Sensitivity Analysis for Dynamic Mechanical Systems with Finite Rotations

Olivier Brüls[1] and Peter Eberhard[2]

[1]Department of Aerospace and Mechanical Engineering - University of Liège
Chemin des Chevreuils, 1, B52, 4000 Liège, Belgium
e-mail: o.bruls@ulg.ac.be

[2] Institute of Engineering and Computational Mechanics, University of Stuttgart
Pfaffenwaldring 9, 70569 Stuttgart, Germany
e-mail: eberhard@itm.uni-stuttgart.de

Abstract

This paper presents a sensitivity analysis for dynamic systems with large rotations using a semi-analytical direct differentiation technique. The choice of a suitable time integration strategy for the rotation group appears to be critical for the development of an efficient sensitivity analysis. Three versions of the generalized-α time integration scheme are considered: a residual form, a linear form, and a geometric form. Their consistency is discussed, and we show that the residual form, which is the most direct extension of the generalized-α algorithm defined in structural dynamics, should not be used for problems with large rotations. The sensitivity analysis is performed and close connections are highlighted between the structure of the sensitivity equations and of the linearized dynamic equations. Hence, algorithms developed for the original problem can be directly reused for the sensitivities. Finally, a numerical example is analysed in detail.

Key-words: sensitivity analysis, finite rotation, Lie group, generalized-α scheme, geometric integration.

1 Introduction

Gradient-based optimisation algorithms require efficient algorithms to compute the sensitivities of the simulation results with respect to design parameters. For example, the finite difference technique is based on repeated simulations with disturbed values of each design parameter. Besides its inaccuracy, this method quickly becomes inefficient for large-scale dynamic problems with many design parameters. In order to improve the efficiency, an alternative approach consists in including the sensitivity analysis into the simulation code, so that the sensitivities are computed together with the original solution in a single but extended simulation.

Automatic differentiation [18] provides reliable tools to modify a simulation code for this purpose. However, in some cases, this method may lack computational efficiency. For example, while a Newton procedure converges to high precision, the code generated by automatic differentiation computes the sensitivities at each iteration, which clearly represents a waste of effort. A manual optimisation of the efficiency is thus needed, but this is not a trivial task since the operations for the computation of the dynamic response and of the sensitivities are tightly interlaced in the code generated by automatic differentiation.

Alternatively, semi-analytical methods lead to efficient algorithms that are manually implemented in the simulation code. Typically, those formulations are based either on the direct

differentiation method or on the adjoint variable method. At a first glance, the manual implementation of sensitivity analysis in a very complex software may appear as a tedious and error prone process. However, a thorough understanding of the underlying formulation usually allows to simplify drastically the operations. In particular, for nicely formulated time integrators, the sensitivity of the numerical response can be computed using the same algorithm as for the original problem. Since the sensitivity equations closely resemble the linearized equations that are already implemented for the purpose of Newton iterations, the additional effort required to implement a sensitivity analysis turns out to be quite small. Semi-analytical approaches are already mature in structural dynamics, see [22, 23, 25, 27, 34, 35], as well as in multibody dynamics, see [6, 7, 17, 36]. In contrast, several difficulties arise for models involving the representation of absolute 3D rotations, such as rigid bodies [8, 33], geometrically exact beams [12, 24, 32], shells [4, 31], or flexible multibody systems [3, 11, 21].

Firstly, the design of integrators for such problems still represents a major challenge. Indeed, no global parameterization is available for arbitrary large rotations, so that three approaches are generally possible. One may use a *local parameterization* with a standard time integrator, and reparameterize the motion when appropriate, see Cardona and Géradin [11, 13]. Another strategy is to use a *global but redundant parameterization*, e.g. the four Euler parameters or, as suggested by Betsch and Steinmann [8], the nine components of the director vectors, and to treat the problem as a DAE (this approach will not be considered here). Finally, one can use a *geometric integrator*, which is designed so that the numerical solution inherently remains on the manifold, but which does not require the formulation of the equations of motion as differential equations in local independent coordinates, see Simo and co-authors [32, 33], Bottasso and Borri [10], Bauchau and Bottasso [3], Ibrahimbegovic and Mamouri [24], as well as the Lie group integrators proposed by Crouch and Grossmann [16], Munthe-Kaas [28], and their implementation for rigid body systems by Celledoni and Owren [14]. Among those contributions, let us remark that references [3, 8, 10, 24, 33] address the design of energy conserving or energy decaying schemes.

Secondly, for the group of rotations $SO(3)$, the geometric nature of a sensitivity, i.e. a tangent vector to the manifold, is quite different from the nature of the rotation itself, so that any attempt to use the same integrator for both objects may a priori sound inconsistent. In addition, sensitivities at two successive rotations cannot be directly combined since they belong to different tangent spaces. Tools from differential geometry and Lie group theory are required to define meaningful algorithms, see e.g. [9]. Actually, those tools are also used in geometrically consistent formulations of solid mechanics, and we will show that unified treatments can still be used for the simulation of the dynamic system and for the sensitivity analysis.

Hence, the present paper demonstrates that despite the difficulties inherent to the transient analysis of systems with large rotations, the semi-analytical direct differentiation method allows an accurate computation of the sensitivities of the simulation results. However, the choice of a suitable time integration strategy has major consequences for the development of an efficient sensitivity analysis. Three representative integrators are treated here, which are extensions of the standard generalized-α algorithm to systems with finite rotations. The first two algorithms, namely the residual form and the linear form, are defined in coordinates and require a reparameterization strategy. We show that the residual form, which is the most direct extension of the classical generalized-α scheme defined in structural dynamics, is not second-order accurate when applied to systems with large rotations. In contrast, the linear form, whose construction is based on an analogy with linear multistep methods, is known to be second-order accurate for a wide class of nonlinear dynamic problems [2]. The third algorithm, which has also some similarities with linear multistep methods, is a geometric extension of the generalized-α scheme. Those three algorithms are general and thus applicable to any kind of mechanical elements, such as rigid bodies, beams or shells. Let us note that the present developments could be reproduced without difficulty for other interesting algorithms, such as the nonlinear formulation of the generalized-α method as a generalized mid-point rule, which has been exploited by Kuhl and Crisfield [26].

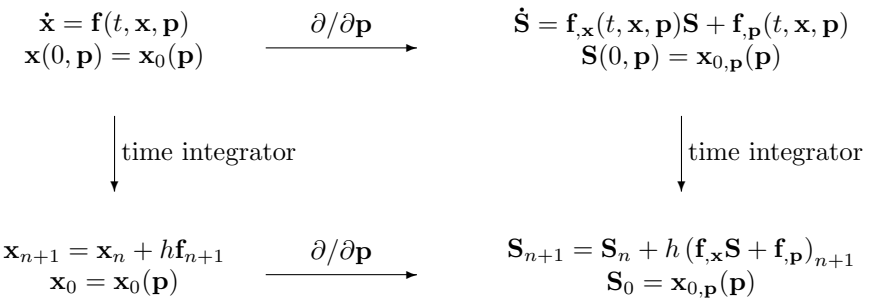

Figure 1: Commutativity of time integration and differentiation (here: implicit Euler). The notation $\mathbf{f}_{n+1}$ means $\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})$.

The paper is organized as follows. Section 2 presents the direct differentiation method for semi-analytical sensitivity analysis of dynamic systems in the vector space $\mathbb{R}^n$, whereas the remaining part of the paper focuses on the treatment of finite rotations. Section 3 describes the basic properties of the group of rotations $SO(3)$. Section 4 develops the dynamic equations of the rigid body, whereas the time integration problem is discussed in Section 5. An example is analysed in Section 6.

2 Direct differentiation for sensitivity analysis in $\mathbb{R}^n$

Let us consider the initial value problem

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{p}), \qquad \mathbf{x}(0, \mathbf{p}) = \mathbf{x}_0(\mathbf{p}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector and $\mathbf{p} \in \mathbb{R}^{n_p}$ is a vector of design parameters. The objective is to compute the solution $\mathbf{x}(t, \mathbf{p})$ as well as the sensitivity matrix

$$\mathbf{S}(t, \mathbf{p}) = \frac{\partial \mathbf{x}(t, \mathbf{p})}{\partial \mathbf{p}}. \tag{2}$$

As illustrated in Figure 1 for the implicit Euler algorithm, the algorithm for the sensitivities can be defined in two equivalent ways: (i) differentiate Equation (1), and integrate the resulting sensitivity equation in time, (ii) integrate Equation (1) in time, and differentiate the numerical solution. The commutativity of this diagram, which is also observed for most properly formulated algorithms, is a consequence of the commutativity of second-order derivatives. As described in Algorithm 1, the code developed for the original problem can be largely reused for the sensitivity analysis performed at the end of the time step. Indeed, using the notations $\mathbf{f}_{,\mathbf{x}} = \partial\mathbf{f}/\partial\mathbf{x}$ and $\mathbf{f}_{,\mathbf{p}} = \partial\mathbf{f}/\partial\mathbf{p}$, the discretized sensitivity equation

$$\mathbf{S}_{n+1} = \mathbf{S}_n + h\left(\mathbf{f}_{,\mathbf{x}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})\mathbf{S}_{n+1} + \mathbf{f}_{,\mathbf{p}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})\right) \tag{3}$$

is a linear equation, that is solved in one single Newton iteration using the same Jacobian matrix $\mathbf{J}$ as for the original problem. However, while an *approximate* expression of the Jacobian matrix is often sufficient to achieve convergence for the original problem, an *exact* expression is necessary to solve the sensitivity problem in one iteration. Let us note that an approximate Jacobian matrix, e.g. the Jacobian matrix computed for the last Newton iteration of the original problem, could still be used provided that an iterative scheme is implemented for the sensitivity problem [25]. In any case, the residual $\mathbf{r}_{,\mathbf{p}}$ associated with Equation (3) requires an

accurate computation of the homogeneous term $\mathbf{f}_{,\mathbf{x}}\mathbf{S}$ and of the pseudo-load $\mathbf{f}_{,\mathbf{p}}$. The pseudo-load is often evaluated according to a finite difference scheme, but the homogeneous term can be implemented analytically in the numerical code. For that purpose, the computation of the full matrix $\mathbf{f}_{,\mathbf{x}}$ followed by the multiplication with the sensitivity matrix $\mathbf{S}$ may demand important memory and CPU resources, especially for large scale problems. Alternatively, the contribution $\mathbf{f}_{,\mathbf{x}}\mathbf{S}$ can be implemented more efficiently at the element level, and then assembled numerically at the system level.

Algorithm 1 $[\mathbf{x}_{n+1}, \mathbf{S}_{n+1}] = \texttt{EulerImplicit}(\mathbf{x}_n, \mathbf{S}_n)$

$\mathbf{x}_{n+1} := \mathbf{x}_n$
for $i = 1$ to i_{max} **do**
  $\mathbf{r} := \mathbf{x}_{n+1} - \mathbf{x}_n - h\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})$
  if $\|\mathbf{r}\| < tol$ **then**
    break
  end if
  $\mathbf{J} := \mathbf{I} - h\mathbf{f}_{,\mathbf{x}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})$
  $\Delta\mathbf{x} := -\mathbf{J}^{-1}\mathbf{r}$
  $\mathbf{x}_{n+1} := \mathbf{x}_{n+1} + \Delta\mathbf{x}$
end for
$\mathbf{S}_{n+1} := \mathbf{S}_n$
$\mathbf{r}_{,\mathbf{p}} := -h\left(\mathbf{f}_{,\mathbf{x}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})\mathbf{S}_{n+1} + \mathbf{f}_{,\mathbf{p}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})\right)$
$\mathbf{J} := \mathbf{I} - h\mathbf{f}_{,\mathbf{x}}(t_{n+1}, \mathbf{x}_{n+1}, \mathbf{p})$
$\Delta\mathbf{S} := -\mathbf{J}^{-1}\mathbf{r}_{,\mathbf{p}}$
$\mathbf{S}_{n+1} := \mathbf{S}_{n+1} + \Delta\mathbf{S}$

In summary, for n_p design parameters, the sensitivity analysis requires only one evaluation and factorization of the Jacobian matrix, one evaluation of the residual $\mathbf{r}_{,\mathbf{p}}$, and one back-substitution for the increment $\Delta\mathbf{S}$. Thus, the impact on the overall computational cost turns out to be much smaller compared to a finite difference method based on repeated simulations. This paradigm is easily extended to multibody systems with kinematic constraints, non-trivial cost-functions, and optimisation constraints, see for example [6, 7, 17, 36].

3 Characterization of the rotation group $SO(3)$

This section recalls basic definitions and properties of the group $SO(3)$. In particular, we focus on the concept of the directional derivative and on the parameterization problem.

Let us consider the motion of a rigid body around its center of gravity. Its orientation with respect to a fixed reference frame is represented using the 3×3 rotation matrix $\mathbf{R}$ which belongs to $SO(3)$, the group of proper orthogonal linear transformations

$$SO(3) = \{\mathbf{R} : \mathbb{R}^3 \to \mathbb{R}^3 \mid \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det\mathbf{R} = +1\}. \tag{4}$$

The group $SO(3)$ has the structure of a 3-dimensional manifold, and it can be locally parameterized using three independent coordinates. The group multiplication is the composition of rotations, i.e. the matrix product.

We consider that the rotation matrix is a function of time t and of a design parameter p, i.e. $\mathbf{R} = \mathbf{R}(t, p)$, and we adopt the convention

$$\frac{d}{dt}(\cdot) = \dot{(\cdot)}, \qquad \frac{d}{dp}(\cdot) = (\cdot)'. \tag{5}$$

By differentiation of the orthogonality condition in Equation (4) with respect to t, one observes that the 3×3 matrix $\widetilde{\mathbf{\Omega}} = \mathbf{R}^T\dot{\mathbf{R}}$ is skew-symmetric. Since $\dot{\mathbf{R}}$ belongs to the tangent space $T_{\mathbf{R}}SO(3)$, the relation

$$\dot{\mathbf{R}} = \mathbf{R}\widetilde{\mathbf{\Omega}} \tag{6}$$

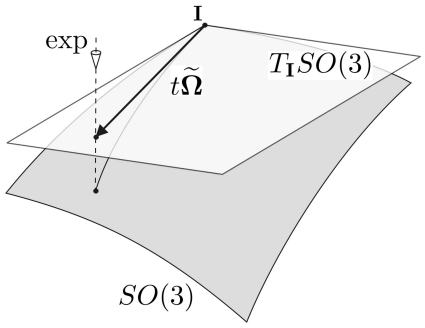

Figure 2: The exponential map.

defines an isomorphism between $T_{\mathbf{R}}SO(3)$ and the set of skew-symmetric matrices $\mathfrak{so}(3)$

$$\mathfrak{so}(3) = \{\widetilde{\mathbf{\Omega}} \mid \widetilde{\mathbf{\Omega}} + \widetilde{\mathbf{\Omega}}^T = \mathbf{0}\}. \tag{7}$$

In particular, $\mathfrak{so}(3)$ is identified with the tangent space at the identity $T_{\mathbf{I}}SO(3)$. Equipped with a bracket operation defined by the matrix commutator $[\widetilde{\mathbf{\Omega}}_1, \widetilde{\mathbf{\Omega}}_2] = \widetilde{\mathbf{\Omega}}_1\widetilde{\mathbf{\Omega}}_2 - \widetilde{\mathbf{\Omega}}_2\widetilde{\mathbf{\Omega}}_1$, $\mathfrak{so}(3)$ forms the Lie algebra associated with the Lie group $SO(3)$.

Moreover, the axial vector $\mathbf{\Omega} \in \mathbb{R}^3$, which represents the material angular velocities, is associated to the skew-symmetric matrix $\widetilde{\mathbf{\Omega}}$ by the isomorphic relation

$$\widetilde{\mathbf{\Omega}} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} = \text{spin}(\mathbf{\Omega}), \qquad \mathbf{\Omega} = \begin{bmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \text{vect}(\widetilde{\mathbf{\Omega}}). \tag{8}$$

We thus write $\mathfrak{so}(3) \simeq \mathbb{R}^3$. The Lie bracket in $\mathfrak{so}(3)$ and the cross product in $\mathbb{R}^3$ are related by $[\widetilde{\mathbf{\Omega}}_1, \widetilde{\mathbf{\Omega}}_2] = \text{spin}\,(\mathbf{\Omega}_1 \times \mathbf{\Omega}_2)$, and we have $\widetilde{\mathbf{\Omega}}\mathbf{a} = \mathbf{\Omega} \times \mathbf{a}$ for any vector $\mathbf{a} \in \mathbb{R}^3$.

Like $\mathbf{R}^T\dot{\mathbf{R}}$, the matrix $\dot{\mathbf{R}}\mathbf{R}^T$ is a skew-symmetric matrix $\widetilde{\boldsymbol{\omega}}$

$$\dot{\mathbf{R}} = \widetilde{\boldsymbol{\omega}}\mathbf{R} \tag{9}$$

and the axial vector $\boldsymbol{\omega}$ represents the spatial angular velocities. For the sake of conciseness, this paper focuses on the material representation of rotations, which leads to simpler expressions for the inertia forces of the rigid body. However, most developments could be reproduced without difficulty for the spatial counterparts.

If $\mathbf{\Omega}$ is known, Equation (6) is a differential equation for $\mathbf{R}$, and the exponential map represents its fundamental solution. Indeed, the curve $\mathbf{R}(t) = \exp(t\widetilde{\mathbf{\Omega}})$ is defined as the solution of Equation (6) for the initial condition $\mathbf{R}(0) = \mathbf{I}$ with a constant angular velocity $\mathbf{\Omega}$. For matrix groups, such as $SO(3)$, the exponential map admits the series expansion

$$\exp(t\widetilde{\mathbf{\Omega}}) = \mathbf{I} + \frac{t\widetilde{\mathbf{\Omega}}}{1!} + \frac{(t\widetilde{\mathbf{\Omega}})^2}{2!} + \dots . \tag{10}$$

As illustrated in Figure 2, this defines a mapping between $\mathfrak{so}(3) = T_{\mathbf{I}}SO(3)$ and $SO(3)$.

Likewise, for the increment $\Delta\mathbf{R}$ (respectively, the sensitivity $\mathbf{R}'$), one can also define an angular increment vector $\Delta\mathbf{\Theta}$ (respectively, an angular sensitivity vector $\mathbf{W}$)

$$\Delta\mathbf{\Theta} = \text{vect}(\mathbf{R}^T\Delta\mathbf{R}), \qquad \mathbf{W} = \text{vect}(\mathbf{R}^T\mathbf{R}'). \tag{11}$$

3.1 Directional derivative in $SO(3)$

First of all, let us recall the definition of the (Gâteaux) directional derivative in a vector space, e.g. in $\mathbb{R}^3$: the derivative of any object $f(\boldsymbol{\phi})$ (f can define a scalar, a vector, etc) at a point $\boldsymbol{\phi} \in \mathbb{R}^3$ in the direction of the vector $\mathbf{v} \in T_{\boldsymbol{\phi}}\mathbb{R}^3 = \mathbb{R}^3$ is defined by

$$Df(\boldsymbol{\phi}) \cdot \mathbf{v} \triangleq \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} f(\boldsymbol{\phi} + \epsilon \mathbf{v}) = \sum_{i=1}^{3} \frac{\partial f(\boldsymbol{\phi})}{\partial \phi_i} v_i \tag{12}$$

where ϕ_i and v_i are the components of the vectors $\boldsymbol{\phi}$ and $\mathbf{v}$. By construction, we have the property $\dot{f} = Df(\boldsymbol{\phi}) \cdot \dot{\boldsymbol{\phi}}$. If $f(\boldsymbol{\phi}, t)$ explicitly depends on t, we have

$$\dot{f} = Df(\boldsymbol{\phi}, t) \cdot \left(\frac{d\boldsymbol{\phi}}{dt}, \frac{dt}{dt} \right) = D_1 f(\boldsymbol{\phi}, t) \cdot \dot{\boldsymbol{\phi}} + f_{,t}. \tag{13}$$

Here, $D_1 f(\boldsymbol{\phi}, t) \cdot \dot{\boldsymbol{\phi}}$ denotes the directional derivative with respect to the first argument (for simplicity, it shall be noted $Df(\boldsymbol{\phi}, t) \cdot \dot{\boldsymbol{\phi}}$), and $f_{,t}$ denotes the partial derivative with respect to t.

In $SO(3)$, the derivative of a function $f(\mathbf{R})$ can be defined at the point $\mathbf{R}$ in the direction specified by a tangent vector $\mathbf{R}\widetilde{\mathbf{V}} \in T_{\mathbf{R}}SO(3)$, with $\mathbf{V} \in \mathbb{R}^3$. Without entering into technical definitions [9], the directional derivative can be computed using the following construction. We choose a curve $\gamma(\cdot) : \mathbb{R} \to SO(3),\ \epsilon \mapsto \mathbf{R}_\epsilon = \gamma(\epsilon)$ which passes through $\mathbf{R}$ at 0, with the tangent $\gamma'(0) = \mathbf{R}\widetilde{\mathbf{V}}$. A convenient choice is $\mathbf{R}_\epsilon = \mathbf{R}\exp(\epsilon\widetilde{\mathbf{V}})$. With a slight abuse of notations, the directional derivative of f at the point $\mathbf{R}$ in the direction $\mathbf{V}$ satisfies

$$Df(\mathbf{R}) \cdot \mathbf{V} = \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} f(\mathbf{R}_\epsilon). \tag{14}$$

The result does not depend on the particular choice of the curve γ, but only on the direction imposed for its tangent. By construction, the directional derivative satisfies the properties

$$\dot{f} = Df(\mathbf{R}) \cdot \boldsymbol{\Omega}, \qquad f' = Df(\mathbf{R}) \cdot \mathbf{W}, \qquad \delta f = Df(\mathbf{R}) \cdot \delta\boldsymbol{\Theta}. \tag{15}$$

For example, the directional derivative of the rotation operator can be computed using the series expansion of the exponential map in Equation (10)

$$D\mathbf{R} \cdot \mathbf{V} = \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} \mathbf{R}\exp(\epsilon\widetilde{\mathbf{V}}) = \mathbf{R}\widetilde{\mathbf{V}}. \tag{16}$$

More complex developments are required to compute the directional derivative of $\widetilde{\boldsymbol{\Omega}}$

$$\begin{aligned} D\widetilde{\boldsymbol{\Omega}} \cdot \mathbf{V} &= \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} \mathbf{R}_\epsilon^T \dot{\mathbf{R}}_\epsilon = \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} \left(\mathbf{R}\exp(\epsilon\widetilde{\mathbf{V}}) \right)^T \left(\dot{\mathbf{R}}\exp(\epsilon\widetilde{\mathbf{V}}) + \mathbf{R}\frac{d}{dt}\exp(\epsilon\widetilde{\mathbf{V}}) \right) \\ &= (\mathbf{R}\widetilde{\mathbf{V}})^T\dot{\mathbf{R}} + \mathbf{R}^T\left(\dot{\mathbf{R}}\widetilde{\mathbf{V}} + \mathbf{R}\dot{\widetilde{\mathbf{V}}} \right) = -\widetilde{\mathbf{V}}\widetilde{\boldsymbol{\Omega}} + \widetilde{\boldsymbol{\Omega}}\widetilde{\mathbf{V}} + \dot{\widetilde{\mathbf{V}}} \\ &= \dot{\widetilde{\mathbf{V}}} + [\widetilde{\boldsymbol{\Omega}}, \widetilde{\mathbf{V}}]. \end{aligned} \tag{17}$$

Unlike $\widetilde{\boldsymbol{\Omega}}\widetilde{\mathbf{V}}$ or $\widetilde{\mathbf{V}}\widetilde{\boldsymbol{\Omega}}$, the matrices $[\widetilde{\boldsymbol{\Omega}}, \widetilde{\mathbf{V}}]$ and $\dot{\widetilde{\mathbf{V}}}$ are skew-symmetric and belong to $\mathfrak{so}(3)$. In terms of axial vectors, this leads to the fundamental relation

$$D\boldsymbol{\Omega} \cdot \mathbf{V} = \dot{\mathbf{V}} + \boldsymbol{\Omega} \times \mathbf{V} \tag{18}$$

which is a general formula to compute increments or derivatives of the angular velocity vector. At the acceleration level, one obtains after time-differentiation

$$D\dot{\boldsymbol{\Omega}} \cdot \mathbf{V} = \ddot{\mathbf{V}} + \boldsymbol{\Omega} \times \dot{\mathbf{V}} + \dot{\boldsymbol{\Omega}} \times \mathbf{V}. \tag{19}$$

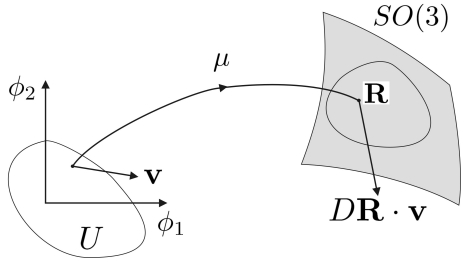

Figure 3: The coordinate map.

In particular, the derivative in the direction of the angular sensitivities $\mathbf{W}$ gives

$$\boldsymbol{\Omega}' = \dot{\mathbf{W}} + \boldsymbol{\Omega} \times \mathbf{W}, \tag{20}$$

$$\dot{\boldsymbol{\Omega}}' = \ddot{\mathbf{W}} + \boldsymbol{\Omega} \times \dot{\mathbf{W}} + \dot{\boldsymbol{\Omega}} \times \mathbf{W}. \tag{21}$$

Let us remark in Equation (20) that the difference between the sensitivity of the angular velocity $\boldsymbol{\Omega}'$ and the time derivative of the angular sensitivity $\dot{\mathbf{W}}$ is measured by the Lie bracket. The derivative of $\boldsymbol{\Omega}$ and $\dot{\boldsymbol{\Omega}}$ in the direction of an angular increment $\Delta\boldsymbol{\Theta}$ is useful for the linearization of the dynamic equations and for the formulation of variational principles

$$\Delta\boldsymbol{\Omega} = \Delta\dot{\boldsymbol{\Theta}} + \boldsymbol{\Omega} \times \Delta\boldsymbol{\Theta}, \tag{22}$$

$$\Delta\dot{\boldsymbol{\Omega}} = \Delta\ddot{\boldsymbol{\Theta}} + \boldsymbol{\Omega} \times \Delta\dot{\boldsymbol{\Theta}} + \dot{\boldsymbol{\Omega}} \times \Delta\boldsymbol{\Theta}. \tag{23}$$

Remark 1 *If $\boldsymbol{\Omega}$ is considered as a vector of $\mathbb{R}^3$, its derivative with respect to p makes perfect sense*

$$\boldsymbol{\Omega}' = \lim_{\epsilon \to 0} \frac{\boldsymbol{\Omega}(p+\epsilon) - \boldsymbol{\Omega}(p)}{\epsilon}. \tag{24}$$

By construction of the directional derivative, this expression is equivalent to $D\boldsymbol{\Omega}\cdot\mathbf{W}$. However, Equation (24) *can also be interpreted as an implicit operation between two tangent vectors that belong to different tangent spaces $T_{\mathbf{R}_\epsilon}SO(3)$ and $T_{\mathbf{R}}SO(3)$. This operation is made possible since both tangent vectors are transported from their respective spaces to the Lie algebra $\mathfrak{so}(3) \simeq \mathbb{R}^3$, using left-translation maps. In Riemannian geometry, the affine connection provides an alternative mechanism to transport a tangent vector from one point to a neighbour. This leads to the definition of the covariant derivative of the vector field $\dot{\mathbf{R}}$ in the direction of the tangent vector $\mathbf{R}'$, which is noted as $\nabla_{\mathbf{R}'}\dot{\mathbf{R}} \in T_{\mathbf{R}}SO(3)$. For example, for a particular choice of the connection, Simo [30] has shown that $\nabla_{\mathbf{R}'}\dot{\mathbf{R}} = \mathbf{R}[\widetilde{\boldsymbol{\Omega}}, \widetilde{\mathbf{W}}]/2 \neq \mathbf{R}(D\widetilde{\boldsymbol{\Omega}}\cdot\mathbf{W})$. The fundamental difference between the directional and the covariant derivative has several implications in solid mechanics, as discussed by this author. In the present paper, we are interested in the derivative of $\boldsymbol{\Omega}$ as a vector of $\mathbb{R}^3$, which justifies the use of the directional derivative.*

3.2 Parameterization of $SO(3)$

The rotation group can be locally parameterized using a vector of three independent coordinates collected in a vector $\boldsymbol{\phi} \in \mathbb{R}^3$. The local coordinate map μ, illustrated in Figure 3, associates to any coordinate vector $\boldsymbol{\phi}$ the corresponding rotation $\mathbf{R}$

$$\mu : U \to SO(3), \quad \boldsymbol{\phi} \mapsto \mathbf{R} = \mu(\boldsymbol{\phi}) \tag{25}$$

where $U \subset \mathbb{R}^3$ represents the validity domain of the parameterization. The derivative of the coordinate map in the direction $\mathbf{v} \in \mathbb{R}^3$

$$D\mathbf{R}\cdot\mathbf{v} = D\mu(\boldsymbol{\phi})\cdot\mathbf{v} \tag{26}$$

defines an angular variation $\widetilde{\mathbf{V}} \in \mathfrak{so}(3)$

$$\widetilde{\mathbf{V}} = \mathbf{R}^T \left(D\mathbf{R} \cdot \mathbf{v}\right) = \mu^T(\boldsymbol{\phi}) \left(D\mu(\boldsymbol{\phi}) \cdot \mathbf{v}\right). \tag{27}$$

The relation between the coordinate variation $\mathbf{v}$ and the angular variation $\mathbf{V}$, which is sometimes noted $d\mu_{\boldsymbol{\phi}}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^3,\ \mathbf{v} \mapsto \mathbf{V} = d\mu_{\boldsymbol{\phi}}(\mathbf{v})$, is necessarily linear with respect to $\mathbf{v}$

$$\mathbf{V} = \mathbf{T}(\boldsymbol{\phi})\mathbf{v}, \tag{28}$$

and the 3×3 tangent operator $\mathbf{T}$ is invertible in the validity range of the parameterization. Particular instances of this relation are

$$\boldsymbol{\Omega} = \mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}, \tag{29}$$
$$\mathbf{W} = \mathbf{T}(\boldsymbol{\phi})\boldsymbol{\phi}', \tag{30}$$
$$\Delta\boldsymbol{\Theta} = \mathbf{T}(\boldsymbol{\phi})\Delta\boldsymbol{\phi}. \tag{31}$$

Equation (28) can be further differentiated with respect to t

$$\dot{\mathbf{V}} = \mathbf{T}\dot{\mathbf{v}} + \dot{\mathbf{T}}\mathbf{v}, \tag{32}$$
$$\ddot{\mathbf{V}} = \mathbf{T}\ddot{\mathbf{v}} + 2\dot{\mathbf{T}}\dot{\mathbf{v}} + \ddot{\mathbf{T}}\mathbf{v} \tag{33}$$

where $\dot{\mathbf{T}}$ and $\ddot{\mathbf{T}}$ can be computed using the concept of directional derivative, see Appendix A.

Several choices are possible for the definition of coordinates $\boldsymbol{\phi}$. A particularly interesting set of coordinates is the Cartesian rotation vector $\boldsymbol{\psi}$, see for example [1], whose definition comes from the Euler theorem: "any rigid motion leaving a point fixed may be represented by a rotation about a suitable axis passing through that point". The vector $\boldsymbol{\psi}$ is oriented in the direction of this axis, and its amplitude is simply the amplitude of the rotation. The coordinate map μ is given by the Rodrigues formula

$$\mathbf{R}(\boldsymbol{\psi}) = \mathbf{I} + \frac{\sin\psi}{\psi}\widetilde{\boldsymbol{\psi}} + \frac{1-\cos\psi}{\psi^2}\widetilde{\boldsymbol{\psi}}\widetilde{\boldsymbol{\psi}} \tag{34}$$

with $\psi \triangleq \|\boldsymbol{\psi}\|$. It can be demonstrated that this expression is equivalent to the exponential map (10), i.e. $\mu(\boldsymbol{\psi}) = \exp(\widetilde{\boldsymbol{\psi}})$. Therefore, the exponential map of the group $SO(3)$ has a closed form mathematical expression. The tangent operator $\mathbf{T}$ is then given by

$$\mathbf{T}(\boldsymbol{\psi}) = \mathbf{I} + \frac{\cos\psi - 1}{\psi^2}\widetilde{\boldsymbol{\psi}} + \left(1 - \frac{\sin\psi}{\psi}\right)\frac{\widetilde{\boldsymbol{\psi}}\widetilde{\boldsymbol{\psi}}}{\psi^2}. \tag{35}$$

It is invertible in the validity range of the parameterization $\psi \in]-2\pi, 2\pi[$, and we note that $\mathbf{T}(\mathbf{0}) = \mathbf{I}$ and $D\mathbf{T}(\mathbf{0}) \cdot \mathbf{v} = -\frac{1}{2}\widetilde{\mathbf{v}}$.

A second interesting parameterization is the conformal rotation vector $\mathbf{c}$, sometimes called the Wiener-Milenkovic parameters. It is defined by its relation with the rotation vector $\boldsymbol{\psi}$

$$\mathbf{c} = \frac{4}{\psi}\tan\left(\frac{\psi}{4}\right)\boldsymbol{\psi}. \tag{36}$$

The corresponding coordinate map is

$$\mathbf{R}(\mathbf{c}) = \frac{1}{(4-c_0)^2}\left[(c_0^2 + 8c_0 - 16)\mathbf{I} + 2\mathbf{c}\mathbf{c}^T + 2c_0\widetilde{\mathbf{c}}\right], \qquad c_0 = 2\frac{\|\mathbf{c}\|^2}{8} \tag{37}$$

and the tangent operator is

$$\mathbf{T}(\mathbf{c}) = \frac{2}{(4-c_0)^2}\left(c_0\mathbf{I} + \frac{1}{4}\mathbf{c}\mathbf{c}^T - \widetilde{\mathbf{c}}\right). \tag{38}$$

As the rotation vector, the conformal rotation vector is valid for amplitudes $\psi \in]-2\pi, 2\pi[$, and we have $\mathbf{T}(\mathbf{0}) = \mathbf{I}$ and $D\mathbf{T}(\mathbf{0}) \cdot \mathbf{v} = -\frac{1}{2}\widetilde{\mathbf{v}}$. For additional details about the parameterization of rotations, we refer to [5, 21].

The advantage of the rotation vector comes from its direct geometric interpretation, but the formula for $\mathbf{R}$ and $\mathbf{T}$ are simpler for the conformal rotation vector, since they do not involve any transcendental function. This property is highly valuable for the computation of first and second derivatives of the operator $\mathbf{T}$, as discussed in Appendix A.

3.3 Reparameterization strategies

Both aforementioned parameterizations are regular over more than one full rotation. Hence, they can be used for the simulation of systems with arbitrary large rotations provided that the coordinate vector does not leave the validity domain U. This condition can be guaranteed if the coordinate vector is rescaled after each time step. Therefore, an inverse coordinate map is defined, which takes its value in a set $U^* \subset U$ such that U^* is far away from the singularities, i.e. $\mu^{-1} : SO(3) \to U^*$, $\mathbf{R} \mapsto \boldsymbol{\phi} = \mu^{-1}(\mathbf{R})$. In order to remain in U^*, the parameters are rescaled according to

$$\boldsymbol{\phi} := \mu^{-1}(\mu(\boldsymbol{\phi})). \tag{39}$$

While this operation leaves unchanged any vector inside the domain U^*, it is discontinuous at the border of U^*. A restart of the integration procedure should then be implemented using a similar algorithm as described later in Section 5.3.

An alternative strategy has been proposed by Cardona and Géradin [11, 13], and it relies on a change of coordinates at the end of each converged time step. The idea is to treat the current rotation as an increment with respect to the rotation at the previous time step

$$\mathbf{R} = \mathbf{R}_{ref}\mathbf{R}_{inc} \tag{40}$$

where the reference rotation $\mathbf{R}_{ref}$ is considered as a given data, and it is only necessary to parameterize the incremental rotation $\mathbf{R}_{inc}$. The angular velocity

$$\widetilde{\boldsymbol{\Omega}} = \mathbf{R}^T\dot{\mathbf{R}} = \mathbf{R}_{inc}^T\dot{\mathbf{R}}_{inc} \tag{41}$$

is not affected by this strategy. Since increments of rotation are always reasonably small between two time steps, the singularities are naturally avoided and the linearity of the problem is improved. At the end of each converged time step, the change of coordinates is defined by

$$\widehat{\mathbf{R}}_{ref} = \mathbf{R}_{ref}\mathbf{R}_{inc}, \qquad \widehat{\mathbf{R}}_{inc} = \mathbf{I} \tag{42}$$

where $\widehat{(\cdot)}$ denotes the quantities after update.

In the simulation code, the reference rotation $\mathbf{R}_{ref}$ represents an additional variable whose numerical value may also depend on the design parameter p. Hence, the sensitivity of $\mathbf{R}_{ref}$ should be taken into account in the analysis. Differentiation of Equation (40) leads to

$$\mathbf{R}' = \mathbf{R}'_{ref}\mathbf{R}_{inc} + \mathbf{R}_{ref}\mathbf{R}'_{inc} \tag{43}$$

which can be multiplied by $\mathbf{R}^T$, so that a relation between the axial vectors $\mathbf{W}_{ref} = \text{vect}(\mathbf{R}_{ref}^T\mathbf{R}'_{ref})$ and $\mathbf{W}_{inc} = \text{vect}(\mathbf{R}_{inc}^T\mathbf{R}'_{inc})$ is obtained

$$\mathbf{W} = \mathbf{R}_{inc}^T\mathbf{W}_{ref} + \mathbf{W}_{inc}. \tag{44}$$

In contrast, the reference configuration is a frozen variable with respect to time, i.e. $\dot{\mathbf{W}}_{ref} = \mathbf{0}$, and the time differentiation of Equation (44) leads to

$$\dot{\mathbf{W}} = \dot{\mathbf{R}}_{inc}^T\mathbf{W}_{ref} + \dot{\mathbf{W}}_{inc} = -\boldsymbol{\Omega} \times \left(\mathbf{R}_{inc}^T\mathbf{W}_{ref}\right) + \dot{\mathbf{W}}_{inc} \tag{45}$$

so that Equation (20) becomes

$$\dot{\mathbf{W}}_{inc} = \boldsymbol{\Omega}' + \mathbf{W}_{inc} \times \boldsymbol{\Omega}. \tag{46}$$

The same expression could have been obtained by differentiation of Equation (41) with respect to p.

4 Equations of motion

This section presents the equations of motion of the rigid body in standard form (Euler equations) and in parameterized form. Both forms will be considered later for the development of time integrators. As in Algorithm 1, the differentiation of the equations of motion is required both for the linearization of the dynamic equilibrium and for the computation of the sensitivities.

4.1 Standard form of rigid-body dynamics

Let us first assume that the center of mass is fixed at the origin of the reference frame (an extension to more general situations will be discussed in Section 6). The rotational kinetic energy of the rigid body is expressed by

$$\mathcal{K} = \frac{1}{2}\boldsymbol{\Omega}^T \mathbf{J}\boldsymbol{\Omega} \tag{47}$$

where $\mathbf{J}$ is the material inertia tensor. Using Equation (22), the variation $\delta\mathcal{K}$ becomes

$$\delta\mathcal{K} = \delta\boldsymbol{\Omega}^T \mathbf{J}\boldsymbol{\Omega} = \delta\dot{\boldsymbol{\Theta}}^T \mathbf{J}\boldsymbol{\Omega} - \delta\boldsymbol{\Theta}^T(\boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega}) = \frac{d}{dt}\left(\delta\boldsymbol{\Theta}^T\mathbf{J}\boldsymbol{\Omega}\right) - \delta\boldsymbol{\Theta}^T\left(\mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega}\right). \tag{48}$$

An external force, with spatial representation $\mathbf{f}(t)$, is applied to a material point P of the rigid body with material position $\mathbf{X}$ and spatial position $\mathbf{x} = \mathbf{R}\mathbf{X}$. If $\mathbf{f}$ is a constant vector, the system is conservative. In the general case, the virtual work is

$$\delta\mathcal{W} = \delta\mathbf{x}^T\mathbf{f}(t) = (\delta\mathbf{R}\mathbf{X})^T\mathbf{f}(t) = (\mathbf{R}\delta\widetilde{\boldsymbol{\Theta}}\mathbf{X})^T\mathbf{f}(t) = \delta\boldsymbol{\Theta}^T\widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t) \tag{49}$$

where $\mathbf{R}^T\mathbf{f}(t)$ is interpreted as a material force, and $\widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t)$ as a material torque.

According to the Hamilton principle, the actual motion satisfies

$$\int_{t_1}^{t_2}(\delta\mathcal{K} + \delta\mathcal{W})dt = 0 \tag{50}$$

which, using Equations (48) and (49), leads to

$$\left[\delta\boldsymbol{\Theta}^T\mathbf{J}\boldsymbol{\Omega}\right]_{t_1}^{t_2} - \int_{t_1}^{t_2}\delta\boldsymbol{\Theta}^T(\mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega} - \widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t))dt = 0. \tag{51}$$

For arbitrary admissible variations $\delta\boldsymbol{\Theta}(t)$ such that $\delta\boldsymbol{\Theta}(t_1) = \delta\boldsymbol{\Theta}(t_2) = \mathbf{0}$, the Euler equation follows as

$$\mathbf{G} \triangleq \mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega} - \widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t) = \mathbf{0}. \tag{52}$$

Given some initial conditions $(\mathbf{R}(0), \boldsymbol{\Omega}(0)) = (\mathbf{R}_0, \boldsymbol{\Omega}_0) \in TSO(3)$, the trajectory $(\mathbf{R}(t), \boldsymbol{\Omega}(t))$ can be computed by solving this equation together with Equation (6). It is clear that for the exact solution, $\mathbf{R}(t) \in SO(3)$, $\forall t \geq 0$. The geometric integration algorithm which will be presented in Section 5.4 is constructed so that the numerical solution inherently remains on the manifold.

For completeness, we mention an alternative semi-parameterized form of the initial value problem, which is obtained by inverting Equation (29)

$$\dot{\boldsymbol{\phi}} = \mathbf{T}^{-1}(\boldsymbol{\phi})\boldsymbol{\Omega}. \tag{53}$$

Equations (52) and (53) form a set of first-order differential equations defined in the vector space $\mathbb{R}^3\times\mathbb{R}^3$. Given a set of initial conditions $(\boldsymbol{\phi}(0), \boldsymbol{\Omega}(0)) = (\boldsymbol{\phi}_0, \boldsymbol{\Omega}_0)$, it can be solved in the time domain for $(\boldsymbol{\phi}(t), \boldsymbol{\Omega}(t)), \forall t > 0$, using classical integration algorithms, as described in [10] and [28].

Differentiation of the Euler equation

Let us differentiate the Euler equation (52)

$$D\mathbf{G}\cdot\mathbf{V} = \mathbf{J}(D\dot{\boldsymbol{\Omega}}\cdot\mathbf{V}) + \mathbf{C}_t\,(D\boldsymbol{\Omega}\cdot\mathbf{V}) + \mathbf{K}_t\,(D\boldsymbol{\Theta}\cdot\mathbf{V}) \tag{54}$$

where the tangent stiffness and damping matrices are defined by

$$\mathbf{K}_t = -\widetilde{\mathbf{X}}\widetilde{\mathbf{R}^T\mathbf{f}}, \qquad \mathbf{C}_t = \widetilde{\boldsymbol{\Omega}}\mathbf{J} - \widetilde{\mathbf{J}\boldsymbol{\Omega}}. \tag{55}$$

In particular, the equation for the increments is

$$\Delta\mathbf{G} = D\mathbf{G}\cdot\Delta\boldsymbol{\Theta} = \mathbf{J}\Delta\dot{\boldsymbol{\Omega}} + \mathbf{C}_t\Delta\boldsymbol{\Omega} + \mathbf{K}_t\Delta\boldsymbol{\Theta}, \tag{56}$$

whereas, using Equation (44), the sensitivities satisfy

$$\mathbf{G}' = D\mathbf{G}\cdot\mathbf{W} + \mathbf{G}_{,p} = \mathbf{J}\dot{\boldsymbol{\Omega}}' + \mathbf{C}_t\boldsymbol{\Omega}' + \mathbf{K}_t\mathbf{W}_{inc} + \mathbf{K}_t\mathbf{R}_{inc}^T\mathbf{W}_{ref} + \mathbf{G}_{,p} = \mathbf{0}. \tag{57}$$

Equations (46) and (57) form a set of differential equations for the sensitivities. Given some initial conditions $(\mathbf{W}_{inc}(0), \boldsymbol{\Omega}'(0)) = (\mathbf{W}_{inc0}, \boldsymbol{\Omega}'_0)$, it can be solved in the time domain for $(\mathbf{W}_{inc}(t), \boldsymbol{\Omega}'(t))$, $\forall t > 0$. The sensitivity of the reference $\mathbf{W}_{ref}$ contributes to Equation (57) as an additional pseudo-load.

4.2 Fully-parameterized form of rigid-body dynamics

After introducing $\delta\boldsymbol{\Theta} = \mathbf{T}(\boldsymbol{\phi})\delta\boldsymbol{\phi}$ into the Hamilton principle (51), we obtain

$$\left[\delta\boldsymbol{\phi}^T\mathbf{T}^T\mathbf{J}\boldsymbol{\Omega}\right]_{t_1}^{t_2} - \int_{t_1}^{t_2}\delta\boldsymbol{\phi}^T\mathbf{T}^T\mathbf{G}dt = 0 \tag{58}$$

which leads, for admissible variations $\delta\boldsymbol{\phi}(t_1) = \delta\boldsymbol{\phi}(t_2) = 0$, to the dynamic equations

$$\mathbf{g} \triangleq \mathbf{T}^T\mathbf{G} = \mathbf{0}. \tag{59}$$

Using Equations (29) and (32), $\mathbf{g}$ can be developed to be

$$\mathbf{g}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, \ddot{\boldsymbol{\phi}}, t) = \mathbf{T}^T(\boldsymbol{\phi})\left(\mathbf{J}\mathbf{T}(\boldsymbol{\phi})\ddot{\boldsymbol{\phi}} + \mathbf{J}\dot{\mathbf{T}}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}} + (\mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}) \times (\mathbf{J}\mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}) - \widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t)\right). \tag{60}$$

Given a set of initial conditions $(\boldsymbol{\phi}(0), \dot{\boldsymbol{\phi}}(0)) = (\boldsymbol{\phi}_0, \dot{\boldsymbol{\phi}}_0)$, the second-order differential equation (59) should be solved in the time domain for $\boldsymbol{\phi}(t)$, $\forall t > 0$. One may reformulate this equation as a first-order equation, and apply any general integrator, such as multistep or Runge-Kutta schemes. Alternatively, the generalized-α method, which directly exploits the second-order structure of the equations of motion, is considered in this paper.

Differentiation of the parameterized equations

First of all, Equation (54) can be developed using Equations (18) and (19)

$$D\mathbf{G}\cdot\mathbf{V} = \mathbf{J}\dot{\mathbf{V}} + \left(\mathbf{J}\widetilde{\boldsymbol{\Omega}} + \mathbf{C}_t\right)\dot{\mathbf{V}} + \left(\mathbf{J}\dot{\widetilde{\boldsymbol{\Omega}}} + \mathbf{C}_t\widetilde{\boldsymbol{\Omega}} + \mathbf{K}_t\right)\mathbf{V}. \tag{61}$$

Given a coordinate variation $\mathbf{v}$, the variation of $\mathbf{G}$ is derived using Equations (28), (32) and (33)

$$D\mathbf{G}\cdot\mathbf{v} = \mathbf{J}(\mathbf{T}\dot{\mathbf{v}} + 2\dot{\mathbf{T}}\dot{\mathbf{v}} + \ddot{\mathbf{T}}\mathbf{v}) + \left(\mathbf{J}\widetilde{\boldsymbol{\Omega}} + \mathbf{C}_t\right)(\mathbf{T}\dot{\mathbf{v}} + \dot{\mathbf{T}}\mathbf{v}) + \left(\mathbf{J}\dot{\widetilde{\boldsymbol{\Omega}}} + \mathbf{C}_t\widetilde{\boldsymbol{\Omega}} + \mathbf{K}_t\right)\mathbf{T}\mathbf{v}. \tag{62}$$

Moreover, we have

$$D\mathbf{g}\cdot\mathbf{v} = \left(D\mathbf{T}^T\cdot\mathbf{v}\right)\mathbf{G} + \mathbf{T}^T\left(D\mathbf{G}\cdot\mathbf{v}\right) = \mathbf{k_G}(\boldsymbol{\phi},\mathbf{G})\mathbf{v} + \mathbf{T}^T\left(D\mathbf{G}\cdot\mathbf{v}\right) \tag{63}$$

where we have introduced the 3×3 operator $\mathbf{k_G}(\boldsymbol{\phi},\mathbf{G})\mathbf{v} \triangleq \left(D\mathbf{T}^T\cdot\mathbf{v}\right)\mathbf{G}$. Clearly, at the equilibrium $\mathbf{G}=\mathbf{0}$, the contribution $\mathbf{k_G}$ vanishes. Finally,

$$D\mathbf{g}\cdot\mathbf{v} = \mathbf{j}\dot{\mathbf{v}} + \mathbf{c}_t\mathbf{v} + \mathbf{k}_t\mathbf{v} \tag{64}$$

holds with

$$\mathbf{k}_t = \mathbf{k_G}(\boldsymbol{\phi},\mathbf{G}) + \mathbf{T}^T\left(\mathbf{J}(\ddot{\mathbf{T}} + \widetilde{\boldsymbol{\Omega}}\dot{\mathbf{T}} + \widetilde{\dot{\boldsymbol{\Omega}}}\mathbf{T}) + \mathbf{C}_t(\dot{\mathbf{T}} + \widetilde{\boldsymbol{\Omega}}\mathbf{T}) + \mathbf{K}_t\mathbf{T}\right), \tag{65}$$

$$\mathbf{c}_t = \mathbf{T}^T\left(\mathbf{J}(2\dot{\mathbf{T}} + \widetilde{\boldsymbol{\Omega}}\mathbf{T}) + \mathbf{C}_t\mathbf{T}\right), \tag{66}$$

$$\mathbf{j} = \mathbf{T}^T\mathbf{J}\mathbf{T}. \tag{67}$$

We remark that $\mathbf{j}(\boldsymbol{\phi})$ is symmetric and that it depends on the configuration $\boldsymbol{\phi}$.

If $\mathbf{v}$ represents an increment $\Delta\boldsymbol{\phi}$, we obtain the linearized equation

$$\Delta\mathbf{g} = \mathbf{j}\Delta\ddot{\boldsymbol{\phi}} + \mathbf{c}_t\Delta\dot{\boldsymbol{\phi}} + \mathbf{k}_t\Delta\boldsymbol{\phi}, \tag{68}$$

but for the sensitivities we get

$$\mathbf{g}' = \mathbf{j}\ddot{\boldsymbol{\phi}}' + \mathbf{c}_t\dot{\boldsymbol{\phi}}' + \mathbf{k}_t\boldsymbol{\phi}' + \mathbf{g}_{ref} + \mathbf{g}_{,p} = \mathbf{0} \tag{69}$$

with $\mathbf{g}_{ref} \triangleq \mathbf{T}^T\mathbf{K}_t\mathbf{R}_{inc}^T\mathbf{W}_{ref}$.

Splitted residual equation

The standard formulation of the generalized-α algorithm is based on a splitted expression of the dynamic equilibrium (59)

$$\mathbf{g} = \mathbf{f}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}}) + \mathbf{f}_c(\boldsymbol{\phi},\dot{\boldsymbol{\phi}},t) \tag{70}$$

where $\mathbf{f}_a$ represents the contribution of the accelerations

$$\mathbf{f}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}}) \triangleq \mathbf{j}(\boldsymbol{\phi})\ddot{\boldsymbol{\phi}} \tag{71}$$

whereas the complementary force vector $\mathbf{f}_c$ is defined by $\mathbf{f}_c = \mathbf{g} - \mathbf{f}_a$.

After differentiation, we get

$$D\mathbf{f}_a\cdot\mathbf{v} = \mathbf{j}\dot{\mathbf{v}} + \left(D\mathbf{j}\cdot\mathbf{v}\right)\ddot{\boldsymbol{\phi}} = \mathbf{j}\dot{\mathbf{v}} + \mathbf{k}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}})\mathbf{v} \tag{72}$$

where the matrix $\mathbf{k}_a$, defined by $\mathbf{k}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}})\mathbf{v} \triangleq (D\mathbf{j}\cdot\mathbf{v})\ddot{\boldsymbol{\phi}}$, is a tangent stiffness proportional to the acceleration. This equation can be particularized

$$\Delta\mathbf{f}_a = \mathbf{j}\Delta\ddot{\boldsymbol{\phi}} + \mathbf{k}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}})\Delta\boldsymbol{\phi}, \tag{73}$$

$$\mathbf{f}_a' = \mathbf{j}\ddot{\boldsymbol{\phi}}' + \mathbf{k}_a(\boldsymbol{\phi},\ddot{\boldsymbol{\phi}})\boldsymbol{\phi}' + (\mathbf{f}_a)_{,p}. \tag{74}$$

Using $\Delta\mathbf{f}_c = \Delta\mathbf{g} - \Delta\mathbf{f}_a$ with Equations (68) and (73), as well as $\mathbf{f}_c' = \mathbf{g}' - \mathbf{f}_a'$ with Equations (69) and (74), one obtains

$$\Delta\mathbf{f}_c = \mathbf{c}_t\Delta\dot{\boldsymbol{\phi}} + (\mathbf{k}_t - \mathbf{k}_a)\Delta\boldsymbol{\phi}, \tag{75}$$

$$\mathbf{f}_c' = \mathbf{c}_t\dot{\boldsymbol{\phi}}' + (\mathbf{k}_t - \mathbf{k}_a)\boldsymbol{\phi}' + \mathbf{g}_{ref} + (\mathbf{f}_c)_{,p}. \tag{76}$$

5 Time integration

The generalized-α family of implicit algorithms has been developed for second-order differential equations in $\mathbb{R}^n$ occuring in structural dynamics. Considering the fully parameterized problem

$$\mathbf{j}(\boldsymbol{\phi})\ddot{\boldsymbol{\phi}} + \mathbf{f}_c(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}}, t) = \mathbf{0}, \qquad (\boldsymbol{\phi}(0), \dot{\boldsymbol{\phi}}(0)) = (\boldsymbol{\phi}_0, \dot{\boldsymbol{\phi}}_0) \tag{77}$$

two definitions are possible for the algorithm: the linear form and the residual form. For a constant mass matrix $\mathbf{j}$, they turn out to be equivalent. The residual form is widely used in linear structural dynamics, but unlike the linear form, it does not have guaranteed convergence properties for systems with a non-constant mass matrix. In the following, we show that the linear form is not more complicated nor computationally more expensive than the residual form. After the description of those algorithms, a geometric extension of the generalized-α method is formulated in a coordinate-free setting.

5.1 Residual form of the generalized-α method

The Newmark integration formulae are obtained from a Taylor series expansion of the displacements and velocities with respect to the time-step size h, as explained in [20],

$$\begin{aligned} \boldsymbol{\phi}_{n+1} &= \boldsymbol{\phi}_n + h\dot{\boldsymbol{\phi}}_n + h^2(\tfrac{1}{2} - \beta)\mathbf{a}_n + h^2\beta\mathbf{a}_{n+1}, \\ \dot{\boldsymbol{\phi}}_{n+1} &= \dot{\boldsymbol{\phi}}_n + h(1-\gamma)\mathbf{a}_n + h\gamma\mathbf{a}_{n+1}, \end{aligned} \tag{78}$$

where the constants β and γ are numerical parameters. The vector $\mathbf{a}$ represents the acceleration, and the reason why we do not use the notation $\ddot{\boldsymbol{\phi}}$ will become clear later on. In the residual form of the generalized-α method, the dynamic equilibrium is replaced by a weighted combination of the acceleration forces $\mathbf{f}_a$ and of the complementary forces $\mathbf{f}_c$ at two successive time steps, i.e.

$$(1-\alpha_m)(\mathbf{f}_a)_{n+1} + \alpha_m(\mathbf{f}_a)_n + (1-\alpha_f)(\mathbf{f}_c)_{n+1} + \alpha_f(\mathbf{f}_c)_n = \mathbf{0}, \qquad (\mathbf{f}_a)_0 + (\mathbf{f}_c)_0 = \mathbf{0} \tag{79}$$

where α_m and α_f are additional numerical parameters, $(\mathbf{f}_a)_n = \mathbf{f}_a(\boldsymbol{\phi}_n, \mathbf{a}_n)$, and $(\mathbf{f}_c)_n = \mathbf{f}_c(\boldsymbol{\phi}_n, \dot{\boldsymbol{\phi}}_n, t_n)$. The optimal choice of the algorithmic parameters $\beta, \gamma, \alpha_f, \alpha_m$ is discussed in [15]. Let us note that the generalized-α method is equivalent to the Hilber-Hughes-Taylor method if $\alpha_m = 0$ and to the Newmark method if $\alpha_m = \alpha_f = 0$.

At each time step, Equations (78) and (79) are solved for the unknowns $\boldsymbol{\phi}_{n+1}$, $\dot{\boldsymbol{\phi}}_{n+1}$ and $\mathbf{a}_{n+1}$ by Newton iterations. Considering the following relationships,

$$\partial\mathbf{f}_a(\boldsymbol{\phi}_{n+1}, \mathbf{a}_{n+1})/\partial\boldsymbol{\phi}_{n+1} = \mathbf{k}_a(\boldsymbol{\phi}_{n+1}, \mathbf{a}_{n+1}), \tag{80}$$

$$\partial\mathbf{f}_a(\boldsymbol{\phi}_{n+1}, \mathbf{a}_{n+1})/\partial\mathbf{a}_{n+1} = \mathbf{j}(\boldsymbol{\phi}_{n+1}), \tag{81}$$

$$\partial\mathbf{f}_c(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, t_{n+1})/\partial\boldsymbol{\phi}_{n+1} = \mathbf{k}_t(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, \mathbf{a}_{n+1}, t_{n+1}) - \mathbf{k}_a(\boldsymbol{\phi}_{n+1}, \mathbf{a}_{n+1}), \tag{82}$$

$$\partial\mathbf{f}_c(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, t_{n+1})/\partial\dot{\boldsymbol{\phi}}_{n+1} = \mathbf{c}_t(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}), \tag{83}$$

and the definition of the auxiliary force covector

$$\mathbf{g}_\alpha = \alpha_m\mathbf{f}_a(\boldsymbol{\phi}_{n+1}, \mathbf{a}_{n+1}) + \alpha_f\mathbf{f}_c(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, t_{n+1}), \tag{84}$$

the complete procedure is described in Algorithm 2. For the sake of notational efficiency, the argument lists of the operators $\mathbf{f}_a()$, $\mathbf{f}_c()$, $\mathbf{j}()$, $\mathbf{c}_t()$, $\mathbf{k}_a()$ and $\mathbf{k}_t()$ have been dropped; it is assumed that they are all evaluated at time t_{n+1}.

The sensitivities can be computed by direct differentiation of the time integration algorithm. Differentiation of the Newmark formula (78) with respect to p yields

$$\begin{aligned} \boldsymbol{\phi}'_{n+1} &= \boldsymbol{\phi}'_n + h\dot{\boldsymbol{\phi}}'_n + h^2(\tfrac{1}{2} - \beta)\mathbf{a}'_n + h^2\beta\mathbf{a}'_{n+1}, \\ \dot{\boldsymbol{\phi}}'_{n+1} &= \dot{\boldsymbol{\phi}}'_n + h(1-\gamma)\mathbf{a}'_n + h\gamma\mathbf{a}'_{n+1} \end{aligned} \tag{85}$$

Algorithm 2 $[\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, \mathbf{a}_{n+1}, \mathbf{g}_\alpha] = \texttt{ResidualAlpha}\left(\boldsymbol{\phi}_n, \dot{\boldsymbol{\phi}}_n, \mathbf{a}_n, \mathbf{g}_\alpha, \mathbf{f}_a(), \mathbf{f}_c(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t(), \mathbf{k}_a()\right)$

$\boldsymbol{\phi}_{n+1} := \boldsymbol{\phi}_n + h\dot{\boldsymbol{\phi}}_n + h^2(0.5-\beta)\mathbf{a}_n$
$\dot{\boldsymbol{\phi}}_{n+1} := h(1-\gamma)\mathbf{a}_n$
$\mathbf{a}_{n+1} := \mathbf{0}$
for $i = 1$ to i_{max} **do**
  $\mathbf{r} := (1-\alpha_m)\mathbf{f}_a() + (1-\alpha_f)\mathbf{f}_c() + \mathbf{g}_\alpha$
  if $\|\mathbf{r}\| < tol$ **then**
    break
  end if
  $\mathbf{S} := (1-\alpha_m)\left(\frac{1}{\beta h^2}\mathbf{j}() + \mathbf{k}_a()\right) + (1-\alpha_f)\left(\frac{\gamma}{\beta h}\mathbf{c}_t() + \mathbf{k}_t() - \mathbf{k}_a()\right)$
  $\Delta\boldsymbol{\phi} := -\mathbf{S}^{-1}\mathbf{r}$
  $\boldsymbol{\phi}_{n+1} := \boldsymbol{\phi}_{n+1} + \Delta\boldsymbol{\phi}$
  $\dot{\boldsymbol{\phi}}_{n+1} := \dot{\boldsymbol{\phi}}_{n+1} + \gamma/(\beta h)\Delta\boldsymbol{\phi}$
  $\mathbf{a}_{n+1} := \mathbf{a}_{n+1} + 1/(\beta h^2)\Delta\boldsymbol{\phi}$
end for
$\mathbf{g}_\alpha := \alpha_m\mathbf{f}_a + \alpha_f\mathbf{f}_c$

whereas the differentiation of Equation (79) leads to

$$(1-\alpha_m)(\mathbf{f}_a')_{n+1} + \alpha_m(\mathbf{f}_a')_n + (1-\alpha_f)(\mathbf{f}_c')_{n+1} + \alpha_f(\mathbf{f}_c')_n = \mathbf{0}, \qquad (\mathbf{f}_a')_0 + (\mathbf{f}_c')_0 = \mathbf{0} \tag{86}$$

where $\mathbf{f}_a'()$ and $\mathbf{f}_c'()$ are given by Equations (74) and (76). Thus, the sensitivity problem is solved using the same algorithm as for the original problem

$$[\boldsymbol{\phi}_{n+1}', \dot{\boldsymbol{\phi}}_{n+1}', \mathbf{a}_{n+1}', \mathbf{g}_\alpha'] = \texttt{ResidualAlpha}\left(\boldsymbol{\phi}_n', \dot{\boldsymbol{\phi}}_n', \mathbf{a}_n', \mathbf{g}_\alpha', \mathbf{f}_a'(), \mathbf{f}_c'(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t(), \mathbf{k}_a()\right). \tag{87}$$

5.2 Linear form of the generalized-α method

The linear form of the generalized-α method is also based on the Newmark formula, but the acceleration variable $\mathbf{a}$ is now defined by the recursion

$$(1-\alpha_m)\mathbf{a}_{n+1} + \alpha_m\mathbf{a}_n = (1-\alpha_f)\ddot{\boldsymbol{\phi}}_{n+1} + \alpha_f\ddot{\boldsymbol{\phi}}_n, \qquad \mathbf{a}_0 = \ddot{\boldsymbol{\phi}}_0. \tag{88}$$

Clearly, $\mathbf{a}$ is not equal to the true acceleration, which satisfies the original residual equation at each time step

$$\mathbf{g}(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, \ddot{\boldsymbol{\phi}}_{n+1}, t) = \mathbf{j}(\boldsymbol{\phi}_{n+1})\ddot{\boldsymbol{\phi}}_{n+1} + \mathbf{f}_c(\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, t) = \mathbf{0}. \tag{89}$$

It is well-known that the generalized-α method is equivalent to a three-step linear formula for the displacements and to a two-step linear formula for the velocities [19]. Unlike the residual form, the linear form is based on this analogy, and it is thus a correct extension to systems with a non-constant mass matrix, see [2] for a detailed discussion. One can easily check that the linear and the residual form of the generalized-α method are equivalent if the matrix $\mathbf{j}$ is constant. Indeed, a multiplication of Equation (88) by $\mathbf{j}$ together with Equation (89) leads to the modified residual Equation (79).

The implementation of the linear form is described in Algorithm 3, which computes the true acceleration $\ddot{\boldsymbol{\phi}}$ without the need of any additional storage resources (the vector $\mathbf{a}$ is an auxiliary vector as the vector $\mathbf{g}_\alpha$ in the residual form). The sensitivity problem is simply solved by

$$[\boldsymbol{\phi}_{n+1}', \dot{\boldsymbol{\phi}}_{n+1}', \ddot{\boldsymbol{\phi}}_{n+1}', \mathbf{a}'] = \texttt{LinearAlpha}\left(\boldsymbol{\phi}_n', \dot{\boldsymbol{\phi}}_n', \ddot{\boldsymbol{\phi}}_n', \mathbf{a}', \mathbf{g}'(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t()\right) \tag{90}$$

where $\mathbf{g}'()$ is given by Equation (69).

Algorithm 3 $[\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, \ddot{\boldsymbol{\phi}}_{n+1}, \mathbf{a}] = \texttt{LinearAlpha}\Big(\boldsymbol{\phi}_n, \dot{\boldsymbol{\phi}}_n, \ddot{\boldsymbol{\phi}}_n, \mathbf{a}, \mathbf{g}(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t()\Big)$

$\boldsymbol{\phi}_{n+1} := \boldsymbol{\phi}_n + h\dot{\boldsymbol{\phi}}_n + h^2(0.5-\beta)\mathbf{a}$
$\dot{\boldsymbol{\phi}}_{n+1} := \dot{\boldsymbol{\phi}}_n + h(1-\gamma)\mathbf{a}$
$\mathbf{a} := 1/(1-\alpha_m)(\alpha_f \ddot{\boldsymbol{\phi}}_n - \alpha_m \mathbf{a})$
$\boldsymbol{\phi}_{n+1} := \boldsymbol{\phi}_{n+1} + \beta h^2 \mathbf{a}$
$\dot{\boldsymbol{\phi}}_{n+1} := \dot{\boldsymbol{\phi}}_{n+1} + \gamma h \mathbf{a}$
$\ddot{\boldsymbol{\phi}}_{n+1} := \mathbf{0}$
for $i = 1$ to i_{max} **do**
  $\mathbf{r} := \mathbf{g}()$
  if $\|\mathbf{r}\| < tol$ **then**
    break
  end if
  $\mathbf{S} := \dfrac{1}{\beta h^2}\dfrac{1-\alpha_m}{1-\alpha_f}\mathbf{j}() + \dfrac{\gamma}{\beta h}\mathbf{c}_t() + \mathbf{k}_t()$
  $\Delta\boldsymbol{\phi} := -\mathbf{S}^{-1}\mathbf{r}$
  $\boldsymbol{\phi}_{n+1} := \boldsymbol{\phi}_{n+1} + \Delta\boldsymbol{\phi}$
  $\dot{\boldsymbol{\phi}}_{n+1} := \dot{\boldsymbol{\phi}}_{n+1} + \gamma/(\beta h)\Delta\boldsymbol{\phi}$
  $\ddot{\boldsymbol{\phi}}_{n+1} := \ddot{\boldsymbol{\phi}}_{n+1} + 1/(\beta h^2)((1-\alpha_m)/(1-\alpha_f)\Delta\boldsymbol{\phi})$
end for
$\mathbf{a} := \mathbf{a} + (1-\alpha_f)/(1-\alpha_m)\ddot{\boldsymbol{\phi}}_{n+1}$

5.3 Reparameterization algorithms

In both preceding algorithms, $\boldsymbol{\phi}_{n+1}$ represents an incremental rotation and the rotation matrix $\mathbf{R}$ is computed according to $\mathbf{R} = \mathbf{R}_{ref}\mu(\boldsymbol{\phi}_{n+1})$. As discussed in Section 3.3, the update operation at the end of the time step is defined by

$$\widehat{\mathbf{R}}_{ref} = \mathbf{R}_{ref}\mu(\boldsymbol{\phi}), \qquad \widehat{\boldsymbol{\phi}} = \mathbf{0}. \tag{91}$$

By imposing that the geometric quantities $\boldsymbol{\Omega}$ and $\dot{\boldsymbol{\Omega}}$ are not modified by the change of coordinates, Equations (29) and (32) lead to the transformation rule at velocity and acceleration levels

$$\dot{\widehat{\boldsymbol{\phi}}} = \mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}, \tag{92}$$

$$\ddot{\widehat{\boldsymbol{\phi}}} = \mathbf{T}(\boldsymbol{\phi})\ddot{\boldsymbol{\phi}} + \dot{\mathbf{T}}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}}. \tag{93}$$

The parameter $\mathbf{a}$, which has no obvious geometrical meaning, may be treated as an auxiliary acceleration at the current time. Hence, the variable $\mathbf{A} \triangleq \mathbf{T}(\boldsymbol{\phi})\mathbf{a} + \dot{\mathbf{T}}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}}$ is a geometric quantity that should also be preserved by the reparameterization,

$$\widehat{\mathbf{a}} = \mathbf{T}(\boldsymbol{\phi})\mathbf{a} + \dot{\mathbf{T}}(\boldsymbol{\phi}, \dot{\boldsymbol{\phi}})\dot{\boldsymbol{\phi}}. \tag{94}$$

For the residual form of the algorithm, the auxiliary force covector $\mathbf{g}_\alpha$ is updated by imposing that its virtual work with respect to any variation $\delta\boldsymbol{\phi}$ is unchanged, which leads to

$$\widehat{\mathbf{g}}_\alpha = \mathbf{T}^{-T}\mathbf{g}_\alpha. \tag{95}$$

In other words, $\mathbf{g}_\alpha$ is the coordinate expression of a material covector $\mathbf{G}_\alpha \triangleq \mathbf{T}^{-T}\mathbf{g}_\alpha$, which is preserved by the reparameterization.

Transformation rules for the sensitivities

The sensitivities are also affected by the change of coordinates. Transformation rules are obtained by direct differentiation of Equations (91) to (95)

$$\begin{aligned} \hat{\mathbf{W}}_{ref} &= \mathbf{R}_{inc}^T \mathbf{W}_{ref} + \mathbf{T}\boldsymbol{\phi}', & \hat{\mathbf{W}}_{inc} &= \widehat{\boldsymbol{\phi}}' = \mathbf{0}, \\ \dot{\widehat{\boldsymbol{\phi}}}' &= \mathbf{T}\dot{\boldsymbol{\phi}}' + \mathbf{T}'\dot{\boldsymbol{\phi}}, & \ddot{\widehat{\boldsymbol{\phi}}}' &= \mathbf{T}\ddot{\boldsymbol{\phi}}' + \mathbf{T}'\ddot{\boldsymbol{\phi}} + \dot{\mathbf{T}}\dot{\boldsymbol{\phi}}' + \dot{\mathbf{T}}'\dot{\boldsymbol{\phi}}, \\ \widehat{\mathbf{a}}' &= \mathbf{T}\mathbf{a}' + \mathbf{T}'\mathbf{a} + \dot{\mathbf{T}}\dot{\boldsymbol{\phi}}' + \dot{\mathbf{T}}'\dot{\boldsymbol{\phi}}, & \widehat{\mathbf{g}}'_\alpha &= \mathbf{T}^{-T}\mathbf{g}'_\alpha + (\mathbf{T}^{-T})'\mathbf{g}_\alpha. \end{aligned} \tag{96}$$

In the first line, we observe that the sensitivity of the increment $\boldsymbol{\phi}'$ is transferred to the sensitivity of the reference configuration $\mathbf{W}_{ref}$.

However, since several combinations of $\mathbf{W}_{ref}$ and $\mathbf{W}_{inc}$ can represent the same geometric vector $\mathbf{W} = \mathbf{R}_{inc}^T \mathbf{W}_{ref} + \mathbf{W}_{inc}$, other transformation rules could be defined so that the contribution $\mathbf{W}_{ref}$ is forced to zero. The proposed reference-free update algorithm is based on the requirement that the geometric sensitivities, which can be represented either by $(\mathbf{W}, \boldsymbol{\Omega}', \dot{\boldsymbol{\Omega}}')$ or by $(\mathbf{W}, \dot{\mathbf{W}}, \ddot{\mathbf{W}})$, are preserved during the transformation. Therefore, the geometric sensitivities are first computed from $(\boldsymbol{\phi}', \dot{\boldsymbol{\phi}}', \ddot{\boldsymbol{\phi}}')$ using

$$\left\{ \begin{array}{l} \mathbf{W} = \mathbf{T}\boldsymbol{\phi}', \\ \boldsymbol{\Omega}' = \mathbf{T}\dot{\boldsymbol{\phi}}' + \mathbf{T}'\dot{\boldsymbol{\phi}}, \\ \dot{\boldsymbol{\Omega}}' = \mathbf{T}\ddot{\boldsymbol{\phi}}' + \mathbf{T}'\ddot{\boldsymbol{\phi}} + \dot{\mathbf{T}}\dot{\boldsymbol{\phi}}' + \dot{\mathbf{T}}'\dot{\boldsymbol{\phi}}, \end{array} \right. \quad \text{or} \quad \left\{ \begin{array}{l} \mathbf{W} = \mathbf{T}\boldsymbol{\phi}', \\ \dot{\mathbf{W}} = \mathbf{T}\dot{\boldsymbol{\phi}}' + \dot{\mathbf{T}}\boldsymbol{\phi}', \\ \ddot{\mathbf{W}} = \mathbf{T}\ddot{\boldsymbol{\phi}}' + 2\dot{\mathbf{T}}\dot{\boldsymbol{\phi}}' + \ddot{\mathbf{T}}\boldsymbol{\phi}'. \end{array} \right. \tag{97}$$

Given $(\widehat{\boldsymbol{\phi}}, \dot{\widehat{\boldsymbol{\phi}}}, \ddot{\widehat{\boldsymbol{\phi}}})$, the transformed sensitivities $(\widehat{\boldsymbol{\phi}}', \dot{\widehat{\boldsymbol{\phi}}}', \ddot{\widehat{\boldsymbol{\phi}}}')$ are then obtained by inversion of the same relations (97) (this inversion is always possible since $\mathbf{T}(\widehat{\boldsymbol{\phi}}) = \mathbf{I}$). The treatment can be extended to the auxiliary variables $\mathbf{a}$ and $\mathbf{g}_\alpha$ since the sensitivity of the geometric quantities $\mathbf{A}$ and $\mathbf{G}_\alpha$, i.e.,

$$\mathbf{A}' = \mathbf{T}\mathbf{a}' + \mathbf{T}'\mathbf{a} + \dot{\mathbf{T}}\dot{\boldsymbol{\phi}}' + \dot{\mathbf{T}}'\dot{\boldsymbol{\phi}}, \tag{98}$$

$$\widehat{\mathbf{G}}'_\alpha = \mathbf{T}^{-T}\mathbf{g}'_\alpha + (\mathbf{T}^{-T})'\mathbf{g}_\alpha \tag{99}$$

should also be preserved during the reparameterization.

5.4 Geometric generalized-α method

The time integration algorithms described in the previous sections were based on a local parameterization of the rotation group and on a reparameterization strategy at each time step. We have seen that the implementation of a sensitivity analysis in that context becomes rather complex and technical. In contrast, the present section shows that a geometric integrator leads to simpler and computationally more efficient algorithms. Let us recall the initial value problem in standard form

$$\dot{\mathbf{R}} = \mathbf{R}\widetilde{\boldsymbol{\Omega}}, \tag{100}$$

$$\mathbf{G}(\mathbf{R}, \boldsymbol{\Omega}, \dot{\boldsymbol{\Omega}}, t) = \mathbf{0} \tag{101}$$

with $(\mathbf{R}(0), \boldsymbol{\Omega}(0)) = (\mathbf{R}_0, \boldsymbol{\Omega}_0) \in TSO(3)$. Following the idea of Crouch and Grossman [16], a Lie group integrator can be constructed by replacing the right-hand-side $\mathbf{R}\widetilde{\boldsymbol{\Omega}}$ by a simpler expression. In particular, assuming that $\boldsymbol{\Omega}(t)$ is replaced by a constant approximation $\overline{\boldsymbol{\Omega}}$ during the time step ($\overline{\boldsymbol{\Omega}}$ is a frozen vector field), the numerical solution is given by the exponential map

$$\mathbf{R}_{n+1} = \mathbf{R}_n \exp(h\widetilde{\overline{\boldsymbol{\Omega}}}). \tag{102}$$

Table 1: Polymorphic definition of the operators π, exp and $d\exp_{\boldsymbol{\psi}}$.

Rotations	Rotation sensitivities	Parameters (and sensitivities)
$\pi(\mathbf{R}_n, \mathbf{R}_{inc}) \triangleq \mathbf{R}_n\mathbf{R}_{inc}$	$\pi(\mathbf{W}_n, \mathbf{W}_{inc}) \triangleq \mathbf{R}_{inc}^T\mathbf{W}_n + \mathbf{W}_{inc}$	$\pi(\boldsymbol{\phi}_n, \boldsymbol{\phi}_{inc}) \triangleq \boldsymbol{\phi}_n + \boldsymbol{\phi}_{inc}$
$\exp(\boldsymbol{\psi}) \triangleq \exp(\widetilde{\boldsymbol{\psi}})$	$\exp(\boldsymbol{\psi}') \triangleq \mathbf{T}(\boldsymbol{\psi})\boldsymbol{\psi}'$	$\exp(\boldsymbol{\phi}_{inc}) \triangleq \boldsymbol{\phi}_{inc}$
$d\exp_{\boldsymbol{\psi}} \triangleq \mathbf{T}(\boldsymbol{\psi})$	$d\exp_{\boldsymbol{\psi}'} \triangleq \mathbf{T}(\boldsymbol{\psi})$	$d\exp_{\boldsymbol{\phi}_{inc}} \triangleq \mathbf{I}$

In order to avoid any confusion between $\overline{\boldsymbol{\Omega}}$ and $\boldsymbol{\Omega}(t)$, the variable $\overline{\boldsymbol{\Omega}}$ shall be replaced by the auxiliary variable $\boldsymbol{\psi} = h\overline{\boldsymbol{\Omega}}$. According to Equation (102), $\boldsymbol{\psi}$ is the rotation vector associated with the incremental rotation from $\mathbf{R}_n$ to $\mathbf{R}_{n+1}$. An extension of the generalized-α method to Lie groups is thus defined by

$$\begin{aligned}
\boldsymbol{\psi} &= h\overline{\boldsymbol{\Omega}} = h\boldsymbol{\Omega}_n + (0.5-\beta)h^2\mathbf{A}_n + \beta h^2\mathbf{A}_{n+1}, && (103)\\
\boldsymbol{\Omega}_{n+1} &= \boldsymbol{\Omega}_n + (1-\gamma)h\mathbf{A}_n + \gamma h\mathbf{A}_{n+1}, && (104)\\
(1-\alpha_m)\mathbf{A}_{n+1} + \alpha_m\mathbf{A}_n &= (1-\alpha_f)\dot{\boldsymbol{\Omega}}_{n+1} + \alpha_f\dot{\boldsymbol{\Omega}}_n && (105)
\end{aligned}$$

and the dynamic equilibrium $\mathbf{G}_{n+1} = \mathbf{0}$ should be satisfied at the current time step. In the particular case $\alpha_m = \alpha_f = 0$, this corresponds to the algorithm proposed by Simo and Vu-Quoc [32]. The numerical procedure is described in Algorithm 4, and a proof of second-order consistency is given in Appendix B.

Denoting $\mathbf{R}_{inc} = \exp(\widetilde{\boldsymbol{\psi}})$, the differentiation of Equation (102) leads to

$$\mathbf{V}_{n+1} = \mathbf{R}_{inc}^T\mathbf{V}_n + \mathbf{V}_{inc} = \mathbf{R}_{inc}^T\mathbf{V}_n + \mathbf{T}(\boldsymbol{\psi})\mathbf{v}. \tag{106}$$

where $\mathbf{V}_n$, $\mathbf{V}_{n+1}$ and $\mathbf{V}_{inc}$ are the angular variations associated with $\mathbf{R}_n$, $\mathbf{R}_{n+1}$ and $\mathbf{R}_{inc}$, respectively, and $\mathbf{v}$ is the variation of $\boldsymbol{\psi}$. For the Newton iterations, the rotation at the previous time step is fixed, so that

$$\Delta\boldsymbol{\Theta}_{n+1} = \mathbf{T}(\boldsymbol{\psi})\Delta\boldsymbol{\psi} \tag{107}$$

whereas the equation of the sensitivities is

$$\mathbf{W}_{n+1} = \mathbf{R}_{inc}^T\mathbf{W}_n + \mathbf{T}(\boldsymbol{\psi})\boldsymbol{\psi}'. \tag{108}$$

Hence, Algorithm 4 is an extension of the `LinearAlpha` algorithm which can be used to solve four different problems

$$\begin{aligned}
[\mathbf{R}_{n+1}, \boldsymbol{\Omega}_{n+1}, \dot{\boldsymbol{\Omega}}_{n+1}, \mathbf{A}] &= \texttt{GeometricAlpha}\left(\mathbf{R}_n, \boldsymbol{\Omega}_n, \dot{\boldsymbol{\Omega}}_n, \mathbf{A}, \mathbf{G}(), \mathbf{J}(), \mathbf{C}_t(), \mathbf{K}_t()\right),\\
[\mathbf{W}_{n+1}, \boldsymbol{\Omega}'_{n+1}, \dot{\boldsymbol{\Omega}}'_{n+1}, \mathbf{A}'] &= \texttt{GeometricAlpha}\left(\mathbf{W}'_n, \boldsymbol{\Omega}'_n, \dot{\boldsymbol{\Omega}}'_n, \mathbf{A}', \mathbf{G}'(), \mathbf{J}(), \mathbf{C}_t(), \mathbf{K}_t()\right),\\
[\boldsymbol{\phi}_{n+1}, \dot{\boldsymbol{\phi}}_{n+1}, \ddot{\boldsymbol{\phi}}_{n+1}, \mathbf{a}] &= \texttt{GeometricAlpha}\left(\boldsymbol{\phi}_n, \dot{\boldsymbol{\phi}}_n, \ddot{\boldsymbol{\phi}}_n, \mathbf{a}, \mathbf{g}(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t()\right),\\
[\boldsymbol{\phi}'_{n+1}, \dot{\boldsymbol{\phi}}'_{n+1}, \ddot{\boldsymbol{\phi}}'_{n+1}, \mathbf{a}'] &= \texttt{GeometricAlpha}\left(\boldsymbol{\phi}'_n, \dot{\boldsymbol{\phi}}'_n, \ddot{\boldsymbol{\phi}}'_n, \mathbf{a}', \mathbf{g}'(), \mathbf{j}(), \mathbf{c}_t(), \mathbf{k}_t()\right)
\end{aligned}$$

provided that the operators π (composition), exp (exponential), and $d\exp_{\boldsymbol{\psi}}$ (tangent operator) receive the polymorphic definition of Table 1. Using this implementation, the same algorithm can be applied to problems defined either in a vector space or in a Lie group.

Let us remark that the geometric algorithm involves the operators $\mathbf{G}()$, $\mathbf{J}()$, $\mathbf{C}_t()$ and $\mathbf{K}_t()$, which have a much simpler expression than $\mathbf{g}()$, $\mathbf{j}()$, $\mathbf{c}_t()$ and $\mathbf{k}_t()$, and that it does not require any reparameterization strategy. Hence, it allows a simpler implementation and a better computational efficiency than the algorithm based on a local parameterization.

Algorithm 4 $[\mathbf{R}_{n+1}, \boldsymbol{\Omega}_{n+1}, \dot{\boldsymbol{\Omega}}_{n+1}, \mathbf{A}] = \texttt{GeometricAlpha}\Big(\mathbf{R}_n, \boldsymbol{\Omega}_n, \dot{\boldsymbol{\Omega}}_n, \mathbf{A}, \mathbf{G}(), \mathbf{J}(), \mathbf{C}_t(), \mathbf{K}_t()\Big)$

$\boldsymbol{\psi} := h\boldsymbol{\Omega}_n + h^2(0.5-\beta)\mathbf{A}$
$\boldsymbol{\Omega}_{n+1} := \boldsymbol{\Omega}_n + h(1-\gamma)\mathbf{A}$
$\mathbf{A} := 1/(1-\alpha_m)(\alpha_f\dot{\boldsymbol{\Omega}}_n - \alpha_m\mathbf{A})$
$\boldsymbol{\psi} := \boldsymbol{\psi} + \beta h^2\mathbf{A}$
$\boldsymbol{\Omega}_{n+1} := \boldsymbol{\Omega}_{n+1} + \gamma h\mathbf{A}$
$\dot{\boldsymbol{\Omega}}_{n+1} := \mathbf{0}$
for $i = 1$ to i_{max} **do**
  $\mathbf{R}_{n+1} := \pi(\mathbf{R}_n, \exp(\boldsymbol{\psi}))$
  $\mathbf{r} := \mathbf{G}()$
  if $\|\mathbf{r}\| < tol$ **then**
    break
  end if
  $\mathbf{S} := \dfrac{1}{\beta h^2}\dfrac{1-\alpha_m}{1-\alpha_f}\mathbf{J}() + \left(\dfrac{\gamma}{\beta h}\mathbf{C}_t() + \mathbf{K}_t() d\exp_{\boldsymbol{\psi}}\right)$
  $\Delta\boldsymbol{\psi} := -\mathbf{S}^{-1}\mathbf{r}$
  $\boldsymbol{\psi} := \boldsymbol{\psi} + \Delta\boldsymbol{\psi}$
  $\boldsymbol{\Omega}_{n+1} := \boldsymbol{\Omega}_{n+1} + \gamma/(\beta h)\Delta\boldsymbol{\psi}$
  $\dot{\boldsymbol{\Omega}}_{n+1} := \dot{\boldsymbol{\Omega}}_{n+1} + 1/(\beta h^2)((1-\alpha_m)/(1-\alpha_f)\Delta\boldsymbol{\psi})$
end for
$\mathbf{A} := \mathbf{A} + (1-\alpha_f)/(1-\alpha_m)\dot{\boldsymbol{\Omega}}_{n+1}$

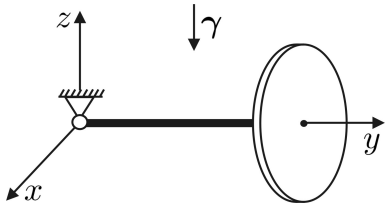

Figure 4: The spinning top.

6 Example: the heavy top

Let us analyse the motion of a top in the gravity field, as illustrated in Figure 4. Following the formalism for multibody systems [21], the motion of a rigid body is described by the absolute translation of the centre of gravity and the absolute rotation around this point. The motion of any other point of the body has to satisfy non-deformation conditions, which are represented by kinematic constraints. If $\mathbf{x}$ is the absolute translation of the centre of gravity with respect to the fixed point of the top, m the mass, and $\boldsymbol{\gamma}$ the gravity acceleration, the equations of motion of the top are

$$m\ddot{\mathbf{x}} - \boldsymbol{\lambda} = m\boldsymbol{\gamma}, \tag{109}$$

$$\mathbf{J}\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}\times\mathbf{J}\boldsymbol{\Omega} + \widetilde{\mathbf{X}}\mathbf{R}^T\boldsymbol{\lambda} = \mathbf{0}, \tag{110}$$

$$-\mathbf{x} + \mathbf{R}\mathbf{X} = \mathbf{0}. \tag{111}$$

Due to the kinematic constraint (111), those equations have a differential algebraic structure, and the vector of Lagrange multipliers $\boldsymbol{\lambda}$ is interpreted as a spatial reaction force applied at the fixed point. It is well-known that the generalized-α method can be used for the simulation of differential algebraic systems provided the introduction of a small amount of high-frequency numerical damping [2, 13].

The numerical data come from references [11, 13]. In standard units, the value of the mass is $m = 15$, the inertia tensor is $\mathbf{J} = \text{diag}\{0.234375, 0.46875, 0.234375\}$, the position of the

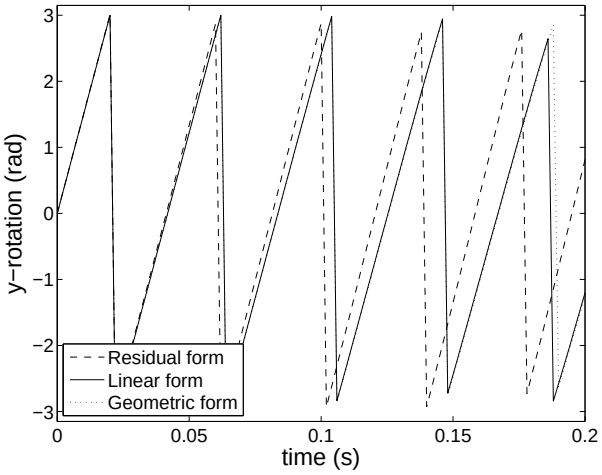

Figure 5: Simulation results for the spinning top: rotation ψ_2.

centre of mass is $\mathbf{X} = [0.\ 1.\ 0.]^T$, the gravity acceleration is $\boldsymbol{\gamma} = [0.\ 0.\ -9.81]^T$, and the initial conditions are $\mathbf{R}_0 = \mathbf{I}$, and $\boldsymbol{\Omega}_0 = [0.\ 150.\ 4.61538]^T$. In this case, the top moves through the bottom position $z = -1$. The initial angular acceleration satisfies the equilibrium of angular momentum with respect to the fixed point

$$\dot{\boldsymbol{\Omega}}_0 = \left(\mathbf{J} - m\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}\right)^{-1} \left(\mathbf{X} \times m\boldsymbol{\gamma} - \boldsymbol{\Omega}_0 \times ((\mathbf{J} - m\widetilde{\mathbf{X}}\widetilde{\mathbf{X}})\boldsymbol{\Omega}_0)\right). \tag{112}$$

Consistent initial values for $\mathbf{x}_0$, $\dot{\mathbf{x}}_0$ and $\ddot{\mathbf{x}}_0$ satisfy the constraints at position, velocity and acceleration level

$$\mathbf{x}_0 = \mathbf{X}, \qquad \dot{\mathbf{x}}_0 = \boldsymbol{\Omega}_0 \times \mathbf{X}, \qquad \ddot{\mathbf{x}}_0 = \dot{\boldsymbol{\Omega}}_0 \times \mathbf{X} + \boldsymbol{\Omega}_0 \times (\boldsymbol{\Omega}_0 \times \mathbf{X}). \tag{113}$$

Various versions of the time-integration algorithm are tested, and the algorithmic coefficients β, γ, α_f and α_m are always selected according the CH-α method [15], i.e.,

$$\alpha_f = \frac{\rho_\infty}{\rho_\infty + 1}, \quad \alpha_m = \frac{2\rho_\infty - 1}{\rho_\infty + 1}, \quad \beta = \frac{1}{4}(1 + \alpha_f - \alpha_m)^2, \quad \gamma = \frac{1}{2} + \alpha_f - \alpha_m \tag{114}$$

where ρ_∞ represents the desired value of the spectral radius at infinite frequencies, in this example $\rho_\infty = 0.8$. The default value of the time step is $h = 2.\text{e-}3$ s.

Figures 5, 6, and 7 represent the transient response for the three algorithms. We note that the range of the abscissa is different in each case. In the simulation results, the global rotation is represented by the rotation vector $\boldsymbol{\psi} = \mu^{-1}(\mathbf{R})$, and the discontinuities observed in Figure 5 are a consequence of the discontinuity of the function μ^{-1}, see Section 3.3. In Figure 6, the residual form of the generalized-α method yields a significant error, since the trajectory does not go through the bottom point $z = -1$. Figure 7 shows the initial high-frequency oscillations of the Lagrange multipliers, which are progressively damped due to the presence of high-frequency numerical dissipation. This phenomenon is typical when a differential algebraic system is solved using a linear multistep method (recall that the linear generalized-α method is equivalent to a linear three-step method). However, the oscillations are significantly higher for the residual form of the algorithm, whereas the linear form exhibits a much better behaviour.

The convergence of the numerical response for decreasing values of h is shown in Figure 8. For each algorithm, the reference solution is computed using a smaller step-size $h_{ref} = 5.\text{e-}6$ s, and the relative error is estimated by the expression

$$\frac{\|\mathbf{x}_h(t^*) - \mathbf{x}_{ref}(t^*)\|}{\|\mathbf{x}_{ref}(t^*)\|} \tag{115}$$

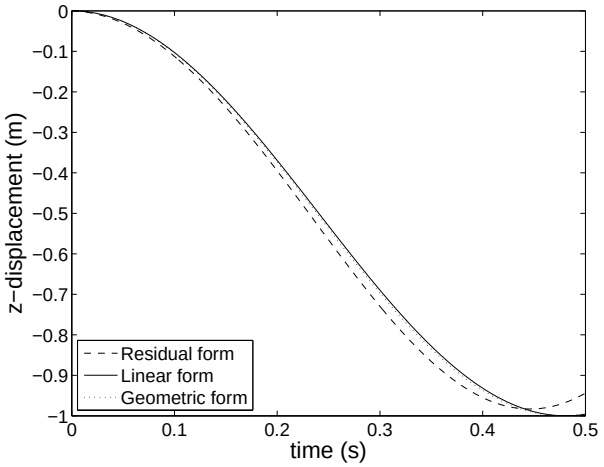

Figure 6: Simulation results for the spinning top: vertical translation x_3.

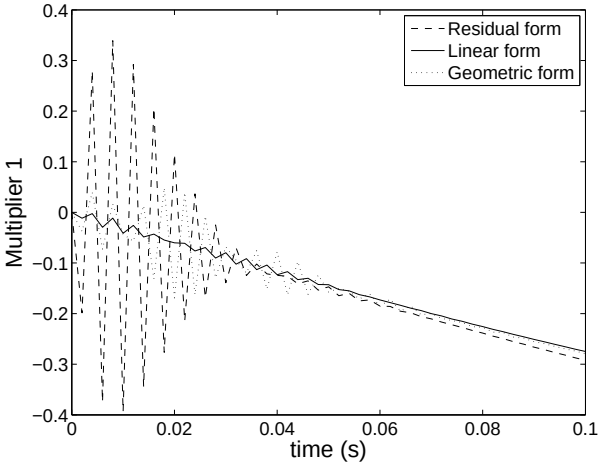

Figure 7: Simulation results for the spinning top: Lagrange multiplier λ_1.

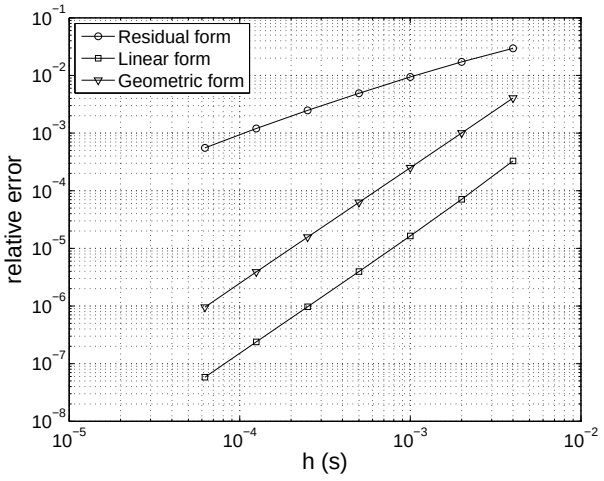

Figure 8: Convergence of the simulation results.

Table 2: Relative errors for the sensitivities (Reference algorithms).

Algorithm	Relative sensitivity errors
`ResidualAlpha`	2.1e-10
`LinearAlpha`	3.5e-10
`GeometricAlpha`	2.2e-10

with $t^* = 0.1$ s. While the linear and the geometric algorithms exhibit a clear second-order convergence, only first-order convergence is observed for the residual algorithm. This confirms that the residual form of the generalized-α method should not be used for systems with a non-constant mass matrix, in particular for systems with large rotations. In contrast, the linear form allows a correct extension to this case with a high level of accuracy. Even though the geometric form also yields second-order convergence, the error constant is higher than for the linear form in this example.

The sensitivities of the numerical response with respect to the mass m have also been computed. The initial values are obtained by differentiation of the initial values for the initial problem, which leads to $\mathbf{R}'_0 = \mathbf{0}$, $\mathbf{\Omega}'_0 = \mathbf{0}$, $\mathbf{x}'_0 = \mathbf{0}$, $\dot{\mathbf{x}}'_0 = \mathbf{0}$, and

$$\dot{\mathbf{\Omega}}'_0 = \left(\mathbf{J} - m\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}\right)^{-1} \left(\mathbf{X} \times \boldsymbol{\gamma} + \mathbf{X} \times (\mathbf{X} \times \dot{\mathbf{\Omega}}_0) + \mathbf{\Omega}_0 \times (\mathbf{X} \times (\mathbf{X} \times \mathbf{\Omega}_0))\right), \tag{116}$$

$$\ddot{\mathbf{x}}'_0 = \dot{\mathbf{\Omega}}'_0 \times \mathbf{X}. \tag{117}$$

Since the equations of motion depend linearly on m, the finite difference scheme implemented for the pseudo-load in the sensitivity equation ($\mathbf{G}_{,m}$, $\mathbf{g}_{,m}$, $(\mathbf{f}_a)_{,m}$ and $(\mathbf{f}_c)_{,m}$) does not introduce any numerical error.

The results x'_{sa} of the semi-analytical algorithm are compared to the results of a finite difference scheme x'_{fd} based on repeated simulations. As shown in Table 2, the relative error

$$\frac{\|\mathbf{x}'_{sa}(t^*) - \mathbf{x}'_{fd}(t^*)\|}{\|\mathbf{x}'_{fd}(t^*)\|} \tag{118}$$

is quite small for each algorithm. Table 3 illustrates the importance of the finite difference scheme used for the computation of $\mathbf{x}'_{fd}$. In theory, the backward difference, the central difference, and the five-point stencil lead to first, second, and fourth order approximations. Since

Table 3: Relative errors for the sensitivities (influence of the finite difference scheme).

	Backward difference ϵ=1.e-7	Central difference ϵ=1.e-5	Five-point stencil ϵ=1.e-3
`ResidualAlpha`	0.9e-7	2.1e-10	8.3e-12
`LinearAlpha`	1.8e-7	3.5e-10	2.3e-12
`GeometricAlpha`	1.1e-7	2.2e-10	3.6e-12

Table 4: Relative errors for the sensitivities (Simplified algorithms).

Simplified algorithm	h=2.e-3 s	h=2.e-4 s
`ResidualAlpha` - reference-free update	6.4e-4	6.6e-5
`LinearAlpha` - reference-free update	1.1e-4	1.0e-6
`ResidualAlpha` - without $\mathbf{k_G}$	1.1e-4	1.0e-6
`LinearAlpha` - without $\mathbf{k_G}$	3.5e-10	5.0e-10
`GeometricAlpha` - with symmetric $\mathbf{K}_t$	1.0e-4	1.0e-4

the finite difference computation is also sensitive to roundoff errors, the relative perturbation ϵ used to compute $\mathbf{x}'_{fd}$ has been selected so as to minimize the difference with respect to $\mathbf{x}'_{sa}$ in each case. Clearly, when the accuracy of the finite difference scheme increases, $\mathbf{x}'_{fd}$ converges to $\mathbf{x}'_{sa}$, which indicates that $\mathbf{x}'_{sa}$ is very close to the exact value of the sensitivity.

The results presented in both Tables 2 and 3 were obtained using the transformation rules (96) which come from the direct differentiation of the update formulae. In Table 4, the results in the first two lines have been obtained using the reference-free update algorithm. A non-negligible error is observed, which however decreases as $\mathcal{O}(h)$ for the residual form and as $\mathcal{O}(h^2)$ for the linear form. Indeed, even though the exact solution is parameterization-independent, the numerical solution is affected by parameterization-dependent numerical errors, which are not taken into account by the reference-free update algorithm. This explains why the resulting errors in the sensitivities decrease as the integration errors of the original problem. In conclusion, physical considerations can be used to simplify an algorithm obtained by direct differentiation, provided that the numerical solution is close enough to the exact solution.

Another simplification is related with the contribution of $\mathbf{k_G}$ to the tangent stiffness matrix in Equation (65). Since $\mathbf{k_G}$ vanishes at the equilibrium $\mathbf{G} = \mathbf{0}$, it seems interesting to neglect it when computing the tangent stiffness. For the `LinearAlpha` algorithm, the equilibrium $\mathbf{G} = \mathbf{0}$ is satisfied at convergence of the nominal problem, so that the contribution $\mathbf{k_G}$ does not significantly affect the convergence of the Newton iterations, and it does not influence the computation of the sensitivities at all. Rigorously, this simplification is not possible for the `ResidualAlpha` algorithm since $\mathbf{G} \neq \mathbf{0}$ at convergence, see Equation (79). In particular, this contribution should not be omitted for the computation of the sensitivities, since it has a direct impact on their accuracy, as shown in Table 4. Again, even though this does not really alter the convergence of the Newton iterations, it has a direct influence on the accuracy of the sensitivities. However, the errors resulting from this abusive simplification quickly decrease with the step-size h.

It can be demonstrated that the parameterized stiffness matrix $\mathbf{k}_t$ is symmetric for static problems. In contrast, the geometric stiffness matrix $\mathbf{K}_t$ is not symmetric. For the implementation of more efficient geometric algorithms, it is a common practice to symmetrize $\mathbf{K}_t$, and a theoretical justification of this operation is developed in [30]. For the sensitivity analysis the symmetrization of $\mathbf{K}_t$ yields a significant error, which does not decrease with the step-size (Table 4). Hence, it is important to use the non-symmetric operator in order to compute the

correct sensitivities.

7 Conclusions

In this paper, a sensitivity analysis has been developed for dynamic systems with finite rotations using a direct differentiation approach. Since no global parameterization is available for arbitrarily large rotations, a consistent geometric framework has been exploited.

Three extensions of the generalized-α algorithm have been considered for dynamic systems evolving on the Lie group $SO(3)$. The residual form and the linear form are based on local parameterizations of $SO(3)$, whereas the geometric form does not require the formulation of the equations of motion as differential equations in local independent coordinates. We have shown that the residual form, which is the most direct extension of the generalized-α method defined in structural dynamics, is no more second-order accurate when applied to systems with large rotations. In contrast, second-order convergence is observed for the linear form. Both algorithms are combined with a reparameterization strategy implemented at the end of each converged time-step. The advantages of the geometric form come from its simple implementation and its efficiency, since the dynamic equilibrium takes a simpler expression and no reparameterization is required. As the linear form, it achieves second-order accuracy but a higher error constant is observed for the heavy top benchmark. We note that more sophisticated geometric extensions of the generalized-α scheme are possible, which might lead to higher levels of accuracy. In any case, the choice of a suitable algorithm should be based on the desired compromise between efficiency and accuracy.

Then, a sensitivity analysis has been derived for those three integration schemes. The concept of directional derivative plays a critical role in this analysis, and close connexions are highlighted between the structure of the sensitivity equations and of the linearized equations. Since the latter are usually available in the simulation code for the purpose of Newton iterations, the additional effort required for the implementation of the sensitivity analysis is quite small. After each converged time-step, the computation of the sensitivities only requires one additional iteration, so that the impact on the overall CPU time is limited. Provided that exact (non-symmetric) tangent operators are used, very high levels of accuracy can be achieved for the sensitivities.

For integrators based on local parameterizations, the reparameterization strategy has to be taken into account in the sensitivity analysis. A simplified reference-free update algorithm has been proposed, but it leads to numerical errors in the sensitivities which are of the same order as the integration errors for the original dynamic problem. In contrast, the choice of a geometric integrator allows a much simpler and computationally more efficient algorithm for the sensitivities.

Acknowledgements

The authors would like to thank Prof. P. Duysinx for stimulating discussions during the preparation of this paper. O. Brüls is supported by the Belgian National Fund for Scientific Research (FNRS) which is gratefully acknowledged. Part of the presented work has been done during his one-year stay at the University of Stuttgart, Germany. This work also presents research results of the Belgian Program on Inter-University Poles of Attraction initiated by the Belgian state, Prime Minister's office, Science Policy Programming. The scientific responsibility rests with its authors.

Appendix A: Derivatives of the operator T

For the implementation of the parameterized algorithms `ResidualAlpha` and `LinearAlpha`, the conformal rotation vector has been selected for the local parameterization of $SO(3)$. In addition

to a limited level of nonlinearity, the coordinate map μ and the tangent operator $\mathbf{T}$ only involve algebraic relations, which leads to simpler and more efficient computations.

The first-order derivative of $\mathbf{T}$ is given by

$$D\mathbf{T}\cdot\mathbf{v} = \frac{2}{(4-c_0)^2}\left(\frac{\mathbf{c}\mathbf{v}^T}{4}+\frac{\mathbf{v}\mathbf{c}^T}{4}-\widetilde{\mathbf{v}}\right)-\frac{\mathbf{c}^T\mathbf{v}}{(4-c_0)^3}\left((2+\frac{c_0}{2})\mathbf{I}+\frac{\mathbf{c}\mathbf{c}^T}{4}-\widetilde{\mathbf{c}}\right). \tag{119}$$

This expression allows to compute $\dot{\mathbf{T}} = D\mathbf{T}\cdot\dot{\mathbf{c}}$, as well as $\mathbf{T}' = D\mathbf{T}\cdot\mathbf{c}'$. For the second-order derivative, we have

$$\begin{aligned} D(D\mathbf{T}\cdot\mathbf{v})\cdot\mathbf{w} \;=\; & \frac{2}{(4-c_0)^2}\left(\frac{\mathbf{w}\mathbf{v}^T+\mathbf{v}\mathbf{w}^T+\mathbf{c}(D\mathbf{v}\cdot\mathbf{w})^T+(D\mathbf{v}\cdot\mathbf{w})\mathbf{c}^T}{4}-\widetilde{D\mathbf{v}\cdot\mathbf{w}}\right) \\ & -\frac{\mathbf{c}^T\mathbf{w}}{(4-c_0)^3}\left(\frac{\mathbf{c}\mathbf{v}^T+\mathbf{v}\mathbf{c}^T}{4}-\widetilde{\mathbf{v}}\right)-\frac{\mathbf{c}^T\mathbf{v}}{(4-c_0)^3}\left(\frac{\mathbf{c}\mathbf{w}^T+\mathbf{w}\mathbf{c}^T}{4}-\widetilde{\mathbf{w}}-\frac{1}{8}\mathbf{I}\mathbf{c}^T\mathbf{w}\right) \\ & +\frac{1}{(4-c_0)^3}\left(\frac{3}{4}\frac{(\mathbf{c}^T\mathbf{v})(\mathbf{c}^T\mathbf{w})}{4-c_0}-\mathbf{w}^T\mathbf{v}-\mathbf{c}^T(D\mathbf{v}\cdot\mathbf{w})\right)\left((2+\frac{c_0}{2})\mathbf{I}+\frac{\mathbf{c}\mathbf{c}^T}{4}-\widetilde{\mathbf{c}}\right). \end{aligned}$$

For $\mathbf{v}=\mathbf{w}=\dot{\mathbf{c}}$, this expression gives the value of $\ddot{\mathbf{T}}$ (in this case, $D\mathbf{v}\cdot\mathbf{w}=\ddot{\mathbf{c}}$), whereas for $\mathbf{v}=\dot{\mathbf{c}}$ and $\mathbf{w}=\mathbf{c}'$, it gives the value of $\dot{\mathbf{T}}'$ (in this case, $D\mathbf{v}\cdot\mathbf{w}=\dot{\mathbf{c}}'$).

An expression can also be obtained for $\mathbf{k_G}$. Indeed, using Equation (119), we get

$$(D\mathbf{T}^T\cdot\mathbf{v})\mathbf{G} = \frac{2}{(4-c_0)^2}\left(\frac{\mathbf{c}\mathbf{v}^T}{4}+\frac{\mathbf{v}\mathbf{c}^T}{4}+\widetilde{\mathbf{v}}\right)\mathbf{G}-\frac{\mathbf{c}^T\mathbf{v}}{(4-c_0)^3}\left((2+\frac{c_0}{2})\mathbf{I}+\frac{\mathbf{c}\mathbf{c}^T}{4}+\widetilde{\mathbf{c}}\right)\mathbf{G} \tag{120}$$

so that

$$\mathbf{k_G}(\mathbf{c},\mathbf{G}) = \frac{2}{(4-c_0)^2}\left(\frac{\mathbf{c}\mathbf{G}^T}{4}+\frac{\mathbf{c}^T\mathbf{G}}{4}\mathbf{I}-\widetilde{\mathbf{G}}\right)-\frac{1}{(4-c_0)^3}\left((2+\frac{c_0}{2})\mathbf{G}+\frac{\mathbf{c}^T\mathbf{G}}{4}\mathbf{c}+\widetilde{\mathbf{c}}\mathbf{G}\right)\mathbf{c}^T. \tag{121}$$

For a parameterization based on the rotation vector $\boldsymbol{\psi}$, the first and second derivatives of the operator $\mathbf{T}$ are developed in [29].

Appendix B: Consistency of the `GeometricAlpha` algorithm

For an algorithm directly defined on $SO(3)$, such as `GeometricAlpha`, the local error should be analysed in any local coordinate system $\boldsymbol{\phi}=\mu^{-1}(\mathbf{R})$. If the exact solution is denoted by $(\boldsymbol{\phi}(t),\boldsymbol{\Omega}(t))$ and the numerical solution after one time step $(\boldsymbol{\phi}_1(h),\boldsymbol{\Omega}_1(h))$, the algorithm is consistent of order two provided that the local errors at position and velocity level satisfy $\boldsymbol{\phi}(h)-\boldsymbol{\phi}_1(h)=\mathcal{O}(h^3)$ and $\boldsymbol{\Omega}(h)-\boldsymbol{\Omega}_1(h)=\mathcal{O}(h^3)$, respectively.

Theorem 1 *If the algorithmic parameters satisfy*

$$\gamma = 0.5+\alpha_f-\alpha_m, \tag{122}$$

and considering a coordinate system $\boldsymbol{\phi}$ centered on $\mathbf{R}_0$ (i.e. $\boldsymbol{\phi}(0)=\mathbf{0}$ and $\mathbf{R}(\mathbf{0})=\mathbf{R}_0$) such that

$$\mathbf{T}(\mathbf{0})=\mathbf{I},\quad D\mathbf{T}(\mathbf{0})\cdot\mathbf{v}=-0.5\widetilde{\mathbf{v}}, \tag{123}$$

the local errors are characterized by

$$\boldsymbol{\phi}(h)-\boldsymbol{\phi}_1(h)=\mathcal{O}(h^3),\quad \boldsymbol{\Omega}(h)-\boldsymbol{\Omega}_1(h)=\mathcal{O}(h^3). \tag{124}$$

We observe that an incremental parameterization based on the rotation vector or on the conformal rotation vector verifies the condition (123). In addition, Equation (122) is the standard second-order accuracy requirement for the generalized-α method in structural dynamics, which is satisfied by the HHT-α and the CH-α schemes.

Proof. The present analysis focuses on the position level condition, since the analysis of the velocity level condition is straightforward. If we consider the Taylor series expansion

$$\boldsymbol{\phi}(h) = \boldsymbol{\phi}(0) + h\frac{d}{dh}\boldsymbol{\phi}(0) + \frac{h^2}{2}\frac{d^2}{dh^2}\boldsymbol{\phi}(0) + \mathcal{O}(h^3), \tag{125}$$

$$\boldsymbol{\phi}_1(h) = \boldsymbol{\phi}_1(0) + h\frac{d}{dh}\boldsymbol{\phi}_1(0) + \frac{h^2}{2}\frac{d^2}{dh^2}\boldsymbol{\phi}_1(0) + \mathcal{O}(h^3). \tag{126}$$

where $\boldsymbol{\phi}_1(0) = \boldsymbol{\phi}(0) = \mathbf{0}$ by construction, the second-order conditions are

$$\frac{d}{dh}\boldsymbol{\phi}_1(0) = \frac{d}{dh}\boldsymbol{\phi}(0), \quad \frac{d^2}{dh^2}\boldsymbol{\phi}_1(0) = \frac{d^2}{dh^2}\boldsymbol{\phi}(0). \tag{127}$$

Let us demonstrate that they are verified if $\boldsymbol{\phi}_1$ is computed by the `GeometricAlpha` algorithm.

Firstly, the exact solution $\boldsymbol{\phi}(h)$ satisfies

$$\boldsymbol{\phi}(h) = \mu^{-1}\left(\mathbf{R}(h)\right) \tag{128}$$

and the first and second differentiation of this expression yield

$$\mathbf{T}_\mu(\boldsymbol{\phi}(h))\frac{d}{dh}\boldsymbol{\phi}(h) = \boldsymbol{\Omega}(h), \tag{129}$$

$$\mathbf{T}_\mu(\boldsymbol{\phi}(h))\frac{d^2}{dh^2}\boldsymbol{\phi}(h) + \frac{d}{dh}\mathbf{T}_\mu(\boldsymbol{\phi}(h))\frac{d}{dh}\boldsymbol{\phi}(h) = \frac{d}{dh}\boldsymbol{\Omega}(h) = -\mathbf{F}(\mathbf{R}(h), \boldsymbol{\Omega}(h), h) \tag{130}$$

where the operator $\mathbf{F}$ is defined by $\mathbf{F}(\mathbf{R}, \boldsymbol{\Omega}, t) = \mathbf{J}^{-1}\left(\boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega} - \widetilde{\mathbf{X}}\mathbf{R}^T\mathbf{f}(t)\right)$. Using Equation (123), this leads to

$$\frac{d}{dh}\boldsymbol{\phi}(0) = \boldsymbol{\Omega}_0, \quad \frac{d^2}{dh^2}\boldsymbol{\phi}(0) = -\mathbf{F}(\mathbf{R}_0, \boldsymbol{\Omega}_0, 0). \tag{131}$$

Secondly, the numerical solution $\boldsymbol{\phi}_1(h)$ is given by

$$\mu(\boldsymbol{\phi}_1(h)) = \exp(\boldsymbol{\psi}(h)) \tag{132}$$

with $\boldsymbol{\psi}(0) = \mathbf{0}$. The first and second differentiations of this expression yield

$$\mathbf{T}_\mu(\boldsymbol{\phi}_1)\frac{d}{dh}\boldsymbol{\phi}_1 = \mathbf{T}_{\exp}(\boldsymbol{\psi})\frac{d}{dh}\boldsymbol{\psi}, \tag{133}$$

$$\mathbf{T}_\mu(\boldsymbol{\phi}_1)\frac{d^2}{dh^2}\boldsymbol{\phi}_1 + \frac{d}{dh}\mathbf{T}_\mu(\boldsymbol{\phi}_1)\frac{d}{dh}\boldsymbol{\phi}_1 = \mathbf{T}_{\exp}(\boldsymbol{\psi})\frac{d^2}{dh^2}\boldsymbol{\psi} + \frac{d}{dh}\mathbf{T}_{\exp}(\boldsymbol{\psi})\frac{d}{dh}\boldsymbol{\psi}. \tag{134}$$

Using Equation (123), we obtain

$$\frac{d}{dh}\boldsymbol{\phi}_1(0) = \frac{d}{dh}\boldsymbol{\psi}(0), \quad \frac{d^2}{dh^2}\boldsymbol{\phi}_1(0) = \frac{d^2}{dh^2}\boldsymbol{\psi}(0). \tag{135}$$

Then, we observe that the value $\boldsymbol{\psi}$ corresponds to the numerical solution obtained when the classical generalized-α method is applied to the fictitious second-order differential equation in $\mathbb{R}^3$

$$\ddot{\boldsymbol{\psi}} + \mathbf{F}(\exp(\boldsymbol{\psi}), \dot{\boldsymbol{\psi}}, t) = \mathbf{0}, \quad (\boldsymbol{\psi}(0), \dot{\boldsymbol{\psi}}(0)) = (\mathbf{0}, \boldsymbol{\Omega}_0). \tag{136}$$

Since the classical generalized-α method with the condition (122) is second-order accurate for nonlinear problems in $\mathbb{R}^n$, see [19], we have

$$\frac{d}{dh}\boldsymbol{\psi}(0) = \boldsymbol{\Omega}_0, \quad \frac{d^2}{dh^2}\boldsymbol{\psi}(0) = -\mathbf{F}(\mathbf{R}_0, \boldsymbol{\Omega}_0, 0). \tag{137}$$

From Equations (131), (135) and (137), we conclude that the second-order conditions (127) are satisfied for the `GeometricAlpha` algorithm. Q.E.D.

References

[1] J. Argyris. An excursion into large rotations. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**:85–155.

[2] M. Arnold and O. Brüls. Convergence of the generalized-α scheme for constrained mechanical systems, *Multibody System Dynamics* 2007; **18**:185–202.

[3] O.A. Bauchau and C.L. Bottasso. On the design of energy preserving and decaying schemes for flexible nonlinear multi-body systems. *Computer Methods in Applied Mechanics and Engineering* 1999; **169**:61–79.

[4] O.A. Bauchau, J.-Y. Choi, and C.L. Bottasso. Time integrators for shells in multibody dynamics. *Computers and Structures* 2002; **80**:871–889.

[5] O. Bauchau and L. Trainelli. The vectorial parameterization of rotation. *Nonlinear Dynamics* 2003; **32**:71–92.

[6] D. Bestle and P. Eberhard. Analyzing and optimizing multibody systems. *Mechanics of Structures and Machines* 1992; **20**:67–92.

[7] D. Bestle and J. Seybold. Sensitivity analysis of constrained multibody systems. *Archive of Applied Mechanics* 1992; **62**:181–190.

[8] P. Betsch and P. Steinmann. Constrained integration of rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering* 2001; **191**:467–488.

[9] W.M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry* (2nd edn). Academic Press, 2003.

[10] C.L. Bottasso and M. Borri. Integrating finite rotations. *Computer Methods in Applied Mechanics and Engineering* 1998; **164**:307–331.

[11] A. Cardona. *An Integrated Approach to Mechanism Analysis.* PhD thesis, Université de Liège, 1989.

[12] A. Cardona and M. Géradin. A beam finite element non-linear theory with finite rotations. *International Journal for Numerical Methods in Engineering* 1988; **26**:2403–2438.

[13] A. Cardona and M. Géradin. Time integration of the equations of motion in mechanism analysis. *Computers and Structures* 1989; **33**:801–820.

[14] E. Celledoni and B. Owren. Lie group methods for rigid body dynamics and time integration on manifolds. *Computer Methods in Applied Mechanics and Engineering* 2003; **192**(3-4):421–438.

[15] J. Chung and G.M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. *ASME Journal of Applied Mechanics* 1993; **60**:371–375.

[16] P.E. Crouch and R. Grossman. Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science* 1993; **3**:1–33.

[17] J.M.P. Dias and M.S. Pereira. Sensitivity analysis of rigid-flexible multibody systems. *Multibody System Dynamics* 1997; **1**(3):303–322.

[18] P. Eberhard and C. Bischof. Automatic Differentiation of numerical integration algorithms. *Mathematics of Computation* 1999; **68**:717–731.

[19] S. Erlicher, L. Bonaventura, and O.S. Bursi. The analysis of the generalized-α method for non-linear dynamic problems. *Computational Mechanics* 2002; **28**:83–104.

[20] M. Géradin and D. Rixen. *Mechanical Vibrations: Theory and Application to Structural Dynamics*. John Wiley & Sons, 1997.

[21] M. Géradin and A. Cardona. *Flexible Multibody Dynamics: A Finite Element Approach*. John Wiley & Sons, New York, 2001.

[22] R.T. Haftka and H.M. Adelman. Recent developments in structural sensitivity analysis. *Structural Optimization* 1989; **1**:137–151.

[23] E.J. Haug, K.K. Choi, and V. Komkov. *Design Sensitivity Analysis of Structural Systems*. Academic Press, Orlando, 1986.

[24] A. Ibrahimbegovic and S. Mamouri. Energy conserving/decaying implicit time-stepping scheme for nonlinear dynamics of three-dimensional beams undergoing finite rotations. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**:4241–4258.

[25] P. Kowalczyk. Design sensitivity analysis in large deformation elasto-plastic and elasto-viscoplastic problems. *International Journal for Numerical Methods in Engineering* 2006; **66**:1234–1270.

[26] D. Kuhl and M. Crisfield. Energy-conserving and decaying algorithms in non-linear structural dynamics. *International Journal for Numerical Methods in Engineering* 1999; **45**:569–599.

[27] P. Michaleris, D.A. Tortorelli, and C.A Vidal. Tangent operators and design sensitivity formulations for transient non-linear coupled problems with applications to elastoplasticity. *International Journal for Numerical Methods in Engineering* 1994; **37**:2471–2499.

[28] H. Munthe-Kaas. Lie-Butcher theory for Runge-Kutta methods. *BIT* 1995; **35**:572–587.

[29] M. Ritto-Corrêa and D. Camotim. On the differentiation of the Rodrigues formula and its significance for the vector-like parameterization of Reissner-Simo beam theory. *International Journal for Numerical Methods in Engineering* 2002; **55**:1005–1032.

[30] J.C. Simo. The (symmetric) Hessian for geometrically nonlinear models in solid mechanics: Intrinsic definition and geometric interpretation. *Computer Methods in Applied Mechanics and Engineering* 1992; **96**:189–200.

[31] J.C. Simo and D.D. Fox. On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization. *Computer Methods in Applied Mechanics and Engineering* 1989; **72**:267–304.

[32] J.C. Simo and L. Vu-Quoc. On the dynamics in space of rods undergoing large motions - a geometrically exact approach. *Computer Methods in Applied Mechanics and Engineering* 1988; **66**:125–161.

[33] J.C. Simo and K. Wong. Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *International Journal for Numerical Methods in Engineering* 1991; **31**:19–52.

[34] D.A. Tortorelli. Sensitivity analysis for non-linear constrained elastoplastic systems. *International Journal for Numerical Methods in Engineering* 1992; **33**:1643–1660.

[35] F. Van Keulen, R.T. Haftka, and N. Kim. Review of options for structural design sensitivity analysis. Part 1: Linear systems. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:3213–3243.

[36] T.M. Wasfy and K. Noor. Modeling and sensitivity analysis of multibody systems using new solid, shell and beam elements. *Computer Methods in Applied Mechanics and Engineering* 1996; **138**:187–211.