group Method: Munthe–Kaas Method

For a differential equation on a matrix Lie group G given by (26), it is well known that the solution is formally given by [12, Lemma 3.1]

$$Y(t) = \exp\left(\Theta(t)\right) Y_0 \tag{30}$$

where $\Theta \in \mathfrak{g}$ satisfies the differential equation

$$\dot{\Theta}(t) = \operatorname{dexp}^{-1}_{\Theta(t)} A(t, Y) := F\left(t, Y(t), \Theta(t)\right). \tag{31}$$

This can be easily seen as

$$\dot{Y}(t) = \operatorname{dexp}_{\Theta(t)} \dot{\Theta}(t) \exp\left(\Theta(t)\right) Y_0 = A(t, Y) Y(t).$$

Inverting $\operatorname{dexp}_{\Theta(t)}$ yields (31).

Note that (31) is a differential equation on a Lie algebra. Compared with solving the differential equation on a Lie group, one can enjoy the advantage that the differential equation on a Lie algebra can be integrated using the conventional RK methods, since the Lie algebra is isomorphic to $\mathbb{R}^n$. This is the key idea of MK [11] in implementing a Lie-group integrator, referred to as the MK method. The algorithm is implemented as follows. For $i = 1, 2, \ldots, s$

$$\Theta^{(i)} = h \sum_{j=1}^{i-1} a_{ij} F^{(j)} \tag{32a}$$

$$Y^{(i)} = \exp\left(\Theta^{(i)}\right) Y_k \tag{32b}$$

$$F^{(i)} = \operatorname{dexp}^{-1}_{\Theta^{(i)}} A\left(t_k + h c_i, Y^{(i)}\right) \tag{32c}$$

$$\Theta = h \sum_{i=1}^{s} b_i F^{(i)} \tag{32d}$$

$$Y_{k+1} = \exp(\Theta) Y_k. \tag{32e}$$

The differential equation (31) is integrated numerically as (32d) by an RK method of a desired order. For example, the standard fourth-order RK coefficients, shown in Table III, can be used to implement the fourth-order MK method. It was shown that the order condition of the underlying RK methods still holds for the MK method, even if $\operatorname{dexp}^{-1}_{\Theta^{(i)}} A(t_k + h c_i, Y^{(i)})$ is approximated by some finite terms [11]. In particular, those up to the heading $p - 1$ terms are sufficient for the pth-order implementation. For example, for the fourth-order implementation, it is approximated by $\operatorname{dexp}^{-1}_{\Theta} A \approx A - (1/2)[\Theta, A] + (1/12)[\Theta, [\Theta, A]]$, requiring two evaluations of commutator. In addition, four evaluations of exponential maps are needed.

TABLE IV
COMPLEXITY PER STEP OF CG, CF, AND MK METHODS FOR A GENERAL LIE GROUP [28]

order	3			4		
operation	exp[a]	com[b]	mmult[c]	exp	com	mult
CG	6	0	6	15	0	15
CF	3	0	3	5	0	5
MK	3	1	3	4	2	4

[a] exponential map
[b] commutator
[c] matrix multiplication

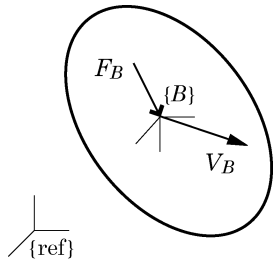


Fig. 1. Single rigid body.

Table IV compares the complexity of each method implemented for a general matrix Lie group. It should be noted that in our implementation of the Lie-group integrators, commutator computation is not necessary, as we have the closed-form expression of $\operatorname{dexp}^{-1}_{\Theta}$ for $\Theta \in \mathfrak{se}(3)$. Exponential map is also computed using the closed-form formula. A more detailed analysis of complexity will be given later. It is worth recapitulating that all these Lie-group integrators evolve the approximated solutions on the Lie group.

IV. A SINGLE RIGID BODY

A. Rigid Body Kinematics and Dynamics

The displacement and attitude of a body frame attached to the body, relative to a reference frame, is expressed as a homogeneous transformation matrix

$$T = \begin{bmatrix} R & r \\ 0 & 1 \end{bmatrix}$$

where r denotes the displacement vector of the origin of the body frame, say, $\{B\}$ in Fig. 1, with respect to the reference frame {ref}, and R is the rotation matrix of the body frame relative to the reference frame. The homogeneous transformation matrix is the representation of the special Euclidean group $SE(3)$.

Body twists are the geometric entity embodying the Lie algebra of $SE(3)$, denoted by $\mathfrak{se}(3)$. It consists of the translational and rotational velocity, denoted, respectively, by $v \in \mathbb{R}^3$ and $w \in \mathbb{R}^3$, of the body frame relative to the reference frame, which are represented with respect to the body frame. The body twist, denoted by V, is defined by

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix}.$$

The body twist has the relationship given by (5) with the time derivative of the homogeneous transformation. Equivalently, we have

$$v = R^T \dot{r} \tag{33a}$$
$$\lceil \omega \rceil = R^T \dot{R}. \tag{33b}$$

The body wrench consists of the force and moment applied to the body, at the origin of the body frame, which are expressed with respect to the body frame. It is defined by

$$F = \begin{bmatrix} f \\ n \end{bmatrix}$$

where $f \in \mathbb{R}^3$ and $n \in \mathbb{R}^3$ stand for the force and moment.

In terms of the body twist and wrench pair, the motion of the body is described by the following equation of motion [6, Exercise 4.6]:

$$F = A\dot{V} + BV \tag{34}$$

where

$$A = \begin{bmatrix} mI_3 & 0 \\ 0 & \mathcal{I} \end{bmatrix}, \quad B = \begin{bmatrix} m\lceil \omega \rceil & 0 \\ 0 & -\lceil \mathcal{I}\omega \rceil \end{bmatrix}$$

provided that the body frame is located at the center of mass of the body. In the equation, m and $\mathcal{I}$ are the mass and the rotational inertia of the body at the body frame. It should be noted that this particular form of the matrices is only valid for the body frame located at the center of mass of the body. Only then, the off-diagonal blocks become zero. If not, the matrices should be properly transformed to the body frame [6].

B. Geometric Integration Algorithm on $SE(3)$

First, rewrite the equations of motion (34) together with (5) in the following form:

$$\dot{T}(t) = T(t)\lceil V(t) \rceil \tag{35a}$$
$$\dot{V}(t) = \mathcal{K}(t, T(t), V(t)) \tag{35b}$$

where

$$\mathcal{K}(t, T(t), V(t)) = -A^{-1}B(V(t))V(t) + A^{-1}F(t). \tag{36}$$

Note that the body inertia A is constant, while the body bias matrix B depends on $\omega(t)$.

The key insight into the equations is that they are differential equations on a Lie group $SE(3) \times \mathfrak{se}(3) \simeq SE(3) \times \mathbb{R}^6$. The states consisting of $T \in SE(3)$ and $V \in \mathfrak{se}(3) \simeq \mathbb{R}^6$ form a Lie group, since the direct product of two Lie groups is also a Lie group.

Now we can apply the Lie-group integrators in a straightforward manner. As the underlying Lie group is a direct product of $SE(3)$ and $\mathfrak{se}(3) \simeq \mathbb{R}^6$, implementing a Lie-group integrator on the product group amounts to implementing a Lie-group integrator separately on each group and combining the both.

1) CG Method on $SE(3)$: Given $T(t_k)$ and $V(t_k)$ at time t_k, the CG method being applied to (35a) yields

$$T(t_{k+1}) = T(t_k) \prod_{i=1}^{s} \exp\left(hb_i V^{(i)}\right) \tag{37a}$$

and to (35b)

$$V(t_{k+1}) = V(t_k) + \sum_{i=1}^{s} hb_i \mathcal{K}^{(i)} \tag{37b}$$

as $\mathfrak{se}(3)$ is isomorphic to $\mathbb{R}^6$. The intermediate updates for $i = 1, 2, \cdots, s$ are computed by

$$\mathcal{K}^{(i)} = \mathcal{K}\left(t_k + c_i h, T^{(i)}, V^{(i)}\right) \tag{37c}$$
$$V^{(i)} = V(t_k) + \sum_{j=1}^{i-1} ha_{ij}\mathcal{K}^{(j)} \tag{37d}$$
$$T^{(i)} = T(t_k) \prod_{j=1}^{i-1} \exp\left(ha_{ij}V^{(j)}\right). \tag{37e}$$

Note that the exponential map is computed by (10). For any square matrices A_k, the post-product operator is defined as follows:

$$\prod_{k=m}^{n} A_k = \begin{cases} I, & \text{for } m > n \\ A_m \left(\prod_{k=m+1}^{n} A_k\right), & \text{for } m \le n. \end{cases} \tag{38}$$

Notice that due to the left trivialization of the Lie algebra, (35a) is in the form of $\dot{Y} = Y\,A(t, Y)$, not in the same form as (26). This is reflected in the right multiplication of the exponential map in the update operation.

2) CF Method on $SE(3)$: The CF method can also be applied to $SE(3) \times \mathfrak{se}(3)$. For (35a), a general CF method produces the approximation by

$$T(t_{k+1}) = T(t_k) \prod_{\nu=1}^{\kappa} \exp\left(h \sum_{j=1}^{s} \beta_j^{[\nu]} V^{(j)}\right) \tag{39a}$$

where

$$T^{(i)} = T(t_k) \prod_{\nu=1}^{\kappa} \exp\left(h \sum_{j=1}^{i-1} \alpha_{ij}^{[\nu]} V^{(j)}\right). \tag{39b}$$

When it is applied to a differential equation on $\mathfrak{se}(3) \simeq \mathbb{R}^6$, the exponential map is the identity map. Hence, partitioned rows in coefficient tableaux in each stage evaluation are summed as a single row. In particular, (29a) is simplified to

$$\begin{aligned} V^{(i)} &= V(t_k) + \sum_{\nu=1}^{\kappa} \sum_{j=1}^{i-1} h\alpha_{ij}^{[\nu]} K^{(j)} \\ &= V(t_k) + \sum_{j=1}^{i-1} h \left(\sum_{\nu=1}^{\kappa} \alpha_{ij}^{[\nu]}\right) K^{(j)}. \end{aligned} \tag{39c}$$

Similar reduction holds for (29c), that is

$$V(t_{k+1}) = V(t_k) + \sum_{j=1}^{i=1} h \left(\sum_{\nu=1}^{\kappa} \beta_j^{[\nu]} \right) K^{(j)}. \tag{39d}$$

It is worth noting that the coefficient tableaux in Table II become equal to those in Table III upon this kind of reduction being made. In other words, $b_j = \sum_{\nu=1}^{\kappa} \beta_j^{[\nu]}$ and $a_{ij} = \sum_{\nu=1}^{\kappa} \alpha_{ij}^{[\nu]}$.

3) MK Method on $SE(3)$: In applying the MK method to (35a), one has to accordingly formulate the differential equation on the Lie algebra such as (31). Consider the differential equation on a Lie group G

$$\dot{Y} = YA(t,Y),\ Y(0) = Y_0,\ A(t,Y) \in \mathfrak{g}. \tag{40}$$

Assuming the solution $Y(t) = Y_0 \exp(\Theta(t))$ with $\Theta \in \mathfrak{g}$ yields the differential equation on the Lie algebra $\mathfrak{g}$

$$\dot{\Theta}(t) = \operatorname{dexp}^{-1}_{-\Theta(t)} A(t,Y) := F\left(t, Y(t), \Theta(t)\right). \tag{41}$$

The following formula will be of use in deriving the equation:

$$\begin{aligned} \frac{d}{dt}\exp\left(A(t)\right) &= \operatorname{dexp}_{A(t)}\left(\dot{A}(t)\right)\exp\left(A(t)\right) \\ &= \exp\left(A(t)\right)\operatorname{dexp}_{-A(t)}\left(\dot{A}(t)\right). \end{aligned}$$

As mentioned, (35b) can be integrated just by the RK method, as the exponential map is the identity and dexp^{-1} is also the identity matrix. Hence, the MK method is implemented as follows. Given $T(t_k), V(t_k)$ at time t_k

$$\Theta = \sum_{i=1}^{s} h b_i F^{(i)} \tag{42a}$$

$$T(t_{k+1}) = T(t_k)\exp(\Theta) \tag{42b}$$

$$V(t_{k+1}) = V(t_k) + \sum_{i=1}^{s} h b_i \mathcal{K}^{(i)} \tag{42c}$$

where for $i = 1, 2, \ldots, s$

$$\mathcal{K}^{(i)} = \mathcal{K}\left(t_k + c_i h, T^{(i)}, V^{(i)}\right) \tag{42d}$$

$$V^{(i)} = V(t_k) + \sum_{j=1}^{i-1} h a_{ij} \mathcal{K}^{(j)} \tag{42e}$$

and they are used to compute

$$\Theta^{(i)} = \sum_{j=1}^{i-1} h a_{ij} F^{(j)} \tag{42f}$$

$$T^{(i)} = T(t_k)\exp\left(\Theta^{(i)}\right) \tag{42g}$$

$$F^{(i)} = \operatorname{dexp}^{-1}_{-\Theta^{(i)}} V^{(i)}. \tag{42h}$$

It was mentioned that to save computations of exponentials in the CG methods, the CF method was developed [28]. Commutator evaluations can also be reduced by a clever transformation on $\mathfrak{g}$, making use of the notion of the graded free Lie algebra [29], [30]. Fortunately, when $SE(3)$ is concerned, there is no need to approximate the exponential and the inverse of its differential, since the explicit expressions are available. In particular, *Lemma 1* provides the closed-form expression of $\operatorname{dexp}_V^{-1}$ for $V \in \mathfrak{se}(3)$. Therefore, the issue of computational burden is not of much concern in $SE(3)$.

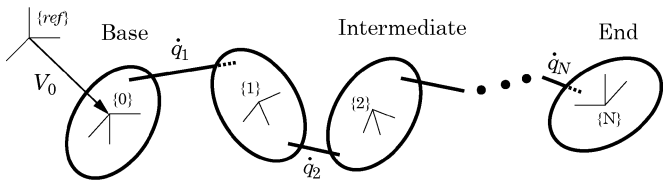


Fig. 2. Chain structure of articulated multiple bodies.

V. Articulated Multibody Systems

A. Equations of Motion

Systems having one branch from the base to the end without any loop are called chain-structured.[7] In Fig. 2, a chain-structured articulated multibody system consisting of K bodies with N joints is shown.

As there are K bodies, we have to consider K body wrenches, denoted by $F_0, F_1, \ldots, F_{K-1}$. For each of N joints, the joint torque is applied, and it is denoted by τ_i for $i = 1, 2, \ldots, N$. Then, we can show that the equations of motion are of the following form [31]:

$$W(T,q)F + L(T,q)\tau = M(T,q)\begin{bmatrix}\dot{V}\\ \ddot{q}\end{bmatrix} + C(T,V,q,\dot{q})\begin{bmatrix}V\\ \dot{q}\end{bmatrix} \tag{43}$$

where $T \in SE(3)$ and $V \in \mathfrak{se}(3)$ are the homogeneous transformation matrix and the body twist of the base body, i.e., body 0, and $\dot{q}_j$ and $\ddot{q}_j$ are time derivatives of q_j, the jth joint value. The vectors in the equation are written as

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \in \mathbb{R}^N,\ \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_N \end{bmatrix} \in \mathbb{R}^N,\ F = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{K-1} \end{bmatrix} \in \mathbb{R}^{6K}.$$

B. Integration Algorithm

The above equations of motion can be written as the differential equations in the following form:

$$\dot{T}(t) = T(t)\lceil V(t) \rceil \tag{44a}$$

$$\dot{V}(t) = \mathcal{K}_V\left(t, T(t), V(t), q(t), \dot{q}(t); F(t), \tau(t)\right) \tag{44b}$$

$$\dot{q}(t) = \dot{q}(t) \tag{44c}$$

$$\ddot{q}(t) = \mathcal{K}_q\left(t, T(t), V(t), q(t), \dot{q}(t); F(t), \tau(t)\right) \tag{44d}$$

[7]General open-loop multibody systems are classified into chain structure and tree structure. In this paper, we only focus on chain-structured open-loop articulated systems. However, the proposed method can be applied to any multibody systems, as long as the equations of motion have the same form as in (43). As a matter of fact, any open-loop multibody system has the equation of motion of that special form.

where

$$\begin{bmatrix}\mathcal{K}_V(\cdot)\\ \mathcal{K}_q(\cdot)\end{bmatrix} = -M^{-1}(T,q)C(T,V,q,\dot{q})\begin{bmatrix}V\\ \dot{q}\end{bmatrix} + M^{-1}(T,q)\left(W(T,q)F(t)+L(T,q)\tau\right). \quad (45)$$

It is not difficult to see that these are the differential equations on the Lie group $SE(3)\times\mathfrak{se}(3)\times\mathbb{R}^N\times\mathbb{R}^N \simeq SE(3)\times\mathbb{R}^6\times\mathbb{R}^N\times\mathbb{R}^N$, consisting of $(T,V,q,\dot{q})$.

1) CG Method: The explicit algorithm is described as follows. Given $T(t_k)$, $V(t_k)$, $q(t_k)$, and $\dot{q}(t_k)$ at time t_k

$$T(t_{k+1}) = T(t_k)\prod_{i=1}^{s}\exp\left(hb_iV^{(i)}\right) \quad (46a)$$

$$q(t_{k+1}) = q(t_k) + \sum_{i=1}^{s} hb_i\dot{q}^{(i)} \quad (46b)$$

$$V(t_{k+1}) = V(t_k) + \sum_{i=1}^{s} hb_i\mathcal{K}_V^{(i)} \quad (46c)$$

$$\dot{q}(t_{k+1}) = \dot{q}(t_k) + \sum_{i=1}^{s} hb_i\mathcal{K}_q^{(i)} \quad (46d)$$

where

$$\mathcal{K}_V^{(i)} = \mathcal{K}_V\left(t_k + c_ih, T^{(i)}, V^{(i)}, q^{(i)}, \dot{q}^{(i)}\right) \quad (46e)$$

$$\mathcal{K}_q^{(i)} = \mathcal{K}_q\left(t_k + c_ih, T^{(i)}, V^{(i)}, q^{(i)}, \dot{q}^{(i)}\right) \quad (46f)$$

$$V^{(i)} = V(t_k) + \sum_{j=1}^{i-1} ha_{ij}\mathcal{K}_V^{(j)} \quad (46g)$$

$$\dot{q}^{(i)} = \dot{q}(t_k) + \sum_{j=1}^{i-1} ha_{ij}\mathcal{K}_q^{(j)} \quad (46h)$$

$$T^{(i)} = T(t_k)\prod_{j=1}^{i-1}\exp\left(ha_{ij}V^{(j)}\right) \quad (46i)$$

$$q^{(i)} = q(t_k) + \sum_{j=1}^{i-1} ha_{ij}\dot{q}^{(j)}. \quad (46j)$$

The third- and fourth-order CG methods can be implemented using the tableaux given in Table I.

2) CF Method: The CF methods of orders 3 and 4 can be similarly implemented using the tableaux in Table II.

3) MK Method: The explicit algorithm is described as follows. Given $T(t_k)$, $V(t_k)$, $q(t_k)$, and $\dot{q}(t_k)$ at time t_k, instead of (46a), use

$$\Theta = \sum_{i=1}^{s} hb_iF^{(i)} \quad (47a)$$

$$T(t_{k+1}) = T(t_k)\exp(\Theta) \quad (47b)$$

and use

$$\Theta^{(i)} = \sum_{j=1}^{i-1} ha_{ij}F^{(j)} \quad (47c)$$

$$T^{(i)} = T(t_k)\exp\left(\Theta^{(i)}\right) \quad (47d)$$

$$F^{(i)} = \mathrm{dexp}^{-1}_{-\Theta^{(i)}}V^{(i)} \quad (47e)$$

instead of (46i).

The third- and fourth-order MK methods can be implemented using the tableaux given in Table III.

TABLE V
COMPLEXITY OF ALGORITHMS IN TERMS OF ELEMENTARY OPERATIONS

Method	FDyn	add[a]	mult	div	sqrt	trig
CG2	2	191/299	270/378	6	3	6
CF2	2	191/299	270/378	6	3	6
MK2	2	201/309	265/373	7	2	4
CG3	3	381/597	540/756	12	6	12
CF3	3	208/352	286/430	6	3	6
MK3	3	334/478	430/574	12	3	6
CG4	5	950/1490	1350/1890	30	15	30
CF4	4	411/771	542/902	10	5	10
MK4	4	493/745	633/885	17	4	8

[a] The numbers in the column add and mult denote the complexity in the case of $N = 2$ and $N = 20$, respectively, where N is the joint DOF.

C. Comparison of Computational Complexity

Computational complexity of each algorithm is summarized in terms of elementary operations in Table V. More detailed analysis is listed in Table X in Appendix I-B, where FDyn stands for the complexity required for the forward dynamics computation, or $\mathcal{K}_q$ and $\mathcal{K}_V$ in (45). As a matter of fact, this complexity is by far heavier than the other. In this sense, CG4 is computationally more demanding than the other methods. The difference is not significant for second-order implementations. For the third-order implementations, CF3 is superior to the other methods. Among the fourth-order implementations, CF4 is also most efficient for lower joint degrees of freedom (DOFs). However, as the joint DOF gets higher, MK4 becomes more efficient than CF4.

There is another point which should be taken into account. In implementing higher order methods of each Lie-group integrator, one has to obtain the suitable coefficient tableaux. The coefficient tableau has been found for the fifth-order CG method requiring nine stages. This is implemented at the cost of 45 evaluations of exp [27]. In practice, coefficients have not been reported for CG methods with higher orders than five. For the CF methods of fifth-order and higher implementations, the coefficient tableaux have not been found [28]. However, the MK methods of higher order can be easily implemented, as we have the coefficient tableaux for higher order RK methods. For example, the fifth-order MK method can be implemented in six stages, requiring six evaluations of exp and dexpinv per step. As a matter of fact, RK methods of 6th order in 7 stages, 7th order in 9 stages, 8th order in 11 stages, and finally, 10th order in 17 stages have been implemented explicitly [5]. Therefore, from the theoretical viewpoint, the higher order MK methods can be implemented using those coefficient tableaux.

VI. NUMERICAL EXAMPLE

Consider the articulated multibody system consisting of three bodies and two revolute joints, shown in Fig. 3. Every body frame and every joint motion are illustrated in the figure. For the sake of convenience, every body is illustrated by spherical body of radius 1 m and mass 10 kg. The generalized coordinates of the system is (T, q_1, q_2), where T is the homogeneous transformation matrix of the base body indexed by 1. Note that body 1 is free to move.

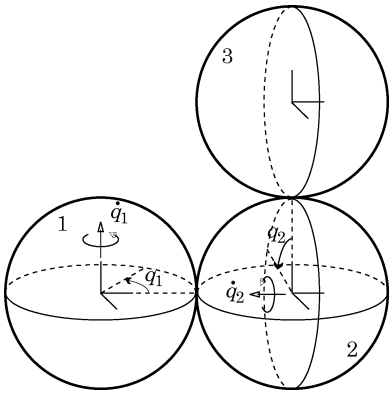


Fig. 3. Three bodies with two revolute joints at $t = 0$.

TABLE VI
ABSOLUTE VALUE OF KINETIC ENERGY DRIFT

Method	$h = 1$	$h = 0.1$	$h = 0.01$	$h = 0.001$
CG2	3.4673E−2	1.1679E−3	1.2151E−5	1.2197E−7
CG3	3.9707E−2	4.2896E−5	4.3020E−8	4.3059E−11
GG4	1.5120E−3	2.0035E−7	2.0421E−11	1.7764E−14
CF2	3.4673E−2	1.1679E−3	1.2151E−5	1.2197E−7
CF3	5.3524E−2	5.0535E−5	5.0334E−8	5.0292E−11
CF4	3.6308E−3	3.2133E−7	3.1989E−11	1.7764E−14
MK2	3.4673E−2	1.1679E−3	1.2151E−5	1.2197E−7
MK3	5.3524E−2	5.0535E−5	5.0334E−8	5.0292E−11
MK4	3.6308E−3	3.2133E−7	3.1996E−11	1.4211E−14

The objective of the simulation is validation of the proposed numerical integration algorithm, though the algorithm is firmly based on theoretical argument. To this end, we conducted the following simulations in order to examine conservation behaviors[8] of integral invariants, i.e., the total kinetic energy and the linear/angular momentum [32]. It assumes zero joint torques and zero body wrenches, i.e., $\tau_j = 0$ and $F_k = 0$, but with nontrivial initial conditions $T(0) = I$, $q(0) = 0$, and nonzero $V(0)$ and $\dot{q}(0)$.

First, we let $V(0) = 0$, $\dot{q}(0) = (0.4, 0.4)^T$ rad/s. Integration is run until $t = 1$ s for each method of orders two, three, and four by reducing the time step, denoted by h, as 1, 0.1, 0.01, and 0.001 s. At time $t = 1$, the norm of kinetic energy drift, the norm of linear momentum drift, and the norm of angular momentum drift were computed and summarized in Tables VI–VIII.

As a matter of fact, the CG and CF methods result in the identical method in the case of second-order implementation. Though the second-order MK method, i.e., MK2, is not identical to the others, almost the same drift behavior has been produced.

This table can give a rough idea of tradeoff between a higher order implementation with sparser time step and a lower order one with a denser time step, in terms of accuracy and complexity. For example, it can be observed from the tables that the drifts of the third-order methods with $h = 0.001$ are almost comparable with those of the fourth-order methods with $h = 0.01$. To compare the actual complexity between two cases, note that the case $h = 0.001$ leads to 1000 steps of integration,

[8]As a matter of fact, the system of our particular concern is the general articulated multibody system, for which neither energy and momentum are preserved. Though the simulations employ the free dynamics, where the energy and momentum are conserved, this is purely for validation of integration methods in an indirect way, because we do not have the exact solution which can be used as the ground truth for comparison.

TABLE VII
NORM OF THE LINEAR MOMENTUM DRIFT

Method	$h = 1$	$h = 0.1$	$h = 0.01$	$h = 0.001$
CG2	5.5613E−1	5.2947E−3	5.2633E−5	5.2600E−7
CG3	5.6523E−2	4.7138E−5	4.6619E−8	4.6510E−11
GG4	8.4583E−3	8.2850E−7	8.3399E−11	3.2901E−13
CF2	5.5613E−1	5.2947E−3	5.2633E−5	5.2600E−7
CF3	5.1921E−2	5.2437E−5	5.2982E−8	5.2981E−11
CF4	1.5953E−2	1.5814E−6	1.5782E−10	5.7011E−14
MK2	5.5613E−1	5.3178E−3	5.2893E−5	5.2864E−7
MK3	5.2334E−2	5.3244E−5	5.3800E−8	5.3802E−11
MK4	1.5910E−2	1.5903E−6	1.5877E−10	9.0256E−14

TABLE VIII
NORM OF THE ANGULAR MOMENTUM DRIFT

Method	$h = 1$	$h = 0.1$	$h = 0.01$	$h = 0.001$
CG2	8.3715E−1	8.1705E−3	8.3278E−5	8.3449E−7
CG3	2.0265E−1	1.9643E−4	1.9513E−7	1.9481E−10
GG4	2.3172E−2	2.6873E−6	2.8057E−10	2.7154E−12
CF2	8.3715E−1	8.1705E−3	8.3278E−5	8.3449E−7
CF3	2.8075E−1	2.8154E−4	2.8084E−7	2.8059E−10
CF4	3.6510E−2	3.6710E−6	3.6619E−10	3.9933E−13
MK2	8.3715E−1	8.1218E−3	8.2721E−5	8.2885E−7
MK3	2.7813E−1	2.8401E−4	2.8359E−7	2.8344E−10
MK4	3.9118E−2	3.8141E−6	3.8025E−10	1.9654E−13

TABLE IX
COMPARISON OF COMPLEXITY BETWEEN 1000 TIMES OF THIRD-ORDER METHODS AND 100 TIMES OF FOURTH-ORDER METHODS

Method	FDyn	add	mult	div	sqrt	trig
CG3 ×1000	3000	381000	540000	12000	6000	12000
CG4 ×100	500	95000	135000	3000	1500	3000
CF3 ×1000	3000	208000	286000	6000	3000	6000
CF4 ×100	400	41100	54200	1000	500	1000
MK3 ×1000	3000	334000	430000	12000	3000	6000
MK4 ×100	400	49300	63300	1700	400	800

while the case $h = 0.01$ leads to 100 steps. Table IX summarizes the actual computational complexity between third-order and fourth-order methods. Actual complexity to get a desired accuracy with lower order implementations becomes by far heavier than that of higher order methods.

To illustrate the behavior of the drift, the plots in Figs. 4–6 compare the drifts in log scale for each order implementation of each method. One can see that the fourth-order behavior of CG4 is not so strictly enforced, compared with CF4 and MK4, since the rate of decrease is reduced as the time step decreases, as shown in Fig. 4. In addition, the CG methods produce more drifts, compared with other methods. This will be more clearly seen when one compares the long-term drift behaviors, as in the next simulation.

Now, we simulated the system with

$$V(0) = (0.1, 0.1, 0.1, 0.3, 0.3, 0.3)^T \text{ m/s or rad/s}$$

and $\dot{q}(0) = (0.4, 0.4)^T$ rad/s. Integration is run until $t = 200$ for fixed time step $h = 0.05$ to examine the long-term drift behavior. We applied the CG, CF, and MK methods of orders three and four.

Fig. 7 summarizes the drift behavior of CG3, CF3, and MK3, while Fig. 8 summarizes CG4, CF4, and MK4. The behavioral difference becomes conspicuous in the fourth-order implementations. In particular, the CG4 method results in much faster drift of energy and momenta, while CF4 and MK4 behave

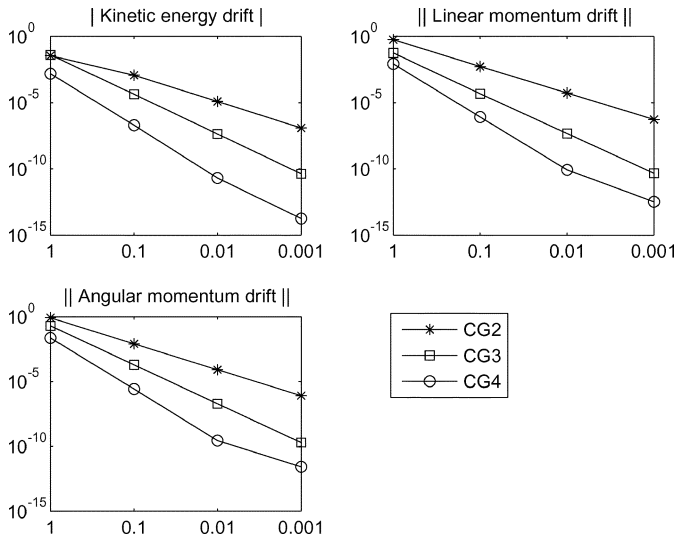


Fig. 4. CG methods. Drift of conserved quantities at $t = 1.0$ s.

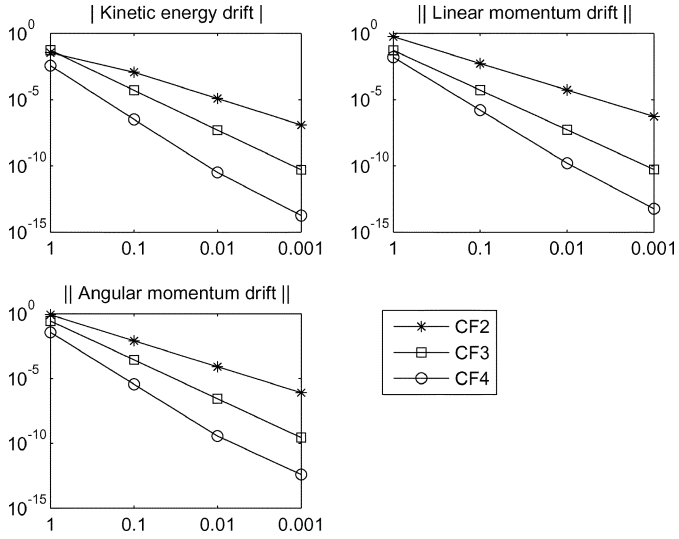


Fig. 5. CF methods. Drift of conserved quantities at $t = 1.0$ s.

almost similarly, though only slight advantage is observed in MK4. This simulation shows that the third- and fourth-order CG methods perform worse than the other two methods. Note that no attempt to stabilize or dampen the numerical integration has been done during simulation. The algorithm was used just as described in this paper.

VII. Concluding Remark

Numerical integration methods based on the Lie-group theoretic geometric approach were applied for dynamical simulation of articulated multibody systems. The proposed Lie-group integrators have the advantage that the approximated solutions always evolve on the underlying group. An additional point of advocating the method is that no additional parametrization of the equations of motion is necessary. Exploiting the structure of the equations of motion of the general articulated multibody systems, we can focus on the Lie-group integrators evolving on a Lie group including the special Euclidean group $SE(3)$.

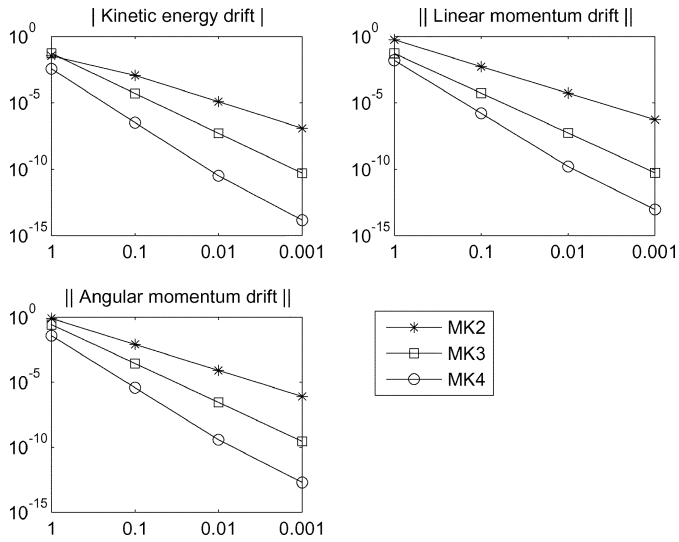


Fig. 6. MK methods. Drift of conserved quantities at $t = 1.0$ s.

In particular, the CG, CF, and MK methods were formulated. The Rodrigues formula presents the closed-form expression of the exponential map on $SE(3)$, eliminating the need of finite term approximation. We also developed the closed-form formulas for the differential of the exponential map and its inverse on $SE(3)$, which are particularly useful in implementing the MK method. Recapitulating, the computational burden required for approximating the exponential and the inverse of its differential has been virtually eliminated in implementing the proposed Lie-group integrators for articulated multibody systems. Through numerical experiments, we verified the performance of the methods, and concluded that the CF or MK methods perform better and are implemented more efficiently than the CG methods, especially in the higher order implementations.

Appendix I
Computational Complexity

A. *Complexity of* $\texttt{exp}$ *and* $\texttt{dexp}^{-1}$ *on* $SE(3)$

Let us consider the additional computational burden to compute the inverse of the differential of the exponential map on $SE(3)$.

First, let us analyze the computational cost for one evaluation of the exponential map on $SE(3)$. For a given $\omega = (\omega_1, \omega_2, \omega_3)^T$ the following six basic terms are computed:

$$\omega_1^2, \quad \omega_1\omega_2, \quad \omega_1\omega_3, \quad \omega_2^2, \quad \omega_2\omega_3, \quad \text{and} \quad \omega_3^3$$

which requires six scalar multiplications. Then, one can compute

$$\|\omega\|^2 \quad \text{and} \quad \|\omega\|$$

at the cost of two scalar additions and one square-root operation. By computing the half of $\|\omega\|$, i.e., $\|\omega\|/2$ with one more scalar multiplication, one can compute the common coefficients s and c in (14) by two trigonometric function evaluations, and one additional scalar division for s. Further, the common terms α and β should be computed at the cost of two scalar multiplications. Then, the exponential map on $SO(3)$ defined by (12) is written

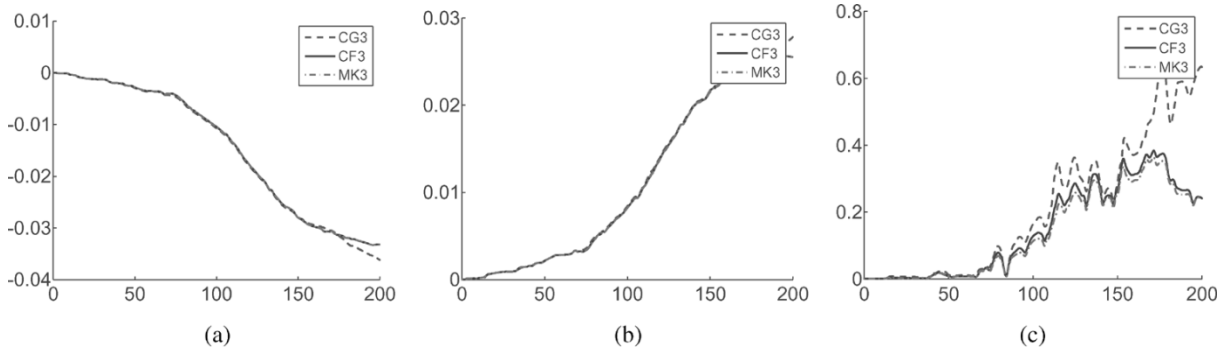


Fig. 7. CG3/CF3/RKMK3 with $h = 0.05$: drift behavior. (a) Kinetic energy variation. (b) Norm of linear momentum drift. (c) Norm of angular momentum drift.

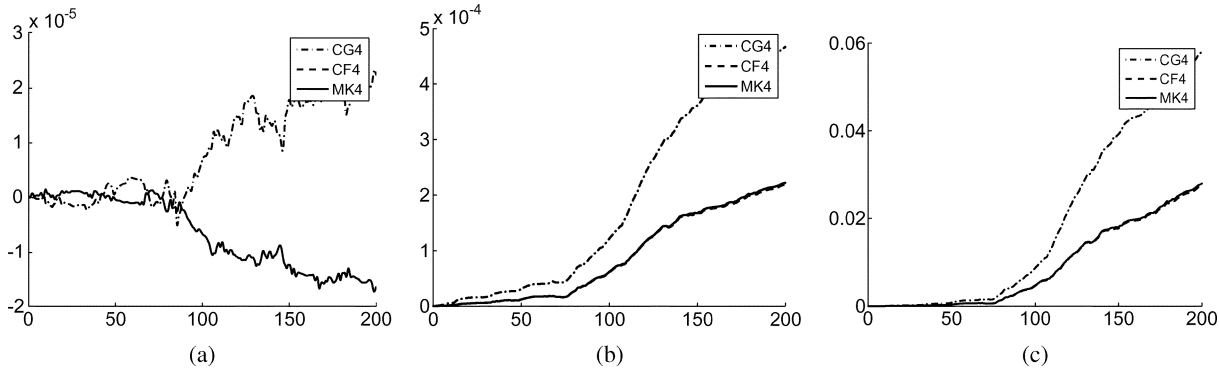


Fig. 8. CG4/CF4/RKMK4 with $h = 0.05$: drift behavior. (a) Kinetic energy variation. (b) Norm of linear momentum drift. (c) Norm of angular momentum drift.

in the form of $I + a\lceil\omega\rceil + b\lceil\omega\rceil^2$, which can be efficiently computed by

$$I + a\lceil\omega\rceil + b\lceil\omega\rceil^2 = \begin{bmatrix} 1 - b\left(\omega_2^2 + \omega_3^2\right) & -a\omega_3 + b\omega_1\omega_2 & a\omega_2 + b\omega_1\omega_3 \\ a\omega_3 + b\omega_1\omega_2 & 1 - b\left(\omega_1^2 + \omega_3^2\right) & -a\omega_1 + b\omega_2\omega_3 \\ -a\omega_2 + b\omega_1\omega_3 & a\omega_1 + b\omega_2\omega_3 & 1 - b\left(\omega_1^2 + \omega_2^2\right) \end{bmatrix}.$$

In total, 9 scalar multiplications (in computing $b\omega_i\omega_j$ and $a\omega_k$) and 12 scalar additions are enough to compute the expression. Therefore, computation of $\mathtt{exp}(\omega)$ needs 9 scalar multiplications and 12 scalar additions, as well as 1 more scalar multiplication for $\beta/2$. The upper off-diagonal block of $\mathtt{exp}(V)$ can be written as

$$\begin{aligned} &-\frac{1}{\|\omega\|^2}\left(\alpha\left(\omega^T v\omega - \|\omega\|^2 v\right) - \frac{\beta}{2}\|\omega\|^2\omega \times v\right) + \frac{1}{\|\omega\|^2}\omega\omega^T v \\ &\quad = -\frac{1}{\|\omega\|^2}\alpha(\omega^T v)\omega + \alpha v + \frac{\beta}{2}\omega \times v + \frac{1}{\|\omega\|^2}(\omega^T v)\omega \\ &\quad = \alpha v + \frac{1-\alpha}{\|\omega\|^2}(\omega^T v)\omega + \frac{\beta}{2}\omega \times v. \end{aligned}$$

The following common terms are computed:

$$\omega_1 v_1, \quad \omega_1 v_2, \quad \omega_1 v_3, \quad \omega_2 v_1, \quad \omega_2 v_2, \quad \omega_2 v_3, \quad \omega_3 v_1, \quad \omega_3 v_2, \quad \text{and} \quad \omega_3 v_3$$

by nine scalar multiplications. Then, $\omega^T v$ and $\omega \times v$ can be computed at the cost of five scalar additions. Then, to compute the off-diagonal term requires one scalar division, one scalar addition, one scalar multiplication (for computing the coefficient of ω), then three `svmult` operation of dim. 3 and two `vadd` operation of dim. 3, where `svmult` denotes the operation of scalar-vector multiplication, and `vadd` the operation of vector addition. In total, one scalar division, seven scalar additions, and ten scalar multiplications are necessary.

Summarizing, one computation of $\exp(V)$ requires 26 scalar additions, 38 scalar multiplications, 2 scalar divisions as well as 1 square-root operation, and 2 trigonometric function evaluations.

To compute $\mathrm{dexp}_\omega^{-1}$ by (19), first the coefficient of $\lceil\omega\rceil^2$ is computed by one scalar addition and one scalar division, as well as one scalar division for the common term γ. Then, since it has the same form as $\exp(\omega)$, it can be computed by 9 scalar multiplications, and 12 scalar additions. The upper off-diagonal block requires computation of D. In order to compute $\lceil v, \omega\rceil$, we use the following explicit expression:

$$\begin{aligned} &\lceil v, \omega\rceil \\ &\quad = v\omega^T + \omega v^T - 2v^T\omega I \\ &\quad = \begin{bmatrix} -2(v_2\omega_2 + v_3\omega_3) & v_1\omega_2 + \omega_1 v_2 & v_1\omega_3 + \omega_1 v_3 \\ v_1\omega_2 + \omega_1 v_2 & -2(v_1\omega_1 + v_3\omega_3) & v_2\omega_3 + \omega_2 v_3 \\ v_1\omega_3 + \omega_1 v_3 & v_2\omega_3 + \omega_2 v_3 & -2(v_1\omega_1 + v_2\omega_2) \end{bmatrix} \end{aligned}$$

which requires six scalar additions as well as three scalar multiplication, owing to the symmetry of the matrix. In order to evaluate the coefficient for $\lceil\omega\rceil^2$, one scalar division and two scalar additions (for $1/\beta + \gamma - 2$), one scalar multiplication (for $1/\|\omega\|^4$), and additional two scalar multiplications are necessary. Since $\lceil\omega\rceil^2 = \omega\omega^T - \omega^T\omega I$

$$\lceil\omega\rceil^2 = \begin{bmatrix} -\left(\omega_2^2 + \omega_3^2\right) & \omega_1\omega_2 & \omega_1\omega_3 \\ \omega_1\omega_2 & -\left(\omega_1^2 + \omega_3^2\right) & \omega_2\omega_3 \\ \omega_1\omega_3 & \omega_2\omega_3 & -\left(\omega_1^2 + \omega_2^2\right) \end{bmatrix}$$

TABLE X
COMPLEXITY OF EACH ALGORITHM

Method	FDyn	svmultadd(6)	svmultadd(N)	svmult(6)	exp	hmmult	add	dexpinv	mvmult(6×6)
CG2	2	3	6	3	3	3	2	0	0
CF2	2	3	6	3	3	3	2	0	0
MK2	2	3	6	3	2	2	2	1	1
CG3	3	6	12	6	6	6	3	0	0
CF3	3	5	8	3	3	3	3	0	0
MK3	3	5	8	4	3	3	3	2	2
CG4	5	15	30	15	15	15	5	0	0
CF4	4	17	20	5	5	5	4	0	0
MK4	4	10	14	7	4	4	4	3	3

and only three scalar multiplications are enough to compute it. Then, D can be computed at the cost of two smmult and one madd of dim. 3×3, where smmult denotes the operation of scalar-matrix multiplication and madd the operation of matrix addition. Taking into account the symmetry, only 12 scalar multiplications and 6 additions are enough. Then, the off-diagonal block can be computed by an additional six scalar additions with three scalar multiplications for $-\lceil v/2 \rceil + D$.

The additional computation necessary to compute $\mathrm{dexp}_{\omega}^{-1}$ is summarized to be 33 scalar additions, the same number of multiplications, and 3 scalar divisions.

B. Comparison of Complexity Lie Group Methods

See Table X. In particular, FDyn stands for the complexity to compute the forward dynamics. $\mathtt{svmult}(n)$ and $\mathtt{svmultadd}(n)$ denote the operations of scalar-vector multiplication and scalar-vector multiplication with vector addition of dim n. hmmult is the homogeneous matrix multiplication, and $\mathtt{mvmult}(m \times n)$ is the matrix-vector multiplication of dimension $m \times n$.

REFERENCES

[1] R. Featherstone, *Robot Dynamics Algorithms*. Norwell, MA: Kluwer, 1987.

[2] K. W. Lilly, *Efficient Dynamic Simulation of Robotic Mechanisms*. Norwell, MA: Kluwer, 1993.

[3] D. S. Bae and E. J. Haug, "A recursive formulation for constrained mechanical system dynamics: Part 1, Open-loop systems," *Mechan. Struct. Mach.*, vol. 15, no. 3, pp. 359–382, 1987.

[4] ——, "A recursive formulation for constrained mechanical system dynamics: Part 2, Closed-loop systems," *Mechan. Struct. Mach.*, vol. 15, no. 4, pp. 481–506, 1987.

[5] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, New York: Springer-Verlag, 1993.

[6] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC, 1994.

[7] V. S. Varadarajan, *Lie Groups, Lie Algebras, and Their Representations*, New York: Springer-Verlag, 1984.

[8] E. J. Haug, *Intermediate Dynamics*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[9] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. New York: Springer, 2002.

[10] P. E. Crouch and R. Grossman, "Numerical integration of ordinary differential equations on manifolds," *J. Nonlinear Sci.*, vol. 3, pp. 1–33, 1993.

[11] H. Munthe–Kaas, "High order Runge–Kutta methods on manifolds," *Appl. Numer. Math.*, vol. 29, no. 1, pp. 115–127, 1999.

[12] A. Iserles, H. Z. Munthe–Kaas, S. P. Norsett, and A. Zanna, "Lie-group methods," *Acta Numerica*, pp. 215–365, 2000.

[13] E. Celledoni and B. Owren, "Lie group methods for rigid body dynamics and time integration on manifolds," *Computer Methods Appl. Mechan. Eng.*, vol. 192, pp. 421–438, 2003.

[14] J. C. Simo and K. K. Wong, "Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum," *Int. J. Numerical Methods Eng.*, vol. 31, pp. 19–52, 1991.

[15] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, pp. 357–514, 2001.

[16] L. O. Jay, "Structure preservation for constrained dynamics with super partitioned additive Runge–Kutta methods," *SIAM J. Sci. Comput.*, vol. 20, no. 2, pp. 416–446, 1998.

[17] J. C. Simo, N. Tarnow, and K. K. Wong, "Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics," *Comput. Methods Appl. Mechan. Eng.*, vol. 100, pp. 63–116, 1992.

[18] D. Lewis and J. C. Simo, "Conserving algorithms for the dynamics of Hamiltonian systems on Lie groups," *J. Nonlinear Sci.*, vol. 4, pp. 253–299, 1994.

[19] K. Engø and A. Marthinsen, "Modeling and solution of some mechanical problems on lie groups," *Multibody Syst. Dyn.*, vol. 2, pp. 71–88, 1998.

[20] S. R. Buss, "Accurate and efficient simulation of rigid-body rotations," *J. Computat. Phys.*, vol. 164, pp. 377–406, 2000.

[21] A. Zanna, K. Engø, and H. Z. Munthe–Kaas, "Adjoint and self adjoint Lie-group methods," *BIT Numer. Math.*, vol. 41, no. 2, pp. 395–421, 2001.

[22] B. Owren and A. Marthinsen, "The Newton iteration on Lie groups," *BIT Numer. Math.*, vol. 40, no. 1, pp. 121–145, 2000.

[23] H. Rehbinder and B. K. Ghosh, "Pose estimation using line-based dynamic vision and inertial sensors," *IEEE Trans. Autom. Control*, vol. 48, no. 2, pp. 186–199, Feb. 2003.

[24] S. Gwak, J. Kim, and F. C. Park, "Numerical optimization on the Euclidean group with applications to camera calibration," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 65–74, Feb. 2003.

[25] P. J. Olver, *Applications of Lie Groups to Differential Equations*, 2nd ed. New York: Springer-Verlag, 1993.

[26] B. Owren and A. Marthinsen, "Runge–Kutta methods adapted to manifolds and based on rigid frames," *BIT Numer. Math.*, vol. 39, pp. 116–142, 1999.

[27] Z. Jackiewicz, A. Marthinsen, and B. Owren, "Construction of Runge–Kutta methods of Crouch–Grossman type of high order," *Adv. Computat. Math.*, vol. 13, no. 4, pp. 405–415, 2000.

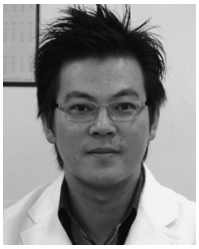
[28] E. Celledoni, A. Marthinsen, and B. Owren. (2002) Commutator-free Lie group methods. Norwegian Univ. Sci. Technol., Trondheim, Norway. [Online]. Available: http://www.math.ntnu.no/num/synode/papers, Tech. Rep. 1/02

[29] H. Munthe–Kaas and B. Owren. (1999) Computations in a free Lie algebra. Dept. Informatics, Univ. Bergen, Bergen, Norway. [Online]. Available: http://www.ii.uib.no/~hans/papers, Tech. Rep. 148

[30] F. Casas and B. Owren, "Cost efficient Lie group integrators in the RKMK class," *BIT Numer. Math.*, vol. 43, pp. 723–742, 2003.

[31] J. Park. Principle of dynamical balance for multibody systems. *Multibody Syst. Dyn.* [Online]. Available: http://rnb.postech.ac.kr/publication

[32] D. T. Greenwood, *Principles of Dynamics*. Englewood Cliffs, NJ: Prentice-Hall, 1988.



Jonghoon Park (M'01) received the B.S., M.S., and Ph.D. degrees from the Department of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 1992, 1994, and 1999, respectively.

He was then a Postdoctoral Researcher with the Robotics Lab., and ARC (Automation Research Center), POSTECH, until January 2000. From then until June 2001, he was with Hiroshima University, Hiroshima, Japan, as a Visiting Researcher funded by the Japan Society of Promotion of Science (JSPS). He was a Research Assistant Professor with the Department of Mechanical Engineering, POSTECH, from 2001 until October 2003. Since then, he has been a Senior Researcher there. His research interests include manipulation analysis and synthesis, especially for enveloping grasp system, simulation of multi-rigid-body dynamical system in frictional contact, nonlinear control techniques for Euler–Lagrange system using the nonlinear H-infinity optimal control technique, and kinematic/dynamic analysis and synthesis of control for general multibody systems having redundancy, such as humanoid.

Wan-Kyun Chung (M'86) received the B.S. degree in mechanical design from Seoul National University, Seoul, Korea, in 1981, and the M.S. degree in mechanical engineering and the Ph.D. degree in production engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1983 and 1987, respectively.

He is currently a Professor with the School of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea, where he joined the faculty in 1987. In 1988, he was a Visiting Professor at the Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA. In 1995, he was a Visiting Scholar at the University of California, Berkeley. His research interests include the localization and navigation for mobile robots, underwater robots, and the development of a robust controller for precision motion control. He is also a Director of the National Research Laboratory for Intelligent Mobile Robot Navigation.