



Evaluation and implementation of Lie group integration methods for rigid multibody systems

Stefan Holzinger¹ · Martin Arnold² · Johannes Gerstmayr¹

Received: 20 March 2023 / Accepted: 30 January 2024
© The Author(s) 2024

Abstract

As commonly known, standard time integration of the kinematic equations of rigid bodies modeled with three rotation parameters is infeasible due to singular points. Common workarounds are reparameterization strategies or Euler parameters. Both approaches typically vary in accuracy depending on the choice of rotation parameters. To efficiently compute different kinds of multibody systems, one aims at simulation results and performance that are independent of the type of rotation parameters. As a clear advantage, Lie group integration methods are rotation parameter independent. However, few studies have addressed whether Lie group integration methods are more accurate and efficient compared to conventional formulations based on Euler parameters or Euler angles. In this paper, we close this gap using the $\mathbb{R}^3 \times SO(3)$ Lie group formulation and several typical rigid multibody systems. It is shown that explicit Lie group integration methods outperform the conventional formulations in terms of accuracy. However, it turns out that the conventional Euler parameter-based formulation is the most accurate one in the case of implicit integration, while the Lie group integration method is computationally the more efficient one. It also turns out that Lie group integration methods can be implemented at almost no extra cost in an existing multibody simulation code if the Lie group method used to describe the configuration of a body is chosen accordingly.

Keywords Multibody systems · Rigid body · Lie group · Numerical integration · Rotation parameters · Singularities · Absolute coordinates · Exudyn

✉ J. Gerstmayr
johannes.gerstmayr@uibk.ac.at

S. Holzinger
stefan.holzinger@uibk.ac.at

M. Arnold
martin.arnold@mathematik.uni-halle.de

¹ Department of Mechatronics, University of Innsbruck, Technikerstrasse 13, Innsbruck, 6020, Tyrol, Austria

² Institute of Mathematics, Martin Luther University Halle-Wittenberg, Theodor-Lieser-Str. 5, Halle (Saale), 06120, Saxony-Anhalt, Germany

1 Introduction

Rigid bodies are commonly used to model complex multibody systems (MBSs). Since rigid bodies have six degrees of freedom, it is obvious to use three parameters to model translations and three parameters to model rotations. As commonly known, modeling spatial rotations with three rotation parameters is problematic due to singular points [1–3]. Common workarounds are reparameterization strategies [4, 5] or the use of four Euler parameters optionally with unorthodox normalization or in combination with differential-algebraic equations [6]. The latter approaches are often referred to as classical parameterization-based formulations [2, 4] and typically vary in accuracy depending on the choice of rotation parameters. To be able to efficiently compute a variety of different MBSs, one aims at simulation results and performance that are independent of the choice of rotation parameters. Time integration algorithms based on Lie group methods, referred to in the following as Lie group integration methods (LGIMs), are rotation parameter independent. Moreover, if the Lie group method used to describe the configuration of a body is chosen accordingly, only a few adaptations are needed in classical integration algorithms to convert them into LGIMs, which motivates the present work and will be illustrated in this paper.

Let us briefly review LGIMs. LGIMs have now been developed and successfully applied for the simulation of MBSs for more than 30 years; see, e.g., Refs. [7, 8]. They not only enable a singularity-free representation of spatial rotations [9], they also allow the use of three rotation parameters to model MBSs [10, 11] and even make the unit length constraint equation obsolete when integrating Euler parameters [3, 12]. Moreover, LGIMs allow the modeling of MBSs with local velocity coordinates [13] and the so-called “local frame approach” [14]. Another advantage of LGIMs is that no parameterization is required for the formulation of the EOMs (see, e.g., [2]), and thus parameterization effects can be avoided.

In the case of rigid body systems, the EOMs are typically formulated on either a Lie group defined as multiples of the direct product (\times) or the semi-direct product (\rtimes) of the group of translations \mathbb{R}^3 and the group of special orthogonal transformations $SO(3)$, i.e., $\mathbb{R}^3 \times SO(3)$, $\mathbb{R}^3 \rtimes SO(3) =: SE(3)$, or the group of Euler parameters (unit quaternions) $S^3 \subseteq \mathbb{R}^4$, i.e., $\mathbb{R}^3 \times S^3$, $\mathbb{R}^3 \rtimes S^3$; see, e.g., [8, 15]. For simulating rigid bodies, all of the latter formulations can be used. It is reported in [16] that for a general MBS, the $\mathbb{R}^3 \times SO(3)$ formulation yields the same accuracy as the $SE(3)$ formulation and in [17] that the $\mathbb{R}^3 \times SO(3)$ formulation should be used as the $SE(3)$ formulation is numerically more complex than the $\mathbb{R}^3 \rtimes SO(3)$ formulation. However, formulations based on the semi-direct product allow to represent rigid body motions and are also beneficial for simulating flexible MBSs; see, e.g., [14, 18]. For both formulations, time integration schemes are reported in the literature. Runge–Kutta (RK) methods for Lie groups have been introduced by Munthe-Kaas in [19] and [20]. Multistep methods of BDF type have been extended to Lie groups in [21, 22]. A RATTLE inspired integration scheme for Lie groups was proposed in [8], and in [1, 9, 12, 23, 24] the generalized- α methods [25, 26] were extended to Lie groups.

LGIMs are conceptually coordinate-free, which makes them difficult to be incorporated into existing multibody simulation packages that are based on “absolute coordinates,” i.e., a set of coordinates that describe the absolute position and orientation of the individual bodies with respect to an inertial frame [6]. Recently, an interface technique has been proposed [6, 27] that allows MBSs to be modeled with absolute coordinates while using Lie group time integration methods. However, the latter studies did not deal with computational aspects such as the practical implementation of the technique and computational performance.

It is surprising that despite their benefits and long history in multibody dynamics (MBD), few studies have addressed whether LGIMs are more accurate and computationally efficient

Table 1 Rigid multibody examples. “perf.” stands for performance and “accu.” stands for accuracy

Nr.	Example	Expl. Lie	Impl. Lie	Comparison with non-Lie
1	Free rotating rigid body [1, 3, 10, 11]	✓	✓	perf.: – accu.: [3, 10, 11]
2	Heavy top with kinematic constraints [1, 2, 9, 17, 22, 23, 28–30]	✓	✓	perf.: [30] accu.: [1]
3	Heavy top without kinematic constraints [1, 3, 10, 11, 29]	✓	✓	perf.: – accu.: [1, 3, 10, 11]
4	Rotating body with follower torque [1]	–	✓	perf.: – accu.: [1]
5	Slider–crank mechanism (3D) [15]	–	✓	perf.: – accu.: –
6	Spherical double pendulum [17, 29]	✓	–	perf.: – accu.: –
7	Flyball governor with compliant joints [10]	✓	–	perf.: – accu.: [10]
8	Spacecraft with reaction wheels [3]	✓	–	perf.: [3] accu.: [3]
9	Orthogonal Bricard mechanism [30]	✓	–	perf. [30] accu.: –
10	Two quadrotors transporting a mass point [29]	✓	–	perf.: – accu.: [29]
11	Satellite with 5-DOF manipulator [28]	✓	–	perf.: – accu.: –
12	Interconnected floating bodies [17]	✓	–	perf.: – accu.: –
13	Closed loop spherical 3-bar linkage [17]	✓	–	perf.: – accu.: –
14	Dual-spin satellite [31]	✓	–	perf.: – accu.: –
15	Three bodies with two revolute joints [32]	✓	–	perf.: – accu.: –

compared to classical parameterization-based formulations, even though the answer could be crucial for the decision to equip a multibody simulation package with LGIMs. Table 1 lists numerical examples used in past work on LGIMs for rigid body systems and the comparisons made concerning accuracy and computational performance. As can be seen in Table 1, up to now, LGIMs have mostly been compared with other LGIMs, but barely with classical parameterization-based formulations based on Euler parameters or Euler angles, especially in combination with implicit integration.

In this paper, we compare the accuracy and computational performance of explicit and implicit LGIMs with classical parameterization-based formulations based on Euler angles and Euler parameters using a set of rigid body systems. We also show the transition from absolute coordinate-based integration methods to LGIMs, with specific focus on the implications for algorithms and implementation. The implementation is shown for formalisms that employ explicit RK methods and the implicit generalized- α method [25, 26]. In the present approach, both classical parameterization-based formulations and LGIMs can be mixed, which is advantageous for general purpose simulation codes. For the evaluation of

the accuracy and computational performance of LGIMs, we restrict our investigations to the $\mathbb{R}^3 \times SO(3)$ formulation. The evaluation of the accuracy and computational performance of $SE(3)$ formulations is left for future work.

The remaining part of the paper is organized as follows: In Sect. 2 the EOMs of two classical parameterization-based formulations are recapitulated and their relation to the EOMs on the Lie group $\mathbb{R}^3 \times SO(3)$ is explained. Subsequently, we show in Sect. 3 how MBS formalisms that employ explicit RK methods as well as the generalized- α method [26] have to be modified to become Lie group methods, i.e., Runge–Kutta–Munthe-Kaas (RKMK) methods [19, 20] or Lie group generalized- α [1]. Then, in Sect. 4, the accuracy and computational performance of an LGIM are evaluated using five examples of rigid body systems. Lastly, we draw conclusions from the study in Sect. 5.

2 Equations of motion of rigid body systems

In this section we recapitulate two classical parameterization-based formulations of the EOMs of rigid body systems and show how these can be turned into EOMs on the Lie group $\mathbb{R}^3 \times SO(3)$. The section is divided into two parts. In the first part the classical formulations are addressed, and in the second part the transition to the Lie group formulation is addressed.

2.1 Classical parameterization-based formulations

The EOMs of an MBS with N rigid bodies can be expressed in the general form

$$\mathbf{M}_\bullet(\mathbf{q}_\bullet)\dot{\mathbf{v}}_\bullet = \mathbf{g}_\bullet(\mathbf{q}_\bullet, \mathbf{v}_\bullet, t) - \mathbf{B}_\bullet^T(\mathbf{q}_\bullet)\boldsymbol{\lambda}, \quad (1)$$

$$\mathbf{v}_\bullet = \mathbf{H}_\bullet(\mathbf{q}_\bullet)\dot{\mathbf{q}}_\bullet, \quad (2)$$

$$\Phi_\bullet(\mathbf{q}_\bullet) = \mathbf{0}, \quad (3)$$

where Eq. (1) represents k equilibrium equations,¹ Eq. (2) k kinematic equations and Eq. (3) j linearly independent holonomic constraints.² Note that the quantities in Eqs. (1)–(3), marked with the symbol \bullet , are different in each of the two classical formulations but have the same meaning and are therefore only placeholders at this point. In the following, the general meaning of the quantities in Eqs. (1)–(3) is first described and then explicitly given for the respective formulation. The symbols assigned to the respective formulation are given in Table 2. In this paper, the time derivative of a quantity \mathbf{c} is denoted by $\dot{\mathbf{c}}$.

The system states are represented by

$$\mathbf{q}_\bullet = \left[(\mathbf{q}_\bullet^{(1)})^T, \dots, (\mathbf{q}_\bullet^{(N)})^T \right]^T, \quad \mathbf{v}_\bullet = \left[(\mathbf{v}_\bullet^{(1)})^T, \dots, (\mathbf{v}_\bullet^{(N)})^T \right]^T, \quad (4)$$

and the states for the i -th rigid body by

$$\mathbf{q}_\bullet^{(i)} = \begin{bmatrix} \mathbf{x}_\bullet^{(i)} \\ \boldsymbol{\rho}_\bullet^{(i)} \end{bmatrix}, \quad \mathbf{v}_\bullet^{(i)} = \begin{bmatrix} \mathbf{v}_{\mathbf{x}_\bullet}^{(i)} \\ \mathbf{v}_{\boldsymbol{\rho}_\bullet}^{(i)} \end{bmatrix}. \quad (5)$$

¹For the classical formulation 1 we have $k = 6N$ and for the classical formulation 2 we have $k = (3 + l)N$, where l is the number of rotation parameters used to describe the orientation of the individual rigid bodies in the system. In the case of Euler angles or the rotation vector, $l = 3$, and in the case of Euler parameters, $l = 4$.

²Note that the constraints (3) can also be formulated in terms of \mathbf{v}_\bullet or $\dot{\mathbf{q}}_\bullet$.

Table 2 Assignment list of the placeholder quantities and the quantities in the respective classical formulation

Quantity	q_\bullet	x_\bullet	ρ_\bullet	v_\bullet	$v_{x,\bullet}$	$v_{\rho,\bullet}$	G_\bullet	H_\bullet	M_\bullet	Φ_\bullet	B_\bullet	$g_{\bullet,gyr}$
Formulation 1	\mathbf{q}	\mathbf{x}	ρ	\mathbf{v}	\mathbf{v}_x	\mathbf{v}_ρ	\mathbf{G}	\mathbf{H}	\mathbf{M}	Φ	\mathbf{B}	\mathbf{g}_{gyr}
Formulation 2	$\widehat{\mathbf{q}}$	$\widehat{\mathbf{x}}$	$\widehat{\rho}$	$\widehat{\mathbf{v}}$	$\widehat{\mathbf{v}}_x$	$\widehat{\mathbf{v}}_\rho$	$\widehat{\mathbf{G}}$	$\widehat{\mathbf{H}}$	$\widehat{\mathbf{M}}$	$\widehat{\Phi}$	$\widehat{\mathbf{B}}$	$\widehat{\mathbf{g}}_{gyr}$

In Eq. (5), $\mathbf{x}_\bullet^{(i)} \in \mathbb{R}^3$ represents the i -th rigid body’s position relative to an inertial frame, $\rho_\bullet^{(i)} \in \mathbb{R}^l$ the body’s l rotation parameters and $\mathbf{v}_{x,\bullet}^{(i)} \in \mathbb{R}^3$ and $\mathbf{v}_{\rho,\bullet}^{(i)} \in \mathbb{R}^l$ its translational and rotational velocities. The block diagonal matrix $\mathbf{H}_\bullet(\mathbf{q}_\bullet) \in \mathbb{R}^{6N \times (3+l)N}$

$$\mathbf{H}_\bullet(\mathbf{q}_\bullet) = \text{blockdiag}_{1 \leq i \leq N} (\mathbf{H}_\bullet^{(i)}(\mathbf{q}_\bullet^{(i)})) \tag{6}$$

relates $\dot{\mathbf{q}}_\bullet$ to \mathbf{v}_\bullet , where

$$\mathbf{H}_\bullet^{(i)}(\mathbf{q}_\bullet^{(i)}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_\bullet(\rho_\bullet^{(i)}) \end{bmatrix} \in \mathbb{R}^{6 \times (3+l)}. \tag{7}$$

In Eq. (7), matrix $\mathbf{G}_\bullet(\rho_\bullet^{(i)}) \in \mathbb{R}^{3 \times l}$ relates $\dot{\rho}_\bullet^{(i)}$ to $\mathbf{v}_{\rho,\bullet}^{(i)}$ and the matrices \mathbf{I} and $\mathbf{0}$ represent identity and null matrices of proper size. The mass matrix $\mathbf{M}_\bullet(\mathbf{q}_\bullet) \in \mathbb{R}^{k \times k}$

$$\mathbf{M}_\bullet(\mathbf{q}_\bullet) = \text{blockdiag}_{1 \leq i \leq N} (\mathbf{M}_\bullet^{(i)}(\mathbf{q}_\bullet^{(i)})) \tag{8}$$

contains in its block diagonal the mass matrices of the N rigid bodies, is symmetric, not necessarily constant, and may depend on \mathbf{q}_\bullet . The vector $\mathbf{g}_\bullet \in \mathbb{R}^k$ in Eq. (1) collects for each rigid body the applied forces³ \mathbf{g}_a and gyroscopic forces $\mathbf{g}_{\bullet,gyr}$

$$\mathbf{g}_\bullet = \left[(\mathbf{g}_\bullet^{(1)})^T, \dots, (\mathbf{g}_\bullet^{(N)})^T \right]^T, \quad \mathbf{g}_\bullet^{(i)} = \mathbf{g}_a^{(i)} + \mathbf{g}_{\bullet,gyr}^{(i)}, \quad \mathbf{g}_a^{(i)} = \begin{bmatrix} \mathbf{f}^{(i)} \\ \mathbf{t}^{(i)} \end{bmatrix}. \tag{9}$$

In Eq. (9), the vector $\mathbf{f}^{(i)} \in \mathbb{R}^3$ represents external forces expressed in the inertial frame and the vector $\mathbf{t}^{(i)} \in \mathbb{R}^3$ represents external torques expressed in the body-attached frame [4]. Note that both $\mathbf{f}^{(i)}$ and $\mathbf{t}^{(i)}$ result, for example, from gravity or from springs and dampers. The holonomic constraints (3) are coupled to the equilibrium equations by constraint forces $-\mathbf{B}_\bullet^T(\mathbf{q}_\bullet)\lambda$. The vector of Lagrange multipliers is denoted by $\lambda \in \mathbb{R}^j$ and is multiplied by the transpose of the constraint matrix $\mathbf{B} \in \mathbb{R}^{j \times k}$ that represents the constraint gradients

$$\mathbf{B}_\bullet(\mathbf{q}_\bullet) = \frac{\partial \dot{\Phi}(\mathbf{q}_\bullet)}{\partial \mathbf{v}_\bullet}. \tag{10}$$

2.1.1 Classical formulation 1

In the classical formulation 1, the velocity coordinates of Eq. (5), the mass matrix and the gyroscopic forces for the i -th rigid body read

$$\mathbf{v}_x^{(i)} = \dot{\mathbf{x}}^{(i)}, \quad \mathbf{v}_\rho^{(i)} = \boldsymbol{\omega}^{(i)}, \quad \mathbf{M}^{(i)} = \begin{bmatrix} m^{(i)}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}^{(i)} \end{bmatrix}, \quad \mathbf{g}_{gyr}^{(i)} = \begin{bmatrix} \mathbf{0} \\ \widetilde{\boldsymbol{\omega}}^{(i)}\mathbf{J}^{(i)}\boldsymbol{\omega}^{(i)} \end{bmatrix}. \tag{11}$$

³The vector of the applied forces \mathbf{g}_a respectively $\mathbf{g}_a^{(i)}$ is identical in both formulations. Therefore, they are not marked with the placeholder symbol \bullet .

In Eq. (11), $\boldsymbol{\omega}^{(i)}, \mathbf{v}_\rho^{(i)} \in \mathbb{R}^3$ represents the angular velocity vector expressed in the body-attached frame, $m^{(i)}$ is the mass of the rigid body, $\mathbf{J}^{(i)} \in \mathbb{R}^{3 \times 3}$ is the inertia tensor expressed in the body-attached frame and $\tilde{\boldsymbol{\omega}}^{(i)}$ is the skew-symmetric matrix such that $\boldsymbol{\omega} \times \mathbf{y} = \tilde{\boldsymbol{\omega}} \mathbf{y}$ for $\boldsymbol{\omega}, \mathbf{y} \in \mathbb{R}^3$.

2.1.2 Classical formulation 2

In the classical formulation 2, Eqs. (1)–(2) are reformulated into k second-order differential equations.⁴ To this end, Eq. (2) is inserted into Eq. (1) and Eq. (1) is then multiplied with $\widehat{\mathbf{H}}^T(\hat{\mathbf{q}})$ from the left hand side, which yields

$$\begin{aligned} \widehat{\mathbf{M}}(\hat{\mathbf{q}})\ddot{\hat{\mathbf{q}}} &= \widehat{\mathbf{H}}^T(\hat{\mathbf{q}}) (\widehat{\mathbf{g}}(\hat{\mathbf{q}}, \dot{\hat{\mathbf{q}}}, t) - \widehat{\mathbf{B}}^T(\hat{\mathbf{q}})\boldsymbol{\lambda}), \\ \widehat{\boldsymbol{\Phi}}(\hat{\mathbf{q}}) &= \mathbf{0}. \end{aligned} \quad (12)$$

For the i -th rigid body, the velocity coordinates Eq. (5), the mass matrix and gyroscopic forces read⁵

$$\widehat{\mathbf{v}}_{\mathbf{x}}^{(i)} = \dot{\mathbf{x}}^{(i)}, \quad \widehat{\mathbf{v}}_\rho^{(i)} = \widehat{\mathbf{G}}(\boldsymbol{\rho}^{(i)})\dot{\boldsymbol{\rho}}^{(i)}, \quad (13)$$

$$\widehat{\mathbf{M}}^{(i)} = \begin{bmatrix} m^{(i)}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widehat{\mathbf{G}}^{(i),T}\mathbf{J}^{(i)}\widehat{\mathbf{G}}^{(i)} \end{bmatrix}, \quad \widehat{\mathbf{g}}_{\text{gyr}}^{(i)} = \begin{bmatrix} \mathbf{0} \\ \left(\mathbf{J}^{(i)}\widehat{\mathbf{G}}^{(i)} + \tilde{\boldsymbol{\omega}}^{(i)}\mathbf{J}^{(i)}\widehat{\mathbf{G}}^{(i)} \right) \dot{\boldsymbol{\rho}}^{(i)} \end{bmatrix}. \quad (14)$$

2.2 Lie group formulation

Before we illustrate the transition step from the classical parameterization-based formulations of the EOMs into a Lie group formulation, we first need to introduce the kinematic description for rigid bodies on $\mathbb{R}^3 \times SO(3)$.

2.2.1 Rigid body kinematics on $\mathbb{R}^3 \times SO(3)$

On $\mathbb{R}^3 \times SO(3)$, the configuration of a rigid body is denoted by the pair

$$q = (\mathbf{x}, \mathbf{R}), \quad (15)$$

where the rigid body's position relative to an inertial frame is represented by $\mathbf{x} \in \mathbb{R}^3$ and its orientation by the rotation matrix $\mathbf{R} \in SO(3)$

$$SO(3) := \{\mathbf{R} \in \mathbb{R}^{3 \times 3} \mid \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = +1\}. \quad (16)$$

The Lie group $G := \mathbb{R}^3 \times SO(3)$ is defined as the direct product of the linear space \mathbb{R}^3 with the Lie group $SO(3)$ and composition operation \circ ,

$$q_a \circ q_b = (\mathbf{x}_a, \mathbf{R}_a) \circ (\mathbf{x}_b, \mathbf{R}_b) = (\mathbf{x}_a + \mathbf{x}_b, \mathbf{R}_a\mathbf{R}_b) \in G, \quad (17)$$

⁴Note that certain time integration methods such as Newmark-type methods, for example, require that $\mathbf{v}_\bullet = \dot{\mathbf{q}}_\bullet$ in Eqs. (1)–(3).

⁵Note that formulation 1 is closer to the Lie group formulation than formulation 2 but requires that $\mathbf{v}_\rho^{(i)} \neq \dot{\boldsymbol{\rho}}^{(i)}$. However, explicit time integration methods can directly be applied to formulation 1, whereas implicit time integration methods are only directly applicable when $\dim(\dot{\boldsymbol{\rho}}^{(i)}) = \dim(\mathbf{v}_\rho^{(i)})$.

which results in the kinematic equations [2, 9]

$$\dot{\mathbf{x}} = \mathbf{u} \quad \text{and} \quad \dot{\mathbf{R}} = \mathbf{R}\tilde{\boldsymbol{\omega}}. \tag{18}$$

In Eq. (18), $\mathbf{u} \in \mathbb{R}^3$ represents the translational velocity in the inertial frame. The kinematic equations (18) can be written in the compact form [2, 9]

$$\dot{\mathbf{Q}} = \mathbf{Q}\tilde{\mathbf{v}} \tag{19}$$

if the pair (\mathbf{x}, \mathbf{R}) in Eq. (15) is represented by a 7×7 matrix \mathbf{Q} ,

$$\mathbf{Q} := \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{x} \\ \mathbf{0}_{1 \times 3} & \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \in G, \quad \tilde{\mathbf{v}} := \begin{bmatrix} \tilde{\boldsymbol{\omega}} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{u} \\ \mathbf{0}_{1 \times 3} & \mathbf{0}_{1 \times 3} & 0 \end{bmatrix}. \tag{20}$$

For an efficient implementation, we do not work with the rotation matrices \mathbf{R} directly but with an isomorphic representation in terms of rotation parameters⁶ $\boldsymbol{\rho} \in \mathbb{R}^l$. To this end, we express Eq. (15) in terms of rotation parameters $\boldsymbol{\rho}$,

$$q = (\mathbf{x}, \boldsymbol{\rho}) \cong \mathbf{q}^{(i)}. \tag{21}$$

The composition operation \circ then reads

$$q_a \circ q_b = (\mathbf{x}_a, \boldsymbol{\rho}_a) \circ (\mathbf{x}_b, \boldsymbol{\rho}_b) = (\mathbf{x}_a + \mathbf{x}_b, \boldsymbol{\rho}_a \diamond \boldsymbol{\rho}_b). \tag{22}$$

In Eq. (22), the symbol \diamond denotes the composition operation for rotation parameters, whose explicit form depends on the respective rotation parameters. For LGIMs, the composition operation for Euler angles is given in [11], for the rotation vector in, e.g., [11, 33] and for Euler parameters in, e.g., [3, 12].

2.2.2 Equations of motion on $\mathbb{R}^3 \times SO(3)$

In this section, we introduce the EOMs of rigid body systems on the Lie group $\mathbb{R}^3 \times SO(3)$ and explain their connection with the classical formulations introduced earlier in Sect. 2.1.

The EOMs on $\mathbb{R}^3 \times SO(3)$ are closely related to the EOMs of the classical formulation 1. On $\mathbb{R}^3 \times SO(3)$, the equilibrium conditions, the constraint forces and the holonomic constraints are the same as in the classical formulation 1. Only the kinematic equations (2) are different; cf. Sect. 2.2.1. Thus, the EOMs on $\mathbb{R}^3 \times SO(3)$ read⁷

$$\begin{aligned} \mathbf{M}\dot{\mathbf{v}} &= \mathbf{g}(\mathbf{q}, \mathbf{v}, t) - \mathbf{B}^T(\mathbf{q})\boldsymbol{\lambda}, \\ \dot{\mathbf{Q}} &= \mathbf{Q}\tilde{\mathbf{v}}, \\ \boldsymbol{\Phi}(\mathbf{q}) &= \mathbf{0}. \end{aligned} \tag{23}$$

The equations in (23), more specifically the kinematic reconstruction equations (19), are integrated using LGIMs; see Sect. 3.1. This ensures that the orthogonality of the rotation matrix is preserved when solving $\dot{\mathbf{R}} = \mathbf{R}\tilde{\boldsymbol{\omega}}$ (Eq. (18)). The derivatives $\dot{\mathbf{Q}}$ are never explicitly evaluated in the numerical procedure, which will be illustrated in Sect. 3. The same applies to the matrix \mathbf{Q} ; only absolute coordinates \mathbf{q} are considered.

⁶We understand $\boldsymbol{\rho}$ as a ‘‘compressed storage of \mathbf{R} ,’’ as $\mathbf{R} = \mathbf{R}(\boldsymbol{\rho})$ holds.

⁷Note that for MBSs, $\dot{\mathbf{Q}} = \text{blockdiag}_{1 \leq i \leq N} \left(\dot{\mathbf{Q}}^{(i)} \right)$.

3 Embedding Lie group methods in an MBS formalism

In this section, we show how MBS formalisms that employ explicit RK methods [34], as well as the generalized- α method [25, 26] for the time integration of the EOMs of the classical formulation 2 (12), have to be modified to become Lie group methods, i.e., RKMK methods [19, 20] and Lie group generalized- α [1], respectively.

To this end, we first show for a single time step $t_n \rightarrow t_{n+1}$ how Lie group time integration methods solve the kinematic reconstruction equations (18). We then show how the classical time integration methods can be transformed into Lie group time integration methods and discuss implementation details.

3.1 Lie group time integration step

This section deals with solving the kinematic reconstruction equations (18) using LGIMs. The methodology is illustrated for a single rigid body.

LGIMs for absolute coordinates compute an update for the coordinates $\mathbf{q}(t)$ at time $t \in [t_n, t_{n+1}]$ within time step $n + 1$ in terms of local coordinates $\Delta\mathbf{q}(t)$ and initial values $\mathbf{q}(t_n)$. The transition from local coordinates and initial values to coordinates \mathbf{q} is performed by using the so-called *local-global transition* (LGT) map $\boldsymbol{\tau}$,

$$\begin{aligned} \boldsymbol{\tau} : \mathbb{R}^{3+l} \times \mathbb{R}^6 &\rightarrow \mathbb{R}^{3+l}, \\ (\mathbf{q}(t_n), \Delta\mathbf{q}(t)) &\mapsto \mathbf{q}(t) = \boldsymbol{\tau}(\mathbf{q}(t_n), \Delta\mathbf{q}(t)), \end{aligned} \quad (24)$$

which was introduced recently in [6, 27]. We determine the local coordinates $\Delta\mathbf{q}(t)$ by numerical integration of the ordinary differential equation

$$\Delta\dot{\mathbf{q}}(t) = \mathbf{T}_{\text{exp}}^{-1}(\Delta\mathbf{q}(t))\mathbf{v}(t) \quad (25)$$

with the initial conditions $\Delta\mathbf{q}(t_n) = \mathbf{0}$. In Eq. (25), $\mathbf{T}_{\text{exp}}^{-1} \in \mathbb{R}^{6 \times 6}$ represents the inverse of the tangent operator $\mathbf{T}_{\text{exp}} \in \mathbb{R}^{6 \times 6}$ on $\mathbb{R}^3 \times SO(3)$ and is given in closed form in Appendix A.1.

It is important to note that the LGT map (24) involves the composition operation (22). Thus, in Eq. (24), different rotation parameters can be used, such as Euler parameters, Euler angles or the rotation vector, for example.

In this paper, we use the three components of the rotation vector $\boldsymbol{\psi} \in \mathbb{R}^3$ as rotation parameters, i.e., $\boldsymbol{\rho} = \boldsymbol{\psi}$. The rotation vector is defined as the vector

$$\boldsymbol{\psi} = \varphi \mathbf{n}, \quad (26)$$

which has the direction of the rotation axis $\mathbf{n} = \boldsymbol{\psi} / \|\boldsymbol{\psi}\|$ and a length equal to the rotation angle $\varphi = \|\boldsymbol{\psi}\|$ [4]. Thus, in our case, the absolute coordinates $\mathbf{q}^{(i)} \in \mathbb{R}^6$ for the i -th rigid body are

$$\mathbf{q}^{(i)} = \begin{bmatrix} \mathbf{x}^{(i)} \\ \boldsymbol{\psi}^{(i)} \end{bmatrix}. \quad (27)$$

For implementation purposes, it is convenient to divide the vector $\Delta\mathbf{q} \in \mathbb{R}^6$ into the parts $\Delta\mathbf{x} \in \mathbb{R}^3$ and $\Delta\boldsymbol{\psi} \in \mathbb{R}^3$,

$$\Delta\mathbf{q} = \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\psi} \end{bmatrix}, \quad (28)$$

Algorithm 1 Numerical algorithm for updating absolute coordinates. The symbol “#” represents start of comments

```

function UAC(q, Δq)
1: for  $i = 1$  to  $n$  do # loop over all nodes  $n$ 
2:   if type( $\mathbf{q}^{(i)}$ ) = LieGroupNode then
3:      $\mathbf{q}^{(i)} := \begin{bmatrix} \boldsymbol{\tau}_x(\mathbf{x}^{(i)}, \Delta\mathbf{x}^{(i)}) \\ \boldsymbol{\tau}_\psi(\boldsymbol{\psi}^{(i)}, \Delta\boldsymbol{\psi}^{(i)}) \end{bmatrix}$  # apply composition operations
4:   else
5:      $\mathbf{q}^{(i)} := \mathbf{q}^{(i)} + \Delta\mathbf{q}^{(i)}$  # “conventional” update
6:   end if
7: end for
8: return q
end function

```

and to separate the LGT map (24) into an LGT map for translations $\boldsymbol{\tau}_x$,

$$\boldsymbol{\tau}_x : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3, \tag{29}$$

$$(\mathbf{x}(t_n), \Delta\mathbf{x}(t)) \mapsto \mathbf{x}(t) = \mathbf{x}(t_n) + \Delta\mathbf{x}(t),$$

and an LGT map for rotations $\boldsymbol{\tau}_\psi$,

$$\boldsymbol{\tau}_\psi : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}^3, \tag{30}$$

$$(\boldsymbol{\psi}(t_n), \Delta\boldsymbol{\psi}(t)) \mapsto \boldsymbol{\psi}(t) = \boldsymbol{\psi}(t_n) \diamond \Delta\boldsymbol{\psi}(t).$$

The symbol \diamond in Eq. (30) denotes the composition operation for rotation vectors, which is given in Appendix A.3.

At this point, we would like to emphasize that the easy transition from classical integration methods to LGIMs is facilitated by the fact that Eq. (24) can be used to directly calculate an update for the rotation parameters without the need for history variables [11]. Thus, the data structures available in an MBS package based on absolute coordinates can be used directly. Note also that beside rigid body nodes, other components in an MBS code might also add coordinates $\mathbf{q}^{(i)}$ to the vector of overall absolute coordinates \mathbf{q} , which might have a different meaning compared to rigid bodies, e.g., minimal coordinates of a subsystem or flexible coordinates. As the integration step for such coordinates has to be performed in the “conventional way”

$$\mathbf{q}^{(i)}(t) = \mathbf{q}^{(i)}(t_n) + \Delta\mathbf{q}^{(i)}(t), \tag{31}$$

it is necessary to distinguish between nodes of the type “Lie group” and the type “conventional.” To this end, we use an algorithm UAC (Update Absolute Coordinates) that performs the latter distinction; see Algorithm 1. The algorithm UAC represents the practical implementation of the LGT map proposed in [6, 27], but also includes linear spaces and therefore allows an arbitrary combination of Lie group and other formulations within a single simulation.

3.2 Explicit time integration

In this section, we show how MBS formalisms that employ explicit RK methods for the time integration of the EOMs (12) have to be modified to become Lie group methods, i.e., RKMK methods [19, 20].

Algorithm 2 Numerical algorithm for computing one step with an explicit RK method with s stages. The symbol “#” represents start of comments

function RKStep($\hat{\mathbf{q}}_n, \mathbf{w}_n, t_n$)

```

1: # evaluate slopes at the first stage, i.e., at  $i = 1$ 
2:  $\mathbf{K}_1 = \mathbf{w}_n$ 
3:  $\mathbf{k}_1 = \widehat{\mathbf{M}}^{-1} \widehat{\mathbf{g}}(\hat{\mathbf{q}}_n, \mathbf{w}_n, t_n)$ 
4: for  $i = 2, \dots, s$  do
5:   # evaluate solution vectors at stage  $i$ 
6:    $\hat{\mathbf{q}}_i := \hat{\mathbf{q}}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{K}_j$ 
7:    $\bar{\mathbf{v}}_i := \mathbf{w}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j$ 
8:   # evaluate slopes at stage  $i$  with solution vectors at stage  $i$ 
9:    $\mathbf{K}_i := \bar{\mathbf{v}}_i$ 
10:   $\mathbf{k}_i := \widehat{\mathbf{M}}^{-1} \widehat{\mathbf{g}}(\hat{\mathbf{q}}_i, \bar{\mathbf{v}}_i, t_n + c_i h)$ 
11: end for
12:  $\hat{\mathbf{q}}_{n+1} := \hat{\mathbf{q}}_n + h \sum_{j=1}^s b_j \mathbf{K}_j$ 
13:  $\mathbf{w}_{n+1} := \mathbf{w}_n + h \sum_{j=1}^s b_j \mathbf{k}_j$ 
14: return  $\hat{\mathbf{q}}_{n+1}, \mathbf{w}_{n+1}$ 

```

end function

To enable explicit time integration of the EOMs (12), we convert Eq. (12) without constraints to first-order equations,

$$\begin{aligned} \dot{\mathbf{w}} &= \widehat{\mathbf{M}}^{-1} \widehat{\mathbf{g}}(\hat{\mathbf{q}}, \mathbf{w}, t), \\ \hat{\dot{\mathbf{q}}} &= \mathbf{w}. \end{aligned} \quad (32)$$

In classical MBS formalisms, the equations (32) are solved for a single time step $t_n \rightarrow t_{n+1}$ as shown in Algorithm 2. The modifications required to transmit the explicit RK method shown in Algorithm 2 into an RKMK method are given in the subsequent listing and are illustrated in Algorithm 3.

1. Instead of calculating the absolute coordinates as described in line 6 and line 12 in Algorithm 2, Lie group time integration methods require \mathbf{q} to be determined via the LGT map (24). For this, the local coordinates $\Delta \mathbf{q}$ must be calculated in addition to \mathbf{q} . This modification is shown in lines 6–7 and 13–14 in Algorithm 3.
2. The kinematic equation $\hat{\dot{\mathbf{q}}}$ in Eq. (32) must be replaced by the kinematic equation (25). This modification is shown in line 10 in Algorithm 3.
3. Instead of calculating the slopes with Eq. (32) (see lines 3 and 10 in Algorithm 2), Lie group time integration methods require the slopes to be calculated using the equilibrium conditions (23). This modification is shown in lines 3 and 11 in Algorithm 3.

3.2.1 Automatic step size control for RKMK methods

Advanced explicit time integration methods such as ODE23 [35] and DOPRI5 [36] include automatic step size control and are well established in MBD for solving Eq. (32). Variable step size implementations for solving Eq. (23) with RKMK methods have been discussed in a recent paper of Celledoni et al.; see [29]. In this paper, we follow a slightly different approach, which is summarized in Appendix B and [37].

Algorithm 3 Numerical algorithm for computing one step with an explicit RKMK method with s stages. The symbol “#” represents start of comments

[h]

```

function LieRKStep( $\mathbf{q}_n, \mathbf{v}_n, t_n$ )
1: # evaluate slopes at the first stage, i.e., at  $i = 1$ 
2:  $\mathbf{K}_1 = \mathbf{v}_n$ 
3:  $\mathbf{k}_1 = \mathbf{M}^{-1} \mathbf{g}(\mathbf{q}_n, \mathbf{v}_n, t_n)$ 
4: for  $i = 2, \dots, s$  do
5:   # evaluate solution vectors at stage  $i$ 
6:    $\Delta \bar{\mathbf{q}}_i := h \sum_{j=1}^{i-1} a_{ij} \mathbf{K}_j$ 
7:    $\bar{\mathbf{q}}_i := \text{UAC}(\mathbf{q}_n, \Delta \bar{\mathbf{q}}_i)$ 
8:    $\bar{\mathbf{v}}_i := \mathbf{v}_n + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j$ 
9:   # evaluate slopes at stage  $i$  with solution vectors at stage  $i$ 
10:   $\mathbf{K}_i := \mathbf{T}_{\text{exp}}^{-1}(\Delta \bar{\mathbf{q}}_i) \bar{\mathbf{v}}_i$ 
11:   $\mathbf{k}_i := \mathbf{M}^{-1} \mathbf{g}(\bar{\mathbf{q}}_i, \bar{\mathbf{v}}_i, t_n + c_i h)$ 
12: end for
13:  $\Delta \mathbf{q}_n := h \sum_{j=1}^s b_j \mathbf{K}_j$ 
14:  $\mathbf{q}_{n+1} := \text{UAC}(\mathbf{q}_n, \Delta \mathbf{q}_n)$  # update absolute coordinates
15:  $\mathbf{v}_{n+1} := \mathbf{v}_n + h \sum_{j=1}^s b_j \mathbf{k}_j$ 
16: return  $\mathbf{q}_{n+1}, \mathbf{v}_{n+1}$ 
end function

```

3.3 Implicit time integration

In this section, we show how an MBS formalism that employs the generalized- α method [25, 26] for the time integration of the EOMs in parameterized form (12) has to be modified to obtain Lie group generalized- α [1]. In [26], a function AlphaStep has been introduced that solves the EOMs in parameterized form (12) at time t_{n+1} using the generalized- α method. In the following, we show how AlphaStep can be modified to obtain Lie group generalized- α .

The function AlphaStep (see Algorithm 4) represents a Newton method with unknowns $\hat{\mathbf{q}}$ and λ , and solves Eq. (12) at time t_{n+1} and involves the parameters

$$\beta' = \frac{1 - \alpha_m}{h^2 \beta (1 - \alpha_f)}, \quad \gamma' = \frac{\gamma}{h \beta}, \tag{33}$$

the residual vector

$$\hat{\mathbf{r}}(\hat{\mathbf{q}}, \hat{\mathbf{q}}, \hat{\mathbf{q}}, \lambda, t) = \hat{\mathbf{M}}(\hat{\mathbf{q}}) \hat{\mathbf{q}} - \hat{\mathbf{H}}^T(\hat{\mathbf{q}}) (\hat{\mathbf{g}}(\hat{\mathbf{q}}, \hat{\mathbf{q}}, t) - \hat{\mathbf{B}}^T(\hat{\mathbf{q}}) \lambda) \tag{34}$$

and the iteration matrix⁸

$$\hat{\mathbf{S}}_t = \begin{bmatrix} \hat{\mathbf{M}}(\hat{\mathbf{q}}) \beta' + \hat{\mathbf{C}}_t(\hat{\mathbf{q}}, \hat{\mathbf{q}}, t) \gamma' + \hat{\mathbf{K}}_t(\hat{\mathbf{q}}, \hat{\mathbf{q}}, \lambda, t) & \hat{\mathbf{H}}^T(\hat{\mathbf{q}}) \hat{\mathbf{B}}^T(\hat{\mathbf{q}}) \\ \hat{\mathbf{H}}(\hat{\mathbf{q}}) \hat{\mathbf{B}}(\hat{\mathbf{q}}) & \mathbf{0} \end{bmatrix}, \tag{35}$$

which is also denoted as Jacobian. The numerical parameters $\alpha_m, \alpha_f, \beta$ and γ in Eqs. (33)–(35) can be selected by choosing the numerical damping $\rho_\infty \in [0, 1]$ in order

⁸Note that $\hat{\mathbf{S}}_t$ may be singular for formulations with three rotation parameters at singular points.

Algorithm 4 Numerical algorithm for computing a Newton update for generalized- α according to [26]

```

function AlphaStep( $\hat{\mathbf{q}}_n, \hat{\dot{\mathbf{q}}}_n, \hat{\ddot{\mathbf{q}}}_n, \mathbf{a}$ )
1:  $\hat{\mathbf{q}}_{n+1} := \mathbf{0}$ 
2:  $\lambda_{n+1} := \mathbf{0}$ 
3:  $\bar{\mathbf{a}} := (\alpha_f \hat{\ddot{\mathbf{q}}}_n - \alpha_m \mathbf{a}) / (1 - \alpha_m)$ 
4:  $\hat{\dot{\mathbf{q}}}_{n+1} := \hat{\dot{\mathbf{q}}}_n + h(1 - \gamma) \mathbf{a} + h\gamma \bar{\mathbf{a}}$ 
5:  $\hat{\mathbf{q}}_{n+1} := \hat{\mathbf{q}}_n + h \hat{\dot{\mathbf{q}}}_n + h^2(0.5 - \beta) \mathbf{a} + h^2 \beta \bar{\mathbf{a}}$ 
6: for  $k = 1$  to  $k_{max}$  do
7:    $\mathbf{res} := \begin{bmatrix} \hat{\mathbf{r}}(\hat{\mathbf{q}}_{n+1}, \hat{\dot{\mathbf{q}}}_{n+1}, \hat{\ddot{\mathbf{q}}}_{n+1}, \lambda_{n+1}, t_{n+1}) \\ \hat{\Phi}(\hat{\mathbf{q}}_{n+1}) \end{bmatrix}$ 
8:   if  $\|\mathbf{res}\| < tol$  then
9:     break
10:  end if
11:   $\hat{\mathbf{S}}_t := \hat{\mathbf{S}}_t(\hat{\mathbf{q}}_{n+1}, \hat{\dot{\mathbf{q}}}_{n+1}, \hat{\ddot{\mathbf{q}}}_{n+1}, \lambda_{n+1}, t_{n+1})$ 
12:   $\begin{bmatrix} \Delta \hat{\mathbf{q}} \\ \Delta \lambda \end{bmatrix} := -\hat{\mathbf{S}}_t^{-1} \mathbf{res}$ 
13:   $\hat{\mathbf{q}}_{n+1} := \hat{\mathbf{q}}_{n+1} + \Delta \hat{\mathbf{q}}$ 
14:   $\hat{\dot{\mathbf{q}}}_{n+1} := \hat{\dot{\mathbf{q}}}_{n+1} + \gamma' \Delta \hat{\mathbf{q}}$ 
15:   $\hat{\ddot{\mathbf{q}}}_{n+1} := \hat{\ddot{\mathbf{q}}}_{n+1} + \beta' \Delta \hat{\mathbf{q}}$ 
16:   $\lambda_{n+1} := \lambda_{n+1} + \Delta \lambda$ 
17: end for
18:  $\mathbf{a} := \bar{\mathbf{a}} + (1 - \alpha_f) / (1 - \alpha_m) \hat{\ddot{\mathbf{q}}}_{n+1}$ 
19: return  $\hat{\mathbf{q}}_{n+1}, \hat{\dot{\mathbf{q}}}_{n+1}, \hat{\ddot{\mathbf{q}}}_{n+1}, \lambda_{n+1}, \mathbf{a}$ 
end function

```

to have suitable accuracy and stability properties [26]. In Eq. (35), the tangent stiffness and damping matrices $\hat{\mathbf{K}}_t$ and $\hat{\mathbf{C}}_t$ are calculated by

$$\hat{\mathbf{K}}_t(\hat{\mathbf{q}}, \hat{\dot{\mathbf{q}}}, \hat{\ddot{\mathbf{q}}}, \lambda, t) = \left. \frac{\partial \hat{\mathbf{r}}(\mathbf{z}, \hat{\mathbf{q}}, \hat{\dot{\mathbf{q}}}, \lambda, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\hat{\mathbf{q}}} \quad (36)$$

$$\approx - \left. \frac{\partial \hat{\mathbf{H}}^T(\mathbf{z}) \hat{\mathbf{g}}(\mathbf{z}, \hat{\mathbf{q}}, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\hat{\mathbf{q}}}, \quad (37)$$

$$\hat{\mathbf{C}}_t(\hat{\mathbf{q}}, \hat{\dot{\mathbf{q}}}, t) = - \left. \frac{\partial \hat{\mathbf{H}}^T(\hat{\mathbf{q}}) \hat{\mathbf{g}}(\hat{\mathbf{q}}, \mathbf{z}, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\hat{\mathbf{q}}}, \quad (38)$$

where we use Eq. (37) in our implementation. Note that in Algorithm 4, the auxiliary variable \mathbf{a} is set to⁹ $\mathbf{a}_0 = \hat{\ddot{\mathbf{q}}}_0$ for time $t = 0$; see [26].

The modifications required to transform AlphaStep (Algorithm 4) into a Lie group method, i.e., SolveTimeStep [1], are illustrated in Algorithm 5 and are given in the subsequent listing.

1. The unknowns in the Newton method are now $\Delta \mathbf{q}$ and λ ; cf. Sect. 3.1.

⁹An optimal initialization scheme for Lie group generalized- α , which guarantees second-order convergence in all solution components, is presented in [9], which is not considered here, however.

Algorithm 5 Numerical algorithm computing a Newton update for Lie generalized- α . The symbol “#” represents start of comments

```

function LieAlphaStep( $\mathbf{q}_n, \mathbf{v}_n, \dot{\mathbf{v}}_n, \mathbf{a}$ )
1:  $\dot{\mathbf{v}}_{n+1} := \mathbf{0}$ 
2:  $\boldsymbol{\lambda}_{n+1} := \mathbf{0}$ 
3:  $\bar{\mathbf{a}} := (\alpha_f \dot{\mathbf{v}}_n - \alpha_m \mathbf{a}) / (1 - \alpha_m)$ 
4:  $\mathbf{v}_{n+1} := \mathbf{v}_n + h(1 - \gamma)\mathbf{a} + h\gamma\bar{\mathbf{a}}$ 
5:  $\Delta\mathbf{q}_n := h\mathbf{v}_n + h^2(0.5 - \beta)\mathbf{a} + h^2\beta\bar{\mathbf{a}}$ 
6:  $\mathbf{q}_{n+1} := \text{UAC}(\mathbf{q}_n, \Delta\mathbf{q}_n)$  # initial guess for absolute coordinates
7: for  $k = 1$  to  $k_{max}$  do
8:    $\mathbf{res} := \begin{bmatrix} \mathbf{r}(\mathbf{q}_{n+1}, \mathbf{v}_{n+1}, \dot{\mathbf{v}}_{n+1}, \boldsymbol{\lambda}_{n+1}, t_{n+1}) \\ \boldsymbol{\Phi}(\mathbf{q}_{n+1}) \end{bmatrix}$ 
9:   if  $\|\mathbf{res}\| < tol$  then
10:     break
11:   end if
12:    $\mathbf{S}_t := \mathbf{S}_t(\mathbf{q}_{n+1}, \Delta\mathbf{q}, \mathbf{v}_{n+1}, \dot{\mathbf{v}}_{n+1}, \boldsymbol{\lambda}_{n+1}, t_{n+1})$ 
13:    $\begin{bmatrix} \Delta\mathbf{q} \\ \Delta\boldsymbol{\lambda} \end{bmatrix} := -\mathbf{S}_t^{-1}\mathbf{res}$ 
14:    $\Delta\mathbf{q}_n := \Delta\mathbf{q}_n + \Delta\mathbf{q}$ 
15:    $\mathbf{q}_{n+1} := \text{UAC}(\mathbf{q}_n, \Delta\mathbf{q}_n)$  # update absolute coordinates
16:    $\mathbf{v}_{n+1} := \mathbf{v}_{n+1} + \gamma'\Delta\mathbf{q}$ 
17:    $\dot{\mathbf{v}}_{n+1} := \dot{\mathbf{v}}_{n+1} + \beta'\Delta\mathbf{q}$ 
18:    $\boldsymbol{\lambda}_{n+1} := \boldsymbol{\lambda}_{n+1} + \Delta\boldsymbol{\lambda}$ 
19: end for
20:  $\mathbf{a} := \bar{\mathbf{a}} + (1 - \alpha_f) / (1 - \alpha_m) \dot{\mathbf{v}}_{n+1}$ 
21: return  $\mathbf{q}_{n+1}, \mathbf{v}_{n+1}, \dot{\mathbf{v}}_{n+1}, \boldsymbol{\lambda}_{n+1}, \mathbf{a}$ 
end function

```

2. The absolute coordinates \mathbf{q} have to be determined via the LGT map (24). For this, the vector $\Delta\mathbf{q}$ must be calculated in addition to \mathbf{q} . This modification is shown in lines 5–6 and 14–15 in Algorithm 5. Note that the latter modification might be also achieved by (1) introducing an additional data coordinate $\hat{\mathbf{q}}_n$ that stores \mathbf{q}_n , i.e., $\hat{\mathbf{q}}_n = \mathbf{q}_n$, and (2) by setting $\mathbf{q}_n = \mathbf{0}$. The data coordinate $\hat{\mathbf{q}}_n$ may then be used instead of \mathbf{q}_n to update the absolute coordinates in lines 6 and 15.
3. The residual vector Eq. (34) has to be computed with the equilibrium conditions in Eq. (23), which now reads

$$\mathbf{r}(\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, t) = \mathbf{M}(\mathbf{q})\dot{\mathbf{v}} - \mathbf{g}(\mathbf{q}, \mathbf{v}, t) + \mathbf{B}^T(\mathbf{q})\boldsymbol{\lambda}. \tag{39}$$

This modification is shown in line 8 in Algorithm 5.

4. The iteration matrix Eq. (35) has to be computed differently. For Lie group generalized- α , the iteration matrix is given by [1, 9, 23]

$$\mathbf{S}_t(\mathbf{q}, \Delta\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, t) = \begin{bmatrix} \mathbf{M}(\mathbf{q})\beta' + \mathbf{C}_t(\mathbf{q}, \mathbf{v}, t)\gamma' + \mathbf{K}_t(\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, t)\mathbf{T}_{\text{exp}}(\Delta\mathbf{q}) & \mathbf{B}^T(\mathbf{q}) \\ \mathbf{B}(\mathbf{q})\mathbf{T}_{\text{exp}}(\Delta\mathbf{q}) & \mathbf{0} \end{bmatrix}. \tag{40}$$

This modification is shown in line 12 in Algorithm 5. In this paper, the tangent stiffness matrix¹⁰ \mathbf{K}_t , reads

$$\mathbf{K}_t(\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, t) = \left. \frac{\partial \mathbf{r}(\text{UAC}(\mathbf{q}, \mathbf{z}), \mathbf{v}, \dot{\mathbf{v}}, \boldsymbol{\lambda}, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{0}} \quad (41)$$

$$\approx - \left. \frac{\partial \mathbf{g}(\text{UAC}(\mathbf{q}, \mathbf{z}), \mathbf{v}, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{0}}, \quad (42)$$

where we use Eq. (42) in our implementation.¹¹ The damping matrix \mathbf{C}_t in Eq. (40) reads

$$\mathbf{C}_t(\mathbf{q}, \mathbf{v}, t) = - \left. \frac{\partial \mathbf{g}(\mathbf{q}, \mathbf{z}, t)}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{v}}. \quad (43)$$

Practical experiments have shown that the tangent operator $\mathbf{T}_{\text{exp}}(\Delta \mathbf{q}) = \mathbf{I} + \mathcal{O}(\|\Delta \mathbf{q}\|)$ can be neglected in Eq. (40) for small time steps, which we do throughout this paper and will illustrate in the numerical examples in Sect. 4. Note however that neglecting the tangent operator in Eq. (40) may lead to stability problems but may prevent the destruction of sparsity structures of the system matrices in the simulation code.

Both iteration matrices Eq. (35) and Eq. (40) become severely ill conditioned for small time step sizes h [9, 26]. To avoid this phenomenon, we use the scaling method proposed by Bottasso et al. in [38] in our implementation. The scaling of the iteration matrix for the generalized- α method and its Lie group counterpart are discussed, for example, in [9, 26, 39].

3.4 Modified Newton method

Evaluating the iteration matrix (35) respectively (40) might be time-consuming in the full Newton method, where the Jacobian is computed in each Newton iteration. Therefore, to increase computational performance, we compute an update and factorization of the iteration matrix in the so-called “modified Newton method” only if needed. Therein, the iteration matrix is computed at the beginning of the simulation and updated during the iteration process only if either a predefined maximum number of Newton iterations is reached or the error in each Newton iteration could not be reduced by a predefined tolerance [37, 40].

3.5 Effects of round-off errors in \mathbf{T}_{exp} and $\mathbf{T}_{\text{exp}}^{-1}$

In our numerical experiments, we observed effects of round-off errors in the evaluation of the tangent operator \mathbf{T}_{exp} respectively its inverse near $\|\Delta \mathbf{q}\| = \mathbf{0}$. The effects of round-off errors when evaluating tangent operators have also been reported in [41] and are termed there as a loss of significance. To obtain accurate and precise results, we use Taylor approximations for those terms that cause the loss of significance. The approximation strategy is presented in Appendix A.2.

¹⁰The multiplication of the tangent stiffness matrix \mathbf{K}_t with the matrix \mathbf{T}_{exp} in Eq. (40) can be avoided by evaluating the directional derivative (41) at $\mathbf{z} = \Delta \mathbf{q}$.

¹¹Using Eq. (42) yields an improvement of the computational performance compared to Eq. (40) as the linearization of the mass matrix and constraint forces is avoided.

4 Numerical examples

In this section, we compare the classical formulations 1 and 2 with an LGIM in terms of accuracy and computational performance using five typical rigid body systems. In the LGIM, the rotation vector is used to represent the orientation of a rigid body; see Sect. 3.1. In both classical formulations, Cardan/Tait–Bryan angles (123-Euler angles, cf. [37, 42]) are used as rotation parameters. Euler parameters are used only in the classical formulation 2 in combination with implicit integration and by imposing the Euler parameter unit length condition as an additional constraint equation on position level; cf. [37]. Both the LGIM and the classical formulation 1 are used in combination with explicit and implicit time integration methods. In all numerical examples, the following abbreviations are used:

- “Rxyz” means Cardan/Tait–Bryan angles (classical formulation 2),
- “RV” means rotation vector (LGIM),
- “EP” means Euler parameter (classical formulation 2).

The abbreviations used in the following for the explicit RK and RKMK methods, such as RK44, follow the designations used in [34].

All rigid body systems considered are set up and simulated in Exudyn¹² [37, 43] using version 1.4.5.dev1 and Python 3.7 64bit. Note that the LGIM has been implemented in C++ focusing on efficiency. All simulations have been performed in Windows 10 on an Intel Core i7-6600U 2.60 GHz processor. The absolute and relative error tolerances a_{tol} and r_{tol} for automatic step size control have been chosen in each numerical example as $r_{tol} = 10^{-2}/(t_{End}/h)^p$ and $a_{tol} = r_{tol} \times 10^{-2}$, where t_{End} represents the simulation period and p the order of the integration method. The spectral radius is set to $\rho_\infty = 0.9$ for generalized- α and Lie group generalized- α for all examples.

4.1 Heavy top

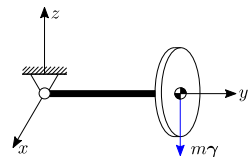
The first rigid body system we consider is the benchmark example “heavy top”; see Fig. 1. Two simulation cases are considered, which are described in the following two sections. In both cases, we simulate the motion of the heavy top over a period of $t = 1$ s. Model parameters and initial conditions are summarized in Table 3.

4.1.1 Heavy top without kinematic constraints

In the first investigations, we consider the heavy top without kinematic constraints according to Refs. [1, 3, 10, 11, 29].

In Fig. 2, the time histories of the heavy top’s center of mass (COM) position coordinates (left plot) and translational velocity coordinates (right plot) computed with Lie group generalized- α are compared with reference solutions computed with the multibody simulation software RecurDyn.¹³ As can be seen in Fig. 2, the time histories obtained with Lie

Fig. 1 Benchmark problem
Heavy top; see, e.g., Ref. [1]



¹²<https://github.com/jgerstmayr/EXUDYN>.

¹³<https://support.functionbay.com/en/page/single/2/recurdyn-overview>, accessed on February 23, 2023.

Table 3 Model parameters and initial conditions of the heavy top

Parameter	Value	Notes
m in kg	15	mass of the heavy top
J_{xx} in kg m^2	0.234375	moment of inertia w.r.t. COM for x -axis
J_{yy} in kg m^2	0.468750	moment of inertia w.r.t. COM for y -axis
J_{zz} in kg m^2	0.234375	moment of inertia w.r.t. COM for z -axis
\mathbf{x} in m	[0.0, 1.0, 0.0]	position vector of the COM w.r.t. the fixing point represented in the body-attached frame
$\boldsymbol{\gamma}$ in m/s^2	[0.0, 0.0, -9.81]	vector of gravity acceleration
$\boldsymbol{\omega}_0$ in rad/s	[0.0, 150.0, -4.61538]	initial angular velocity
$\boldsymbol{\psi}_0$ in rad	[0.0, 0.0, 0.0]	initial orientation (rotation vector)

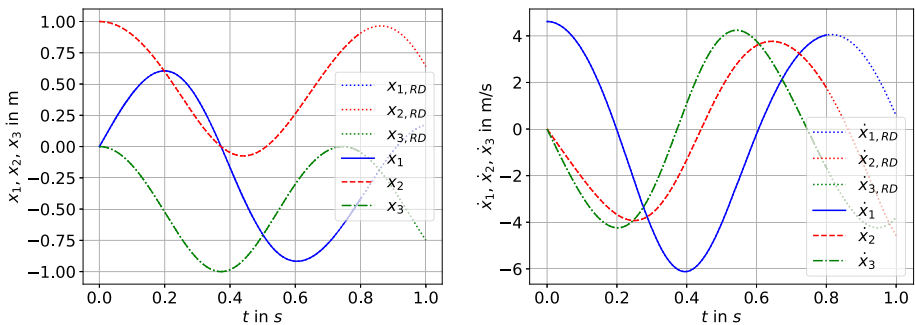


Fig. 2 Time histories of the heavy top's center of mass position (left plot) and translational velocity coordinates (right plot) expressed in the global frame. Reference solutions were computed over a simulation time of $t = 1$ s with a time step size of $h = 2.5 \times 10^{-5}$ s using RecurDyn. The heavy top's center of mass position and translational velocity coordinates obtained with Euler parameters and generalized- α are computed for comparison purposes over $t = 0.8$ s

group generalized- α visually match the reference solutions. The absolute deviation of the heavy top's position and velocity coordinates at $t = 1$ s is approximately 1.0×10^{-5} m respectively 1.0×10^{-5} m/s. Similar results have been obtained for explicit Lie group time integration.

In Fig. 3, the convergence in the norm of the COM position error $\|\mathbf{x}_{Ref} - \mathbf{x}\|$ is illustrated for explicit (left plot) and implicit time integration (right plot). The fixed time step sizes used in the convergence analysis are calculated by $h_n = (1.0 \times 10^{-4} \times 2^{(1-n)})_{n=1,2,\dots,8}$. Convergence is investigated at $t = 1$ s. The reference solution for the convergence analysis \mathbf{x}_{Ref} was computed using the LGIM and an explicit RK method with order six using a time step size of $h_{Ref} = 3.90625 \times 10^{-6}$ s. As can be seen in Fig. 3, in case of explicit time integration, LGIM outperforms the conventional approach in terms of accuracy. In contrast, the conventional Euler parameter approach is more accurate than Lie group time integration for implicit integration; see Fig. 3 (right plot).

At this point, we conclude that Euler parameters only give nonlinearities with low-order polynomials, e.g., in the rotation matrix, which seem to be more suitable for the second-order time integrator than the Lie group method.

In Table 4 the average number of Newton iterations per time step needed for the simulation of the heavy top is given. As can be seen in Table 4, the average number of Newton

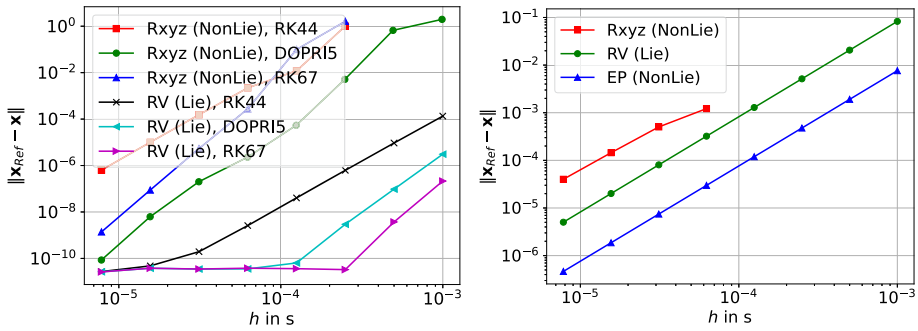


Fig. 3 Convergence in the norm of the maximum error of the heavy top without kinematic constraints COM position for different rotation parameters and time integration methods. *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

Table 4 Average number of Newton iterations per time step needed in generalized- α for different rotation parameters: benchmark heavy top without kinematic constraints

h in s	Rxyz (non-Lie)	RV (Lie)	EP (non-Lie)
1.00×10^{-3}	–	1.96	3.0
5.00×10^{-4}	–	1.9	2.52
2.50×10^{-4}	–	1.8	2.0
1.25×10^{-4}	–	1.6	2.0
6.25×10^{-5}	2.36	1.0	2.0
3.12×10^{-5}	2.09	1.0	2.0
1.56×10^{-5}	2.01	1.0	2.0
7.81×10^{-6}	1.96	1.0	2.0

iterations per time step is lower for Lie group time integration as compared to the conventional approaches, which is due to the reduced nonlinearities in the EOMs for Lie group methods.

The CPU time needed for simulation is shown in Fig. 4 for explicit (left plot) and implicit integration (right plot). As can be seen in Fig. 4, the CPU time is on average shorter in the case of Lie group time integration, both explicit and implicit, than in the conventional approaches. In the case of implicit integration, the latter observation is in line with our expectations, since the lower number of Newton iterations required by the LGIM compared to the two conventional approaches should also be reflected in the CPU time. In contrast to the expectation, in case of explicit integration, the additional computational effort required by the LGIM¹⁴ compared to the conventional approach (compare Algorithm 2 with Algorithm 3) does not seem to have a big effect on the CPU time. The shorter CPU time of the LGIM stems from the fact that the mass matrix is constant and therefore does not need to be recomputed and factorized in each time step. However, this is not the case for the conventional Rxyz formulation. Similar behavior is observed in all other numerical examples considered, as will be shown in the remainder of this paper.

Finally, Fig. 5 shows the work–precision diagrams for explicit and implicit integration. As illustrated in Fig. 5, the CPU time required to achieve, for example, an accuracy of

¹⁴Evaluating the integration formulas requires about six times more CPU time for the LGIM than in the conventional Rxyz approach in this example.

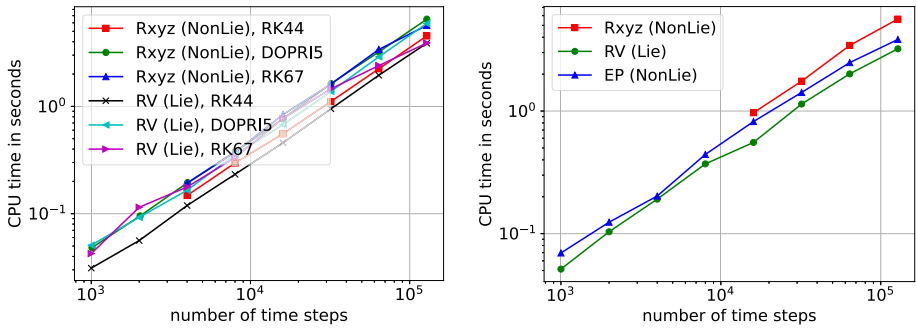


Fig. 4 CPU time of “heavy top without kinematic constraints” for different rotation parameters and time integration methods. *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

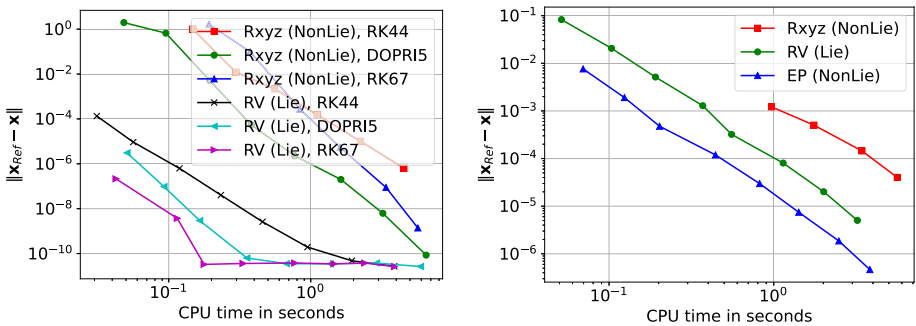


Fig. 5 Work–precision diagrams for explicit and implicit time integration methods for the example “heavy top without kinematic constraints.” *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

1.0×10^{-6} with Cardan/Tait–Bryan angles and RK44 is about 40 times higher than with the LGIM. In contrast, the conventional Euler parameter approach requires only about a quarter of the CPU time needed by the LGIM to achieve an accuracy of 1.0×10^{-4} in case of implicit integration.

4.1.2 Heavy top with kinematic constraints

In the second investigations, we consider the heavy top with kinematic constraints according to Refs. [1, 2, 9, 17, 22, 23, 28–30].

In Fig. 6, the convergence in the norm of the heavy top’s COM position error $\|\mathbf{x}_{Ref} - \mathbf{x}\|$ is illustrated for implicit time integration and different types of Newton methods; cf. Sect. 3.4. The time step sizes used in the convergence analysis are calculated again by $h_n = (1.0 \times 10^{-4} \times 2^{(1-n)})_{n=1,2,\dots,8}$ and convergence is investigated at $t = 1$ s. The reference solution \mathbf{x}_{Ref} has been computed with the conventional EP approach and generalized- α using a time step size of $h_{Ref} = 1 \times 10^{-7}$ s.

As can be seen in Fig. 6, the conventional EP approach is again more accurate than Lie group time integration both for full Newton and modified Newton. However, the LGIM can simulate the heavy top even at the largest considered time step sizes as compared to both classical approaches.

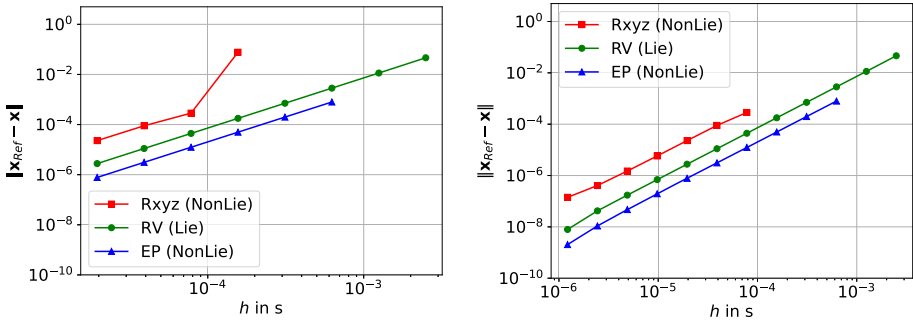


Fig. 6 Convergence in the norm of the maximum error of the heavy top with kinematic constraints COM position for different rotation parameters and generalized- α using different Newton methods. *Left plot*, full Newton; *right plot*, modified Newton

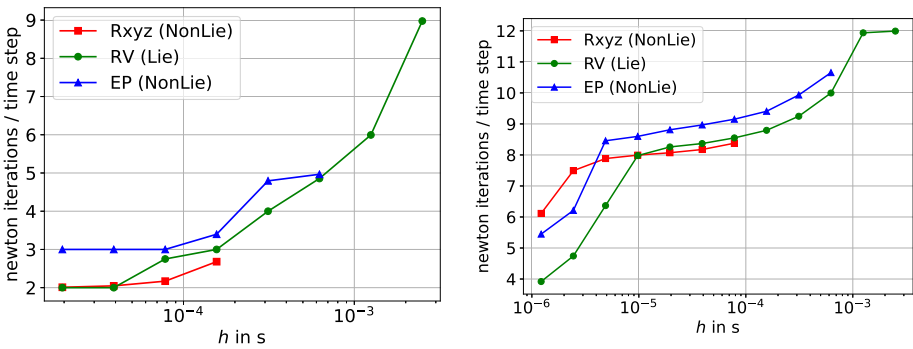


Fig. 7 Number of Newton iterations (generalised- α) required in the “heavy top with kinematic constraints” example for different Newton methods and rotation parameters. *Left plot*, full Newton; *right plot*, modified Newton. For the LGIM (RV), \mathbf{T}_{exp} is included in Eq. (40)

As expected, the number of Newton iterations for “modified Newton” is larger compared to “full Newton”; see Fig. 7. Figure 7 also shows that the number of Newton iterations is lower for Lie group time integration than in the conventional approaches, which results in a shorter CPU time as shown in Fig. 8. Finally, Table 5 shows the average number of Newton iterations required per time step when the tangent operator \mathbf{T}_{exp} is considered or neglected in Eq. (40). As can be seen from Table 5, neglecting \mathbf{T}_{exp} in the iteration matrix (40) leads to a larger number of Newton iterations per time step at time step sizes $h \geq 4.0 \times 10^{-5}$ s, which corresponds at time $t = 1$ s to an incremental rotation of $\|\Delta\psi\| \approx \|h\omega\| \approx 6.0 \times 10^{-3}$ rad. However, for time step sizes $h < 4.0 \times 10^{-5}$ s, neglecting \mathbf{T}_{exp} in Eq. (40) no longer affects the number of Newton iterations needed per time step. As can be seen in Table 5, including \mathbf{T}_{exp} in Eq. (40) within the modified Newton method has hardly any effect on the required Newton iterations per time step; cf. Fig. 7. We observed similar results in all other numerical examples.

4.2 High-speed rotor with flexible supports

We investigate the performance of Lie group time integration methods compared to the conventional approaches considered using a flexibly supported high-speed rotor; see Fig. 9.

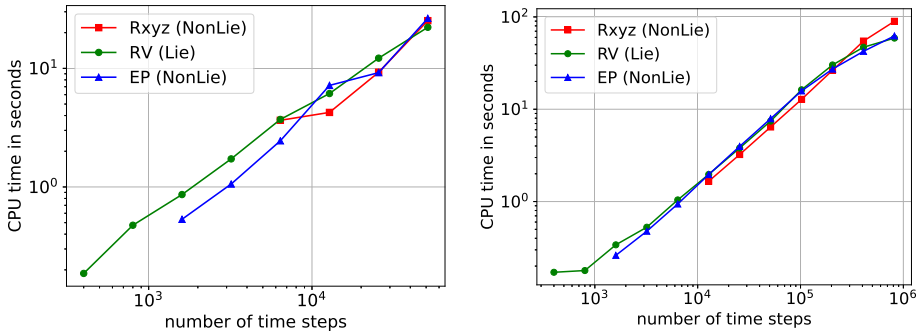
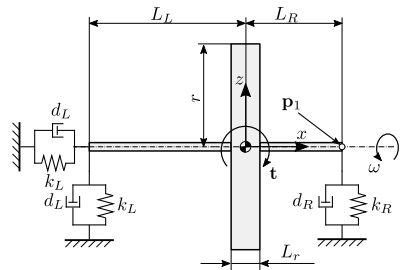


Fig. 8 CPU time (generalised- α) required in the “heavy top with kinematic constraints” example for different Newton methods and rotation parameters. *Left plot*, full Newton; *right plot*, modified Newton

Table 5 Average number of Newton iterations per time step needed in Lie group generalised- α when the tangent operator \mathbf{T}_{exp} is considered or neglected in Eq. (40): benchmark heavy top with kinematic constraints

h in s	With T_{exp} (full Newton)	Without T_{exp} (full Newton)	Without T_{exp} (modified Newton)
2.50×10^{-3}	3.0	8.98	11.99
1.25×10^{-3}	2.41	6.0	11.94
6.25×10^{-4}	2.0	4.86	10.0
3.12×10^{-4}	2.0	4.0	9.25
1.56×10^{-4}	2.0	3.0	8.79
7.81×10^{-5}	2.0	2.75	8.55
3.91×10^{-5}	2.0	2.0	8.37

Fig. 9 Schematic representation of high-speed rotor with flexible supports



The model parameters are given in Table 6. The rotor is subjected to a torque of

$$\mathbf{t} = [0 \quad 0 \quad (\frac{L}{2} - L_L)mg]^T. \tag{44}$$

All simulations are performed over a period of $t = 4$ s.

In Fig. 10, the trajectory of point $\mathbf{p}_1 = [-L_L, 0.0, 0.0]$ in the y - z -plane (left plot) and the time history of its coordinates (right plot) are shown. Figure 10 was obtained using the LGIM in combination with an explicit RK method with order six (RK67) and a time step size of $h = 1.0 \times 10^{-5}$ s. As can be seen in the left plot of Fig. 10, the rotor performs a precession motion, whereby the precession period is about 2.5 s (cf. Fig. 10, right plot), which corresponds to a precession frequency of 0.4 Hz.

Table 6 Model parameters of “high-speed rotor with flexible supports”

Model parameter	Value	Notes
m in kg	1.223	mass of the rotor
r in m	0.05	radius for disk mass distribution
L_r in m	0.02	length of rotor disk
J_{xx} in kg m^2	0.001541	polar moment of inertia
J_{yy} in kg m^2	0.000812	moment of inertia for y-axis
J_{zz} in kg m^2	0.000812	moment of inertia for z-axis
L_L in m	0.11	length of rotor
L_R in m	0.09	length of rotor
n in rpm	200000	rotational speed
ω in rad/s	20944	$\omega = 2\pi n/60$
k_L in N/m	4000	applied in x-y-z-direction
k_R in N/m	4000	applied in y-z-direction
D	0.0001	dimensionless damping
d_L in kg/s	5.165093	$d_L = 2D\omega m$ applied in x-y-z-direction
d_R in kg/s	5.165093	$d_R = 2D\omega m$ applied in y-z-direction
g in m/s^2	9.81	gravitational acceleration

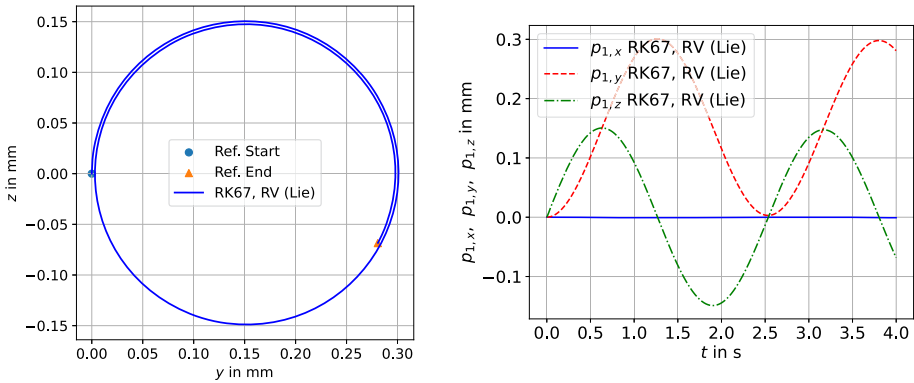


Fig. 10 *Left:* Trajectory of point \mathbf{p}_1 on the rotor located at the position of the left support. *Right:* Time history of the components of the angular velocity vector represented in the local frame. *Left plot,* tilting motion of the rotor; *right plot,* position coordinates of point \mathbf{p}_1

In Fig. 11, the convergence in the norm of the position error $\|\mathbf{p}_{Ref} - \mathbf{p}\|$ of point \mathbf{p}_1 on the rotor is illustrated for explicit time integration (left plot) and for implicit time integration (right plot). The fixed time step sizes used in the convergence analysis are calculated by $h_n = (1.0 \times 10^{-4} \times 2^{(1-n)})_{n=1,2,\dots,8}$. Convergence is investigated at $t = 4$ s. The reference solution \mathbf{p}_{Ref} for the convergence analysis was computed using the conventional approach Rxyz and RK67 using a time step size of $h_{Ref} = 1.0 \times 10^{-6}$ s. As can be seen in Fig. 11, Lie group time integration outperforms the conventional approach in terms of accuracy in case of explicit time integration. In contrast, the conventional EP approach is again more accurate than the LGIM in case of implicit integration; see Fig. 11 (right plot). However, in contrast to the conventional EP approach, both the LGIM and the conventional Rxyz approach can simulate the rotor even at the largest considered time step sizes.

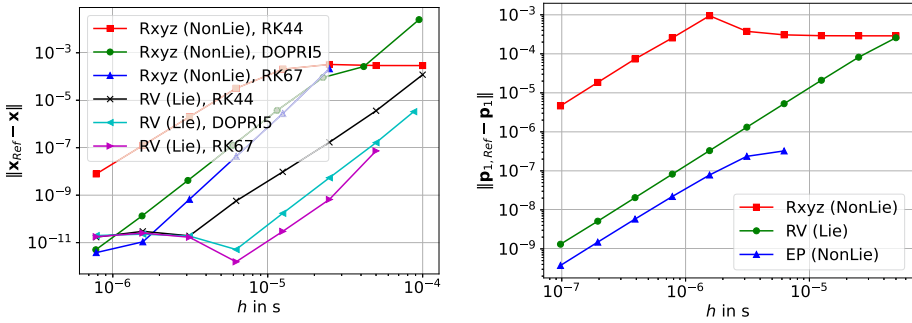


Fig. 11 Convergence in the norm of the maximum error of point \mathbf{p}_1 on the rotor located at the position of the left support for different rotation parameters and time integration methods. *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

Table 7 Average number of Newton iterations per time step needed in generalized- α for different rotation parameters: high-speed rotor with flexible supports

h in s	Rxyz (non-Lie)	RV (Lie)	EP (non-Lie)
5.00×10^{-5}	3.0	2.0	–
2.50×10^{-5}	3.0	2.0	–
1.25×10^{-5}	3.0	2.0	–
6.25×10^{-6}	2.0	2.0	3.17
3.12×10^{-6}	2.0	2.0	3.0
1.56×10^{-6}	2.0	1.0	2.81
7.81×10^{-7}	2.0	1.0	2.0
3.91×10^{-7}	2.0	1.0	2.0
1.95×10^{-7}	2.0	1.0	2.0
9.77×10^{-8}	1.23	1.0	2.0

In Table 7 the average number of Newton iterations per time step needed for the simulation of the rotor is given. As can be seen in Table 7, the average number of Newton iterations per time step is smaller for LGIM as compared to the conventional approaches.

The CPU time needed for simulation of the high-speed rotor is shown in Fig. 12 for explicit integration (right plot) and for implicit integration (left plot). As with the previous numerical examples, the CPU time is in the case of the LGIM on average equal or even smaller as compared to the conventional approaches, both for explicit and implicit integration.

Finally, Fig. 13 shows the work–precision diagrams for explicit (left plot) and implicit (right plot) integration. As illustrated in Fig. 13, the CPU time required to achieve, for example, an accuracy of 1.0×10^{-9} with Cardan/Tait–Bryan angles and DOPRI5 or RK67 is about 10 times higher than with the LGIM. Figure 13 also shows that the LGIM achieves in this example in case of implicit integration a higher performance than the conventional Rxyz approach and a similar performance to the conventional Euler parameter approach.

4.3 Spatial rigid Slider–Crank mechanism

In the next investigations, we compare the accuracy and computational performance of LGIM with both conventional approaches using the slider–crank mechanism shown in Fig. 14. The spatial rigid slider–crank mechanism is an MBD benchmark problem taken

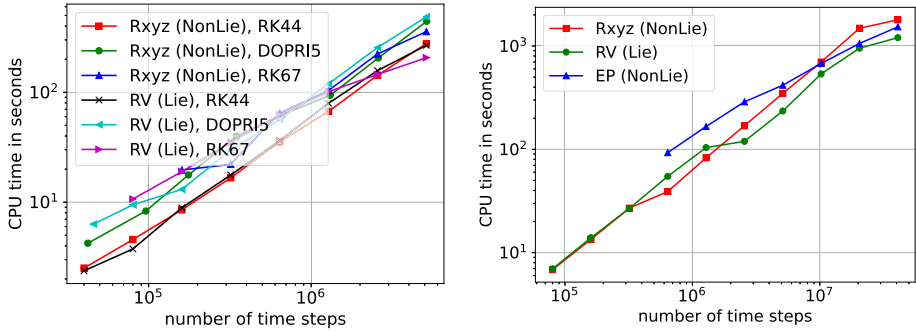


Fig. 12 CPU time of “high-speed rotor with flexible supports” for different rotation parameters and time integration methods. *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

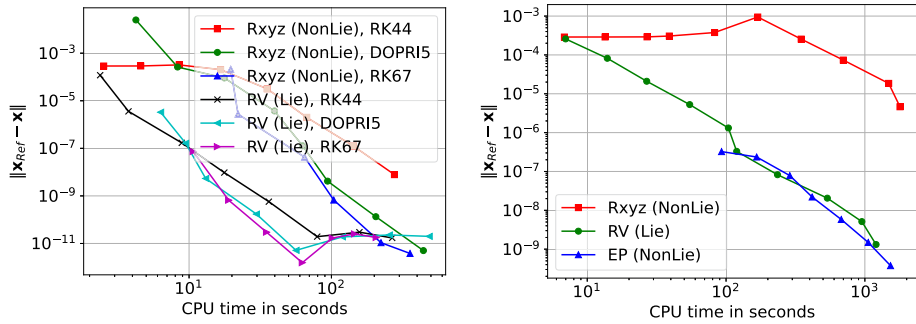
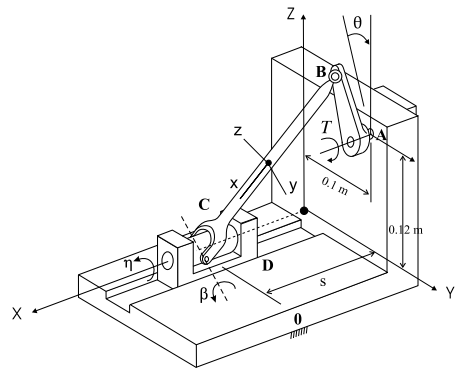


Fig. 13 Work–precision diagrams for explicit and implicit time integration methods for the example “high-speed rotor with flexible supports.” *Left plot*, explicit integration; *right plot*, implicit integration (generalized- α)

Fig. 14 Spatial rigid slider–crank mechanism [45]

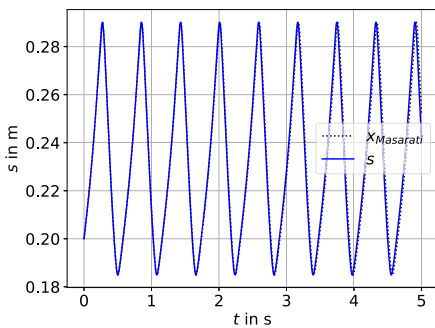


from the library of computational benchmark problems by IFToMM [44] and is illustrated in Fig. 14.

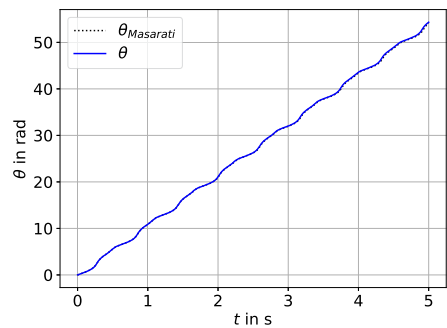
The spatial slider–crank mechanism consists of four rigid bodies: a slider, a crank, a connecting rod and the ground. The crank (AB) has a length of 0.08 m and is connected to the ground by a revolute joint at point A. The crank is driven from initial crank angle $\theta = 0$ rad

Table 8 Model parameters and initial conditions of the slider–crank mechanism. All quantities are expressed with respect to the center of mass of the respective rigid body

Parameter	Value	Notes
m_{cr} in kg	0.12	mass of the crank
m_{co} in kg	0.5	mass of the connecting rod
m_s in kg	2.0	mass of the slider
\mathbf{J}_{cr} in kg m ²	diag(0.0001, 0.00001, 0.0001)	crank inertia
\mathbf{J}_{co} in kg m ²	diag(0.0004, 0.004, 0.004)	connecting rod inertia
\mathbf{J}_s in kg m ²	diag(0.0001, 0.0001, 0.0001)	slider inertia
\mathbf{x}_{cr} in m	[0.0, 0.1, 0.12] ^T	initial position: crank
\mathbf{x}_{co} in m	[0.1, 0.05, 0.1] ^T	initial position: connecting rod
\mathbf{x}_s in m	[0.2, 0.0, 0.0] ^T	initial position: slider
θ in rad	0.0	initial crank angle
ρ_{co}^{Rxyz} in rad	[-0.46364761, 0.84106867, 0.0] ^T	initial attitude: connecting rod
$\dot{\mathbf{x}}_{co}$ in m/s	[0.12, -0.24, 0.0] ^T	initial velocity: connecting rod
$\dot{\mathbf{x}}_s$ in m/s	[0.24, 0.0, 0.0] ^T	initial velocity: slider
ω_{cr} in rad/s	[6.0, 0.0, 0.0] ^T	init. ang. vel.: crank
ω_{co} in rad/s	[1.92, -0.96, 0.48] ^T	init. ang. vel.: connecting rod



(a) Slider Position



(b) Crank Angle

Fig. 15 Time histories of the slider–crank’s slider position and crank angle. The reference solution ($s_{Masarati}$) was taken from the library of computational benchmark problems IFToMM [44]

with an initial angular velocity of 6 rad/s. The connecting rod has a length of 0.3 m and is connected to the crank by a spherical joint at point B and to the slider by a universal joint at point C. The slider is connected to the ground by a prismatic joint at point D with sliding displacement s . All links are subjected to gravity of magnitude 9.81 m/s^2 in the negative z -direction. Studying the dynamic response of the slider–crank mechanism under the gravitational force is the main objective of this benchmark problem. The model parameters and initial conditions are summarized¹⁵ in Table 8.

In Fig. 15, the time histories of the slider–crank’s slider position coordinate s (Fig. 15a) and crank angle θ (Fig. 15b) are compared with reference solutions provided by IFToMM

¹⁵The crank’s initial velocity $\dot{\mathbf{x}}_{AB}$ and the slider’s initial angular velocity ω_s are zero, i.e., $\dot{\mathbf{x}}_{AB} = \omega_s = \mathbf{0}$.

Fig. 16 Convergence in the norm of the maximum error of the slider–crank’s slider position for different rotation parameters and generalized- α

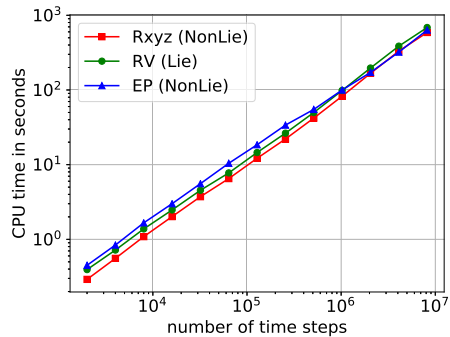
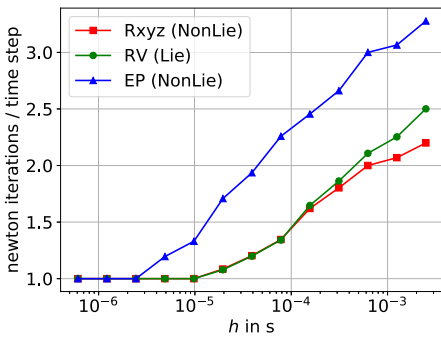
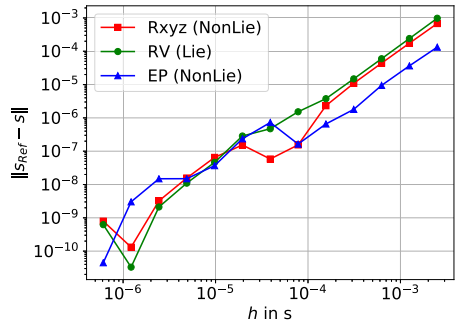


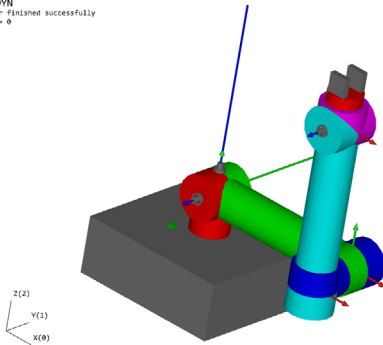
Fig. 17 Average number of Newton iterations per time step and CPU time of the slider–crank simulation for different rotation parameters and time implicit integration methods. *Left plot*, Newton iterations; *right plot*, CPU time

[44]. Figure 15 was obtained using the conventional Rxyz approach and generalized- α with time step size $h = 1.0 \times 10^{-5}$ s. As can be seen in Fig. 15, the time histories obtained with generalized- α and Euler parameters visually match the reference solution. The absolute deviation of the slider–crank’s slider position s at $t = 5$ s is approximately 8.6×10^{-3} m. The difference between our solution and the solution provided by IFToMM is due to the fact that different time integration methods were used; see [44].

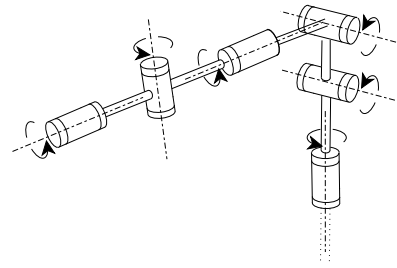
In Fig. 16, the convergence in the norm of the slider position error is shown for generalized- α and different formulations for decreasing values of the time step $h_n = (4.0 \times 10^{-3} \times 2^{(1-n)})_{n=1,2,\dots,13}$. Convergence is investigated at $t = 5$ s. The reference solution s_{Ref} for the convergence analysis was computed using the conventional approach Rxyz and generalized- α using a time step size of $h_{Ref} = 1.0 \times 10^{-6}$ s. As can be seen in Fig. 16, the conventional approaches are more accurate than the LGIM for time step sizes $h > 1.0 \times 10^{-4}$ s. For time step sizes $h \leq 1.0 \times 10^{-5}$ s, the LGIM appears to be partially slightly more accurate than both conventional approaches.

In Fig. 17 (left plot) the average number of Newton iterations per time step needed for the simulation of the slider–crank mechanism is illustrated. As can be seen in Fig. 17, the average number of Newton iterations per time step is lower for the conventional approach Rxyz compared to the LGIM and the Euler parameter approach. The CPU time needed for simulation of the slider–crank mechanism is shown in Fig. 17 (right plot). As expected from the lower number of Newton iterations required in the Rxyz formulation compared to the

EXUDYN
 Solver finished successfully
 time = 0



(a) Puma P560



(b) Kinematic Chain

Fig. 18 Schematic representation of the Puma P560 manipulator and its kinematic chain

Lie group and Euler parameter formulations, the CPU time of the Rxyz formulation turns out to be on average the lowest.

4.4 Puma 560 robot

Finally, we consider the Puma 560 robot (Programmable Universal Manipulator for Assembly), which is illustrated schematically in Fig. 18a. The Puma robot consists of six rigid bodies [46] and its kinematic chain with six revolute joints is illustrated in Fig. 18b. In total, the Puma robot has six degrees of freedom. The robot performs a point-to-point motion using constant joint acceleration profiles, whereby all links are subjected to gravity of magnitude 9.81 m/s^2 in the negative z -direction. The constant joint acceleration profiles are given in Table 9. The point-to-point motion starts with initial joint angles φ^{init} and aims to reach the goal configuration defined by the joint angles φ^{Goal} :

$$\varphi^{init} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \\ \varphi_6 \end{bmatrix} = \begin{bmatrix} 0.0^\circ \\ 22.5^\circ \\ 90.0^\circ \\ 0.0^\circ \\ 22.5^\circ \\ 0.0^\circ \end{bmatrix}, \quad \varphi^{Goal} = \begin{bmatrix} \varphi_1^{Goal} \\ \varphi_2^{Goal} \\ \varphi_3^{Goal} \\ \varphi_4^{Goal} \\ \varphi_5^{Goal} \\ \varphi_6^{Goal} \end{bmatrix} = \begin{bmatrix} -120.0^\circ \\ 60.0^\circ \\ -200.0^\circ \\ 90.0^\circ \\ 90.0^\circ \\ 45.0^\circ \end{bmatrix}. \quad (45)$$

In Fig. 19, the time histories of the robot's joint angles are shown, which were computed using the conventional Euler parameter approach and a time step size of $h = 1.0 \times 10^{-4} \text{ s}$. The Puma P560 robot has been well studied and its parameters are very well known [46]. The standard Denavit–Hartenberg parameters, as well as the physical parameters of the Puma robot, can be found in many textbooks on robotics, such as [46, 47] for example, and are omitted here due to space reasons. In each joint, a PD controller, which is implemented in the simulation model as a spring–damper element with control parameters specified in Table 10, ensures that the prescribed trajectory is followed with minimal error. In the following, we compare the accuracy and computational performance of the LGIM and both conventional approaches with a minimal coordinate formulation [48], which is abbreviated as “Min. Coord.” hereafter.

Table 9 Values of constant angular acceleration profiles used for the point-to-point motion

Link	$\ddot{\psi} \leq t = 0.5$	$\ddot{\psi} > t = 0.5$
1	-8.3776	8.3776
2	4.1888	-4.1888
3	-13.9626	13.9626
4	6.2832	-6.2832
5	6.2832	-6.2832
6	3.1416	-3.1416

Fig. 19 Joint angles

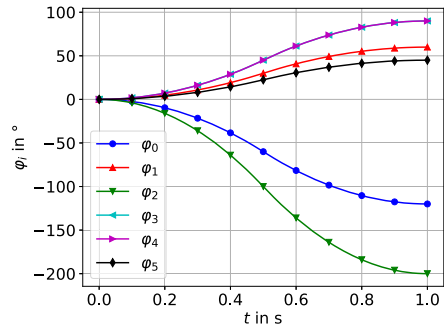
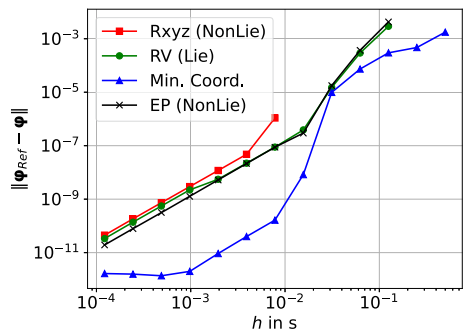


Table 10 PD controller parameters of the Puma P560 manipulator

Link	Proportional (P) term	Derivative (D) term
1	40000.0	400.0
2	40000.0	400.0
3	40000.0	100.0
4	100.0	1.0
5	100.0	1.0
6	10.0	0.1

Fig. 20 Convergence in the norm of the maximum error of the robot's joint angles for different formulations and generalized- α



In Fig. 20, the convergence in the norm of the robot's joint angles is illustrated. The time step sizes used in the convergence analysis are calculated by $h_n = (1.0 \times 10^{-1} \times 2^{(1-n)})_{n=1,2,\dots,11}$. Convergence is investigated at $t = 1$ s and the reference solution φ_{Ref} was

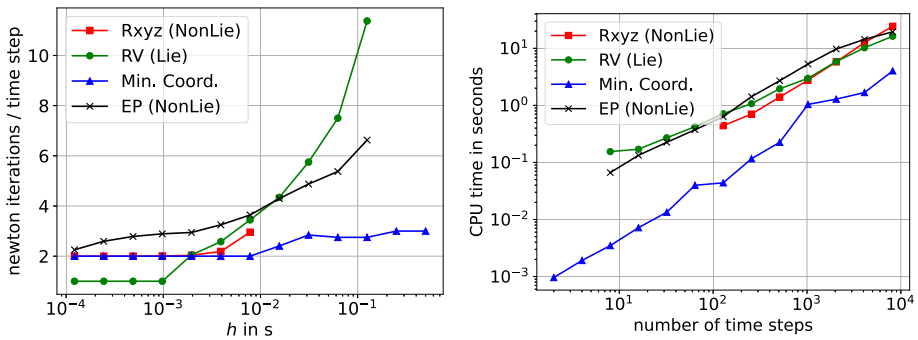


Fig. 21 Average number of Newton iterations per time step and CPU time required by the generalized- α method in the example “serial 6R robot” for different rotation parameters. *Left plot*, Newton iterations (full Newton); *right plot*, CPU time

computed by employing the minimal coordinate formulation and generalized- α using a time step size of $h_{Ref} = 1.0 \times 10^{-6}$ s. As can be seen in Fig. 20, the minimal coordinate formulation is the most accurate for all considered time step sizes. The classical Euler parameter approach and the LGIM exhibit for time step sizes $h > 1.56 \times 10^{-2}$ s virtually the same accuracy, while the classical Euler parameter approach is more accurate at smaller time step sizes. The conventional Rxyz approach is the least accurate. Moreover, both the LGIM and the conventional Euler parameter approach can simulate the Puma at almost the largest considered time step sizes, in contrast to the conventional Rxyz approach. However, the minimal coordinate formulation outperforms both the LGIM and the conventional approaches in this regard. At this point we would like to note that the relative tolerance in “full Newton” had to be increased to 1.0×10^{-5} in order for Newton’s method to converge in the case of the LGIM.

In Fig. 21, the average number of Newton iterations per time step (left plot) and the CPU time (right plot) needed for simulation of the Puma robot are shown. As can be seen in Fig. 21, the LGIM achieves the lowest number of Newton iterations per time step for time step sizes $h \leq 1.0 \times 10^{-3}$ s, while the LGIM requires also the most Newton iterations per time step for time step sizes $h > 3.12 \times 10^{-2}$ s. Similar to the other numerical examples considered, the classical Rxyz approach requires on average fewer Newton iterations per time step than the conventional Euler parameter approach. These observations are reflected in the CPU time required; see Fig. 21 (left plot). It is worth noting that the minimal coordinate formulation achieves the lowest CPU time, which is due to the fact that only six EOMs need to be treated in the minimal coordinate formulation, while in the LGIM and both conventional approaches $6 \times (3 + l)$ equilibrium equations (1) and 6×5 holonomic constraints (3) need to be treated.

5 Conclusion

The accuracy and computational performance of explicit and implicit LGIMs have been compared with classical Euler angle and Euler parameter-based formulations using a set of rigid body systems. It has been found that explicit LGIMs outperform classical parameterization-based formulations in terms of accuracy, while the computational efficiency is almost the same. Especially for systems with high rotational speeds, explicit

LGIMs turn out to be more accurate than the classical formulations. In contrast, in the case of implicit integration, the classical Euler parameter-based formulation outperforms both the classical Euler angle formulation and the LGIM in terms of accuracy. The LGIM appears to be computationally more efficient, as the number of Newton iterations required in each time step is lower compared to the classical formulations. The number of Newton iterations may be further reduced in case of changing from the $\mathbb{R}^3 \times SO(3)$ formulation to the $SE(3)$ formulation, which reduces nonlinearities in equations for one integration step. Furthermore, it has been illustrated how LGIMs can be implemented into an MBS formalism that is based on absolute coordinates. The implementation is shown for formalisms that employ explicit RK methods and the implicit generalized- α method. It turns out that LGIMs can be implemented at almost no extra cost, i.e., it takes only a few changes in the classical integration algorithms to convert them into LGIMs. The straightforward transition from classical integration methods to LGIMs is facilitated by the fact that LGIMs readily allow three rotation parameters to be used for modeling MBSs, which makes the dimension of the residual and the iteration matrix coincide with that of the absolute coordinates and thus existing data structures might be used. Up to now, we restricted our investigations to the $\mathbb{R}^3 \times SO(3)$ formulation. Future work will address the evaluation of the $SE(3)$ formulation in terms of accuracy and computational performance compared to classical formulations using a set of rigid and flexible MBSs.

Appendix A: Implementation details

A.1 Tangent operators

The tangent operator $\mathbf{T}_{\text{exp}} \in \mathbb{R}^{6 \times 6}$ on $\mathbb{R}^3 \times SO(3)$ and its inverse $\mathbf{T}_{\text{exp}}^{-1} \in \mathbb{R}^{6 \times 6}$ are given by [3, 49]

$$\mathbf{T}_{\text{exp}}(\Delta \mathbf{q}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{\text{exp}_{SO(3)}}(\Delta \boldsymbol{\psi}) \end{bmatrix}, \quad \mathbf{T}_{\text{exp}}^{-1}(\Delta \mathbf{q}) = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{\text{exp}_{SO(3)}}^{-1}(\Delta \boldsymbol{\psi}) \end{bmatrix}. \quad (\text{A1})$$

In Eq. (A1), the matrices $\mathbf{T}_{\text{exp}_{SO(3)}} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{T}_{\text{exp}_{SO(3)}}^{-1} \in \mathbb{R}^{3 \times 3}$ are the tangent operators on $SO(3)$ and read [2, 41]

$$\mathbf{T}_{\text{exp}_{SO(3)}}(\Delta \boldsymbol{\psi}) = \begin{cases} \mathbf{I} & \text{if } \|\Delta \boldsymbol{\psi}\| = 0, \\ \mathbf{I} + f_1(\|\Delta \boldsymbol{\psi}\|)\widetilde{\Delta \boldsymbol{\psi}} + f_2(\|\Delta \boldsymbol{\psi}\|)\widetilde{\Delta \boldsymbol{\psi}}^2 & \text{else,} \end{cases} \quad (\text{A2})$$

$$\mathbf{T}_{\text{exp}_{SO(3)}}^{-1}(\Delta \boldsymbol{\psi}) = \begin{cases} \mathbf{I} & \text{if } \|\Delta \boldsymbol{\psi}\| = 0, \\ \mathbf{I} + \frac{1}{2}\widetilde{\Delta \boldsymbol{\psi}} + f_3(\|\Delta \boldsymbol{\psi}\|)\widetilde{\Delta \boldsymbol{\psi}}^2 & \text{else,} \end{cases} \quad (\text{A3})$$

where

$$f_1(\|\Delta \boldsymbol{\psi}\|) = \begin{cases} \frac{\cos \|\Delta \boldsymbol{\psi}\| - 1}{\|\Delta \boldsymbol{\psi}\|^2} & \text{if } \|\Delta \boldsymbol{\psi}\| > 0, \\ -\frac{1}{2} & \text{if } \|\Delta \boldsymbol{\psi}\| = 0, \end{cases} \quad (\text{A4})$$

$$f_3(\|\Delta \boldsymbol{\psi}\|) = \begin{cases} \frac{\|\Delta \boldsymbol{\psi}\| - \sin \|\Delta \boldsymbol{\psi}\|}{\|\Delta \boldsymbol{\psi}\|^3} & \text{if } \|\Delta \boldsymbol{\psi}\| > 0, \\ \frac{1}{6} & \text{if } \|\Delta \boldsymbol{\psi}\| = 0, \end{cases} \quad (\text{A5})$$

$$f_3(\|\Delta\psi\|) = \begin{cases} \frac{1 - \frac{\|\Delta\psi\|}{2} \cot \frac{\|\Delta\psi\|}{2}}{\|\Delta\psi\|^2} & \text{if } \|\Delta\psi\| > 0, \\ \frac{1}{12} & \text{if } \|\Delta\psi\| = 0. \end{cases} \quad (\text{A6})$$

A.2 Handling loss of significance in \mathbf{T}_{exp} and $\mathbf{T}_{\text{exp}}^{-1}$

There is a loss of significance when Eqs. (A4)–(A6) are evaluated near $\|\Delta\psi\| = 0$ on a computer [41]. In order to obtain accurate and precise results from Eqs. (A4)–(A6) for all reasonable values of $\|\Delta\psi\|$, we approximate Eqs. (A4)–(A6) by Taylor polynomials. We choose the intervals where Eqs. (A4)–(A6) are evaluated by its Taylor approximation as proposed in [41]. The improved versions of Eqs. (A4)–(A6) read

$$f_1(\|\Delta\psi\|) = \begin{cases} \frac{\cos \|\Delta\psi\| - 1}{\|\Delta\psi\|^2} & \text{if } \|\Delta\psi\| \geq 1 \times 10^{-2}, \\ -\frac{1}{2} + \frac{\|\Delta\psi\|^2}{24} - \frac{\|\Delta\psi\|^4}{720} & \text{if } 0 \leq \|\Delta\psi\| < 1 \times 10^{-2}, \end{cases} \quad (\text{A7})$$

$$f_2(\|\Delta\psi\|) = \begin{cases} \frac{\|\Delta\psi\| - \sin \|\Delta\psi\|}{\|\Delta\psi\|^3} & \text{if } \|\Delta\psi\| \geq 1 \times 10^{-4}, \\ \frac{1}{6} - \frac{\|\Delta\psi\|^2}{120} + \frac{\|\Delta\psi\|^4}{5040} & \text{if } 0 \leq \|\Delta\psi\| < 1 \times 10^{-4}, \end{cases} \quad (\text{A8})$$

$$f_3(\|\Delta\psi\|) = \begin{cases} \frac{1 - \frac{\|\Delta\psi\|}{2} \cot \frac{\|\Delta\psi\|}{2}}{\|\Delta\psi\|^2} & \text{if } \|\Delta\psi\| \geq 1 \times 10^{-2}, \\ \frac{1}{12} + \frac{\|\Delta\psi\|^2}{720} + \frac{\|\Delta\psi\|^4}{30240} & \text{if } 0 \leq \|\Delta\psi\| < 1 \times 10^{-2}, \end{cases} \quad (\text{A9})$$

which give an absolute error in the Taylor approximation of about 1×10^{-16} and a relative error in the actual function of about 1×10^{-10} .

A.3 Composition operation for rotation vectors

We write the composition operation \diamond for two rotation vectors ψ_0 and $\Delta\psi$ with ψ being the resultant rotation vector as

$$\psi = \psi_0 \diamond \Delta\psi = \frac{v_0 \eta + \eta_0 \mathbf{v} + \mathbf{v} \times \eta}{2 \operatorname{sinc} \varphi}, \quad (\text{A10})$$

where

$$\eta_0 = \cos \frac{\|\psi_0\|}{2}, \quad \eta = \frac{\psi_0}{2} \operatorname{sinc} \frac{\|\psi_0\|}{2}, \quad (\text{A11})$$

$$v_0 = \cos \frac{\|\Delta\psi\|}{2}, \quad \mathbf{v} = \frac{\Delta\psi}{2} \operatorname{sinc} \frac{\|\Delta\psi\|}{2} \quad (\text{A12})$$

and

$$\varphi = \operatorname{acos}(v_0 \eta_0 - \mathbf{v}^T \eta); \quad (\text{A13})$$

cf. [11, 33]. In Eqs. (A10)–(A13), sinc represents the cardinal sine function

$$\operatorname{sinc}(x) = \begin{cases} 1 & \text{if } x = 0, \\ \frac{\sin x}{x} & \text{else,} \end{cases} \quad (\text{A14})$$

which is continuous and computable at $x = 0$ [49]. The composition operation Eq. (A10) enables a singularity-free update of the rotation vector and can be evaluated without restrictions [10, 11].

Appendix B: Automatic step size control

In this paper, we use one and the same approach for automatic step size control for formulations 1 and 2 as well as for the Lie group formulation. To this end, we define the solution vector for explicit solvers as

$$\xi = [\mathbf{q}^T \quad \mathbf{v}^T]^T. \tag{B15}$$

Following [37], we estimate the error of a time step with current step size h by using an embedded RK formula, which includes two approximations of order p and $\hat{p} = p - 1$. The approximations are obtained by using two different integration formulas with common coefficients c_i , but two sets of weights b_i and \hat{b}_i , leading to two approximations ξ and $\hat{\xi}$. These so-called embedded RK formulas are widely used; see for example Hairer et al. [34].

The approximations ξ and $\hat{\xi}$ are used to estimate an error

$$e_j = |\xi_j - \hat{\xi}_j| \tag{B16}$$

for every component j of the solution vector ξ . A scaling e_j is used for every component of the solution vector, evaluating at the beginning (0) and end (1) of the time step

$$s_j = a_{tol} + r_{tol} \cdot \max(|\xi_{0j}|, |\xi_{1j}|). \tag{B17}$$

In Eq. (B17), a_{tol} and r_{tol} represent the absolute respectively relative tolerance for the error control, both prescribed by the user. The relative, scaled, scalar error for the step, which needs to fulfill $err \leq 1$, is computed as

$$err = \sqrt{\frac{1}{n} \sum_{j=1}^n \left(\frac{\xi_{1j} - \hat{\xi}_{1j}}{s_j} \right)^2}. \tag{B18}$$

The optimal time step size then reads

$$h_{opt} = h \cdot \left(\frac{1}{err} \right)^{\frac{1}{p+1}}. \tag{B19}$$

Currently we use the suggested step size as

$$h_{new} = \min(\min(h_{max}, h \cdot f_{maxInc}), \max(h_{min}, f_{sfty} \cdot h_{opt})) \tag{B20}$$

with the minimum and maximum step size h_{max}

$$h_{max} = \frac{t_{end} - t_{start}}{n_{steps}} \tag{B21}$$

respectively h_{min} , both prescribed by the user. The factor f_{maxInc} limits the increase of the current step size h , and the factor f_{sfty} is a safety factor for limiting the chosen step size relative to the optimal one in order to avoid frequent step rejections. If $h_{opt} \geq h$, the current step is accepted, otherwise the step is recomputed with h_{new} . For more details, see [37].

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under Marie Skłodowska-Curie grant agreement No. 860124. This publication reflects only the authors' view and the Research Executive Agency is not responsible for any use that may be made of the information it contains.

Author contributions Stefan Holzinger wrote the main manuscript text and prepared all figures. Johannes Gerstmayr implemented all time integration methods in Exudyn with contributions from Stefan Holzinger. Stefan Holzinger and Johannes Gerstmayr set up all the numerical examples. Martin Arnold provided minor additions to the literature study as well as contributions to notation and numerical algorithms. All authors reviewed the manuscript.

Funding Open access funding provided by University of Innsbruck and Medical University of Innsbruck.

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Brüls, O., Cardona, A.: On the use of Lie group time integrators in multibody dynamics. *J. Comput. Nonlinear Dyn.* **5**(3), 1–13 (2010). <https://doi.org/10.1115/1.4001370>
2. Brüls, O., Arnold, M., Cardona, A.: Two Lie group formulations for dynamic multibody systems with large rotations. In: Proceedings of IDETC/MSNDC 2011, ASME 2011 International Design Engineering Technical Conferences, Washington, USA (2011). <https://doi.org/10.1115/DETC2011-48132>
3. Terze, Z., Müller, A., Zlatar, D.: Singularity-free time integration of rotational quaternions using non-redundant ordinary differential equations. *Multibody Syst. Dyn.* **38**(3), 201–225 (2016). <https://doi.org/10.1007/s11044-016-9518-7>
4. Géradin, M., Cardona, A.: *Flexible Multibody Dynamics: A Finite Element Approach* 344 p. Wiley, New York (2001)
5. Singla, P., Mortari, D., Junkins, J.L.: How to avoid singularity when using Euler angles? *Adv. Astronaut. Sci.* **119**(SUPPL), 1409–1426 (2005)
6. Müller, A.: Singularity-free Lie group integration and geometrically consistent evaluation of multibody system models described in terms of standard absolute coordinates. *J. Comput. Nonlinear Dyn.* **17**(5), 1–7 (2022). <https://doi.org/10.1115/1.4053368>
7. Simo, J.C., Vu-Quoc, L.: On the dynamics in space of rods undergoing large motions - a geometrically exact approach. *Comput. Methods Appl. Mech. Eng.* **66**(2), 125–161 (1988). [https://doi.org/10.1016/0045-7825\(88\)90073-4](https://doi.org/10.1016/0045-7825(88)90073-4)
8. Hante, S., Arnold, M.: RATTLie: a variational Lie group integration scheme for constrained mechanical systems. *J. Comput. Appl. Math.* **387**, 112492 (2021). <https://doi.org/10.1016/j.cam.2019.112492>
9. Arnold, M., Cardona, A., Brüls, O.: A Lie algebra approach to Lie group time integration of constrained systems. In: Betsch, P. (ed.) *Structure-Preserving Integrators in Nonlinear Structural Dynamics and Flexible Multibody Dynamics*, pp. 91–158. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31879-0_3
10. Holzinger, S., Gerstmayr, J.: Explicit time integration of multibody systems modelled with three rotation parameters. In: Proceedings of the ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE2020, August 17–19. ASME, Virtual, Online (2020)
11. Holzinger, S., Gerstmayr, J.: Time integration of rigid bodies modelled with three rotation parameters. *Multibody Syst. Dyn.* **53**, 345–378 (2021). <https://doi.org/10.1007/s11044-021-09778-w>

12. Arnold, M., Hante, S.: Implementation details of a generalized- α differential-algebraic equation Lie group method. *J. Comput. Nonlinear Dyn.* **12**, 1–8 (2017). <https://doi.org/10.1115/1.4033441>
13. Holzinger, S., Schöberl, J., Gerstmayr, J.: The equations of motion for a rigid body using non-redundant unified local velocity coordinates. *Multibody Syst. Dyn.* **48**(3), 283–309 (2020). <https://doi.org/10.1007/s11044-019-09700-5>
14. Sonnevile, V.: A geometric local frame approach for flexible multibody systems. PhD thesis, University of Liège (2015)
15. Sonnevile, V., Brüls, O.: A formulation on the special Euclidean group for dynamic analysis of multibody systems. *J. Comput. Nonlinear Dyn.* **9**(4), 041002 (2014). <https://doi.org/10.1115/1.4026569>
16. Müller, A., Terze, Z.: The significance of the configuration space Lie group for the constraint satisfaction in numerical time integration of multibody systems. *Mech. Mach. Theory* **82**, 173–202 (2014)
17. Müller, A., Terze, Z.: Is there an optimal choice of configuration space for Lie group integration schemes applied to constrained MBS? In: Proceedings of the ASME Design Engineering Technical Conference & Computers and Information in Engineering Conference IDETC/CIE 2013, August 12–15, 2013. ASME, Portland (2013). <https://doi.org/10.1115/DETC2013-12151>
18. Sonnevile, V., Cardona, A., Brüls, O.: Geometrically exact beam finite element formulated on the special Euclidean group SE(3). *Comput. Methods Appl. Mech. Eng.* **268**, 451–474 (2014). <https://doi.org/10.1016/j.cma.2013.10.008>
19. Munthe-Kaas, H.: Runge-Kutta methods on Lie groups. *BIT Numer. Math.* **38**(111038), 92–111 (1998). <https://doi.org/10.1007/BF02510919>
20. Munthe-Kaas, H.: High order Runge-Kutta methods on manifolds. *Appl. Numer. Math.* **29**(1), 115–127 (1999). [https://doi.org/10.1016/S0168-9274\(98\)00030-0](https://doi.org/10.1016/S0168-9274(98)00030-0)
21. Faltinsen, S., Marthinsen, A., Munthe-Kaas, H.Z.: Multistep methods integrating ordinary differential equations on manifolds. *Appl. Numer. Math.* **39**(3–4), 349–365 (2001). [https://doi.org/10.1016/S0168-9274\(01\)00103-9](https://doi.org/10.1016/S0168-9274(01)00103-9)
22. Wieloch, V., Arnold, M.: BDF integrators for constrained mechanical systems on Lie groups. *J. Comput. Appl. Math.* **387**, 112517 (2021). <https://doi.org/10.1016/j.cam.2019.112517>
23. Brüls, O., Cardona, A., Arnold, M.: Lie group generalized- α time integration of constrained flexible multibody systems. *Mech. Mach. Theory* **48**(1), 121–137 (2012). <https://doi.org/10.1016/j.mechmachtheory.2011.07.017>
24. Arnold, M., Brüls, O., Cardona, A.: Error analysis of generalized- α Lie group time integration methods for constrained mechanical systems. *Numer. Math.* **129**, 149–179 (2015)
25. Chung, J., Hulbert, G.M.: A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized- α method. *J. Appl. Mech.* **60**(2), 371 (1993). <https://doi.org/10.1115/1.2900803>
26. Arnold, M., Brüls, O.: Convergence of the generalized- α scheme for constrained mechanical systems. *Multibody Syst. Dyn.* **18**, 185–202 (2007). <https://doi.org/10.1007/s11044-007-9084-0>
27. Müller, A.: Singularity-free Lie group integration of multibody system models described in absolute coordinates. In: Proceedings of the ASME 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 9: 17th International Conference on Multibody Systems, Nonlinear Dynamics, and Control (MSNDC). Virtual, On (2021). <https://doi.org/10.1115/DETC2021-68186>
28. Terze, Z., Müller, A., Zlatar, D.: Lie-group integration method for constrained multibody systems in state space. *Multibody Syst. Dyn.* **34**(3), 275–305 (2015). <https://doi.org/10.1007/s11044-014-9439-2>
29. Celledoni, E., Çokaj, E., Leone, A., Murari, D., Owren, B.: Lie group integrators for mechanical systems. *Int. J. Comput. Math.* **99**(1), 58–88 (2022). <https://doi.org/10.1080/00207160.2021.1966772>. [arXiv: 2102.12778](https://arxiv.org/abs/2102.12778)
30. Zhou, P., Ren, H.: Stabilized explicit integrators for local parametrization in multi-rigid-body system dynamics. *J. Comput. Nonlinear Dyn.* **17**(10), 1–11 (2022). <https://doi.org/10.1115/1.4054801>
31. Terze, Z., Müller, A., Zlatar, D.: DAE index 1 formulation for multibody system dynamics in Lie-group setting. In: Proceedings of the 2nd Joint International Conference on Multibody System Dynamics, May 29–June 1, Stuttgart, Germany (2012). 2012
32. Park, J., Chung, W.K.: Geometric integration on Euclidean group with application to articulated multibody systems. *IEEE Trans. Robot.* **21**(5), 850–863 (2005). <https://doi.org/10.1109/TRO.2005.852253>
33. Condrache, D., Ciureanu, I.A.: Baker-Campbell-Hausdorff-Dynkin formula for the Lie algebra of rigid body displacements. *Mathematics* **8**(7) (2020). <https://doi.org/10.3390/math8071185>
34. Hairer, E., Lubich, C., Wanner, G.: *Geometric Numerical Integration*, 2nd edn. 644 p. Springer, Heidelberg (2006). <https://doi.org/10.1007/3-540-30666-8>
35. Bogacki, P., Shampine, L.F.: A 3(2) pair of Runge-Kutta formulas. *Appl. Math. Lett.* **2**(4), 321–325 (1989). [https://doi.org/10.1016/0893-9659\(89\)90079-7](https://doi.org/10.1016/0893-9659(89)90079-7)

36. Hairer, E., Nørsett, S., Wanner, G.: Solving Ordinary Differential Equations I, Nonstiff Problems 528 p. Springer, Heidelberg (1987). <https://doi.org/10.1007/978-3-540-78862-1>
37. Gerstmayr, J.: Exudyn – a C++ based Python package for flexible multibody systems. In: The 6th Joint International Conference on Multibody System Dynamics and the 10th Asian Conference on Multibody System Dynamics, New Delhi, India (2022)
38. Bottasso, C.L., Dopico, D.D.: On the optimal scaling of index three DAEs in multibody dynamics. In: Proc. of the European Conference on Computational Mechanics (ECCOMAS-ECCM), Lisbon, Portugal (2006)
39. Arnold, M., Brüls, O., Cardona, A.: Convergence analysis of generalized- α Lie group integrators for constrained systems. In: Proceedings of Multibody Dynamics ECCOMAS Conference (2011)
40. Shampine, L.F.: Evaluation of implicit formulas for the solution of ODEs. BIT Numer. Math. **19**(4), 495–502 (1979). <https://doi.org/10.1007/BF01931266>
41. Hante, S.: Geometric integration of a constrained cosserat beam model. PhD thesis, Martin Luther University Halle-Wittenberg (2022). <https://doi.org/10.25673/91397>
42. Nikravesh, P.E.: Computer-Aided Analysis of Mechanical Systems 370 p. Prentice Hall, New York (1988)
43. Gerstmayr, J.: Exudyn – a C++-based Python package for flexible multibody systems. Multibody Syst. Dyn. (2023). <https://doi.org/10.1007/s11044-023-09937-1>
44. IFToMM Technical Committee for Multibody Dynamics: Library of Computational Benchmark Problems (2015)
45. Masoudi, R.: Spatial rigid slider-Crank mechanism. Taken from “Library of Computational Benchmark Problems” (2022). https://www.iftoMM-multibody.org/benchmark/problem/Spatial_rigid_slider-Crank_mechanism/
46. Corke, P.I.: Robotics, Vision and Control, 2nd edn. 570 p. Springer, Cham (2011). <https://doi.org/10.1007/978-3-319-54413-7>
47. Kim, H.Y., Streit, D.A.: Configuration dependent stiffness of the PUMA 560 manipulator: analytical and experimental results. Mech. Mach. Theory **30**(8), 1269–1277 (1995). [https://doi.org/10.1016/0094-114X\(95\)00017-S](https://doi.org/10.1016/0094-114X(95)00017-S)
48. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics 2nd edn. 2228 p. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-32552-1>
49. Müller, A.: Coordinate mappings for rigid body motions. J. Comput. Nonlinear Dyn. **12**(2) (2017). <https://doi.org/10.1115/1.4034730>

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.