Painless Differentiable Rotation Dynamics

MAGÍ ROMANYÀ-SERRASOLSAS, Universidad Rey Juan Carlos, Spain
JUAN J. CASAFRANCA, Universidad Rey Juan Carlos, Spain
MIGUEL A. OTADUY, Universidad Rey Juan Carlos, Spain

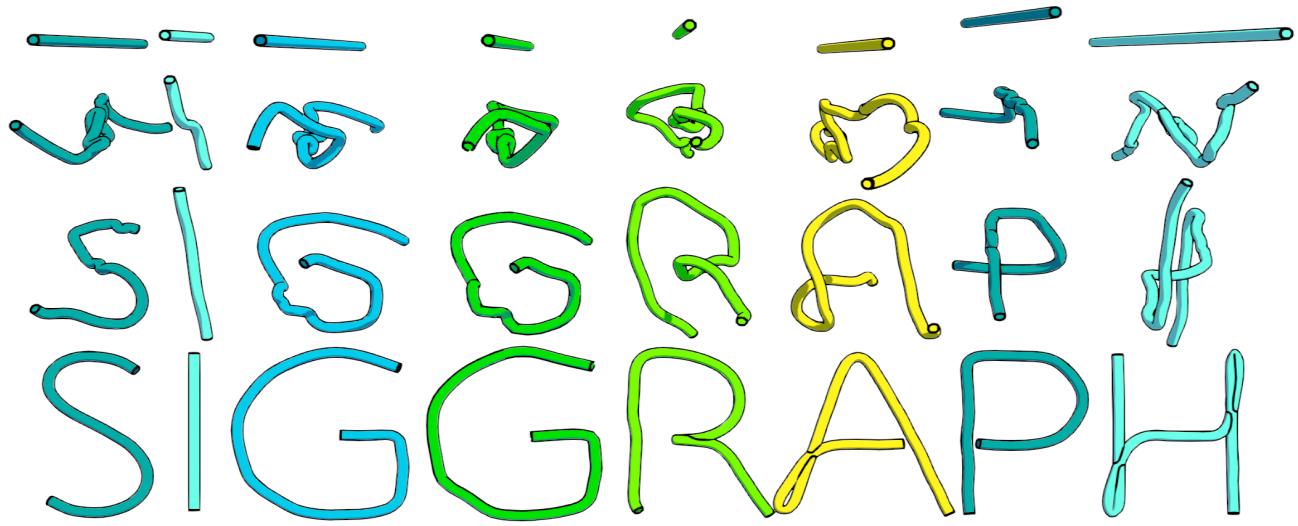

Fig. 1. We optimize the initial velocities of straight rods (top row), such that after a few simulation frames they spell "SIGGRAPH". Our approach based on Lie derivatives of rotations converges in just 1m 21s, 4.8× faster than a formulation based on rotation vectors.

We propose the formulation of forward and differentiable rigid-body dynamics using Lie-algebra rotation derivatives. In particular, we show how this approach can easily be applied to incremental-potential formulations of forward dymamics, and we introduce a novel definition of adjoints for differentiable dynamics. In contrast to other parameterizations of rotations (notably the popular rotation-vector parameterization), our approach leads to painlessly simple and compact derivatives, better conditioning, and higher runtime efficiency. We demonstrate our approach on fundamental rigid-body problems, but also on Cosserat rods as an example of multi-rigid-body dynamics.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Rigid bodies, dynamic simulation, differentiable simulation.

Authors' addresses: Magí Romanyà-Serrasolsas, Universidad Rey Juan Carlos, Tulipán s/n, Móstoles, Madrid, 28933, Spain, magi.romanya@urjc.es; Juan J. Casafranca, Universidad Rey Juan Carlos, Tulipán s/n, Móstoles, Madrid, 28933, Spain, jjcasmar@gmail.com; Miguel A. Otaduy, Universidad Rey Juan Carlos, Tulipán s/n, Móstoles, Madrid, 28933, Spain, miguel.otaduy@urjc.es.

0730-0301/2025/8-ART
https://doi.org/10.1145/3730944

ACM Reference Format:
Magí Romanyà-Serrasolsas, Juan J. Casafranca, and Miguel A. Otaduy. 2025. Painless Differentiable Rotation Dynamics. *ACM Trans. Graph.* 44, 4 (August 2025), 13 pages. https://doi.org/10.1145/3730944

1 INTRODUCTION

It is well known that the nonlinear nature of the space of rotations imposes intrinsic complexity in the representation and computation of rigid-body motion. Unlike translations, which are straightforward to handle in a Euclidean space, rotations are described by the special orthogonal group $SO(3)$. Rotations admit many representations, but all come with challenges. Euler angles suffer singularities; rotation matrices require orthogonality and unit determinant; quaternions require normalization. These challenges become more severe when functions must be differentiated in the space of rotations $SO(3)$. While rotations are linear (resp. quadratic) operations with respect to rotation matrices (resp. quaternions), and apparently easy to differentiate, derivatives must also be constrained, and quantities computed in the differential domain cannot simply be added, due to the nonlinear nature of $SO(3)$.

Rotation vectors (a.k.a. axis-angle) have gained popularity in recent years in computer graphics. The reason is simple: rotation vectors are parameterized in $\mathbb{R}^3$, and do not require additional constraints to maintain validity. Quantities computed in differential rotation-vector space can simply be added together and still define new valid rotations through the exponential map, which is

compactly computed using the Rodrigues rotation formula. Unfortunately, the nonlinearity of the Rodrigues formula leads to "derivative hell". We invite the reader to look at details in appendices and supplementary material of high-impact works on both forward dynamics (notably Rigid IPC [Ferguson et al. 2021]) and differentiable dynamics (notably ADD [Geilinger et al. 2020]). Authors often rely on automatic differentiation, but at the expense of computational cost and special Taylor approximations to handle indeterminate forms.

Interestingly, differentiation of rotations and with respect to rotations can be easily handled with a tool largely overlooked in computer graphics: Lie theory. Admittedly, there are notable exceptions which have successfully used Lie theory for simulation in computer graphics, e.g., [Kobilarov et al. 2009; Soliman et al. 2024]. The fundamental realization is that differentiation of rotations can be carried out on a local vector tangent space of $SO(3)$. Differential quantities are not rotations in themselves, but can easily be converted to rotations through the exponential map and then composed with other rotations. In Section 3 we summarize the fundamentals of Lie-algebra rotation derivatives, and we explore in contrast the intrinsic challenges of rotation-vector differentiation.

In this work, we make two important contributions to forward and differentiable rigid-body dynamics. First, as described in Section 4, we detail the application of Lie derivatives to implicit integration of rigid-body dynamics based on incremental-potential formulations. We show that the approach produces derivatives (gradients and Hessians) that are embarrassingly simple in contrast to rotation vectors.

Our second contribution has its major impact in differentiable rigid-body dynamics, but stems from a small yet crucial choice in forward dynamics. State-of-the-art forward simulation methods formulate the time-step update as the computation of state and velocity, and this turns into state and velocity adjoints in the context of differentiable simulation. Instead, we formulate the time-step update as the computation of state and step (i.e., the state change between two time steps). Thanks to this apparently simple choice, both state and step can be represented using rotations, and we fully unleash the power of Lie derivatives in the computation of state and step adjoints. As we show in Section 5, this produces again embarrassingly simple derivatives.

Our Lie-derivative formulations of forward and differentiable rigid-body dynamics come with several important and fundamental benefits. (i) Simplicity of derivative functions. Derivatives are almost as simple as those of state variables defined in Euclidean space. This allows painless, exact, analytical computation without indeterminate forms, and it also enhances interpretability. (ii) Better numerical conditioning. We avoid conditioning problems and spurious indefiniteness caused by the rotation-vector parameterization. (iii) Computational speed-up. The speed-up arises thanks to a reduction in solver iteration count, as well as far fewer per-iteration operations for the computation of derivatives.

In addition to rigid-body simulation, we demonstrate the application of our method to Cosserat rod dynamics [Pai 2002; Spillmann and Teschner 2007]. Cosserat rods serve as example of multi-rigid-body parameterization of complex kinematics and dynamics. In Section 6 we discuss the application of our method to Cosserat rods and, more generally, elastic and damping potentials dependent on rotations.

We also demonstrate that our method is general and serves all use-cases of differentiable simulation. We have tested the correctness and performance of the method for the optimization of rest shape, initial conditions (e.g., Fig. 1), and control forces. In addition, throughout the paper we compare our formulation to rotation vectors from different perspectives. We analyze basic properties via eigen-analysis of fundamental derivatives, we compare the complexity of simulation gradients and Hessians, and we compare simulation convergence and computational cost.

The full implementation of our formulation and all the examples shown in the paper are available at https://gitlab.com/mslab-urjc/mandos/-/tree/SIGG25. We believe they evidence the painless implementation of both forward and differentiable dynamics.

2 RELATED WORK

2.1 Differentiable Simulation

Optimization subject to dynamics is a well-known problem, which can be formulated in the continuum as a constrained PDE. The adjoint method [Lions 1971] is a classic and general solution, which allows an efficient computation of the optimization gradient by traversing the simulation backwards. The ability to compute simulation gradients wrt optimization parameters has been termed *differentiable simulation* in computer graphics. McNamara et al. [2004] were the first to apply the adjoint method in computer graphics. They did so for fluid control, including free-surface liquids. Wojtan et al. [2006] studied the application of the adjoint method to particle systems with implicit integration. For an introduction to differentiable simulation in computer graphics and an overview of recent research, we recommend the course by Coros et al. [2021]. For the optimization of initial conditions in rigid-body simulations, Twigg and James [2008] introduced an interesting alternative based on backward time-stepping.

Much of the recent work on differentiable simulation has focused on differentiating complex computational processes of the simulation algorithm. Some examples are continuous collision detection and complementarity-based contact [Liang et al. 2019], multi-body dynamics with frictional contact [Geilinger et al. 2020], articulated bodies [Qiao et al. 2021b], projective dynamics [Du et al. 2021], soft articulated bodies with projective dynamics [Qiao et al. 2021a], projective dynamics with dry friction [Li et al. 2022], or extended position-based dynamics (XPBD) [Stuyck and Chen 2023] among others. Huang et al. [2024] present a comprehensive differentiable solver that covers arbitrary optimization arguments, unified treatment of dynamics and static problems, arbitrary temporal and spatial discretization, and an IPC formulation of general frictional contact. Our approach to handle rotation derivatives and formulate the step update for differentiability could complement their method.

The large applicability of differentiable simulation for training AI controllers (e.g., in robotics) has motivated efforts on fully differentiable simulation engines. One notable example is DiffTaichi [Hu et al. 2020], which offers a programming language to express differentiable simulation and automatically compile as GPU-parallel code. A related effort is the combination of differentiable simulation and

differentiable rendering, to produce a fully differentiable system of physics-based image generation [Murthy et al. 2021]. Warp [Macklin 2022] enables this from Python and at an industry scale, building on top of massively parallel GPU-based differentiable simulation [Xu et al. 2022].

Most differentiable simulation algorithms focus on the efficient and accurate computation of gradients. Then, gradients are used in a (possibly stochastic) gradient-based optimization method (e.g. L-BFGS or Adam). There are some exceptions. One is SGN [Zehnder et al. 2021], which uses Gauss-Newton optimization, and avoids the assembly and solve of the dense Gauss-Newton matrix by solving instead a sparse saddle-point problem. Another one is X-Shells [Panetta et al. 2019], which uses full Newton optimization, and computes Hessian-vector products by solving for adjoint variations wrt parameter variations.

2.2 Lie Theory

The study and applications of Lie theory go well beyond the scope of this paper. We largely share the focus of the introductory paper by Solà et al. [2021], who are interested in Lie theory as a tool for differentiation with rotations. Other sources cover Lie theory in more depth and breadth [Howe 1983; Stillwell 2008], but Solà et al. cover just the sufficient material for our use case.

Lie-group integrators are a particular application of Lie theory to design variational integrators on kinematic spaces described by Lie groups [Bou-Rabee and Marsden 2009; Lee et al. 2007]. The fundamental idea is that derivatives are expressed in a local tangent space of the Lie group (i.e., the Lie algebra). For rotations, the time derivative in this local tangent space is nothing else but angular velocity. See Section 3 for a more detailed discussion of Lie-algebra derivatives of rotations.

Some notable works in computer graphics have contributed to Lie-group integration. Kobilarov et al. [2009] designed Lie-group integrators for systems with general nonholonomic constraints, i.e., constraints on time-derivatives of the state. Their detailed derivation is limited to symplectic Euler, while we address implicit integration following the incremental-potential optimization formulation [Kane et al. 2000; Li et al. 2020]. Very recently, Soliman et al. [2024] have used Lie-group integrators for rigid-body dynamics in incompressible media. We go a step further in using Lie theory to differentiate functions of rotations in forward and differentiable simulation.

With the general view of differentiable simulation as a constrained optimization problem, Lie-group integrators have also been used in the context of optimal control [Kobilarov and Marsden 2011; Lee et al. 2008]. Our paper brings a fundamental novelty of expressing step-update constraints as rotations, which allows us to further leverage Lie theory in the derivation of constraint Jacobians needed for adjoint computation.

3 LIE-ALGEBRA ROTATION DERIVATIVES

3.1 Lie Group and Lie Algebra

Rotations are the transformations that preserve length, and are denoted as $SO(3)$ in 3D. Rotations form a group because they admit the operation of composition, i.e., the composition of two rotations is also a rotation. The identity element of the group is the identity rotation. Rotations also form a Lie group, because they define a differentiable manifold. This differentiability property is the essence of Lie algebra and the formulation of rotation derivatives.

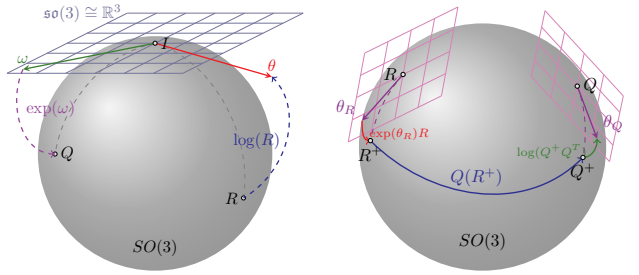

Fig. 2. Schematic representation of the Lie group $SO(3)$ along with its Lie algebra $\mathfrak{so}(3)$. Left: two rotation vectors θ and ω and their relationship with the rotation matrices R and Q through the exponential and log maps. Right: Graphical view of the Lie derivative as defined in equation (2). Note that $SO(3)$ is not the surface of a 3D sphere; here we represent it this way for illustrative purposes.

As anticipated in the introduction, rotations admit various representations. However, the choice of representation comes with challenges. Rotation matrices require the constraint of orthogonality, quaternions require the constraint of normalization, and rotation vectors are not unique. Following regular calculus, one would compute rotation derivatives using regular derivatives with respect to the choice of representation, but this is problematic because one has to enforce constraints on derivatives, or the derivatives are not well behaved, as we will demonstrate for rotation vectors at the end of this section.

Lie algebra is a tool for calculus (i.e., for differentiation) of Lie groups, which defines a robust mechanism for expressing differentiable elements of the group. The Lie algebra of rotations, denoted $\mathfrak{so}(3)$, is the tangent space to $SO(3)$ at the identity rotation. The motivation to define derivatives (i.e., the tangent space) at the identity element can be seen intuitively as follows. A rotation R can be trivially expressed as the composition of identity and itself, $R = I\,R$. By taking a differential rotation at identity, I^+, group composition effectively produces a differential rotation for an arbitrary rotation $R^+ = I^+\,R$.

The Lie algebra of rotations $\mathfrak{so}(3)$ is formed by the space of skew-symmetric matrices. We need two additional operations, to map elements to-from $\mathfrak{so}(3)$ and $SO(3)$. These operations are the exp and log maps. Fig. 2-left depicts the mappings to-from the Lie group and the Lie algebra.

$$\exp : \mathfrak{so}(3) \to SO(3) \qquad \log : SO(3) \to \mathfrak{so}(3) \tag{1}$$

A key feature of the Lie algebra is that it is a linear vector space; for instance, $\mathfrak{so}(3) \cong \mathbb{R}^3$. Moreover, this vector space has dimensions equal to the degrees of freedom of the parent manifold. Specifically, $\mathfrak{so}(3)$ can be parameterized using differential rotation vectors: $\theta \in \mathbb{R}^3 \to \text{skew}(\theta) \in \mathfrak{so}(3)$.

At this point, we gather the four elements necessary for the definition of Lie derivatives:

- The definition of derivatives (i.e., the tangent space) at identity rotations.
- The parameterization of this tangent space using rotation vectors.
- The mapping from tangent space to full elements using the exponential map.
- The composition of rotations to define differentials on full rotaion elements.

3.2 Lie Derivatives

We are interested in the derivatives of functions whose independent variables represent rotations $R \in SO(3)$, and the dependent variables may represent either vectors $u \in \mathbb{R}^n$ or other rotations $Q \in SO(3)$. For an independent-variable rotation R, we define a differential vector $\theta_R \in \mathbb{R}^3$, which parameterizes the differential rotation at identity. Via the exp map and rotation composition, we obtain a new rotation $R^+ = \exp(\theta_R)\, R \in SO(3)$.

Similarly, for a dependent-variable rotation Q, we define another differential vector θ_Q, and we have a new rotation $Q^+ = \exp(\theta_Q)\, Q$. Via the log map, we can extract the differential vector $\theta_Q = \log(Q^+\, Q^T)$. The Lie derivative $\frac{\mathcal{D}Q}{\mathcal{D}R}$ is nothing else but the regular vector-calculus derivative $\frac{\partial \theta_Q}{\partial \theta_R}$ evaluated at $\theta_R = 0$. It can be fully expressed as:

$$\frac{\mathcal{D}Q}{\mathcal{D}R} \equiv \frac{\partial \theta_Q}{\partial \theta_R} = \left.\frac{\partial \log\left(Q(\exp(\theta_R)\, R)\, Q(R)^T\right)}{\partial \theta_R}\right|_{\theta_R=0}. \tag{2}$$

This derivative connects the tangent spaces of R and Q, by mapping variations on the tangent space at R to variations on the tangent space at Q. This is depicted in Fig. 2-right.

When the dependent variable is a vector u, the composition operation is regular vector summation, and the tangent space represents the vector space itself. The Lie derivative $\frac{\mathcal{D}u}{\mathcal{D}R}$ is then nothing else but the regular vector-calculus derivative $\frac{\partial u}{\partial \theta_R}$ at $\theta_R = 0$, and we have:

$$\frac{\mathcal{D}u}{\mathcal{D}R} \equiv \frac{\partial u}{\partial \theta_R} = \left.\frac{\partial u(\exp(\theta_R)\, R)}{\partial \theta_R}\right|_{\theta_R=0}. \tag{3}$$

Throughout the paper, we make use of Lie derivatives in several occasions. It is worth making a few observations about how we obtain those derivatives. For Lie derivatives of rotations, i.e., of the form $\frac{\mathcal{D}Q}{\mathcal{D}R}$, the exponential and log map cancel out after some operations. For Lie derivatives of vectors, i.e., of the form $\frac{\mathcal{D}u}{\mathcal{D}R}$, it suffices to approximate the exponential map $\exp(\theta_R \to 0)$ with the degree of the derivative to be computed. Last, we remark that in our notation we left-multiply by differential rotations, which conveys that differential rotations are expressed in the world frame. Alternatively, one could right-multiply by differential rotations, which conveys that they are expressed in the local frame.

3.3 Comparison with Rotation Vectors

A rotation matrix $R \in SO(3)$ can be parameterized by a rotation vector $r \in \mathbb{R}^3$ via the exponential map, $R = \exp(r)$. Using regular calculus instead of the Lie algebra, any arbitrary function $f(R(r))$ can be differentiated using the chain rule as $\frac{\partial f}{\partial r} = \frac{\partial f}{\partial R}\frac{\partial R}{\partial r}$. The

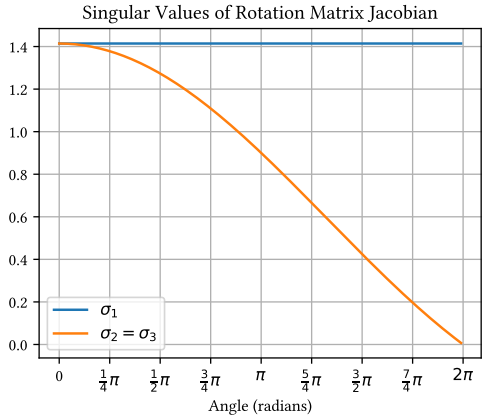

Fig. 3. Singular values σ_1 to σ_3 of the Jacobian of the exponential map $\frac{\partial \text{vec}(\exp(r))}{\partial r}$, for angles $\|r\| \in [0, 2\,\pi]$. The decrease (and full cancellation at $\|r\| = 2\,\pi$) of σ_2 and σ_3 reveals the challenges of rotation vectors.

analysis of $\frac{\partial R}{\partial r}$ reflects the complexity of choosing rotation vectors when differentiability of rotations is required.

We follow Ferguson et al. [2021] to obtain $\frac{\partial R}{\partial r}$. An alternative is the expression by Gallego and Yezzi [2015], which is used by Geilinger et al. [2020]. We can write the Rodrigues rotation formula as (Note that there is a typo in Ferguson's text):

$$R(r) \equiv \exp(r) = I + \sigma(r)\,\text{skew}(r) + \frac{1}{2}\,\sigma^2\left(\frac{r}{2}\right)\,\text{skew}(r)^2, \tag{4}$$

where $\sigma(r) = \text{sinc}(\|r\|)$, and skew() builds an antisymmetric cross-product matrix from a vector. We obtain the derivative:

$$\begin{aligned}\frac{\partial R}{\partial r} &= \text{skew}(r)\,\sigma'(r)\,\frac{r}{\|r\|} + \sigma(r)\,\text{skew}'(r) \\ &+ \frac{1}{2}\,\text{skew}(r)^2\,\sigma\left(\frac{r}{2}\right)\,\sigma'\left(\frac{r}{2}\right)\,\frac{r}{\|r\|} + \sigma^2\left(\frac{r}{2}\right)\,\text{skew}(r)\,\text{skew}'(r).\end{aligned} \tag{5}$$

Ferguson et al. provide details to robustly compute $\text{sinc}'()$ as it becomes indeterminate. $\text{skew}'(r)$ is a trivial constant 3D tensor. $\frac{\partial \text{vec}(R)}{\partial r}$ is a rank-3 9×3 matrix. In Fig. 3 we plot its singular values as a function of the angle of the rotation vector, $\|r\|$. Note how the singular values decrease progressively, and $\sigma_2 = \sigma_3$ even vanish at $\|r\| = 2\,\pi$. Based on this result, in all experimental comparisons in the paper, we express rotation vectors in the range $\|r\| \leq \pi$. The properties of $\frac{\partial \text{vec}(R)}{\partial r}$ explain the complexity of rotation vectors. Not only are derivatives far more complex, but they also hurt solver convergence due to poor conditioning as the rotation vector deviates from 0.

4 FORWARD SIMULATION

Given a framework to compute derivatives with rotations, we wish to apply this to both forward and backward (differentiable) simulation of dynamics. To do this, we start by parameterizing both the state and the step of rotational components of a simulation by rotation matrices. With such parameterization, we can then systematically apply Lie derivatives whenever a function must be differentiated wrt a rotation.

We continue the section with a recap of optimization time integration, following the fundamentals of Ferguson et al. [2021] for rigid-body dynamics, but highlighting differences due to our rotation-matrix parameterization of state and step. Then we show how the nonlinear equations of numerical integration are solved with Newton's method and Lie derivatives. And we conclude by discussing analytical and experimental differences wrt a rotation-vector parameterization.

4.1 State and Step

Let us denote the state of a simulation as q. We pay attention to portions of the state describing 3D frames (with position x and orientation R), but the simulation state may also gather particle positions, FEM nodal discretizations, reduced/subspace representations, etc. In a forward simulation, state is discretized at time samples, which yields a sequence of states $\{q_k\}$.

Let us also denote the step of a simulation as Δq, where the step is the change of state between simulation samples. This also yields a sequence of steps $\{\Delta q_k\}$. For 3D frame positions, particles, FEM discretization nodes, or other Euclidean-space state x, the step is simply the state difference $\Delta x_k = x_k - x_{k-1}$. For rotations, we define the step as the incremental rotation between two states:

$$\Delta R_k = R_k \, R_{k-1}^T. \tag{6}$$

Without loss of generality, each integration step takes as input the state and step from the previous time step $(q_{k-1}, \Delta q_{k-1})$, and yields the new state and step $(q_k, \Delta q_k)$. A common alternative to the step is to use velocity or the time-derivative of state. However, using the step allows us to express all orientation-related kinematics using rotations, and hence to fully leverage the Lie derivatives from Section 3. This will utterly simplify the derivation of adjoints, as we will later see in Section 5.

4.2 Optimization Time Integration

To simulate forward dynamics, we use the optimization formulation of backward Euler integration based on incremental potentials [Gast et al. 2015; Kane et al. 2000; Li et al. 2020; Martin et al. 2011]. Note that this approach can be extended to other integration schemes [Brown et al. 2018]. The computation of the simulation state q_k is formulated as the optimization:

$$q_k = \arg\min \Psi(q_k, q_{k-1}, \Delta q_{k-1}). \tag{7}$$

The objective Ψ gathers the potential energy and the incremental-potential formulations of inertial and dissipative terms [Brown et al. 2018; Ferguson et al. 2021; Li et al. 2020].

Optimality of (7) yields the nonlinear equation system for forward dynamics:

$$\frac{\partial \Psi}{\partial q_k} = 0. \tag{8}$$

For rotation components of the state R_k, the gradient in (8) is the Lie derivative $\frac{\mathcal{D}\Psi}{\mathcal{D}R_k} \equiv \frac{\partial \Psi}{\partial \theta_{R_k}}$. Recall from Section 3 that this implies that rotations are locally parameterized by tangent-space vectors. Then, the physical meaning of the rotational gradient terms is nothing else but the torque, and each rotation component of state naturally yields 3 nonlinear equations in (8). This is a general result of Lie-group integrators [Soliman et al. 2024].

Let us provide some detail about the formulation of inertial rotational terms in the backward-Euler objective Ψ. We follow Ferguson et al. [2021], who express an explicit rotation update $\tilde{R}_k = R_{k-1} + h\,\dot{R}_{k-1}$, where $\dot{R}$ stands for the rotation-matrix derivative and h is the time step. Ferguson et al. also add external forces, but we prefer to exclude them here and add them instead to potential-energy terms. With the explicit rotation update $\tilde{R}_k$ and inertia matrix J as defined by Ferguson et al., each rotation component of the state contributes to the backward-Euler objective Ψ an inertial term:

$$\Psi_R = -\frac{1}{h^2} \operatorname{tr}\left(R_k \, J \, \tilde{R}_k^T\right). \tag{9}$$

Note that we discard the constant term $\frac{1}{2}\operatorname{tr}\left(R_k \, J \, R_k^T\right)$, included by Ferguson et al., as it does not contribute to the result of the optimization. We also divide their objective by h^2 to retain the physical meaning of energy.

Recall from Section 4.1 above that we time-discretize the simulation using step and not velocities. Then, in contrast to Ferguson et al., we rewrite the explicit rotation update as:

$$\tilde{R}_k = \left(2\,I - \Delta R_{k-1}^T\right) R_{k-1}. \tag{10}$$

This expression highlights the dependency of the objective (7) wrt the new state, the old state, and the old step.

4.3 Solution to the Nonlinear Equations

We use Newton's method with line-search to optimize (7) and compute the new state q_k. For each Newton iteration, given a current guess q_k^*, we compute a tentative update δq_k by solving:

$$\frac{\partial^2 \Psi}{\partial {q_k}^2} \, \delta q_k + \frac{\partial \Psi}{\partial q_k}^T = 0. \tag{11}$$

For rotational components of q_k, we formulate the equation using Lie derivatives. This implies (i) using Hessians of the form $\frac{\mathcal{D}^2\Psi}{\mathcal{D}{R_k}^2}$, and (ii) parameterizing the iteration update δ_{R_k} as a rotation tangent-space vector. Then, once (11) is solved, we can apply a line-search weight α on the tentative rotational update δ_{R_k}, and recompute the rotation guess as $R_k^* \leftarrow \exp(\alpha\,\delta R_k)\, R_k^*$.

4.4 Comparison with Rotation Vectors

We argue that Lie derivatives (with their intrinsic tangent-space-vector parameterization of rotations) provide much simpler and better-conditioned derivatives of the incremental-potential objective (7) than rotation vectors. Next, we show this explicitly for the rotational inertial term (9).

With Lie derivatives, we obtain the gradient $\frac{\mathcal{D}\Psi_R}{\mathcal{D}R_k} \equiv \frac{\partial \Psi_R}{\partial \theta_{R_k}}$ (resp. the Hessian $\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}{R_k}^2} \equiv \frac{\partial^2 \Psi_R}{\partial {\theta_{R_k}}^2}$) using regular differentiation rules and a first-order (resp. second-order) approximation of $\exp(\theta_{R_k} \to 0)$. Please see Appendix A for details of the derivation.

$$\frac{\mathcal{D}\Psi_R}{\mathcal{D}R_k} \equiv \frac{\partial \Psi_R}{\partial \theta_{R_k}} = 2\,\operatorname{skew}^{-1}(A_R)^T, \tag{12}$$

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}{R_k}^2} \equiv \frac{\partial^2 \Psi_R}{\partial {\theta_{R_k}}^2} = \operatorname{tr}(S_R)\, I - S_R, \tag{13}$$

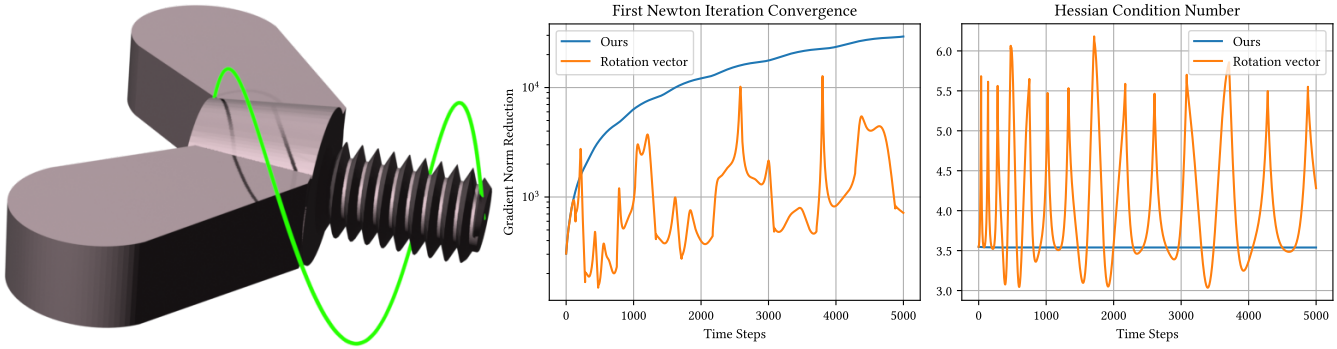

Fig. 4. Comparison of a rigid-body simulation with our rotation derivatives vs. rotation vectors. We simulate gravity-free dynamics of a butterfly bolt (left) with initial velocity, and confirm that it satisfies the tennis-racket theorem with just one Newton iteration per simulation step, while a formulation based on rotation vectors requires two Newton iterations. This result can be justified in two ways: (i: middle) our formulation achieves a larger reduction of the residual of (8) in the first Newton iteration, and (ii: right) our formulation improves the condition number of the Hessian of (9).

where $\frac{1}{h^2} R_k J \tilde{R}_k^T = S_R + A_R$ is a decomposition into symmetric and antisymmetric matrices S_R and A_R, and $\text{skew}^{-1}()$ is the inverse operation of skew() to extract a vector from an antisymmetric matrix.

With rotation vectors r_k, we have $\Psi_R = -\frac{1}{h^2} \text{tr}\left(\exp(r_k) J \tilde{R}_k^T\right)$. This expression can be differentiated by substituting the Rodrigues rotation formula (4) and its derivative (5), and applying the robust derivative of sinc() as proposed by Ferguson et al. However, the derivatives suffer an explosion of complexity in contrast to (12) and (13) above (Please see Appendix B for details of the resulting derivatives). Moreover, Ferguson et al. [2021] report the need for stabilization because of the rotation-vector parameterization, which may turn the Hessian of the inertial term indefinite. This has never been a problem in practice with our Lie derivatives.

We have analyzed experimentally the positive impact of Lie derivatives on forward simulation. Fig. 4-left shows a gravity-free butterfly bolt which we simulate rigid-body dynamics with an initial velocity. With Lie derivatives, the simulation satisfies the tennis racket theorem (i.e., the rotation around the intermediate principal axis of the inertia tensor is not stable) with just one Newton iteration per step. With rotation vectors, on the other hand, the simulation requires two Newton iterations. We have tested this same example on the available implementation of rigid IPC [Ferguson et al. 2021]. The simulation suffers artifacts when the rotation approaches 2π, probably because of a Newton convergence problem reported in their paper, which manifests only in this inertia-dominated scenario. Please see the accompanying video for visual comparisons. Fig. 4-middle compares the reduction in the residual of (8) after the first Newton iteration, and Fig. 4-right compares the condition number of the Hessian. We have observed that Lie derivatives contribute a reduction in Newton iterations in situations where conditioning is dominated by rotations, but not in others dominated by stretch stiffness.

5 DIFFERENTIABLE SIMULATION

We turn our attention now to differentiable simulation, to discuss the role of Lie derivatives in the computation of adjoints for rotational components of the simulation. A differentiable simulation can be regarded, in general terms, as an optimization problem with some objective $g(\{q_k\}, \gamma)$ and optimization parameters γ (e.g., control parameters, material parameters, rest shape, etc). Each simulation step update poses two sets of constraints on the optimization: (i) the optimality condition of discrete forward dynamics (8), and (ii) the computation of the step from old and new states. Let us write the full optimization problem with explicit dependencies of state, step and parameters:

$$\gamma = \arg\min g(\{q_k\}, \gamma), \tag{14}$$

$$\text{s. t. } \frac{\partial \Psi(q_k, q_{k-1}, \Delta q_{k-1}, \gamma)}{\partial q_k} = 0, \ \forall k \tag{15}$$

$$\text{and } \Delta q_k = f(q_k, q_{k-1}), \ \forall k. \tag{16}$$

Next, we discuss the differentiation of the step update (16), the computation of state and step Jacobians wrt optimization parameters, and the role of these Jacobians in the computation of adjoints.

5.1 Differentiation of the Step

The computation of the step (16) is simply $\Delta x_k = x_k - x_{k-1}$ for vector components of the state. Then, the Jacobians of the step update are trivial: $\frac{\partial f}{\partial x_k} = I$ and $\frac{\partial f}{\partial x_{k-1}} = -I$.

For rotational components of the state, the step update is defined as in (6). Using Lie derivatives, we obtain strikingly simple expressions (please see Appendix C): $\frac{\partial f}{\partial R_k} \equiv \frac{\mathcal{D}\Delta R_k}{\mathcal{D} R_k} = I$ and $\frac{\partial f}{\partial R_{k-1}} \equiv \frac{\mathcal{D}\Delta R_k}{\mathcal{D} R_{k-1}} = -\Delta R_k$.

This result reveals the power of our choice of step as parameterization of the time-discretization of the simulation. As mentioned in Section 4.1, a common alternative is to use velocity, e.g., angular velocity ω for rotational components of state. This would yield an update constraint of the form $R_k = \exp(h\,\omega)\, R_{k-1}$. Unfortunately, differentiating this constraint wrt the angular velocity requires differentiating the Rodrigues rotation formula, which increases complexity and weakens conditioning as discussed already in Section 3.3. ADD [Geilinger et al. 2020] suffers a similar issue. The Jacobian of rotation-vector time-derivative wrt angular velocity is

commonplace in their formulation, and it largely complicates their method.

5.2 Jacobians of the Simulation Constraints

By differentiating the dynamics and step constraints (15) and (16) we obtain a linear relationship between the Jacobians of old and new state and step wrt the optimization parameters. This linear relationship is key for the recursive (backward) update of adjoints for differentiable simulation. First, we apply the implicit function theorem to (15), to obtain:

$$\frac{\partial^2 \Psi}{\partial {q_k}^2} \frac{\partial q_k}{\partial \gamma} = -\frac{\partial^2 \Psi}{\partial q_k \, \partial q_{k-1}} \frac{\partial q_{k-1}}{\partial \gamma} - \frac{\partial^2 \Psi}{\partial q_k \, \partial \Delta q_{k-1}} \frac{\partial \Delta q_{k-1}}{\partial \gamma} - \frac{\partial^2 \Psi}{\partial q_k \, \partial \gamma}. \tag{17}$$

Then, we differentiate (16) wrt the optimization parameters and substitute the result $\frac{\partial f}{\partial q_k} = I$ to obtain:

$$\frac{\partial \Delta q_k}{\partial \gamma} = \frac{\partial q_k}{\partial \gamma} + \frac{\partial f}{\partial q_{k-1}} \frac{\partial q_{k-1}}{\partial \gamma}. \tag{18}$$

In the expressions above, we use Lie derivatives whenever we differentiate wrt rotational components of the state or step. Please see Appendix A for the derivatives of the rotational inertial term (9). Note that the step Jacobian $\frac{\partial f}{\partial q_{k-1}}$ bears an interesting geometric interpretation, which becomes apparent in the context of Lie derivatives. It is responsible for transporting back-propagated gradients across tangent spaces at different time steps. For vector quantities, the tangent spaces match and this transport operation is trivial. For rotations, however, Lie derivatives elicit the two tangent spaces across which gradients are transported.

5.3 Adjoints and Objective Gradient

For completeness, we detail the computation of the objective gradient $\frac{\mathrm{d}g}{\mathrm{d}\gamma}$ using the adjoint method recursively, although at this point the formulation does not differ from others. Please see Appendix D for the full derivation.

We define a state adjoint a_k and a step adjoint Δa_k, which are initialized as $a_n = \frac{\partial g}{\partial q_n}$ and $\Delta a_n = 0$, where n is the last simulation step.

State and step adjoints are updated recursively as:

$$a_{k-1} = z_k \frac{\partial^2 \Psi}{\partial q_k \, \partial q_{k-1}} + \Delta a_k \frac{\partial f}{\partial q_{k-1}} + \frac{\partial g}{\partial q_{k-1}}, \tag{19}$$

$$\Delta a_{k-1} = z_k \frac{\partial^2 \Psi}{\partial q_k \, \partial \Delta q_{k-1}}, \tag{20}$$

$$\text{with } z_k \frac{\partial^2 \Psi}{\partial {q_k}^2} = -a_k - \Delta a_k.$$

The objective gradient is computed as:

$$\frac{\mathrm{d}g}{\mathrm{d}\gamma} = \frac{\partial g}{\partial \gamma} + \sum_k z_k \frac{\partial^2 \Psi}{\partial q_k \, \partial \gamma}. \tag{21}$$

6 DERIVATIVES OF OTHER SIMULATION MODELS

6.1 Elasticity and Damping Potentials

In Section 4.2 we have paid attention to the inertial potential term in the optimization formulation of rigid-body dynamics. But (7) also requires other potentials, such as elastic energy, gravity, and incremental-potential formulations of damping and friction. Without loss of generality, these potentials can be expressed as $\Psi_p(p_k = R_k \, \bar{p})$, i.e., some function dependent on point positions p_k, which are themselves the result of rotating undeformed point positions $\bar{p}$. Note that the incremental potentials of damping and friction terms can also be written in this way, as velocities are defined by finite-differencing positions, i.e., $\dot{p}_k = \frac{1}{h}(p_k - p_{k-1})$ [Li et al. 2020].

Lie derivatives of elasticity and damping potential terms can easily be generalized from regular gradient $\frac{\partial \Psi_p}{\partial p}$ and Hessian $\frac{\partial^2 \Psi_p}{\partial p^2}$ as follows:

$$\frac{\mathcal{D}\Psi_p}{\mathcal{D}R_k} = -\frac{\partial \Psi_p}{\partial p} \operatorname{skew}(p), \tag{22}$$

$$\frac{\mathcal{D}^2\Psi_p}{\mathcal{D}{R_k}^2} = D + D^T - \operatorname{skew}(p) \frac{\partial^2 \Psi_p}{\partial p^2} \operatorname{skew}(p), \tag{23}$$

$$D = \frac{1}{2} \operatorname{skew}(p) \operatorname{skew}\left(\frac{\partial \Psi_p}{\partial p}^T\right).$$

We obtained these derivatives using the quadratic approximation of $\exp(\theta_{R_k} \to 0) \, R_k \, \bar{p}$.

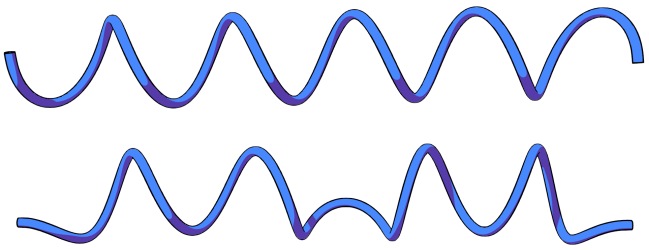

Fig. 5. We use our formulation to simulate the dynamics of Cosserat rods. This figure shows a helical rod with 161 edges, discretized with rotations at edge centers and positions at nodes. Top: The rod being pulled from the sides. Bottom: The rod is straightened to remove its intrinsic twist and then pulled-pushed. This results in helical perversion, as demonstrated by Bergou et al. [2008] for Kirchhoff rods.

6.2 Cosserat Rods

Cosserat rods consider stretch, shear, bending, and twist deformations. This is in contrast to Kirchhoff rods, which only consider bending and twist. With high stretch and shear stiffness, the resulting behavior is practically the same. We use the Cosserat rod model by Spillmann and Teschner [2007], as it employs a multi-body discretization of rod kinematics, allowing direct application of our Lie derivatives for rigid-body dynamics. Fig. 5 shows how our Cosserat rod implementation reproduces the helical perversion behavior demonstrated by Bergou et al. [2008] for Kirchhoff rods.

Rod kinematics are discretized with positions $\{x_i\}$ sampled at nodes and orientations $\{R_i\}$ sampled at edge centers. All elastic energies are defined in the continuum and integrated with quadrature. The discrete stretch energy penalizes edge stretch as $\Psi_s = \sum_i \frac{1}{2} k_s \, l \left(1 - \frac{\|x_{i+1} - x_i\|}{l}\right)^2$, with l the edge rest-length.

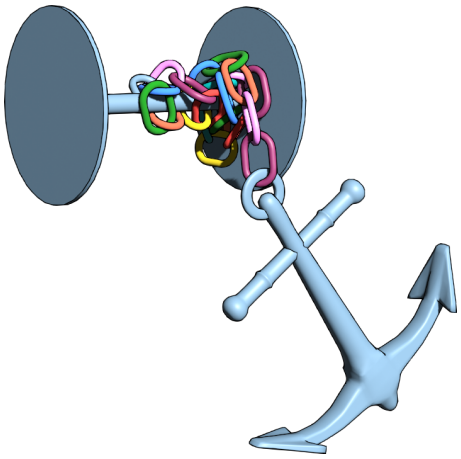

Fig. 6. This demo replicates an example simulated by Ferguson et al. [2021], using our approach, and it demonstrates the ability to handle intricate frictional contact.

The discrete shear energy penalizes the deviation between edge directors and the Z axis of the orientation as $\Psi_x = \sum_i \frac{1}{2}\, k_x\, l\, \left(1 - d_i^T\, R_i\, e_z\right)$, with $d_i = \frac{x_{i+1} - x_i}{\|x_{i+1} - x_i\|}$ the edge director and e_z the unit vector on the Z axis. The combined bending and twist energy is $\Psi_b = \sum_i \frac{1}{2}\, l\, \Delta\omega_i^T\, K_b\, \Delta\omega_i$, with $\Delta\omega_i$ the difference between the deformed ω_i and rest $\bar{\omega}_i$ Darboux vectors. The Darboux vector measures the spatial derivative of the rod's orientation as $R_i' = \hat{R}_i\, \mathrm{skew}(\omega_i)$, and is evaluated at nodes with $\hat{R}_i = \frac{1}{2}(R_i + R_{i-1})$ the average edge orientation and $R_i' = \frac{1}{l}(R_i - R_{i-1})$ the spatial derivative of the rotation matrix. We use the same notation ω for the Darboux vector and the angular velocity as they represent the same concept, albeit in the spatial and temporal domains. K_b is a diagonal stiffness matrix that captures anisotropic bending stiffness and twist stiffness. For inertia, we use a lumped linear inertia (mass) at nodes, and a lumped rotational inertia at edges.

The gradients and Hessians of shear, bending and twist energies rely on Lie derivatives. For convenience, we rewrite as $\Psi_b = \sum_i \frac{1}{2}\, l\, \mathrm{tr}\left(\mathrm{skew}(\Delta\omega_i)\, (K_b - \frac{1}{2}\, \mathrm{tr}(K_b)\, I)\, \mathrm{skew}(\Delta\omega_i)\right)$ the bending and twist energy, and then we can simply substitute $\mathrm{skew}(\omega_i) = \hat{R}_i^T\, R_i'$. Derivatives follow via the chain rule and linear (for the gradient) and quadratic (for the Hessian) approximations of $\exp(\theta_R \to 0)\, R$, as used throughout the paper.

7 RESULTS

In this section, we report results and comparisons of both forward and differentiable simulations. We have validated all Lie derivatives using finite differences, and we have confirmed that, in differentiable simulations, we obtain the same objective gradients (up to numerical precision) with our Lie derivatives, with rotation vectors, and with finite differences. We have used TinyAD [Schmidt et al. 2022] for automatic differentiation wrt rotation vectors, except for the butterfly bolt in Fig. 4, which only has the inertial energy term and we have programmed its analytical gradient and Hessian. We have used the ScyPy implementation of L-BFGS for all our optimization examples of differentiable simulation (with gradients computed in our C++ simulator). All our examples have been executed on an AMD Ryzen 7 6800HS CPU with 8 cores, although our prototype implementation is single-threaded.

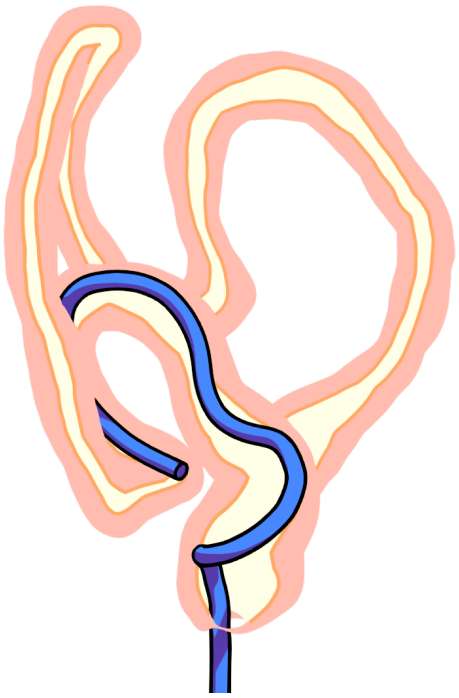

Fig. 7. We simulate a colonoscopy using a Cosserat rod model for the endoscope. Insertion forces are applied at the rectum, and the endoscope advances and deforms due to contact with the colon. We achieve a total speed-up of 4.4× vs. a simulation using rotation vectors. For rendering, we cull the front faces of the colon to highlight the motion of the endoscope.

7.1 Simulation of Rigid Dynamics

We have validated our formulation on forward simulation of frictional rigid-body contact. Fig. 6 shows an example that replicates a benchmark by Ferguson et al. [2021]. Our Lie derivatives handle all potential terms (inertial, elastic and damping/friction) without problem. Our simulation runs 10× faster than the results reported by Ferguson et al., but note that, unlike them, we used quadratic contact potentials based on signed distance fields, not barrier potentials. Our approach can support CCD just as in rigid IPC, using the rotation guess resulting from line-search as described in Section 4.3. But we opted for quadratic potentials over CCD with barrier potentials for ease of implementation.

7.2 Simulation of Cosserat Rod Dynamics

In addition to the helical rod shown in Fig. 5, we demonstrate a colonoscopy simulation in Fig. 7. The characteristics of the endoscope are well captured by a rod model, as the deformation of the cross-section is negligible. The colon model was produced by extracting the colon geometry from medical images (with permission,

Table 1. Complexity (number of degrees of freedom #q, time step h, and number of simulation frames n) and performance comparisons (average assembly and solve time per Newton iteration, and total simulation time) of our forward simulation examples.

	Complexity			Assembly/iter (μs)		Solve/iter (μs)		Total sim (s)	
Example	#q	h	n	rot. vec.	ours	rot. vec.	ours	rot. vec.	ours
Helical perversion (Fig. 5)	969	0.01	1000	2 910	608	195	191	81	13
Colonoscopy (Fig. 7)	969	0.001	2000	2 970	671	753	464	31.1	7.2

but details hidden for anonymity). For contact detection, we sample points along the endoscope's center line, and we use a signed distance field for the colon. We insert the endoscope by applying forces at the rod nodes that fall inside the rectum; no external forces are applied at the tip or elsewhere in the rod.

Table 1 compares the performance of the helical-rod and colonoscopy examples using our formulation vs. rotation vectors. In both cases, the simulation cost is dominated by the gradient and Hessian assembly. The system-solve cost is lower thanks to the narrow-band shape of the system matrix. We achieve speed-ups of $4.4 - 4.8\times$ in the average cost of assembly per Newton iteration, and $4.3 - 6.2\times$ in the total simulation cost. The performance penalty of rotation vectors may be the result of three main factors: (i) more Newton iterations, (ii) the use of automatic differentiation, (iii) the intrinsic complexity of derivative expressions. To understand the effect of this last factor, we have measured the assembly cost on the butterfly bolt of Fig. 4 with analytical derivatives. Rotation derivatives are $2.17\times$ slower per assembly: 505 ns vs 233 ns.

7.3 Control of Frictional Rigid Dynamics

We have developed an example of differentiable frictional rigid dynamics to validate our Lie derivatives within the adjoint method. Fig. 8 shows the resulting animation where we control the initial angular velocity of a die, and we optimize its target orientation and height at the final frame. The optimization converges in just 12 iterations and 565 ms.

7.4 Control of Cosserat Rod Animations

Using Cosserat rods as use-case of multi-rigid-body dynamics, we have validated the generality of our solution on diverse types of optimization problems. Specifically, we have validated the optimization of initial conditions, control forces, and rest shape. While the overall formulation of the adjoint method (Section 5.3) is the same in all cases, they differ in the Jacobian of forces wrt optimization parameters $\frac{\partial^2 \Psi}{\partial q_k \, \partial \gamma}$ in the gradient sum (21). Table 2 lists the simulation settings of all our examples involving Cosserat rods.

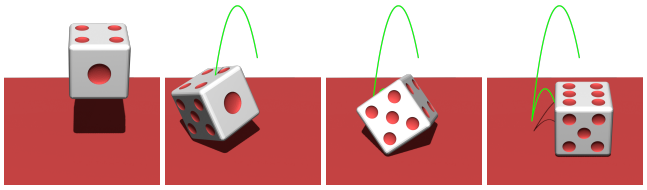

Fig. 8. We throw a die and we optimize its initial angular velocity such that it rolls a 6 and the 5 faces the camera. We express this through objective terms on the orientation and the height at the final frame. The images show some frames along the resulting animation, which was optimized in just 12 iterations and 565 ms.

Optimization of Initial Conditions. This is the easiest case from an implementation perspective, as $\frac{\partial^2 \Psi}{\partial q_k \, \partial \gamma}$ only contributes to the gradient in the first frame. However, it obviously affects the complete trajectory through the recursive contribution of adjoints to z_k.

Fig. 1 shows our benchmark for validation of the optimization of initial conditions. Given straight undeformed rods, we optimize their initial velocities (i.e., the linear velocity of every rod node), such that the rods spell the word SIGGRAPH at the end of the animation. We express the objective (loss) as the L^2 norm of node distances to the target shape, normalized based on the distance of the initial shape.

Fig. 9 compares the convergence rate for the optimization of each letter in the animation. We stop each letter's optimization when the normalized loss reaches 10^{-3} or at 100 iterations. Note that there is no convergence difference between our formulation and rotation vectors, because the objective gradients match up to numerical precision. However, with our formulation the optimization was

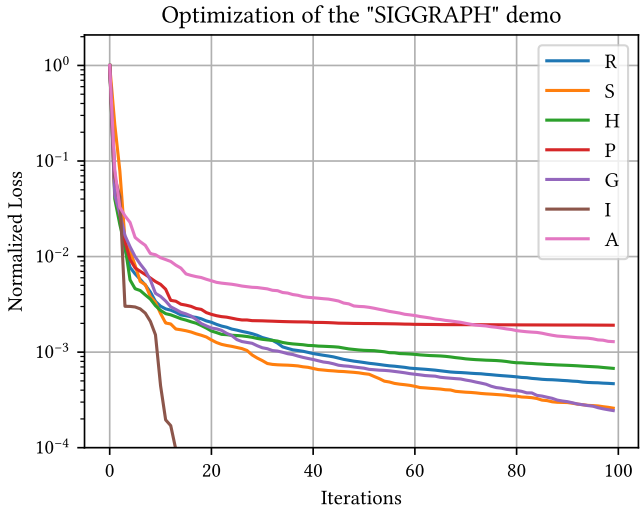

Fig. 9. Convergence rate for the optimization of each letter in the "SIGGRAPH" animation shown in Fig. 1. The objective (loss) is expressed as the L^2 norm of node distances to the target shape, and is normalized based on the distance of the initial shape. There is no convergence difference between our formulation and rotation vectors, but we obtain a total speed-up of $4.8\times$.

Table 2. Settings used in the Cosserat rod examples: mass m, stretch stiffness k_s, bending stiffness K_{b1}, twist stiffness K_{b2}, shear stiffness k_x, number of rigid bodies N and time step h.

Example	m (Kg)	k_s	K_{b1}	K_{b2}	k_x	N	h (ms)
Fig. 5	0.3	2e3	10	25	2e3	162	1
Fig. 7	1.0	5e3	2e3	2e3	1e2	160	0.1
Fig. 10	0.15	5e2	1	1	50	15	10
Fig. 1	0.5 - 0.8	4e2	1e2	1e2	1e2	33 - 85	10

4.8× faster thanks to the speed-up in the forward solves. Table 3 compares the simulation and optimization complexity of each letter, as well as the total optimization time, both with our formulation and with rotation vectors.

Optimization of Control Forces. Given a rod model, we install pulling cables along its sides. Specifically, for the examples shown in Fig. 10, we add 8 cables, each of half-length of the rod, and positioned at 90 degrees on the circumference of the rod's cross-section. Please watch the accompanying video for an illustration of the cables and their forces. Using these cables, we can control the trajectory of the tip of the rod.

Specifically, in the examples in Fig. 10 we optimize the orthographic projection of the trajectory of the tip, such that it matches a given curve. The video shows the convergence and success of the optimization. Similar to all other differentiable simulation examples, we obtain a speed-up wrt rotation vectors thanks to the higher performance of the forward solve, which is the bottleneck. For differentiability, the major direct impact is implementation ease thanks to simplified derivatives.

Optimization of Rest Shape. Finally, we have demonstrated differentiable simulation wrt rest shape, in the example shown in Fig. 10. In this example, we also optimize the trajectory of the tip to follow a given curve. But unlike the previous example, we control the intrinsic Darboux frame of the rod at each discretization edge and frame. Note that we solve a global optimization problem, not independent problems per frame. By doing this, the inertial term of dynamics acts as a regularizer that naturally leads to a smooth animation.

Table 3. Complexity (number of degrees of freedom #q and optimization parameters #γ) and performance (number of iterations N to reach a normalized error of 10^{-3}, but clamped at 100, and total optimization time in seconds) for each letter in the "SIGGRAPH" animation shown in Fig. 1. The table compares optimization performance of our formulation vs. rotation vectors.

Letter	#q	#γ	N	Time (s) rot. vec.	Time (s) ours
S	363	183	28	26.6	5.6
I	99	51	10	1.8	0.5
G	387	195	34	37.7	7.5
R	411	207	39	44.1	9.1
A	291	147	100	75.2	16.1
P	513	258	100	152.2	32.0
H	363	183	55	49.0	10.5

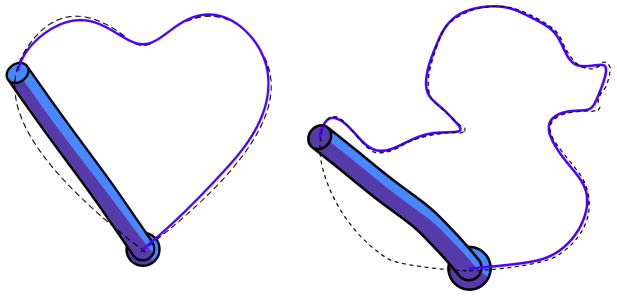

Fig. 10. We optimize the trajectory of the tip of the rod by controlling the forces of pulling cables. The figures show two different trajectories. Please watch the progress of the optimization and the resulting dynamic cable forces in the accompanying video.

8 DISCUSSION AND FUTURE WORK

In this paper, we have presented novel formulations of forward and differentiable rigid-body dynamics, which leverage Lie theory for exact, compact, analytical, well-conditioned and efficient differentiation of and wrt rotations. Our work is in line with Lie-group integrators, but differs from previous work in its focus on implicit integration, and the derivation of adjoints by differentiating a rotation version of the step-update constraint. These contributions make forward and differentiable rigid dynamics easier to implement and more efficient. Efficiency of (differentiable) multi-body dynamics is of high importance, as it is a central ingredient of simulation-based learning for robotics [Xu et al. 2022].

We would like to point out two possible limitations of Lie derivatives. First, in our formulation we only needed derivatives of rotations up to first order, and we have seen that all cases were easy to handle via cancellation of exponential and log maps (See Appendix C). Of course we have used second derivatives (Hessians) with respect to rotations, but for functions in Euclidean space, not for rotation functions. Higher-order derivatives of rotations may require additional complexity.

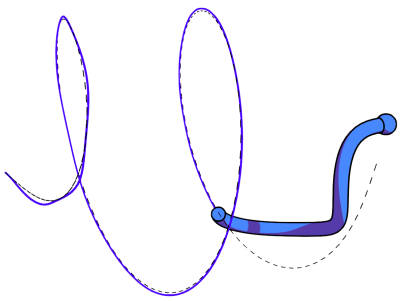

Fig. 11. In this example, we control the rest shape of the rod through its intrinsic Darboux frame, and we optimize the trajectory of the tip to follow a given curve.

Second, off-the-shelf tools for automatic differentiation and numerical optimization do not support Lie derivatives of rotations, as they rely on Euclidean-space differentiation rules. Note that automatic differentiation of vector functions wrt rotations is possible (and we have used it), by first substituting the necessary approximation of the exponential map. We have shown throughout the paper that Lie derivatives are simple, and we have also shown in Section 4.3 that Newton-like optimization methods can easily be applied to rotations. Off-the-shelf optimization tools could be extended to consider the semantics of independent and dependent variables, and apply Lie derivatives accordingly.

While not directly a limitation of our method, we have observed that optimizations of contact problems may converge to local minima, far from the global minimum. The main reason is the discontinuity of contact, which unsurprisingly breaks the assumptions of gradient-based optimization. The use of "leaky gradients" [Turpin et al. 2022] as a way to smooth the objective function could alleviate the problem.

In the paper we have shown the application of our methodology to rigid-body and Cosserat rod dynamics. However, it could be extended to other simulation methods whose kinematic representations rely on rigid transformations. Some tentative use cases are frame-based elasticity models [Gilles et al. 2011], rotation-strain coordinates [Pan et al. 2015], reduced models based on frames [Brandt et al. 2018], or biharmonic coordinates with frame handles [Wang et al. 2015].

As a final remark, we would like to mention an approach different from ours for dealing with the challenges of rotations, which is to formulate rigid-body motion as an affine transformation coupled with a rigidity penalty energy [Lan et al. 2022]. This approach avoids altogether the representation of rotations, at the price of stiffening the simulation problem. It is worth comparing both approaches in terms of convergence, cost, and implementation effort.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive feedback; Maurizio Chiaramonte for tips about related work; Cristian Romero for derivations and implementations with rotation vectors used in the comparisons; Christos Koutras for work on the endoscope simulation; and Pablo Ramón for many discussions during the project. This work was funded in part by the European Commission, grant number 101135082 IRE. The butterfly bolt model was made available by *soonoman* under a Creative Commons license, the anchor model is freely available on TurboSquid, and the colon model was provided by University of Copenhagen in the context of the IRE project.

REFERENCES

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 1–12. https://doi.org/10.1145/1360612.1360662

Nawaf Bou-Rabee and Jerrold E Marsden. 2009. Hamilton–Pontryagin integrators on Lie groups part I: Introduction and structure-preserving properties. *Foundations of computational mathematics* 9, 2 (2009), 197–219.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201387

George E. Brown, Matthew Overby, Zahra Forootaninia, and Rahul Narain. 2018. Accurate dissipative forces in optimization integrators. *ACM Trans. Graph.* 37, 6, Article 282 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275011

Stelian Coros, Miles Macklin, Bernhard Thomaszewski, and Nils Thürey. 2021. Differentiable simulation. In *SIGGRAPH Asia 2021 Courses* (Tokyo, Japan) *(SA '21)*. Association for Computing Machinery, New York, NY, USA, Article 3, 142 pages. https://doi.org/10.1145/3476117.3483433

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics. *ACM Trans. Graph.* 41, 2, Article 13 (Nov. 2021), 21 pages. https://doi.org/10.1145/3490168

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Trans. Graph.* 40, 4, Article 183 (July 2021), 16 pages. https://doi.org/10.1145/3450626.3459802

Guillermo Gallego and Anthony Yezzi. 2015. A compact formula for the derivative of a 3-D rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision* 51 (2015), 378–384.

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (2015), 1103–1115. https://doi.org/10.1109/TVCG.2015.2459687

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph.* 39, 6, Article 190 (Nov. 2020), 15 pages. https://doi.org/10.1145/3414685.3417766

Benjamin Gilles, Guillaume Bousquet, Francois Faure, and Dinesh K. Pai. 2011. Frame-based elastic models. *ACM Trans. Graph.* 30, 2, Article 15 (April 2011), 12 pages. https://doi.org/10.1145/1944846.1944855

Roger Howe. 1983. Very basic Lie theory. *The American Mathematical Monthly* 90, 9 (1983), 600–623.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*.

Zizhou Huang, Davi Colli Tozoni, Arvi Gjoka, Zachary Ferguson, Teseo Schneider, Daniele Panozzo, and Denis Zorin. 2024. Differentiable solver for time-dependent deformation problems with contact. *ACM Trans. Graph.* 43, 3, Article 31 (May 2024), 30 pages. https://doi.org/10.1145/3657648

C. Kane, J. E. Marsden, M. Ortiz, and M. West. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *Internat. J. Numer. Methods Engrg.* 49, 10 (2000), 1295–1325.

Marin Kobilarov, Keenan Crane, and Mathieu Desbrun. 2009. Lie group integrators for animation and control of vehicles. *ACM Trans. Graph.* 28, 2, Article 16 (May 2009), 14 pages. https://doi.org/10.1145/1516522.1516527

Marin B Kobilarov and Jerrold E Marsden. 2011. Discrete geometric optimal control on Lie groups. *IEEE Transactions on Robotics* 27, 4 (2011), 641–655.

Lei Lan, Danny M. Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022. Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. *ACM Trans. Graph.* 41, 4, Article 67 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530064

Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. 2007. Lie group variational integrators for the full body problem. *Computer Methods in Applied Mechanics and Engineering* 196, 29 (2007), 2907–2924. https://doi.org/10.1016/j.cma.2007.01.017

Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. 2008. Optimal attitude control of a rigid body using geometrically exact computations on SO (3). *Journal of Dynamical and Control Systems* 14, 4 (2008), 465–487.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (Aug. 2020), 20 pages. https://doi.org/10.1145/3386569.3392425

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.* 42, 1, Article 2 (Oct. 2022), 20 pages. https://doi.org/10.1145/3527660

Junbang Liang, Ming Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/28f0b864598a1291557bed248a998d4e-Paper.pdf

Jacques L. Lions. 1971. *Optimal Control of Systems Governed by Partial Differential Equations*. Springer-Verlag, New York.

Miles Macklin. 2022. Warp: A high-performance python framework for gpu simulation and graphics. In *NVIDIA GPU Technology Conference (GTC)*.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. *ACM Trans. Graph.* 30, 4, Article 72 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964967

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 449–456. https://doi.org/10.1145/1015706.1015744

J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. 2021. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*. https://openreview.net/forum?id=c_E8kFWfhp0

Dinesh K. Pai. 2002. STRANDS: Interactive Simulation of Thin Solids using Cosserat Models. *Computer Graphics Forum* 21, 3 (2002), 347–352. https://doi.org/10.1111/1467-8659.00594 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00594

Zherong Pan, Hujun Bao, and Jin Huang. 2015. Subspace dynamic simulation using rotation-strain coordinates. *ACM Trans. Graph.* 34, 6, Article 242 (Nov. 2015), 12 pages. https://doi.org/10.1145/2816795.2818090

J. Panetta, M. Konaković-Luković, F. Isvoranu, E. Bouleau, and M. Pauly. 2019. X-Shells: a new class of deployable beam structures. *ACM Trans. Graph.* 38, 4, Article 83 (July 2019), 15 pages. https://doi.org/10.1145/3306346.3323040

Yiling Qiao, Junbang Liang, Vladlen Koltun, and Ming Lin. 2021a. Differentiable simulation of soft multi-body systems. *Advances in Neural Information Processing Systems* 34 (2021), 17123–17135.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. 2021b. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*. PMLR, 8661–8671.

Patrick Schmidt, Janis Born, David Bommes, Marcel Campen, and Leif Kobbelt. 2022. TinyAD: Automatic differentiation in geometry processing made simple. In *Computer graphics forum*, Vol. 41. Wiley Online Library, 113–124.

Yousuf Soliman, Marcel Padilla, Oliver Gross, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2024. Going with the Flow. *ACM Trans. Graph.* 43, 4, Article 57 (July 2024), 12 pages. https://doi.org/10.1145/3658164

Joan Solà, Jeremie Deray, and Dinesh Atchuthan. 2021. A micro Lie theory for state estimation in robotics. arXiv:1812.01537 [cs.RO] https://arxiv.org/abs/1812.01537

J. Spillmann and M. Teschner. 2007. CORDE: Cosserat Rod Elements for the Dynamic Simulation of One-Dimensional Elastic Objects. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, Dimitris Metaxas and Jovan Popovic (Eds.). The Eurographics Association. https://doi.org//10.2312/SCA/SCA07/063-072

John Stillwell. 2008. *Naive lie theory*. Springer Science & Business Media.

Tuur Stuyck and Hsiao-yu Chen. 2023. DiffXPBD: Differentiable Position-Based Simulation of Compliant Constraint Dynamics. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 51 (Aug. 2023), 14 pages. https://doi.org/10.1145/3606923

Dylan Turpin, Liquan Wang, Eric Heiden, Yun-Chun Chen, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. 2022. Grasp'D: Differentiable Contact-rich Grasp Synthesis for Multi-fingered Hands. arXiv:2208.12250 [cs.RO] https://arxiv.org/abs/2208.12250

Christopher D. Twigg and Doug L. James. 2008. Backward steps in rigid body simulation. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 25, 10 pages. https://doi.org/10.1145/1399504.1360624

Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear subspace design for real-time shape deformation. *ACM Trans. Graph.* 34, 4, Article 57 (July 2015), 11 pages. https://doi.org/10.1145/2766952

Chris Wojtan, Peter J. Mucha, and Greg Turk. 2006. Keyframe Control of Complex Particle Systems Using the Adjoint Method. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Marie-Paule Cani and James O'Brien (Eds.). The Eurographics Association. https://doi.org//10.2312/SCA/SCA06/015-023

Jie Xu, Viktor Makoviychuk, Yashraj Narang, Fabio Ramos, Wojciech Matusik, Animesh Garg, and Miles Macklin. 2022. Accelerated Policy Learning with Parallel Differentiable Simulation. In *International Conference on Learning Representations*.

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2021. SGN: Sparse Gauss-Newton for Accelerated Sensitivity Analysis. *ACM Trans. Graph.* 41, 1, Article 4 (Sept. 2021), 10 pages. https://doi.org/10.1145/3470005

A LIE DERIVATIVES OF THE INERTIAL POTENTIAL

We use several general properties of the trace. Let us consider 3D matrices U, V, a 3D symmetric matrix $S = S^T$, a 3D antisymmetric matrix $A = -A^T$, and 3D vectors u, v:

$$\mathrm{tr}(U\,V) = \mathrm{tr}(V\,U). \tag{24}$$

$$\mathrm{tr}(S\,A) = 0. \tag{25}$$

$$\mathrm{tr}(\mathrm{skew}(u)\,U\,\mathrm{skew}(v)) = v^T\ (U - \mathrm{tr}(U)\,I)\ u. \tag{26}$$

To compute the gradient of Ψ_R (9) it suffices to use the linear term of the exponential map. Then, we have:

$$\frac{\mathcal{D}\Psi_R}{\mathcal{D}R_k} \equiv \frac{\partial \Psi_R(\exp(\theta_{R_k})\,R_k)}{\partial \theta_{R_k}} = \frac{\partial \Psi_R(\mathrm{skew}(\theta_{R_k})\,R_k)}{\partial \theta_{R_k}}. \tag{27}$$

If we decompose $\frac{1}{h^2}\,R_k\,J\,\tilde{R}_k^T = M_R = S_R + A_R$ into symmetric and antisymmetric matrices S_R and A_R, we can rewrite the gradient as:

$$\frac{\mathcal{D}\Psi_R}{\mathcal{D}R_k} = \frac{\partial\left(-\mathrm{tr}\left(\mathrm{skew}(\theta_{R_k})\,M_R\right)\right)}{\partial \theta_{R_k}} = \frac{\partial\left(-\mathrm{tr}\left(\mathrm{skew}(\theta_{R_k})\,A_R\right)\right)}{\partial \theta_{R_k}}. \tag{28}$$

And by (26), we have:

$$\frac{\mathcal{D}\Psi_R}{\mathcal{D}R_k} = \frac{\partial\left(2\,\theta_{R_k}^T\,\mathrm{skew}^{-1}(A_R)\right)}{\partial \theta_{R_k}} = 2\,\mathrm{skew}^{-1}(A_R)^T. \tag{29}$$

To compute the Hessian of Ψ_R it suffices to use the quadratic term of the exponential map. Then, we have:

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}{R_k}^2} \equiv \frac{\partial^2 \Psi_R(\exp(\theta_{R_k})\,R_k)}{\partial {\theta_{R_k}}^2} = \frac{\partial^2 \Psi_R(\frac{1}{2}\,\mathrm{skew}^2(\theta_{R_k})\,R_k)}{\partial {\theta_{R_k}}^2}. \tag{30}$$

With the decomposition into S_R and A_R, and since $\mathrm{skew}^2()$ is symmetric, we obtain:

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}{R_k}^2} = \frac{\partial^2\left(-\mathrm{tr}\left(\frac{1}{2}\,\mathrm{skew}^2(\theta_{R_k})\,S_R\right)\right)}{\partial {\theta_{R_k}}^2}. \tag{31}$$

And by (26), we have:

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}{R_k}^2} = \frac{\partial^2\left(\frac{1}{2}\,\theta_{R_k}^T\ (\mathrm{tr}(S_R)\,I - S_R)\ \theta_{R_k}\right)}{\partial {\theta_{R_k}}^2} = \mathrm{tr}(S_R)\,I - S_R. \tag{32}$$

Differentiable simulation also requires mixed Hessians $\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}R_k\,\mathcal{D}R_{k-1}}$ and $\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}R_k\,\mathcal{D}\Delta R_{k-1}}$. To derive these Hessians, it suffices to use the linear terms of exponential maps $\exp(\theta_{R_k})$, $\exp(\theta_{R_{k-1}})$ and $\exp(\theta_{R_{\Delta k-1}})$. Substituting the expression for $\tilde{R}_k$ in Ψ_R, with $M_\Delta = \frac{1}{h^2}\,R_k\,J\,R_{k-1}^T\,\Delta R_{k-1}$, and applying the properties (24)-(26) similarly as before, we reach:

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}R_k\,\mathcal{D}R_{k-1}} = M_R^T - \mathrm{tr}(M_R)\,I + \left(M_\Delta^T - \mathrm{tr}(M_\Delta)\right)\left(I - \Delta R_{k-1}^T\right). \tag{33}$$

$$\frac{\mathcal{D}^2\Psi_R}{\mathcal{D}R_k\,\mathcal{D}\Delta R_{k-1}} = \left(M_\Delta^T - \mathrm{tr}(M_\Delta)\,I\right)\,\Delta R_{k-1}^T. \tag{34}$$

B ROTATION-VECTOR DERIVATIVES OF THE INERTIAL POTENTIAL

We substitute the Rodrigues formula of the exponential map $\exp(r_k)$ (4) into the expression of Ψ_R (9). As we are interested in the gradient and the Hessian, we drop the constant term wrt r_k. We define $M_r = \frac{1}{h^2}\,J\,\tilde{R}_k^T$ and write the revised inertial objective as:

$$\Psi_R = -\sigma(r_k)\,\mathrm{tr}\left(\mathrm{skew}(r_k)\,M_r\right) - \frac{1}{2}\,\sigma^2\left(\frac{r_k}{2}\right)\,\mathrm{tr}\left(\mathrm{skew}(r_k)^2\,M_r\right). \tag{35}$$

With a decomposition into symmetric and antisymmetric matrices $M_r = S_r + A_r$, and using properties of the trace (24)-(26), we

rewrite the objective as:

$$\Psi_R = 2\,\sigma(r_k)\,\mathrm{skew}^{-1}(A_r)^T\,r_k + \frac{1}{2}\,\sigma^2\left(\frac{r_k}{2}\right)\,r_k^T\,(\mathrm{tr}(S_r)\,I - S_r)\,r_k. \tag{36}$$

We can now derive the gradient:

$$\begin{aligned}\frac{\partial \Psi_R}{\partial r_k} &= 2\,\mathrm{skew}^{-1}(A_r)^T\,r_k\,\sigma'(r_k)\,\frac{r_k}{\|r_k\|} + 2\,\sigma(r_k)\,\mathrm{skew}^{-1}(A_r)^T \\ &+ \frac{1}{2}\,r_k^T\,(\mathrm{tr}(S_r)\,I - S_r)\,r_k\,\sigma\left(\frac{r_k}{2}\right)\,\sigma'\left(\frac{r_k}{2}\right)\,\frac{r}{\|r\|} \\ &+ \sigma^2\left(\frac{r_k}{2}\right)\,r_k^T\,(\mathrm{tr}(S_r)\,I - S_r).\end{aligned} \tag{37}$$

This expression evidences the complexity of the rotation-vector parameterization, which is further exacerbated for the Hessian.

C LIE DERIVATIVES OF THE STEP UPDATE

We use some general properties of the exponential map. Let us consider a 3D vector θ and a 3D rotation matrix R:

$$\exp(\theta)^T = \exp(-\theta). \tag{38}$$

$$R\,\exp(\theta) = \exp(R\,\theta)\,R. \tag{39}$$

To differentiate the step update (6), we apply the general expression of Lie derivatives where both the independent and dependent variables are rotations:

$$\frac{\mathcal{D}\Delta R_k}{\mathcal{D}R_k} \equiv \frac{\partial \log\left(\exp(\theta_{R_k})\,\Delta R_k\,\Delta R_k^T\right)}{\partial \theta_{R_k}} = \frac{\partial \theta_{R_k}}{\partial \theta_{R_k}} = I. \tag{40}$$

$$\frac{\mathcal{D}\Delta R_k}{\mathcal{D}R_{k-1}} \equiv \frac{\partial \log\left(\Delta R_k\,\exp(\theta_{R_{k-1}})^T\,\Delta R_k^T\right)}{\partial \theta_{R_{k-1}}} = \frac{\partial\left(-\Delta R_k \theta_{R_{k-1}}\right)}{\partial \theta_{R_{k-1}}} = -\Delta R_k. \tag{41}$$

D ADJOINT METHOD FOR DIFFERENTIABLE SIMULATION

Let us gather the state q_k and step Δq_k in a single vector $y_k = \begin{pmatrix} q_k \\ \Delta q_k \end{pmatrix}$, and the full-simulation state and step in a large vector y. With this notation, the gradient of the objective (14) is expressed as:

$$\frac{\mathrm{d}g}{\mathrm{d}\gamma} = \frac{\partial g}{\partial \gamma} + \sum_k \frac{\partial g}{\partial y_k}\,\frac{\partial y_k}{\partial \gamma} = \frac{\partial g}{\partial \gamma} + \frac{\partial g}{\partial y}\,\frac{\partial y}{\partial \gamma}. \tag{42}$$

The Jacobians of the dynamics and step constraints (17) and (18) can be rewritten in matrix form as:

$$\frac{\partial y_k}{\partial \gamma} + M_k\,\frac{\partial y_{k-1}}{\partial \gamma} = b_k,\ \text{with}\ b_k = \begin{pmatrix} -\frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial \gamma} \\ -\frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial \gamma} \end{pmatrix}, \tag{43}$$

$$M_k = \begin{pmatrix} \frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial q_{k-1}} & \frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial \Delta q_{k-1}} \\ \frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial q_{k-1}} - \frac{\partial f}{\partial q_{k-1}} & \frac{\partial^2 \Psi}{\partial q_k{}^2}^{-1}\,\frac{\partial^2 \Psi}{\partial q_k\,\partial \Delta q_{k-1}} \end{pmatrix},$$

which can be gathered in a full-simulation notation as:

$$M\,\frac{\partial y}{\partial \gamma} = b,\ \text{with}\ M = \begin{pmatrix} & & \cdots & & \\ \cdots & M_{k-1} & I & 0 & \cdots \\ \cdots & 0 & M_k & I & \cdots \\ & & \cdots & & \end{pmatrix}. \tag{44}$$

With the adjoint method, the gradient

$$\frac{\mathrm{d}g}{\mathrm{d}\gamma} = \frac{\partial g}{\partial \gamma} + \frac{\partial g}{\partial y}\,\frac{\partial y}{\partial \gamma},\ \text{with}\ M\,\frac{\partial y}{\partial \gamma} = b, \tag{45}$$

is computed instead as

$$\frac{\mathrm{d}g}{\mathrm{d}\gamma} = \frac{\partial g}{\partial \gamma} + w\,b,\ \text{with}\ w\,M = \frac{\partial g}{\partial y}. \tag{46}$$

It is trivial to see that $w\,b = \frac{\partial g}{\partial y}\,\frac{\partial y}{\partial \gamma} = \frac{\partial g}{\partial y}\,M^{-1}\,b$.

The adjoint formulation allows a recursive (backward) update of adjoint terms:

$$w_{k-1} = -w_k\,M_k + \frac{\partial g}{\partial y_{k-1}}. \tag{47}$$

This backward approach solving for w requires one linear-system solve per step, while a forward approach of solving for $\frac{\partial y}{\partial p}$ requires as many linear-system solves per step as the size of p.

To conclude, we split the adjoint $w_k = (a_k, \Delta a_k)$ into a state adjoint a_k and a step adjoint Δa_k. The adjoint update (47) can then be rewritten separately for the state adjoint and the step adjoint as shown in (19) and (20). Finally, the computation of the gradient (46) can be rewritten as shown in (21).