



**HAL**  
open science

# Implicit time integration simulation of robots with rigid bodies and Cosserat rods based on a Newton-Euler recursive algorithm

Frédéric Boyer, Andrea Gotelli, Philipp Tempel, Vincent Lebastard, Federico Renda, Sébastien Briot

## ► To cite this version:

Frédéric Boyer, Andrea Gotelli, Philipp Tempel, Vincent Lebastard, Federico Renda, et al.. Implicit time integration simulation of robots with rigid bodies and Cosserat rods based on a Newton-Euler recursive algorithm. IEEE Transactions on Robotics, In press, 10.1109/TRO.2023.3334647 . hal-04291835

**HAL Id: hal-04291835**

**<https://hal.science/hal-04291835v1>**

Submitted on 17 Nov 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Implicit time integration simulation of robots with rigid bodies and Cosserat rods based on a Newton-Euler recursive algorithm

Frédéric Boyer, Andrea Gotelli, Philipp Tempel, Vincent Lebastard, Federico Renda, Sébastien Briot

**Abstract**—In this paper, we propose a new algorithm for solving the forward dynamics of multibody systems consisting of rigid bodies connected in arbitrary topologies by localised joints and/or soft links, possibly actuated or not. The simulation is based on the implicit time-integration of the Lagrangian model of these systems, where the soft links are modelled by Cosserat rods parameterised by assumed strain modes. This choice imposes a predictor-corrector structure on the approach, and requires computing both the residual vector and the Jacobian of the residual vector of the dynamics constrained by the time integrator. These additional calculations are handled here with a new Newton-Euler recursive inverse dynamics algorithm and its linearized tangent version. The approach is illustrated with numerical examples from the Cosserat rod literature and from recent robotic applications.

## I. INTRODUCTION

### A. State of the art

With the rise of continuum and soft robotics, many recent robotic designs hybridize rigid and soft bodies in complex topologies actuated in a localized or distributed manner [1], [2]. In this context, the statics and dynamics model of Cosserat rods is gradually becoming one of the standards for the analysis, control and design of these systems [3], [4], [5], [6]. By Cosserat’s rod, we mean a continuous stacking of infinitesimally thin rigid cross-sections along a material line [7]. Cosserat rods theory is a sub-chapter of continuous media mechanics [8]. As is always the case, in this more general context, the medium is modelled by a closed set of partial differential equations (PDEs) or “strong form”, which consists of the equilibrium of its stresses plus their boundary conditions, a definition of the strains, and the constitutive law that relates the stresses and strains [9]. At this primary stage, the formulation is continuous and governs the evolution of the system in an infinite-dimensional configuration space of kinematic fields (for example, the absolute position or displacement field of a 3D medium). In the case of Cosserat rods, the material medium is one-dimensional and we have two independent variables, namely time and a spatial variable, typically the arc-length along the rod. As a consequence of this restriction, whether for a system realised by serial (e.g. a Tendon Actuated Continuum

Robot or TACR) or parallel (e.g. a Continuum Parallel Robot or CPR) connection of rods, its strong form defines a boundary value problem (BVP) in space and an initial value problem (IVP) in time. In this form, the simulation problem can be treated with standard numerical solution methods based on collocation, finite differences, spectral integration, shooting algorithm. In the shooting based approach, the forward static spatial BVP is transformed into a sequence of spatial IVPs whose unknown proximal boundary conditions (BCs), are iteratively approximated in a Newton loop until some of the distal BCs coincide with their required values. Applied at each loading step, this method is well suited to the quasi-static simulation of continuum and soft robots [5]. After this success in statics, Till et al. extended the approach to dynamics [10]. Inspired by the work of the ocean engineering community on towed submarine cables, at each time step of a simulation, the time derivatives of the kinematic fields are removed using an implicit integration scheme. The BVP-IVP in space-time then changes to a simple BVP in space, which can be processed at each of these time steps by the shooting algorithm, in a similar way to the static case. Recently, it has been shown in [11] that: (1) the forward dynamics BVP on which this approach is based derives from a singular optimal control problem, regularised by the implicit time integration scheme; (2) this regularisation works less and less well as the time step and/or beam stiffness decreases; (3) as a consequence, the approach fails in many practical circumstances encountered in soft robotics.

To overcome this issue, we can replace the strong formulation of the original PDEs by the weak formulation of the virtual works in the framework of Lagrangian mechanics [12]. In the most employed method of this framework, the configuration of a rod is parameterized with a set of independent vector fields whose components are decomposed on a truncated basis of functions. The direct dynamics problem is then transformed into a finite-dimensional optimization problem [13], whose stationarity conditions are the Lagrange equations of the system with respect to the coefficients of the basis, which stand for a set of generalised coordinates. Well known as the Ritz, or Rayleigh-Ritz, method [14], this reduction process is the basis of the finite elements method (FEM) of classical three-dimensional (3D) media as that implemented in SOFA [15]. To be applied to Cosserat rods, it was extended from  $\mathbb{R}^3$  to the Lie group  $SO(3)$ , by J. C. Simo and his successors in the framework of the geometrically exact FEM, or GE-FEM [16], [17], [18]. An efficient Lagrangian

F. Boyer and V. Lebastard are with the LS2N lab, Institut Mines Telecom Atlantique, 44307 Nantes, France. e-mail: frederic.boyer@imt-atlantique.fr, vincent.lebastard@imt-atlantique.fr.

A. Gotelli, P. Tempel and S. Briot are with the LS2N lab, CNRS, ECN, 44000 Nantes, France. e-mail: Andrea.Gotelli@ls2n.fr, Philipp.Tempel@ls2n.fr, Sebastien.Briot@ls2n.fr.

F. Renda is with the Khalifa University Center for Autonomous Robotics Systems (KUCARS) and the Department of Mechanical and Nuclear Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE. e-mail: federico.renda@ku.ac.ae.

alternative to GE-FEM has been developed in the computer graphics community. Called discrete elastic rod approach or DER, it has been designed for fast interactive simulation of hairs and other filaments [19], [20]. Unlike other Lagrangian approaches, in DER, the discretization (reduction) process is not applied to fields defined on a continuous rod, but from the outset to the rod itself, which is transformed into a discrete rod, i.e. a set of edges separated by vertices [21]. By redefining all the geometric objects on this discrete manifold and introducing them into the Lagrange equations, we obtain the expected reduced dynamics in a discrete form that resembles that provided by finite differences [19]. However, in contrast to usual finite differences, the approach preserves exactly all the intrinsic properties of the original continuous formulation [21]. First developed for Kirchhoff rods [19], [20] possibly connected to rigid bodies by the fast projection method (FPM) of [22], the DER has recently been extended to Cosserat rods [23]. This context is now implemented in PyElastica, a free and open-source software package for the simulation of slender body assemblies in Python [24].

More recently, some of the authors of this article have proposed an alternative Lagrangian approach to GE-FEM and DER. In this approach based on a Ritz reduction, each of the strain fields of a Cosserat rod (stretch, shear, curvature and torsion) is developed on a truncated basis of assumed strain modes [25], [26], [27]. Named GVS, for Geometric Variable Strain method in [27], the approach has the advantage of providing models in the usual matrix form of rigid manipulators, while offering good accuracy with a small number of modal (strain) coordinates [26]. So far, the approach has been implemented in two different algorithms, depending on whether the generalised forces of the Lagrangian model are calculated with kinematic Jacobian matrices [28], [27], according to a projective matrix process, often referred to as Kane’s method [29], or with a Newton-Euler (NE) inverse dynamic algorithm [26]. Named *IDM* for “inverse dynamic model”, this algorithm generalises the computed torque algorithm used in the 1980s to solve the inverse dynamics of rigid manipulators [30]. While in [26] the approach is restricted to rods serially fixed together, in [27], open, tree and closed loop systems are treated. In the case of kinematic loops, the resulting Lagrangian model takes the form of a set of differential and algebraic equations or DAEs, where the algebraic equations are the geometric constraints of the loop closure [31]. Despite these differences, when applied to forward dynamics (simulation), all these approaches based on modal strain reduction use explicit time integration schemes. That is, schemes known to be much less stable than implicit schemes when applied to the generally “stiff problems” of nonlinear structural dynamics [32].

### B. Contributions of the article

As [27], the present article addresses the dynamic modeling and simulation of hybrid systems composed of Cosserat

rods and rigid bodies in arbitrary topologies with the modal strain reduction. However, unlike all previous work based on this approach, the forward dynamics of soft systems are here integrated in time using an implicit unconditionally stable scheme. This choice strongly structures the simulation algorithm as it requires not only to compute the dynamic model of the robot, but also its tangent dynamics. The first is required to calculate the residual vector of the original DAEs constrained by the integration scheme, and the second to calculate the Jacobian of this vector. This extra effort, compared to explicit integration, has been accomplished by GE-FEM [16] and DER [20]. In the case of strain-parameterized models here considered, this extension is particularly challenging. Indeed, the use of relative variables (strains) instead of the absolute variables of GE-FEM or DER, increases the non-linearities of the model, which are then transferred from the stiffness matrix to the mass matrix. To address this issue, the article refers to rigid robots dynamics, for which efficient algorithms working in relative (joint) coordinates have been developed since the 1980s in the well-known Newton-Euler framework [33]. To this end, the *IDM* of [26] will be hybridized with that of rigid multibody-systems [30] in a single unified algorithm that generalizes the Lie group recursive formulation of rigid robot dynamics [34], [35]. Going one step further, applying an exact differentiation to this *IDM* considered as an input-output map, provides a second algorithm named *TIDM* for “tangent inverse dynamic model”. By feeding these two new algorithms with inputs compatible with the implicit time integration scheme, in this case a one-step integrator of the Newmark class, both the residual vector and its Jacobian matrix can be computed in the Newton correction loop of a general predictor-corrector algorithm. Like in the GE-FEM of [31] and the FPM of [22], the case of closed loops is treated in its index-3 formulation<sup>1</sup>, which, compared to Baumgart’s index-1 method [37], used in [27] after [38], has the advantage of providing solutions that do not depend on the user’s choice of some stabilisation parameters, while satisfying the constraints exactly [39]. At the end, the proposed dynamic simulation algorithm generalizes the quasi-static simulation algorithm of [40] and [41], recently proposed for CPRs and TACRs respectively. In particular, like in [41], the differential properties of the *IDM* and *TIDM* are exploited in order to achieve all the space integrations of generalized forces with spectral methods [42]. The general algorithm of the article is illustrated through numerical examples, both inspired from the GE-FEM literature of Cosserat rods and of some recent robotics applications.

### C. Structure of the article

The article is structured as follows. Section II presents the Lagrangian parametrisation and the corresponding dy-

<sup>1</sup>The index of a system of DAEs is roughly defined as the number of times we need to differentiate the constraints in order to change the DAEs into ODEs (ordinary differential equations). For multibody systems, differentiating once (resp. twice) the geometric constraints, changes the index-3 formulation into an index-2 (resp. index-1) formulation [36].

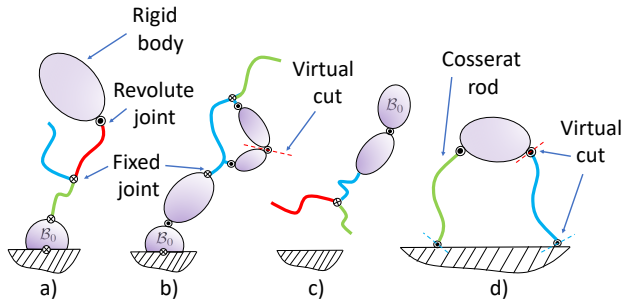


Fig. 1: Different types of SMMS discussed in the article. The ellipsoids and thick lines represent rigid bodies and Cosserrat rods, respectively. From left to right: (a) Tree-shaped manipulator. (b) Manipulator with a closed loop. (c) Locomotor in tree form. (d) Manipulator modelled as a locomotor connected to the ground (see Remark 1). The dashed lines represent the location of the virtual cuts.

dynamic model of systems discussed in the paper systems. In Section III, we present the implicit time integration scheme used by the approach, while section IV presents the general prediction-correction algorithm in which this scheme is integrated for the purpose of dynamic simulation. The central calculation of this algorithm is a Newton loop that requires the calculation of the residual vector of the dynamics and its Jacobian matrix (constrained by the integrator) as explained in Section V. These calculations are based on a Newton-Euler recursive inverse dynamics algorithm, and its differential, or tangent inverse dynamics recursive algorithm. The NE model is presented in Section VI, while the two algorithms are detailed in Section VII and used in section VIII to compute the residual vector and its Jacobian. These algorithms require to space integrate the continuous kinematics and stress along the structure as explained in section IX. Numerical illustrations are reported in Section X, while the paper ends with some perspectives in the conclusion section XI.

## II. LAGRANGIAN MODEL OF A SMMS

### A. Systems addressed

We consider a soft mobile multibody system (SMMS) (see figure 1). Such a system consists of rigid bodies and Cosserrat rods connected by fixed or revolute joints<sup>2</sup> which can be passive or torque actuated. The rods can be passive or internally actuated through distributed actuation systems as cables or pressurized fluids. The system can have a simple open, tree-like or closed topology, with kinematic loops. The system can be a manipulator or a locomotor depending whether it has one of its bodies (named the basis) connected to the ground, or not.

<sup>2</sup>This is assumed for the sake of simplicity, but the approach can be extended with no difficulty to other joints as prismatic or universal ones.

### B. Lagrangian parametrization of a SMMS

If the considered SMMS contains some kinematic loops (see figure 1 (b,d)), then we start by virtually cutting<sup>3</sup> each of these loops at the level of a passive joint (that can be fixed)<sup>4</sup>. Then, the configuration of the so defined tree-like SMMS is parameterized as follows. Any localized joint of revolute type is parameterized by one angle, and the set of all these angles is gathered in the  $(n_r \times 1)$  vector denoted  $q_r$ . Regarding the rods, we use the reduced strain parameterization of [26] i.e., we decompose the allowed strain field  $\epsilon_j$  of each Cosserrat rod of index  $j$  and length at rest  $l_j$ , on a Ritz basis according to:

$$\epsilon_j(X) = \Phi_j(X)q_{\epsilon,j}, \quad \forall X \in [0, l_j], \quad (1)$$

with  $X$  the arc-length along the rod in a resting configuration,  $\Phi_j$  a matrix of functions (typically polynomials), and  $q_{\epsilon,j} \in \mathbb{R}^{n_{\epsilon,j}}$ , the vector of strain (generalized) coordinates of rod  $j$ . Gathering all the strain vectors  $q_{\epsilon,j}$  defines an  $(n_{\epsilon} \times 1)$  vector, noted  $q_{\epsilon}$ . When the SMMS is a manipulator, its configurations or “shapes”, are entirely parameterized by the generalized coordinates vector  $q = (q_r^T, q_{\epsilon}^T)^T$  of size  $(n \times 1)$  with  $n = n_r + n_{\epsilon}$ . In contrast, if the SMMS is a locomotor, it is not only subject to time-variations of its internal degrees of freedom (d.o.f) parameterized by  $q$ , but also to some net rigid motions. These additional external d.o.f require a further set of coordinates that parameterize the pose (position-orientation) of a frame  $\mathcal{F}_0$  attached to one of the rigid bodies of the SMMS, that takes the meaning of a reference body  $\mathcal{B}_0$ . This rigid body can be a full (3D) rigid body, or a rigid cross sections at one of the tips of a Cosserrat rod. The pose of this reference rigid body is parameterized by a homogeneous transformation  $g_0 \in SE(3)$ . Finally, in the most general case, the set of Lagrangian coordinates is defined as a pair  $(g_0, q)$  i.e., an element of the configuration space  $SE(3) \times \mathbb{S}$  with  $\mathbb{S} \cong \mathbb{R}^n$  the shape space of the SMMS. With this definition of the configuration space, the velocities of the SMMS are described at any time  $t$ , with a vector  $(\eta_0^T, \dot{q}^T)^T$  of  $\mathbb{R}^{6+n}$ , where  $\eta_0 = (g_0^{-1}\dot{g}_0)^{\vee} \in se(3) \cong \mathbb{R}^6$  is the twist of  $\mathcal{B}_0$  in its mobile frame  $\mathcal{F}_0$ , and a dot indicates  $d./dt$ .

*Remark 1:* Note that a same closed-loop SMMS can be modelled by different tree-like systems depending on the cuts chosen. To illustrate this, consider a planar CPR like the one in figure 1 (d), where one of the upper joints is actuated, while all the others are passive. The system can be cut at the upper passive joint (shown in red in figure 1 (d)) and the ground defined as the base  $\mathcal{B}_0$  of a two-armed manipulator. Alternatively, all the lower joints can be cut off (shown in blue in figure 1 (d)) and the upper rigid platform defined as the reference body  $\mathcal{B}_0$  of a two legged locomotor ■

<sup>3</sup>A virtual cut is a purely mathematical process. Real cuts, such as the detachment of two bodies, are not covered in this article.

<sup>4</sup>Once again, this condition is imposed for the sake of simplicity, but the approach can be extended to virtual cuts applied to active joints.

### C. Lagrangian dynamic model of a SMMS

Once the configuration space of a general SMMS is defined, applying one of the variational principles of dynamics as D'Alembert's principle of virtual works, or Hamilton's principle of least action, provides its dynamics as a set of differential-algebraic equations of the form [31]:

$$\begin{pmatrix} 0_{6 \times 1} \\ Q_{ad} \end{pmatrix} = \begin{pmatrix} \mathcal{M}_{00} & M_{0q} \\ M_{q0} & M_{qq} \end{pmatrix} \begin{pmatrix} \dot{\eta}_0 \\ \dot{q} \end{pmatrix} + \begin{pmatrix} F \\ Q \end{pmatrix},$$

$$\Phi(g_0, q) = 0_{m \times 1}. \quad (2)$$

The bottom (algebraic) equations define a set of  $m$  independent constraints imposed by the closure loops, which in the general case can depend on both  $g_0$  and  $q$  depending on the topology of the system after cuts and the choice of  $\mathcal{B}_0$  (see Remark 1). The time-differential of these geometric constraints defines the kinematic form of constraints:

$$\dot{\Phi} = \begin{pmatrix} J_0 & J_q \end{pmatrix} \begin{pmatrix} \dot{\eta}_0 \\ \dot{q} \end{pmatrix} = 0_{m \times 1}, \quad (3)$$

where  $J_q(g_0, q)$  and  $J_0(g_0, q)$  are some kinematic Jacobian matrices defined by the usual derivation in  $\mathbb{R}^n$ , for the first:

$$J_q \dot{q} = \left( \frac{\partial \Phi}{\partial q} \right) \dot{q}, \quad (4)$$

and by the directional derivative on  $SE(3)$  for the second:

$$J_0 \dot{\eta}_0 = \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} \Phi(g_0 \exp(\epsilon \hat{\eta}_0), q), \quad (5)$$

with "exp", the exponential map of  $SE(3)$ . In the top (differential equations) of (2), from left to right, one finds: (1) The vector of generalized internal actuation forces  $Q_{ad} = (\tau_d^T, Q_{ade}^T)^T$  including the localized joint torques  $\tau_d(t)$  as well as the distributed actuation generalized forces  $Q_{ade}^T(q_\epsilon, t)$  of a set of cables for instance. (2) The  $q$ -dependent symmetric matrix of generalized inertia. (3) The vector of generalized accelerations. (4) The vector  $(F^T, Q^T)^T = (F_v^T, Q_v^T)^T + (F_c^T, Q_c^T)^T$ , with  $(F_v^T, Q_v^T)^T(q, \eta_0, \dot{q})$  the velocity-dependent generalized forces including the effects of Coriolis and centrifugal accelerations, and  $(F_c^T, Q_c^T)^T$ , the vector of configuration-dependent forces. This vector can be detailed as:

$$\begin{pmatrix} F_c \\ Q_c \end{pmatrix} = \begin{pmatrix} F_g \\ Q_g + Q_e \end{pmatrix} - \begin{pmatrix} J_0^T \\ J_q^T \end{pmatrix} \lambda, \quad (6)$$

where  $(F_g^T, Q_g^T)^T(g_0, q)$  is the vector of gravity forces, while  $Q_e = (0_{1 \times n_r}, Q_\epsilon^T)^T$  stands for the internal restoring (elastic) forces, with  $Q_\epsilon(q_\epsilon) = K_{\epsilon\epsilon} q_\epsilon$ , and  $K_{\epsilon\epsilon}$ , the matrix of generalized (constant) stiffness. In (6),  $J_0^T \lambda$  (respect.  $J_q^T \lambda$ ) is the wrench in  $se(3)^* \cong \mathbb{R}^6$  (respect. the vector of internal generalized forces in  $\mathbb{R}^n$ ) induced by the reaction forces  $\lambda$  imposed by the  $m$  constraints of loops.

*Remark 2:* Although limited to time-independent holonomic constraints of the form (2), all the results of the article can be extended without conceptual difficulty to constraints of the more general form [31]:

$$\Phi(g_0, \eta_0, q, \dot{q}, t) = 0_{m \times 1}, \quad (7)$$

where the components of  $\Phi$  that depend on velocities  $(\eta_0, \dot{q})$  can be non-integrable with respect to time i.e., non-holonomic [43] ■

*Remark 3:* Using notation  $\chi = (g_0, \eta_0, \dot{\eta}_0, q, \dot{q}, \ddot{q}, \lambda)$ , the integration of the dynamics of the SMMS from a set of initial conditions at  $t = t_0$ , consists in finding a time-evolution:  $t \in [t_0, +\infty[ \mapsto \chi(t) \in \mathbb{R}^{3 \times 6 + 3n + m}$ , solution of:

$$\mathcal{R}(\chi(t), t) = 0_{(6+n+m) \times 1}, \quad (8)$$

where  $\mathcal{R}$  defines the dynamic residual vector of the SMMS, which can be deduced from (2) as:

$$\mathcal{R} = \begin{pmatrix} \mathcal{R}_0 \\ \mathcal{R}_q \\ \mathcal{R}_\lambda \end{pmatrix} = \begin{pmatrix} \mathcal{M}_{00} \dot{\eta}_0 + M_{0q} \dot{q} + F \\ M_{q0} \dot{\eta}_0 + M_{qq} \dot{q} + Q - Q_{ad}(t) \\ \Phi \end{pmatrix}. \quad (9)$$

Note that this residual vector is continuous in time and needs to be discretized for numerical simulation purpose. This discretization is based on an implicit integration approach as detailed in next section ■

### III. TIME INTEGRATION

In the context of simulation, a time-integrator allows approximating (8) along a sequence of time instants  $\{t_k\}_{k \in \mathbb{N}^+}$ , with  $t_{k+1} = t_k + \Delta t$  and  $\Delta t$  a time-step that we consider constant. For reasons of stability and simplicity, we adopt an implicit one-step scheme of the Newmark type [44]. Specifically designed for second order ODEs of Newtonian mechanics, this scheme has been refined over the years to meet the specific needs of structural dynamics [32]. Defined at the origin on a vector space, it can be directly applied to the generalized coordinates  $q \in \mathbb{R}^n$  of the internal d.o.f. of a SMMS. In this case, the scheme requires any  $(q, \dot{q}, \ddot{q})$  candidate to be a solution of (8) at a time  $t_{n+1}$ , to satisfy the usual implicit relations:

$$\dot{q} = a(q - q^{(n)}) + f_q^{(n)}, \quad \ddot{q} = b(q - q^{(n)}) + h_q^{(n)}. \quad (10)$$

Here  $q^{(n)}$  denotes the value of  $q$  at  $t_n$  (a notational convention that will be systematically used in what follows),  $a, b$  are two scalars that depend on the time-step  $\Delta t$ , while  $f_q^{(n)} = f(\dot{q}^{(n)}, \ddot{q}^{(n)})$ ,  $h_q^{(n)} = h(\dot{q}^{(n)}, \ddot{q}^{(n)})$ , with  $f$  and  $h$  two vector functions fixed by the scheme and reminded in Appendix 1. As detailed in [11], this integrator can be extended to the external d.o.f. of  $g_0 \in SE(3)$ . To that end, we impose to any  $(g_0, \eta_0, \dot{\eta}_0)$  candidate to be solution of (8) at  $t_{n+1}$ , to fulfill the Newmark scheme on  $SO(3) \times \mathbb{R}^3$ :

$$\begin{pmatrix} \Omega_0 \\ \dot{r}_0 \end{pmatrix} = a \begin{pmatrix} \Theta_0 \\ d_0 \end{pmatrix} + \begin{pmatrix} f_\theta^{(n)} \\ f_r^{(n)} \end{pmatrix},$$

$$\begin{pmatrix} \dot{\Omega}_0 \\ \ddot{r}_0 \end{pmatrix} = b \begin{pmatrix} \Theta_0 \\ d_0 \end{pmatrix} + \begin{pmatrix} h_\theta^{(n)} \\ h_r^{(n)} \end{pmatrix}, \quad (11)$$

where  $g_0 = (R_0, r_0)$ ,  $R_0 \in SO(3)$ ,  $r_0 \in \mathbb{R}^3$ ,  $\eta_0 = (\Omega_0^T, V_0^T)^T$ , with  $\Omega_0$  and  $V_0 = R_0^T \dot{r}_0$  the angular and linear

velocities of  $\mathcal{B}_0$  in the body frame, while we introduced a new vector  $\nu_0 = (\Theta_0^T, d_0^T)^T \in so(3) \times \mathbb{R}^3 \cong \mathbb{R}^6$ :

$$R_0 = R_0^{(n)} \exp(\hat{\Theta}_0), \quad r_0 = r_0^{(n)} + d_0, \quad (12)$$

with “exp” here denoting the exponential map of  $SO(3)$ . Omitting the upper indices, in (11),  $(f_\theta, h_\theta)$  and  $(f_r, h_r)$  are defined similarly to  $(f_q, h_q)$  of (10), but with  $(\Omega_0, \dot{\Omega}_0)$  and  $(\dot{r}_0, \ddot{r}_0)$  replacing  $(\dot{q}, \ddot{q})$  respectively (see Appendix 1). Introducing (11) and (12) into the kinematic definitions of  $(g_0, \eta_0, \dot{\eta}_0)$  in terms of  $R_0$  and  $r_0$ , allows expressing them at any time beyond  $t_n$ , in terms of  $\nu_0$  only [11]:

$$\eta_0 = A(\nu_0), \quad \dot{\eta}_0 = B(\nu_0), \quad g_0 = C(\nu_0), \quad (13)$$

where the detailed expressions of  $A, B, C$  are reminded in Appendix 1. Finally, thanks to the Newmark integrator, the search for  $(g_0, \eta_0, \dot{\eta}_0, q, \dot{q}, \ddot{q}, \lambda)$  at any time  $t_{n+1}$ , is reduced to that of  $(\nu_0, q, \lambda)$ .

*Remark 4:* In the subsequent numerical resolution, one will need to use the increments of velocities and accelerations compatible with the Newmark integration. To get them, it suffices to apply the variation  $\Delta$  to (10):

$$\Delta \dot{q} = a \Delta q, \quad \Delta \ddot{q} = b \Delta q. \quad (14)$$

Similarly, defining the differential of  $g_0$  as  $\Delta \zeta_0 = (g_0^{-1} \Delta g_0)^\vee \in se(3)$ , the  $\Delta$ -variation applied to (13) gives [11]:

$$\begin{aligned} \Delta \eta_0 &= \left( \frac{\partial A}{\partial \nu_0} \right) \Delta \nu_0, \quad \Delta \dot{\eta}_0 = \left( \frac{\partial B}{\partial \nu_0} \right) \Delta \nu_0, \\ \Delta \zeta_0 &= \left( \frac{\partial C}{\partial \nu_0} \right) \Delta \nu_0, \end{aligned} \quad (15)$$

whose detailed expressions are reminded in Appendix 1 ■

#### IV. NUMERICAL RESOLUTION

Based on this approximation, the numerical resolution is achieved by induction as follows. Assuming we know  $\chi^{(n)}$  compatible with the constraints (10-13) and solution of (8) at  $t_n$ , we have to find a  $\chi^{(n+1)}$  compatible with the same constraints, and solution of (8) at  $t_{n+1}$ . To that end, we first introduce (10-13) into (8). This changes the system of DAEs (8), into the algebraic system:

$$\bar{\mathcal{R}}^{(n)}(\bar{\chi}, t_{n+1}) = 0_{(6+n+m) \times 1}, \quad (16)$$

whose root  $\bar{\chi} = (\nu_0, q, \lambda) \in \mathbb{R}^{6+n+m}$  defines  $\bar{\chi}^{(n+1)}$ , and the over-bar indicates that the residual vector is now constrained by the integrator. In details, we have:

$$\bar{\mathcal{R}}^{(n)} = \left( \bar{\mathcal{R}}_0^{(n)T}, \bar{\mathcal{R}}_q^{(n)T}, \Phi^T \right)^T, \quad (17)$$

with upper index  $(n)$  keeping trace of the dependence to  $(g_0^{(n)}, \eta_0^{(n)}, \dot{\eta}_0^{(n)})$  and  $(q^{(n)}, \dot{q}^{(n)}, \ddot{q}^{(n)})$  in (10) and (11). Now remark that (16) is a square system of algebraic nonlinear equations that can be addressed with an iterative root finder,

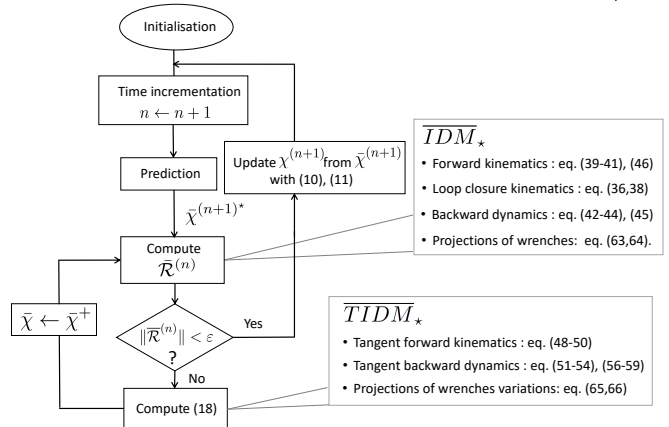


Fig. 2: Flowchart of the “predictor-corrector” simulation algorithm. The calculation of the residual vector and its Jacobian are based on the  $IDM$  and  $TIDM$ . The over-bar indicates that these algorithms are fed by inputs that fulfill the constraints (10-13) and (14,15), imposed by the integration scheme. The lower star defines some adaptations of these two algorithms allowing to calculate directly  $\bar{\mathcal{R}}^{(n)}(\bar{\chi}, t_{n+1})$  and its differential, as detailed in section VIII.

as the Newton algorithm. In this latter case, the algorithm solves at each iteration the linearized system tangent to (16):

$$\bar{\chi}^+ = \bar{\chi} - \left( \frac{\partial \bar{\mathcal{R}}^{(n)}}{\partial \bar{\chi}} \right)^{-1} \bar{\mathcal{R}}^{(n)}(\bar{\chi}, t_{n+1}), \quad (18)$$

where the index “+”, denotes the value of  $\bar{\chi}$  corrected by the current Newton iteration. In this linear system,  $(\partial \bar{\mathcal{R}}^{(n)} / \partial \bar{\chi})$  is the Jacobian matrix of the residual vector constrained by the integrator, which details as:

$$\left( \frac{\partial \bar{\mathcal{R}}^{(n)}}{\partial \bar{\chi}} \right) = \begin{pmatrix} \frac{\partial \bar{\mathcal{R}}_0^{(n)}}{\partial \nu_0} & \frac{\partial \bar{\mathcal{R}}_0^{(n)}}{\partial q} & J_0^T \\ \frac{\partial \bar{\mathcal{R}}_q^{(n)}}{\partial \nu_0} & \frac{\partial \bar{\mathcal{R}}_q^{(n)}}{\partial q} & J_q^T \\ J_0 & J_q & 0 \end{pmatrix}. \quad (19)$$

Note that after convergence of the Newton loop, the full set  $\chi$  is updated from the converged  $\bar{\chi}$  by using (10-13). Furthermore, at each time step  $t_n$ ,  $(g_0, q)$  of  $\bar{\chi} = (g_0, q, \lambda)$  can be initialized in the Newton algorithm, with one of the usual predictors of the Newmark scheme, as for instance, the “inertial” or “ballistic” predictor, which is defined by imposing  $(\dot{\Omega}_0, \ddot{r}_0, \ddot{q}) = (0_{3 \times 1}, 0_{3 \times 1}, 0_{n \times 1})$  in (10) and (11), while for  $\lambda$ , one can use its value at the previous step  $t_{n-1}$ . Finally, the dynamic simulator is structured by two nested loops, one global time-loop (the Newmark loop) and the other (Newton loop) that solves (16) (see figure 2). Note that while this secondary loop runs, the time is frozen and incremented by the Newmark loop only after the Newton loop has converged.

*Remark 5:* At each iteration of the Newton loop, the resolution of (18) requires the computation of  $\bar{\mathcal{R}}^{(n)}$  and  $(\partial \bar{\mathcal{R}}^{(n)} / \partial \bar{\chi})$ . In order to avoid the heavy symbolic computations, we propose to compute  $\bar{\mathcal{R}}^{(n)}$  implicitly, by applying

the Newton-Euler (NE) approach of robotics [33]. To do so, we will use the inverse algorithm of SMMS, named *IDM* for “inverse dynamic model”. Moreover, although the Jacobian  $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})$  can be evaluated numerically by finite difference approximations (as done by defaults, by Matlab’s `fsolve` function, for instance), we propose here a more accurate computation of this matrix, based on another Newton-Euler algorithm named *TIDM*, for “tangent inverse dynamic model” ■

## V. CALCULATION OF THE RESIDUAL VECTOR AND ITS JACOBIAN MATRIX

To introduce the NE computational process of the residual vector and its Jacobian matrix, let us reconsider the upper ODEs of (2), in which  $Q_e$  is moved from the right to the left-hand side, and the vector  $(0_{1\times 6}, (Q_{ad} - Q_e)^T)^T$ , is replaced by a full vector of fictitious actuation forces  $(F_a^T, Q_a^T)^T$ . This defines an (inverse) Lagrangian model of the form:

$$\begin{pmatrix} F_a \\ Q_a \end{pmatrix} = \begin{pmatrix} \mathcal{M}_{00} & M_{0q} \\ M_{q0} & M_{qq} \end{pmatrix} \begin{pmatrix} \dot{\eta}_0 \\ \ddot{q} \end{pmatrix} + \begin{pmatrix} F_v + F_g - J_0^T \lambda \\ Q_v + Q_g - J_q^T \lambda \end{pmatrix}. \quad (20)$$

As this is the case of rigid manipulators [30], we will see soon (Section VII) that the Lagrangian model (20) can be realized alternatively by an inverse Newton-Euler algorithm in two pass, which formally reads:

$$\begin{pmatrix} F_a \\ Q_a \end{pmatrix} = IDM(\chi), \quad (21)$$

where remind that  $\chi = (g_0, \eta_0, \dot{\eta}_0, q, \dot{q}, \ddot{q}, \lambda)$ . Physically, this *IDM* computes for any state  $(g_0, \eta_0, q, \dot{q})$ , the wrench  $F_a$  and the full vector of internal forces  $Q_a$  (applied on all the entries of  $q = (q_r^T, q_e^T)^T$ ), that would be required to ensure the SMMS, considered as fully actuated, to accelerate at  $(\dot{\eta}_0, \ddot{q})$ , while being subject to the external forces  $\lambda$  applied at the tip of the cut branches. Now using the identity (21) in (16), the two top block-components of the residual vector of the SMMS constrained by the Newmark integrator read:

$$\begin{pmatrix} \bar{\mathcal{R}}_0^{(n)} \\ \bar{\mathcal{R}}_q^{(n)} \end{pmatrix}(\bar{\chi}, t_{n+1}) = \overline{IDM}(\bar{\chi}) - \begin{pmatrix} 0_{6\times 1} \\ \Upsilon(q, t_{n+1}) \end{pmatrix}, \quad (22)$$

where the overbar indicates that the *IDM* is now fed with inputs compatible with the integrator constraints (10,11) i.e., with inputs that depend on  $\bar{\chi}$  only, while we used the further notation:

$$\Upsilon(q, t) = Q_{ad}(q_e, t) - Q_e(q_e). \quad (23)$$

In summary, one can compute  $(\bar{\mathcal{R}}_0^{(n)}, \bar{\mathcal{R}}_q^{(n)})(\bar{\chi}, t_{n+1})$  with the *IDM*. To this end, it suffices to force its inputs to fulfill the constraints of the integrator (10,11), and to remove  $(0_{1\times 6}, \Upsilon(q, t_{n+1})^T)^T$  from its vector of outputs. Remarkably, the same idea can be applied to the computation of the

top two block-rows of  $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})$ . Indeed, differentiating (22) provides the identity:

$$\begin{pmatrix} \Delta\bar{\mathcal{R}}_0^{(n)} \\ \Delta\bar{\mathcal{R}}_q^{(n)} \end{pmatrix} = \begin{pmatrix} \frac{\partial\bar{\mathcal{R}}_0^{(n)}}{\partial\bar{\chi}} \\ \frac{\partial\bar{\mathcal{R}}_q^{(n)}}{\partial\bar{\chi}} \end{pmatrix} \Delta\bar{\chi} = \overline{TIDM}(\bar{\chi}, \Delta\bar{\chi}) - \begin{pmatrix} 0_{6\times 1} \\ \Delta\Upsilon(q, \Delta q, t_{n+1}) \end{pmatrix}, \quad (24)$$

where:

$$\Delta\Upsilon(q, \Delta q, t) = \Delta Q_{ad}(q_e, \Delta q_e, t) - \Delta Q_e(q_e, \Delta q_e), \quad (25)$$

with  $\Delta Q_e = K_{e\epsilon} \Delta q_e$ , and  $\Delta Q_{ad} = (\Delta\tau_d(t)^T, \Delta Q_{ade}^T)^T = (0_{1\times n_r}, \Delta Q_{ade}^T)^T$ , since the time is frozen in the correction Newton’s loop, while  $Q_{ade}$  can depend on  $q_e$  [26]. In (24),  $\overline{TIDM}$  denotes the tangent algorithm to the *IDM* or *TIDM*, with the over-bar indicating that it is fed with inputs compatible with the constraints (10,11) and (14,15) imposed by the integrator. Formally, this further NE algorithm is defined by the differential of the input-output map (21):

$$\begin{pmatrix} \Delta F_a \\ \Delta Q_a \end{pmatrix} = TIDM(\chi, \Delta\chi), \quad (26)$$

which for any  $\chi = (g_0, \eta_0, \dot{\eta}_0, q, \dot{q}, \ddot{q}, \lambda)$ , allows to compute the variations of outputs  $\Delta F_a$  and  $\Delta Q_a$  required by imposing variations  $\Delta\chi = (\Delta\zeta_0, \Delta\eta_0, \Delta\dot{\eta}_0, \Delta q, \Delta\dot{q}, \Delta\ddot{q}, \Delta\lambda)$  as inputs. Now let us define  $\delta_i \in \mathbb{R}^{6+n+m}$  as the unit vector, with zero entries, except the  $i^{th}$ , which is equal to one. Then, the identity (24) shows that feeding  $\overline{TIDM}$  with  $\bar{\chi}$  and  $\Delta\bar{\chi} = \delta_i$ ,  $i = 1, 2, \dots, 6 + n + m$ , and removing from its outputs,  $(0_{1\times 6}, \Delta\Upsilon^T)^T$  fed with the same inputs (see next remark), allows to compute column after column, the top two block-rows of  $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})$  in any  $\bar{\chi}$ .

*Remark 6:* Before detailing the *IDM* and *TIDM* algorithms of a SMMS, it is worth mentioning that if we have at our disposal these two algorithms, exploiting the two identities (22) and (24) to compute their left-hand sides, requires to calculate  $\Upsilon(t_{n+1}) = Q_{ad}(t_{n+1}) - Q_e$  and  $\Delta\Upsilon(t_{n+1}) = \Delta Q_{ad}(t_{n+1}) - \Delta Q_e$  (for the sake of concision, we will often indicate the time dependency, only). Moreover, using the identity (22) only provides  $\bar{\mathcal{R}}_0^{(n)}$  and  $\bar{\mathcal{R}}_q^{(n)}$  in (9), and not  $\bar{\mathcal{R}}_\lambda^{(n)} = \Phi(g_0, q)$ . On the other hand, owing to the symmetries of the block matrix (19), calculating its two top rows with the identity (24) is enough to reconstruct the full Jacobian  $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})$ . As we will see later (see section VIII), the missing vectors  $\Upsilon(t_{n+1})$ ,  $\Delta\Upsilon(t_{n+1})$ , and  $\bar{\mathcal{R}}_\lambda^{(n)} = \Phi(g_0, q)$  can also be computed with some of the subparts of the *IDM* and *TIDM*. Therefore, these two algorithms are the key of the computation of the residual vector and the Jacobian of (18). Since they are both based on the Newton-Euler model of SMMS, we will now introduce this model ■

## VI. NEWTON-EULER MODEL OF A SMMS

As per usually, a rigid link is modelled by a rigid 3D body of arbitrary shape (not necessarily rod-shaped). On the other

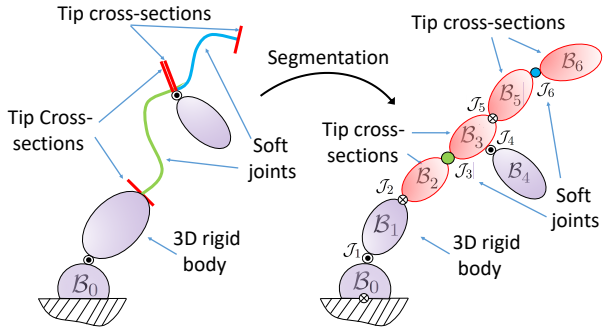


Fig. 3: Segmentation of a SMMS for *IDM* and *TIDM*. The two Cosserat rods are declared as joints  $\mathcal{J}_3$  and  $\mathcal{J}_6$ . Their two tip-cross sections are declared as rigid bodies  $(\mathcal{B}_2, \mathcal{B}_3)$  and  $(\mathcal{B}_5, \mathcal{B}_6)$ .

hand, in order to easily hybridize the *IDM* of rigid multi-body systems [30], with that of Cosserat rods [26] into a single generalized *IDM*, it is convenient to consider a Cosserat rod as two tip cross-sections (i.e., two rigid bodies with no thickness and no inertia), connected by a distributed flexible joint. Therefore, the continuous model of poses, velocities, acceleration and stress of Cosserat rods is used in the model of joints and not bodies. As we will soon see, by adopting this choice, the resulting generalized *IDM* will, at the highest level, take a form similar to that of the rigid case.

#### A. Segmentation of a SMMS

Using the above viewpoint, the tree-like SMMS of section II is first segmented into a rigid body sequence consisting of the original 3D rigid bodies and the tip sections of the Cosserat rods. All these rigid bodies are indexed following the usual Newton-Euler conventions i.e., from the reference body  $\mathcal{B}_0$ , which is the fixed basis for a manipulator, or the free-floating one of a locomotor, to  $\mathcal{B}_N$ , with bodies  $\mathcal{B}_0, \mathcal{B}_1, \dots, \mathcal{B}_N$  numbered increasingly while descending the branches. As in any tree-like system, a body has only one antecedent and possibly several successors. In the subsequent developments,  $j$  preferentially denotes the current body index of the algorithm, while  $k = a(j)$  is its antecedent and  $l$  is the index of one of its successors i.e.,  $l \in s(j)$ , where  $s(j)$  is the set of the indexes of all the successors of  $\mathcal{B}_j$ . As for rigid systems, any two contiguous bodies are separated by a joint. Indexing each of these joints with the index of its succeeding body, one can form the set of joints  $\mathcal{J}_1, \mathcal{J}_2, \dots, \mathcal{J}_N$ , some of them being rigid lumped joints (fixed or revolute) as those met in rigid systems, while others are soft distributed joints modelled by Cosserat rods. This segmentation process is illustrated in figure 3.

#### B. Frames

According to the standard uses of NE's approach, each of the rigid bodies that belongs to the SMMS is equipped

with a frame rigidly attached to it and positioned according to some conventions. With the segmentation introduced above, this concerns both the rigid bodies (3D or rod end cross-sections), and the distributed joints modelled by Cosserat rods. Regarding distributed joints, we adopt for each of them, say  $\mathcal{J}_j$ , the convention of Cosserat rods and attach to each cross-section labelled by its reference arc-length  $X \in [0, l_j]$ , a frame  $\mathcal{F}_j(X) = (O_j, s_j, n_j, a_j)(X)$  where  $O_j(X)$  is centered on the  $X$ -cross-section, and  $s_j(X)$  is its unit normal. Regarding rigid bodies, if  $\mathcal{B}_j$  is a 3D body, we adopt the usual Newton-Euler conventions of rigid systems [45] i.e., a body frame  $\mathcal{F}_j = (O_j, s_j, n_j, a_j)$  is positioned on  $\mathcal{J}_j$ , with  $a_j$ , supported by the joint axis if  $\mathcal{J}_j$  is revolute. If  $\mathcal{B}_j$  is a 2D body, this is one of the two end cross-section of a Cosserat rod, and its body frame is that used in the model of the corresponding distributed joint. If a body  $\mathcal{B}_j$  is at the end of a branch resulting of the virtual cut of a loop, a second frame, other than  $\mathcal{F}_j$ , noted  $\mathcal{F}_j^+ = (O_j^+, s_j^+, n_j^+, a_j^+)$  is attached to  $\mathcal{B}_j$  with  $O_j^+$  located on the cut, and  $a_j^+$  supported by the axis of the cut joint, when it is revolute. Finally, the Euclidean space is equipped with an inertial frame  $\mathcal{F}_e$ . In the rest of the article, for any vector (or tensor)  $\mathbf{V}$  mechanically related to a body  $\mathcal{B}_j$ ,  ${}^iV_j$  denotes the matrix of its components in the frame  $\mathcal{F}_i$ , while for the sake of concision, the upper-left index is omitted when  $\mathcal{F}_i = \mathcal{F}_j$  i.e.,  ${}^jV_j = V_j$ . For similar reasons of concision, the pose of a frame  $\mathcal{F}_j$  with respect to another  $\mathcal{F}_k$  is denoted  ${}^k g_j \in SE(3)$ , while when  $\mathcal{F}_k = \mathcal{F}_e$ , the index  $e$  is omitted i.e.,  ${}^e g_j = g_j$ , and the pose is said "inertial".

*Remark 7:* Note that some conflicts between the frame placement conventions of Cosserat beams and those of rigid robots may occur, e.g. when a rod is preceded by a rotary joint of negligible inertia. However, such conflicts can always be removed by declaring a fictitious rigid body supported by the revolute joint and fixed to the first cross section of the following rod. This explains why we have introduced the *a priori* unnecessary fixed joints in our description ■

#### C. Newton-Euler model of rigid bodies

In the Newton-Euler approach, the configuration of each rigid body  $\mathcal{B}_j$  is defined by the inertial pose of its body frame i.e.,  $g_j = (R_j, r_j) \in SE(3)$ , with  $R_j$  and  $r_j$  the orientation matrix and origin's position of  $\mathcal{F}_j$ , in  $\mathcal{F}_e$ . Introducing the inertial velocity twist of  $\mathcal{B}_j$  in the body frame  $\eta_j = (g_j^{-1} \dot{g}_j)^\vee \in \mathbb{R}^6$ , and applying Newton's laws and Euler's theorem to each body  $\mathcal{B}_j$  isolated in the structure, provides the usual Newton-Euler equations that govern the time evolution of the  $g_j$ s on  $SE(3)$ :

- NE dynamic balance (ODEs) of rigid bodies:

$$\mathcal{M}_j \dot{\eta}_j - ad_{\eta_j}^T \mathcal{M}_j \eta_j = F_{\text{ext},j} + F_j - \sum_{l \in s(j)} {}^j F_l. \quad (27)$$



In these dynamic balances,  $\mathcal{M}_j \in \mathbb{R}^{6 \times 6}$  is the (screw) inertia matrix of an arbitrary shaped rigid body  $\mathcal{B}_j$  (equal to zero if  $\mathcal{B}_j$  is a rod end cross-section),  $F_{\text{ext},j} \in \mathbb{R}^6$  is the wrench of external forces (e.g gravity, external contacts) exerted on it,  $F_j \in \mathbb{R}^6$  is the wrench of internal contacts exerted by the antecedent body  $\mathcal{B}_k$ , ( $k = a(j)$ ) onto  $\mathcal{B}_j$ , all these vectors and tensors being expressed in the mobile body frame  $\mathcal{F}_j$ . With these definitions and our notational conventions, the wrenches in the sum of (27) are contact wrenches transmitted by  $\mathcal{B}_j$  to all its successors  $\mathcal{B}_l$  through the joints that connect them. Therefore they are first defined in the  $\mathcal{F}_l$  frames by  $F_l$ , and then expressed in  $\mathcal{F}_j$  through  ${}^jF_l$ , while action-reaction principle motivates the minus sign.

*Remark 8:* When a rigid body  $\mathcal{B}_j$  is preceded by an actuated lumped (revolute) joint  $\mathcal{J}_j$ , the projection of the contact wrench  $F_j$  transmitted by the joint from  $\mathcal{B}_{j-1}$  to  $\mathcal{B}_j$ , along the joint axis  $a_j$ , is no more than the motor torque:

$$Q_{ar,j} = A_j^T F_j, \quad (28)$$

where  $A_j = (0, 0, 1, 0, 0, 0)^T \in \mathbb{R}^6$  is the unit twist supporting the localized rotation of the revolute joint. Note that according to the common uses in rigid robotics, (28) stands for a constitutive law of the joint, that can be supplemented with a model of friction [45].

#### D. Newton-Euler model of soft joints

In the rest of the article, the temporal and spatial derivations  $\partial./\partial t$  and  $\partial./\partial X$  are designated by a ‘‘dot’’ and a ‘‘prime’’ respectively. In the Newton-Euler approach, the configuration of a Cosserat rod  $\mathcal{J}_j$  is defined by the inertial pose of all its cross-sectional frames  $g_j(X) = (R_j, r_j)(X) \in SE(3)$ ,  $\forall X \in ]0, l_j[$ , whose time and space variations are described by the two twists (in  $\mathbb{R}^6$ ):  $\eta_j(X) = (g_j^{-1}(X)\dot{g}_j(X))^\vee$ , and  $\xi_j(X) = (g_j^{-1}(X)g_j'(X))^\vee$ , both being expressed in the cross-sectional frames. Then, applying Newton’s laws and Euler’s theorem along such a rod, provides the following closed formulation that governs the time-evolution of  $g_j(\cdot)$ :

- NE dynamic balance of  $X$ -cross sections (PDEs),  $\forall X \in ]0, l_j[$ :

$$\mathcal{M}_j \dot{\eta}_j - ad_{\eta_j}^T \mathcal{M}_j \eta_j = \bar{F}_{\text{ext},j} + \Lambda_j' - ad_{\xi_j}^T \Lambda_j, \quad (29)$$

- Boundary conditions:

$$\Lambda_j(0) = -F_{a(j)}, \quad \Lambda_j(l_j) = -F_j. \quad (30)$$

- Active constitutive law along  $[0, l_j]$ :

$$B_j^T \Lambda_j = B_j^T \Lambda_{d,j}(t) + (B_j^T \mathcal{H}_j B_j) \epsilon_j. \quad (31)$$

In these equations,  $\epsilon_j(X) = B_j^T (\xi_j - \xi_j^o)(X)$  is the strain field at  $X$ , with  $B_j$  a matrix (with zero and unit entries) selecting the strain components allowed by the adopted beam kinematics (e.g. Reissner, Kirchhoff...), and  $\xi_j^o(X)$  the value of  $\xi_j(X)$  in a reference (stress-less) configuration indexed by  $o$ .  $\bar{F}_{\text{ext},j}(X) \in \mathbb{R}^6$ ,  $\mathcal{M}_j(X) \in \mathbb{R}^{6 \times 6}$  and

$\mathcal{H}_j(X) \in \mathbb{R}^{6 \times 6}$  are the density of external wrenches, this of inertia (screw) matrix, and the stiffness (screw) matrix along the rod at  $X$ , while  $\Lambda_j(X)$  is the wrench of stress exerted by the piece of rod  $[X, l_j]$  onto the piece  $[0, X]$  across the  $X$ -cross section, all these vectors and tensors being expressed in the cross-sectional frames of the rod. Finally,  $\Lambda_{d,j}(X, t) \in \mathbb{R}^6$ , is a stress wrench modelling the effect of the distributed actuation (if any), across the  $X$ -cross-section of  $\mathcal{J}_j$  as illustrated by the following remark.

*Remark 9:* In the case of a rod  $\mathcal{J}_j$  actuated by a set of  $p_j$  tendons of index ( $\alpha = 1, 2, \dots, p_j$ ), with routings defined by their positions  $D_{j,\alpha}(X) \in \mathbb{R}^3$  in the  $X$ -cross-sectional frames,  $\Lambda_{d,j}$  can be detailed as [26]:

$$\Lambda_{d,j} = \sum_{\alpha=1}^{p_j} \frac{1}{\|\Gamma_{j,\alpha}\|} \begin{pmatrix} D_{j,\alpha} \times \Gamma_{j,\alpha} \\ \Gamma_{j,\alpha} \end{pmatrix} T_{j,\alpha}(t), \quad (32)$$

where  $T_{j,\alpha}$  is the tension exerted on the tendon  $\alpha$ , and  $\Gamma_{j,\alpha} = \Gamma_j + K_j \times D_{j,\alpha} + D_{j,\alpha}'$ , with  $\xi_j = (K_j^T, \Gamma_j^T)^T$ . In general, the model of  $\Lambda_{d,j}$  depends on  $\epsilon_j$  and needs to be derived on the case by case basis ■

*Remark 10:* The above constitutive law can be adapted to approximate the material friction between cross-sections with a simple viscous damping model. To this end, it suffices to change (31) into:

$$B_j^T \Lambda_j = B_j^T \Lambda_{d,j}(t) + (B_j^T \mathcal{D}_j B_j) \dot{\epsilon}_j + (B_j^T \mathcal{H}_j B_j) \epsilon_j, \quad (33)$$

where  $\mathcal{D}_j(X) \in \mathbb{R}^{6 \times 6}$  is the damping matrix along the rod at  $X$ . Using the Kelvin-Voigt model, it is convenient to impose  $\mathcal{D}_j = \mu_j \mathcal{H}_j$ , with  $\mu_j$  a scalar damping coefficient ■

#### E. Kinematic model of joint connections

Using the segmentation of Section VI.A, a joint connecting any two consecutive rigid bodies  $\mathcal{B}_k$  and  $\mathcal{B}_j$  imposes the relation between their poses:

$$g_j = g_k {}^k g_j. \quad (34)$$

In details, according to the standard uses of rigid robots, when  $\mathcal{J}_j$  is a rigid lumped joint, the relative pose  ${}^k g_j$  in (34) is a function of  $q_{r,j}$  or is fixed by design, depending whether the joint is revolute or fixed. When  $\mathcal{J}_j$  is a (distributed) soft joint, following [26], the internal rod kinematics (Reissner, Kirchhoff...) imposes to any two infinitely close cross-sections along the joint, the differential relation between their poses:

$$g_j' = g_j (\xi_j^o + B_j \epsilon_j)^\wedge, \quad (35)$$

where the strain field  $\epsilon_j(\cdot)$  of (35) is reduced on a strain (polynomial) basis with the Ritz approximation (1) in which remind that the vector of strain coordinates of  $\mathcal{J}_j$   $q_{\epsilon,j} \in \mathbb{R}^{n_{\epsilon,j}}$ , defines a sub-part of  $q_\epsilon$ . In the following NE algorithm, (34) and (35) are used to relate the inertial poses  $g_j$  and  $g_j(\cdot)$  to the SMMS generalized coordinates  $q = (q_r^T, q_\epsilon^T)^T$  of the Lagrangian parametrization of Section II. Finally, remark that the cut revolute joint of a closed

loop will not introduce such kinematic parameters, but rather Lagrange multipliers in charge of forcing a set of closure kinematic constraints that we are now going to detail.

### F. Kinematic model of loop closures

Let us consider two terminal bodies  $\mathcal{B}_p$  and  $\mathcal{B}_s$  connected by a revolute joint virtually cut when opening a loop. Then, using the convention that when  $s > p$ ,  $\mathcal{F}_{p+}$  is the leader frame and  $\mathcal{F}_{s+}$  is the follower one, a virtually cut revolute joint imposes to the relative pose  ${}^{p+}g_{s+}$  to be compatible with the five scalar geometric constraints:

$$\Phi_{s|p} = \bar{A}_{p+}^T \begin{pmatrix} \log({}^{p+}R_{s+}) \\ {}^{p+}r_{s+} \end{pmatrix} = \bar{A}_{p+}^T \begin{pmatrix} 0_{3 \times 1} \\ 0_{3 \times 1} \end{pmatrix}, \quad (36)$$

where the log-map stands for the inverse of the exponential in  $SO(3)$  and  $\bar{A}_{p+} \in \mathbb{R}^{6 \times 5}$  is the complementary matrix to the virtually cut revolute joint axis  $A_{p+} = (0, 0, 1, 0_{1 \times 3})^T$  i.e., it is such that  $A_{p+}^T \bar{A}_{p+} = 0_{1 \times 5}$ , and  $\bar{A}_{p+}^T \bar{A}_{p+} = 1_{5 \times 5}$ . This context holds for any type of cut joint where it suffices to replace the 5 constraints above by any number of constraints depending on the nature of the joint. In particular, virtually cutting a fixed joint introduces six constraints instead of five (i.e.  $\bar{A}_{p+} = 1_{6 \times 6}$ ). Time-differentiating (36) provides the kinematic form of the closure loop constraints:

$$\dot{\Phi}_{s|p} = J_{s|p} \eta_{s+/p+} = \bar{A}_{p+}^T 0_{6 \times 1}, \quad (37)$$

where  $\eta_{s+/p+} = ({}^{s+}g_{p+} \quad {}^{p+}\dot{g}_{s+})^\vee = \eta_{s+} - Ad_{s+} g_{p+} \eta_{p+}$  is the relative twist of the follower frame  $\mathcal{F}_{s+}$  with respect to the leader one  $\mathcal{F}_{p+}$ , while  $J_{s|p}$ , is a kinematic Jacobian matrix defined by:

$$J_{s|p} = \bar{A}_{p+}^T \begin{pmatrix} T(\Theta_{s|p})^{-1} & 0_{3 \times 3} \\ 0_{3 \times 3} & {}^{p+}R_{s+} \end{pmatrix}, \quad (38)$$

which depends on  $\Theta_{s|p} = \log({}^{p+}R_{s+}) \in \mathbb{R}^3$ . Finally, note that if the loop is connected to the ground, we simply have  $s = 0$ .

*Remark 11:* Gathering all the equations from (27) to (38) defines the Newton-Euler formulation of the SMMS dynamics. It is here expressed on the Lie group  $SE(3)$  as this was the case for rigid robots in [34], [35]. This formulation is entirely equivalent to the Lagrangian one defined by the set of DAEs (2). As announced in Section V, we are now going to exploit this equivalence in order to calculate the residual of the Lagrangian formulation (17) and its Jacobian matrix (19). This is achieved with two recursive algorithms based on the NE formulation that are detailed in the next section ■

## VII. RECURSIVE IDM AND TIDM OF A SMMS

### A. Newton-Euler IDM of a SMMS

Like for a rigid manipulator [30], the IDM proceeds in two passes. The first, descending (from the base to the ends of the branches), calculates the inertial poses, velocities and accelerations along the branches from the knowledge of  $(g_0, \eta_0, \dot{\eta}_0, q, \dot{q}, \ddot{q})$ . The second, ascending, calculates the interbody wrenches transmitted across the rigid and soft

joints from the knowledge of  $\lambda$  and the inertial kinematics calculated by the first pass. These interbody wrenches are then projected on the d.o.f of joints to calculate  $Q_a$  of (20), while  $F_a$  is the last ‘‘interbody’’ wrench computed by the second pass. As introduced before, thanks to the declaration of rods as distributed joints, at a first (high) level, this algorithm takes the usual form of that of rigid systems, but here applied to the segmentation of section VI.A. The only difference lies in a second (low) level, where the usual discrete forward and backward transfers on inter-body kinematics (poses, velocities, accelerations) and inter-body wrenches, respectively, are replaced by continuous transfers obtained by integrating forward (35) (and its first and second temporal derivatives), and backward (29), when the joint is a soft distributed one.

1) *Forward kinematics:* Time differentiating twice (34) provides the kinematic model that relates the pose, velocities and accelerations of all the rigid bodies  $\mathcal{B}_j$  (including the tip cross sections of soft links) in the form of a recursion on body indexes, initialized by  $(g_0, \eta_0, \dot{\eta}_0)$ :

$$\begin{aligned} \text{For } j = 1, 2, \dots, N, \text{ do :} \\ k = a(j), \\ g_j = g_k {}^k g_j, \\ \eta_j = Ad_{j g_k} \eta_k + \eta_{j/k}, \\ \dot{\eta}_j = Ad_{j g_k} \dot{\eta}_k + ad_{\eta_j} \eta_{j/k} + \dot{\eta}_{j/k}, \end{aligned} \quad (39)$$

where from now on,  $\eta_{j/k} = ({}^k g_j^{-1} {}^k \dot{g}_j)^\vee \in \mathbb{R}^6$  denotes the twist of relative velocities between  $\mathcal{B}_j$  and its antecedent  $\mathcal{B}_k$  expressed in the mobile frame of  $\mathcal{B}_j$ .

Now, when computing (39), we have two cases:

- If  $\mathcal{J}_j$  is a lumped rigid one, one uses the usual relations:

$${}^k g_j = {}^k g_j(q_{rj}), \quad \eta_{j/k} = A_j \dot{q}_{rj}, \quad \dot{\eta}_{j/k} = A_j \ddot{q}_{rj}, \quad (40)$$

where  $A_j$  is the unit twist supporting the localized rotation of a revolute joint as introduced in remark 8, while if the joint is fixed,  $q_{rj}$  is replaced by a constant angle, that depends on the design.

- If  $\mathcal{J}_j$  is a soft joint,  $({}^k g_j, \eta_{j/k}, \dot{\eta}_{j/k})$  is the terminal value of the forward integration from  $X = 0$ , where  $({}^k g_j, \eta_{j/k}, \dot{\eta}_{j/k})(0) = (1_{4 \times 4}, 0_{6 \times 1}, 0_{6 \times 1})$ , to  $l_j$ , of the following system of space ODEs, deduced from (34) and its time derivatives:

$$\frac{d}{dX} \begin{pmatrix} {}^k g_j \\ \eta_{j/k} \\ \dot{\eta}_{j/k} \end{pmatrix} = \begin{pmatrix} {}^k g_j \hat{\xi}_j \\ -ad_{\xi_j} \eta_{j/k} + \dot{\xi}_j \\ -ad_{\xi_j} \dot{\eta}_{j/k} - ad_{\dot{\xi}_j} \eta_{j/k} + \ddot{\xi}_j \end{pmatrix}, \quad (41)$$

where using the strain based parametrization of (1),  $(\xi_j, \dot{\xi}_j, \ddot{\xi}_j) = (\xi_j^g + B_j \Phi_j q_{\epsilon,j}, B_j \Phi_j \dot{q}_{\epsilon,j}, B_j \Phi_j \ddot{q}_{\epsilon,j})$ . To summarize, if  $\mathcal{J}_j$  is rigid, the joint kinematics between  $\mathcal{B}_k$  and  $\mathcal{B}_j$  are directly given by the discrete (lumped) relations (40), while if it is soft, they are defined by the continuum

composition of poses, velocities and accelerations along the distributed joints, with (41). These discrete and continuous kinematic transfers along the joints are illustrated in the top two diagrams of the figure 4.

2) *Backward dynamics*: The model that relates the wrenches transmitted between the  $\mathcal{B}_j$ s (including the tip cross-sections of the rods) consists of the backward recursion on body indexes:

$$\begin{aligned} &\text{For } j = N, N-1, \dots, 1, \text{ do :} \\ &F_j = \mathcal{M}_j \dot{\eta}_j - ad_{\eta_j}^T \mathcal{M}_j \eta_j + F_{\text{ext},j} + \sum_{l \in s(j)} {}^j F_l, \end{aligned} \quad (42)$$

which are initialized by the wrenches applied to the tip of the branches as detailed in the Remark 12 below.

Now when computing (42), we have two cases:

- If  $\mathcal{J}_l$  is rigid, we use the usual relations:

$${}^j F_l = Ad_{l_{g_j}}^T F_l, \quad (43)$$

- If  $\mathcal{J}_l$  is soft,  ${}^j F_l = -\Lambda_l(0)$  is the result of the backward integration from  $X = l_l$  to 0 of:

$$\Lambda_l' = ad_{\xi_l}^T \Lambda_l + \mathcal{M}_l \dot{\eta}_l - ad_{\eta_l}^T \mathcal{M}_l \eta_l - \bar{F}_{\text{ext},l}, \quad (44)$$

starting from initial conditions  $\Lambda_l(l_l) = -F_l$ . In words, if the  $\mathcal{J}_l$  joint is rigid,  $F_l$  is directly transmitted to  $\mathcal{B}_j$  through the rigid transport of (43), while if it is soft, the contact wrench  $-F_l$  is transmitted to  $\mathcal{B}_j$  through the continuum transport of the wrench along the rod that connects  $\mathcal{B}_j$  to  $\mathcal{B}_l$  according to its dynamic balance (44). This context is illustrated in figure 4 (bottom).

*Remark 12*: The initial conditions of the backward dynamics enter through the sum term in (42), when  $j$  is the index, say  $p$ , of a terminal body of a branch. In particular, if such a terminal body does not support any cut joint, it does not contribute to the initial conditions in (42). Otherwise, let us consider two terminal bodies  $\mathcal{B}_p$  and  $\mathcal{B}_s$  connected by a cut revolute joint as in Section II. Then, using the kinematic model of constraints (38) and the duality between twists and wrenches, we have:

$$F_{s+} = J_{s|p}^T \lambda_{s|p}, \quad F_{p+} = -Ad_{s+g_{p+}}^T F_{s+}, \quad (45)$$

where  $\lambda_{s|p} \in \mathbb{R}^5$  is a vector of reaction generalized forces, dual of  $\Phi_{s|p}$  in (38), which defines five components of the Lagrangian vector  $\lambda$  in (6), while  $\lambda_{s|p} \in \mathbb{R}^6$  if the cut joint is fixed ■

*Remark 13*: Note that in (44), one needs to use the inertial fields  $(g_l, \eta_l, \dot{\eta}_l)(\cdot)$ , that can be computed with the formulas:

$$\begin{aligned} g_l(X) &= g_j {}^j g_l(X), \\ \eta_l(X) &= Ad_{jg_l(X)} \eta_j + \eta_{l/j}(X), \\ \dot{\eta}_l(X) &= Ad_{jg_l(X)} (\dot{\eta}_j + ad_{\eta_{l/j}(X)} \eta_j) + \dot{\eta}_{l/j}(X), \end{aligned} \quad (46)$$

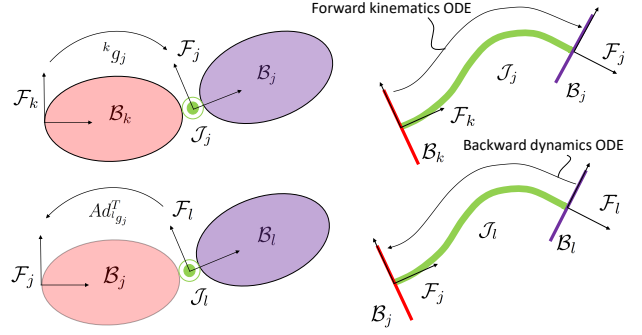


Fig. 4: Forward transport of poses (top) and backward transport of wrenches (bottom) over a rigid (left) and a soft joint (right). The joints are coloured green, the rigid bodies red and purple.

where the inter-body kinematic fields  $({}^k g_j, \eta_{j/k}, \dot{\eta}_{j/k})(\cdot)$  along the rods are known after the first pass through the integration of (41) ■

3) *Summary of the IDM*: The IDM algorithm progresses as follows. First, a forward pass initialized by  $(g_0, \eta_0, \dot{\eta}_0)$  and fed by  $(q, \dot{q}, \ddot{q})$ , computes from the basis  $\mathcal{B}_0$  to all the tip branches, the inertial poses, velocities and accelerations through the composition of the kinematic model of rigid bodies and soft joints (39,40,41) and using (46). These kinematic variables are used to feed a second backward pass that computes the inter-body wrenches from the tip branches to the basis  $\mathcal{B}_0$  through the composition of the dynamic model of rigid bodies and soft joints (42,43,44). According to (45), this second pass is initialized by the contact forces  $\lambda$  transmitted by the cut joints, that are part of the inputs to the algorithm. While running, the second pass provides all the contact wrenches  $F_j$  and  $\Lambda_j(\cdot)$  transmitted across the rigid bodies and soft joints to which we apply the projective relations:

$$Q_{ar,j} = A_j^T F_j, \quad Q_{ae,j} = - \int_0^{l_j} \Phi_j^T \mathcal{B}_j^T \Lambda_j dX, \quad (47)$$

for all lumped revolute and soft joints. Then all the values of (47) are used to fill  $Q_a = (Q_{ar}^T, Q_{ae}^T)^T$ . Moreover, if the system has a free floating basis, the algorithm also provides  $F_0 = F_a$  which completes the outputs  $(F_a^T, Q_a^T)^T$  of the input-output map (21).

### B. Newton-Euler TIDM of a SMMS

The TIDM is derived by calculating the linear perturbations of the outputs of the IDM produced by linear perturbations of its inputs. Practically, it is simply deduced from the IDM by applying the variation  $\Delta$  to all the above expressions and exploiting the properties of the  $Ad$  and  $ad$  maps. Like the IDM, the TIDM consists of two recursions on the body indexes, one (tangent) forward, computes the variations of kinematics, the second, (tangent) backward, computes those of the

internal wrenches transmitted along the structure.

1) *Tangent forward kinematics*: It is deduced from the variation  $\Delta$  of (39):

For  $j = 1, 2, \dots, N$ , do : (48)

$$\begin{aligned} k &= a(j), \\ \Delta\zeta_j &= Ad_{j\ g_k} \Delta\zeta_k + \Delta\zeta_{j/k}, \\ \Delta\eta_j &= Ad_{j\ g_k} \Delta\eta_k + ad_{(Ad_{j\ g_k} \eta_k)} \Delta\zeta_{j/k} + \Delta\eta_{j/k}, \\ \Delta\dot{\eta}_j &= Ad_{j\ g_k} \Delta\dot{\eta}_k + ad_{(Ad_{j\ g_k} \dot{\eta}_k)} \Delta\zeta_{j/k} - ad_{\eta_{j/k}} \Delta\eta_j \\ &\quad + ad_{\eta_j} \Delta\eta_{j/k} + \Delta\dot{\eta}_{j/k}, \end{aligned}$$

which is initialized by the basis kinematic variations  $(\Delta\zeta_0, \Delta\eta_0, \Delta\dot{\eta}_0)$ , and where:

• If  $\mathcal{J}_j$  is a lumped rigid joint, one simply has:

$$(\Delta\zeta_{j/k}, \Delta\eta_{j/k}, \Delta\dot{\eta}_{j/k}) = A_j(\Delta q_{rj}, \Delta\dot{q}_{rj}, \Delta\ddot{q}_{rj}). \quad (49)$$

• If  $\mathcal{J}_j$  is soft,  $(\Delta\zeta_{j/k}, \Delta\eta_{j/k}, \Delta\dot{\eta}_{j/k})$  is the terminal value of the forward integration:

$$\begin{aligned} \frac{d}{dX} \begin{pmatrix} \Delta\zeta_{j/k} \\ \Delta\eta_{j/k} \\ \Delta\dot{\eta}_{j/k} \end{pmatrix} &= \begin{pmatrix} -ad_{\xi_j} \Delta\zeta_{j/k} \\ -ad_{\xi_j} \Delta\eta_{j/k} + ad_{\eta_{j/k}} \Delta\xi_j \\ -ad_{\xi_j} \Delta\dot{\eta}_{j/k} - ad_{\dot{\xi}_j} \Delta\eta_{j/k} \end{pmatrix} \\ &\quad + \begin{pmatrix} \Delta\xi_j \\ \Delta\dot{\xi}_j \\ ad_{\eta_{j/k}} \Delta\dot{\xi}_j + ad_{\dot{\eta}_{j/k}} \Delta\xi_j + \Delta\ddot{\xi}_j \end{pmatrix}, \quad (50) \end{aligned}$$

with initial conditions  $(\Delta\zeta_{j/k}, \Delta\eta_{j/k}, \Delta\dot{\eta}_{j/k})(0) = 0_{6 \times 3}$ , and  $(\Delta\xi_j, \Delta\dot{\xi}_j, \Delta\ddot{\xi}_j) = B_j \Phi_j(\Delta q_{\epsilon,j}, \Delta\dot{q}_{\epsilon,j}, \Delta\ddot{q}_{\epsilon,j})$ .

2) *Tangent backward dynamics*: Applying the variation  $\Delta$  to (42) gives:

For  $j = N, N-1, \dots, 1$ , do : (51)

$$\begin{aligned} \Delta F_j &= \mathcal{M}_j \Delta\dot{\eta}_j - ad_{\eta_j}^T \mathcal{M}_j \Delta\eta_j - ad_{\Delta\eta_j}^T \mathcal{M}_j \eta_j \\ &\quad + \Delta F_{\text{ext},j} + \sum_{l \in s(j)} \Delta^j F_l, \end{aligned}$$

which is initialized by the variations of the wrenches applied onto the terminal bodies of the branches i.e., through some  $\Delta^j F_l$  when  $\mathcal{B}_j$  is a terminal body. These variations take different expressions depending whether the terminal body supports a virtually cut joint or not (see section below).

Once again, we have two cases:

• If  $\mathcal{J}_l$  is a rigid joint, one has:

$$\Delta^j F_l = Ad_{l\ g_j}^T \Delta F_l + \Delta Ad_{l\ g_j}^T F_l, \quad (52)$$

which can be detailed as:

$$\Delta^j F_l = Ad_{l\ g_j}^T \Delta F_l - Ad_{l\ g_j}^T ad_{\Delta\zeta_{l/j}}^T F_l, \quad (53)$$

• If  $\mathcal{J}_l$  is soft,  $\Delta^j F_l = -\Delta\Lambda_l(0)$  is the result of the backward integration from  $X = l_l$  to 0 of:

$$\begin{aligned} \Delta\Lambda'_l &= \mathcal{M}_l \Delta\dot{\eta}_l - ad_{\eta_l}^T \mathcal{M}_l \Delta\eta_l - ad_{\Delta\eta_l}^T \mathcal{M}_l \eta_l \\ &\quad + ad_{\xi_l}^T \Delta\Lambda_l + ad_{\Delta\xi_l}^T \Lambda_l - \Delta\bar{F}_{\text{ext},l}, \quad (54) \end{aligned}$$

starting from initial conditions  $\Delta\Lambda_l(l_l) = -\Delta F_l$ . In these ODEs,  $\Delta\xi_l = B_l \Phi_l \Delta q_{\epsilon,l}$ , while  $\Delta\eta_l, \Delta\dot{\eta}_l$ , and even  $\Delta\zeta_l$  (which can be required by  $\Delta\bar{F}_{\text{ext},l}$ ), are all deduced by varying (46), that provides formulae similar to (50) with  $(l, j)$  instead of  $(j, k)$ .

Once all the  $\Delta F_j$ s and  $\Delta\Lambda_j(\cdot)$ s known, one can calculate the outputs of the *TIDM* i.e.,:

$$\Delta Q_{ar,j} = A_j^T \Delta F_j, \quad \Delta Q_{a\epsilon,j} = - \int_0^{l_j} \Phi_j^T B_j^T \Delta\Lambda_j dX, \quad (55)$$

depending whether it is lumped (rigid) or distributed (soft). Finally, if  $\mathcal{B}_0$  is a free floating basis,  $\Delta F_a = \Delta F_0$ , and the algorithm feeds back all the outputs  $(\Delta F_a^T, \Delta Q_{a,r}^T, \Delta Q_{a,\epsilon}^T)^T$  of the input-output map (26).

3) *Initialization of tangent backward dynamics*: Using the notations of Remark 11, if a terminal body supports a cut, this body can be a leader  $\mathcal{B}_p$  or a follower  $\mathcal{B}_s$  i.e., one can have  $(j, l) = (p, p_+)$  or  $(s, s_+)$  in (51). Then, since  ${}^s g_{s_+}$  and  ${}^p g_{p_+}$  are constant, the initial conditions of (51) are of the form  $\Delta^p F_{p_+} = Ad_{p_+}^T \Delta F_{p_+}$  and  $\Delta^s F_{s_+} = Ad_{s_+}^T \Delta F_{s_+}$ . Thus, we need to calculate  $\Delta F_{p_+}$  and  $\Delta F_{s_+}$ . Regarding  $\Delta F_{s_+}$ , applying  $\Delta$  to the left expression of (45), gives first:

$$\Delta F_{s_+} = J_{s|p}^T \Delta\lambda_{s|p} + (\Delta J_{s|p})^T \lambda_{s|p}, \quad (56)$$

with  $\Delta\lambda_{s|p}$  a sub-vector of  $\Delta\lambda$ , and where we used the variation of the Jacobian matrix (38):

$$\Delta J_{s|p} = \bar{A}_{p_+}^T \begin{pmatrix} \Delta T(\Theta_{s|p})^{-1} & 0_{3 \times 3} \\ 0_{3 \times 3} & p_+ R_{s_+} T(\Theta_{s|p}) \end{pmatrix}, \quad (57)$$

with  $\Delta T$  the second differential of the exponential map of  $SO(3)$  [18]. Regarding  $\Delta F_{p_+}$ , using the properties of  $Ad$  and  $ad$ , the variation  $\Delta$  applied to the right expression of (45) gives:

$$\Delta F_{p_+} = -Ad_{s_+}^T \Delta F_{s_+} - ad_{\Delta\zeta_{s_+/p_+}}^T F_{s_+}, \quad (58)$$

which involves the relative variation:

$$\Delta\zeta_{s_+/p_+} = \Delta\zeta_{s_+} - Ad_{s_+} \Delta\zeta_{p_+}, \quad (59)$$

where  $\Delta\zeta_{s_+} = Ad_{s_+} \Delta\zeta_s$ ,  $\Delta\zeta_{p_+} = Ad_{p_+} \Delta\zeta_p$  are deduced from  $\Delta\zeta_s$  and  $\Delta\zeta_p$  computed by the forward tangent kinematics (48). Therefore, if  $(j, l) = (s, s_+)$ , (51) is initialized with (56,57), and if  $(j, l) = (p, p_+)$ , with (58,59), both being function of  $(\lambda, \Delta\lambda)$ . At last, if a terminal body of a branch does not support any virtual cut, we simply take  $\Delta^j F_l = 0_{6 \times 1}$  as initial condition.

## VIII. CALCULATION OF $\bar{\mathcal{R}}^{(n)}$ AND $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})$

Referring to Remark 5, the calculation of  $\bar{\mathcal{R}}^{(n)}(\bar{\chi}, t_{n+1})$  and  $(\partial\bar{\mathcal{R}}^{(n)}/\partial\bar{\chi})(\bar{\chi}, t_{n+1})$  requires that of  $\bar{\mathcal{R}}_\lambda^{(n)} = \Phi(g_0, q)$  along with  $\Upsilon(q, t_{n+1})$  and  $\Delta\Upsilon(q, \Delta q, t_{n+1})$  in (22) and (24). These additional computations are by-products of *IDM* and *TIDM* which we will now detail.

### A. Calculation of $\bar{\mathcal{R}}_\lambda^{(n)} = \Phi(g_0, q)$

For each loop virtually cut between  $\mathcal{B}_p$  and  $\mathcal{B}_s$ , one can compute a vector of the type  $\Phi_{p|s}$  by feeding the general formula (36) with  ${}^{p+}g_{s+} = g_{p+}^{-1}g_{s+}$  numerically calculated by the forward kinematics (39) along each branch of the SMMS. Note that each of the  $\Phi_{p|s}$ -vectors defines a set of nonlinear algebraic equations depending in general on  $(g_0, q)$ . Gathering all of these  $\Phi_{p|s}$ -vectors, provides the expected  $\Phi$  of the original system of DAEs (2).

### B. Calculation of $\Upsilon(t_{n+1})$ and $\Delta\Upsilon(t_{n+1})$

To finish the computation of the residual vector (22), it remains to compute  $\Upsilon(t_{n+1}) = Q_{ad}(t_{n+1}) - Q_e$ . As regards  $Q_e$ , we have  $Q_e = (0_{1 \times n_r}^T, Q_\epsilon^T)^T$ , with  $Q_\epsilon$  obtained by gathering the vectors of  $\mathbb{R}^{n_{\epsilon,j}}$ :

$$Q_{\epsilon,j} = \left( \int_0^{l_j} \Phi_j^T(B_j^T \mathcal{H}_j B_j) \Phi_j dX \right) q_{\epsilon,j} = K_{\epsilon\epsilon,j} q_{\epsilon,j}, \quad (60)$$

for all soft  $\mathcal{J}_j$ s. Remarking that  $K_{\epsilon\epsilon,j} \in \mathbb{R}^{n_{\epsilon,j} \times n_{\epsilon,j}}$  is constant, we have  $Q_\epsilon = K_{\epsilon\epsilon} q_\epsilon$  where  $K_{\epsilon\epsilon} = \text{diag}(K_{\epsilon\epsilon,j})$  is the constant stiffness matrix of the entire structure. As regards  $Q_{ad}$ , we have  $Q_{ad}(t_{n+1}) = (\tau_d^T, Q_{ade}^T)^T(t_{n+1})$ , where  $Q_{ade}(t_{n+1})$  is obtained by gathering the vectors  $Q_{ade,j} \in \mathbb{R}^{n_{\epsilon,j}}$ , defined by [26]:

$$Q_{ade,j}(t_{n+1}) = - \int_0^{l_j} \Phi_j^T B_j^T \Lambda_{d,j}(t_{n+1}) dX, \quad (61)$$

for all soft  $\mathcal{J}_j$ s. Similarly, one can compute  $\Delta\Upsilon(t_{n+1}) = \Delta Q_{ad}(t_{n+1}) - \Delta Q_e$  by differentiating (60) and (61). The former variation is merely  $\Delta Q_\epsilon = K_{\epsilon\epsilon} \Delta q_\epsilon$ , while the latter is  $\Delta Q_{ad} = (0_{1 \times n_r}, \Delta Q_{ade}^T(t_{n+1}))^T$ , with:

$$\Delta Q_{ade,j}(t_{n+1}) = - \int_0^{l_j} \Phi_j^T B_j^T \Delta \Lambda_{d,j}(t_{n+1}) dX. \quad (62)$$

### C. Adaptation of the *IDM* and *TIDM* to compute the residual and its Jacobian

Finally, to calculate the two top entries of  $\bar{\mathcal{R}}^{(n)}(\bar{\chi}, t_{n+1})$ , one can numerically integrate by quadrature (60) and (61), and remove  $\Upsilon(t_{n+1})$  from  $\overline{IDM}(\bar{\chi})$ , or more directly, replace in the *IDM*, the left side equation of (47) by:

$$\mathcal{R}_j(\chi, t) = A_j^T F_j - \tau_{d,j}(t), \quad (63)$$

if  $\mathcal{J}_j$  is rigid, and the right side one by:

$$\mathcal{R}_j(\chi, t) = \int_0^{l_j} \Phi_j^T B_j^T (\Lambda_{d,j}(t) + \mathcal{H}_j B_j \Phi_j q_{\epsilon,j} - \Lambda_j) dX, \quad (64)$$

if it is soft. Similarly, one can calculate the two top rows of  $(\partial \bar{\mathcal{R}}^{(n)} / \partial \bar{\chi})(\bar{\chi})$ , by replacing in the *TIDM*, the left-side expression of (55) by:

$$\Delta \mathcal{R}_j(\chi, \Delta \chi, t) = A_j^T \Delta F_j, \quad (65)$$

if  $\mathcal{J}_j$  is rigid, and the right-hand side one by:

$$\Delta \mathcal{R}_j(\chi, \Delta \chi, t) = \int_0^{l_j} \Phi_j^T B_j^T (\Delta \Lambda_{d,j}(t) + \mathcal{H}_j B_j \Phi_j \Delta q_{\epsilon,j} - \Delta \Lambda_j) dX, \quad (66)$$

if  $\mathcal{J}_j$  is soft. Finally, feeding the *IDM* and *TIDM*, with inputs compatible with the Newmark scheme (10,13-15) and using as outputs (63,64) and (66,65) instead of (47) and (55) for every  $\mathcal{J}_j$ , allows to compute directly  $\bar{\mathcal{R}}^{(n)}$  and  $\Delta \bar{\mathcal{R}}^{(n)}$ . This defines two further algorithms, noted  $\overline{IDM}_*$  and  $\overline{TIDM}_*$  in figure 2, where the  $\overline{IDM}_*$  also includes the calculation of  $\bar{\mathcal{R}}_\lambda^{(n)} = \Phi(g_0, q)$  of Section VIII.A.

## IX. SPACE INTEGRATION

The algorithm needs to forward (respect. backward) integrate a sequence of space initial value problems (IVPs) defined by (41) (respect. (44)), along the tree, where the initial conditions of one joint, are fixed by the final conditions imposed by its antecedent body (respect. by its successors). This can be achieved with standard finite-difference integrators as ODE45 of Matlab [26]. Nevertheless, let us remark that  $(\xi_j, \dot{\xi}_j, \ddot{\xi}_j)(\cdot)$  being fixed by the inputs  $(q_{r,j}, \dot{q}_{r,j}, \ddot{q}_{r,j})$  of the algorithm, they can be considered as constants in (41) which turn to be linear ODEs with respect to  $(g_j, \eta_j, \dot{\eta}_j)(\cdot)$ . Similarly, once the first pass is achieved,  $(g_j, \eta_j, \dot{\eta}_j)(\cdot)$  can be considered as fixed in the backward ODEs (44), which become linear with respect to the  $\Lambda_j(\cdot)$ s. Parameterizing the rotational component  $R_j(\cdot)$  of  $g_j(\cdot) = (R_j, r_j)(\cdot)$ , with a unit quaternion field<sup>5</sup>  $H_j(\cdot) : X \in [0, l_j] \mapsto H_j(X) \in \mathbb{R}^4$ , this property of the *IDM* allows us to apply efficient spectral methods to the integration of (41) and (44). These methods can be optimized by exploiting a further property of the *IDM* related to its differential structure. Indeed, if one details all the fields of (41,44) into their angular and linear components, with notations:  $(R_j, r_j) = (H_j, r_j)$ ,  $\xi_j = (K_j^T, \Gamma_j^T)^T$ ,  $\eta_j = (\Omega_j^T, V_j^T)^T$ ,  $\dot{\eta}_j = (\dot{\Omega}_j^T, \dot{V}_j^T)^T$  and  $\Lambda_j = (C_j^T, N_j^T)^T$ , it is straightforward to show that each of these two sets of ODEs enjoys a block triangular differential structure which can be solved in cascade, by integrating block after block in an ordered manner, a sequence of linear systems of the canonical state-form:

$$x' = A(X)x + b(X), \quad X \in [0, l_j], \quad (67)$$

where for each soft  $\mathcal{J}_j$ , the state vector  $x \in \mathbb{R}^3$  or  $\mathbb{R}^4$ , successively takes the meaning of  $H_j$ ,  $r_j$ ,  $\Omega_j$ ,  $V_j$ ,  $\dot{\Omega}_j$ ,  $\dot{V}_j$  for the forward ODEs (41), and of  $N_j$ ,  $C_j$  for the backward one (44). Respecting this order, at each step of the cascade resolution, one uses the result of the previous step to feed  $A$  and  $b$  in (67). Therefore, owing to this differential property of the *IDM*, spectral integration based on collocation over a Chebyshev grid, allows to change ODEs (41,44) into algebraic linear systems of triangular structure, that can be solved efficiently with respect to the successive finite dimensional vectors of the values of the fields  $(H_j, r_j, \Omega_j, V_j, \dot{\Omega}_j, \dot{V}_j, C_j, N_j)(\cdot)$  at the Chebyshev nodes of the grids. Finally, by adding to the linear backward ODE systems, the expressions (64) reformulated as ODEs of the form (67) with  $A = 0$ , one completes the spatial integrations required by the computation of the residual vector

<sup>5</sup>Alternatively, one could use quadrature integration based on Magnus expansion in order to remain in  $SO(3)$  as this is done in [28], [46].

$\overline{\mathcal{R}}^{(n)}$ , while a strictly similar process can be applied with the *TIDM* in order to compute  $\Delta\overline{\mathcal{R}}^{(n)}$ , and finally the expected Jacobian of the residual vector. Before concluding this section, let us note that while all the formulas required by *IDM* and *TIDM* were obtained by symbolic calculations on the Lie group  $SE(3)$ , in their numerical implementation, these expressions are integrated on  $SO(3) \times \mathbb{R}^3$  with  $SO(3)$  parameterized by usual quaternions, as was done in [3], [5].

## X. ILLUSTRATIVE EXAMPLES

We now illustrate the above general algorithm on a set of numerical examples and refer the reader to Appendix 2 for a symbolic illustration. We start by two elementary benches well known from the literature on geometrically exact FEM where they are used to validate new numerical methods in the field [26]. These first examples consists of one single rod in different conditions of connection offered by the algorithm. They are here used to compare several methods recently proposed in soft robotics. Once this is achieved, the section continues with three examples more related to robotics. Note that the algorithm presented in the article has been programmed in Matlab, but is not yet optimised, as mentioned in the conclusion.

### A. Single rod benchmarks

We consider single rods in different conditions of connection with the ground. In the first case, a soft and a stiff rod is cantilevered i.e., fixed at one end and free at the other [17]. In the second, another two soft and stiff rods are kinematically free at both ends according to the so-called flying rod bench [16]. Note that these tests are complementary since they illustrate the case of a fixed and a free base system (locomotor and manipulator), while closed loop systems will be illustrated later with a two rod system. These two first benches were used to validate the strain-based parametrization of (1) against GE-FEM in [26] and to study the relationships between the shooting based resolution and optimal control in [11]. We use them here to provide the reader with a very first idea of the robustness and time-consumption of the algorithm in comparison to the explicit time-integration methods of [24], [26] and [27], as well as to the shooting-based approach of [10] and [11] based on implicit integration. Recall that although all these approaches are based on the same model (Cosserat rods), [26] and [27] use the modal strain reduction (1), while [10], [11] and [24] require no configuration space reduction beyond nodal discretization (based on finite differences in the first two cases, and on DER in the third).

1) *Cantilever rod*: The soft rod is that of [17] with  $l = 10$  m,  $\rho = 7800$  kg.m<sup>-3</sup>,  $E = 10^5$  MPa, and a circular cross section with 0.01 m diameter. The stiff rod is that of [11], with  $l = 0.4$  m,  $\rho = 8000$  kg.m<sup>-3</sup>,  $E = 2 \times 10^3$  MPa and 0.002 m diameter. Each of them is first subject to a horizontal static force  $(f, 0, 0)^T$  in the inertial frame, and released in the vertical gravity at  $t = 0$  s. In figure 5, we

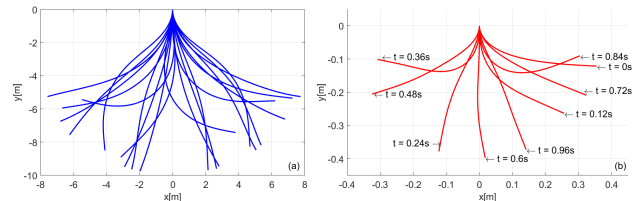


Fig. 5: Clamped soft (a) (blue) and stiff (b) (red) beam released after bending with a horizontal tip force  $f = 50$  N and 10 N respectively. (a) snapshots every 0.5 s for 10 s of simulation. (b) selected snapshots for 1 s of simulation.

reported a set of snapshots simulated with  $f = 50$  N over 10 s, and  $f = 10$  N over 1 s, for the soft and stiff rods respectively. Simulations based on the strain parametrization are performed with the Kirchhoff model and 3 bending modes. Note that the results of all methods that converged are indistinguishable.

2) *Flying rod*: Each of the two rods floats in vacuum without gravity. They are initially inclined and loaded in the inertial plane  $(o, e_1, e_2)$  as indicated in figure 6 (a) and (b). The soft beam is that of Simo [16] i.e.,  $l = 10$  m,  $\mathcal{H} = \text{diag}(5, 5, 5, 10^2, 10^2, 10^2)10^2$ , and  $\mathcal{M} = \text{diag}(10, 10, 10, 1, 1, 1)$  (all in IF units). The stiff rod parameters are those of [11], i.e  $l = 1$  m,  $\rho = 10^3$  kg.m<sup>-3</sup>,  $E = 1$  MPa, for a diameter of 0.1 m. Strain parametrization is performed with the Kirchhoff model with 3 modes for the two curvatures and for torsion. Figures 6 (b) and 6 (c) show a few snapshots of the two rods simulated over 10 s. Note that the results of all the methods that converged are indistinguishable.

With the exception of PyElastica [24], which is programmed in Python, all other methods were implemented in Matlab. All simulations were run on a single computer<sup>6</sup>. In addition, the tuning parameters of each method (time and space-steps, Chebychev grid, quadrature order, etc.) were chosen in such a way as to obtain an accuracy similar to that of the GE-FEM, without penalising any of them in terms of efficiency. In details  $\Delta t = 0.01$  s for the implicit integrators of [11] and that of the paper, tolerance of ODE45 in the time-integration of [26] and [27] is  $10^{-6}$ , while the spatial grids of [27], PyElastica and the present algorithm have 20, 50 and 20 discretization points respectively. Finally, note that in the case of soft rods, the shooting approach of [10] as well as its extension to floating base systems of [11], fail for intrinsic reasons related to the singular character of the underlying optimal control problem from which the spatial BVP is derived [11].

As expected, the simulation times for methods based on explicit integration increase with stiffness since in this case, the time step must be reduced to preserve stability. In

<sup>6</sup>A PC dell (DESKTOP-5F02EKM), Processeur Intel(R) Core(TM) i9-9880H CPU @ 2.30 GHz, RAM 32.0 Go)

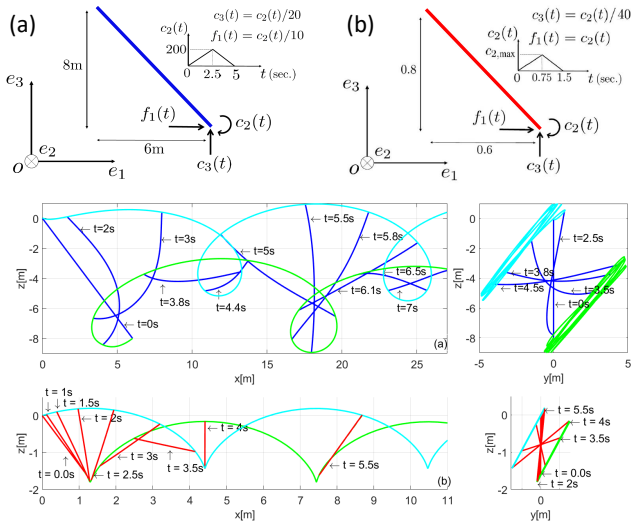


Fig. 6: Top: Conditions of the test for the soft (a) (blue) and stiff (b) (red) rods. Snapshots of the soft (middle) and stiff (bottom) flying rod projected on  $x-z$  (left), and  $y-z$  (right) planes.

Test▼   Method▶	[10], [11]	[24]	[26]	[27]	Article
Stiff cantilever rod	71.9	336	6173	4.5	7.1
Soft cantilever rod	fails	133	1675	1.9	9.1
Stiff flying rod	52.2	36	2806	7.2	8.5
Soft flying rod	fails	1.6	609	1.8	7.9

TABLE I: "Simulation-time over real-time" ratio, for cantilever and flying rod benches obtained with [10], [11], [24] (PyElastica), [26], [27] and the algorithm of the article.

contrast, the article's implicit integrator can be run with a large time step in both cases, which explains why simulation time is independent of rod stiffness. By way of illustration, the critical time step beyond which PyElastica is no longer stable is  $3.4 \times 10^{-6}$ s for the stiff cantilevered rod, and must be divided by  $\sqrt{\alpha}$  when  $E$  is multiplied by  $\alpha \in [0.01, 100]$ , whereas our implicit integrator remains stable over this range of stiffness with a time step of 0.1s. Finally, it should be remembered that these initial estimates of calculation times are far from definitive, and will be systematically studied in the future. For the moment, they tend to show that the extra computations of the Jacobian of the residual vector required by implicit integration are well compensated by the width of the time steps it allows, as it is often the case in nonlinear structural dynamics [47].

### B. Example 1: A mass attached to a rod moved by two flying drones at its two ends

We consider the case of a locomotor, namely a cuboid payload carried by two identical hexarotor flying drones through two identical rods. The ends of the two rods are connected by fixed joints to the drones (UAVs), and to the payload. They are modelled as 3D rigid bodies subject to a lift force of controlled direction and strength. According to our general setting, this SMMS is segmented as indicated in

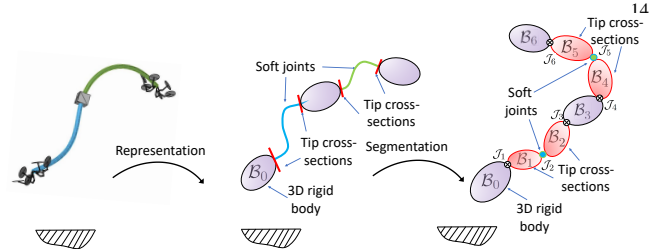


Fig. 7: Segmentation of a cuboid mass attached to two rods moved by two flying drones. The two Cosserat rods are declared as joints  $\mathcal{J}_2$  and  $\mathcal{J}_5$ , respectively. Their two tip-cross sections are declared as rigid bodies ( $\mathcal{B}_1, \mathcal{B}_2$ ) and ( $\mathcal{B}_4, \mathcal{B}_5$ ), respectively.

figure 7 i.e., it contains seven rigid bodies  $\mathcal{B}_{0,1,\dots,6}$ , namely three 3D ones and the four tip cross-sections of the rods. These bodies are connected by six joints  $\mathcal{J}_{1,2,\dots,6}$ , namely four fixed lumped ones and two distributed soft ones. One of the two drones is considered as the reference body  $\mathcal{B}_0$ , and all the other rigid bodies are numbered increasingly along this open chain. The dynamics of the UAV rotors are assumed to be fast enough to impose instant velocity control of external wrenches:

$$j = 0, 6 : F_{\text{ext},j} = k_P e_{\eta_j} + k_I \int e_{\eta_j} dt, \quad (68)$$

with  $k_P$  and  $k_I$  two diagonal  $6 \times 6$  matrices of proportional and integral gains respectively, and  $e_{\eta_j} \in \mathbb{R}^6$  the velocity error defined by:

$$e_{\eta_j} = Ad_{(g_j^{-1} g_j^d)} \eta_j^d - \eta_j, \quad (69)$$

with  $t \mapsto \eta_j^d(t) = ((g_j^d)^{-1} \dot{g}_j^d)^\vee$  a desired velocity time-evolution. In figures 8, 9, we reported the results of a simulation of this system obtained with  $\eta_j^d(t) = (0_{1 \times 3}, V_j^{dT}(t))^T$  and  $V_0^d(t) = (0, 0, \sin(\pi t))^T$ ,  $V_6^d(t) = (0, 0, \sin(\frac{\pi}{2} t))^T$ . The simulation is performed over 10s with a time-step  $\Delta t = 0.01$  s and requires on average, 3 Newton's steps per time-step. The two rods  $\mathcal{J}_2$  and  $\mathcal{J}_5$ , are Kirchhoff, straight at rest, and parameterized with 3 Chebyshev polynomials for each of the three components (two curvatures and torsion) of  $\epsilon_j = K_j$ ,  $j = 2, 5$ . Their geometrical and physical parameters are  $l = 1$  m,  $\rho = 950$  kg.m $^{-3}$ , diameter = 0.07m,  $E = 0.25$  MPa. For the sake of simplicity, gravity is neglected, while we did not seek to optimize the controllers but rather to reveal the richness of the dynamic interactions between the drones and the rods. To this end, taking  $(k_P, k_I) = (\text{diag}(20, 20), \text{diag}(2.5, 2.5))$  in (68), defines a soft controller that keeps the velocity errors limited, while not reacting too much to the elastic restoring torques induced by the rods on the two drones. In Figure 8, snapshots and trajectories of the two drones and the payload  $\mathcal{B}_3$  (see also the multimedia attachment for a video) show how the rods move  $\mathcal{B}_3$  on an almost cyclic circular path. The time evolution of the displacement of these 3 bodies is displayed in figure 9.

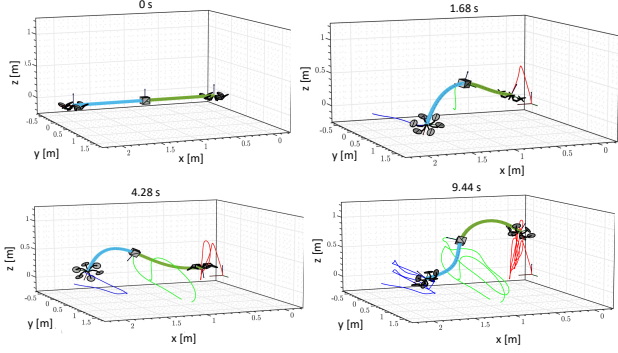


Fig. 8: From left to right and top to bottom: Snapshots and trajectories of the two drones and the cuboid payload between  $t = 0$  s and  $t = 1.68$ ,  $4.22$ , and  $9.44$  s

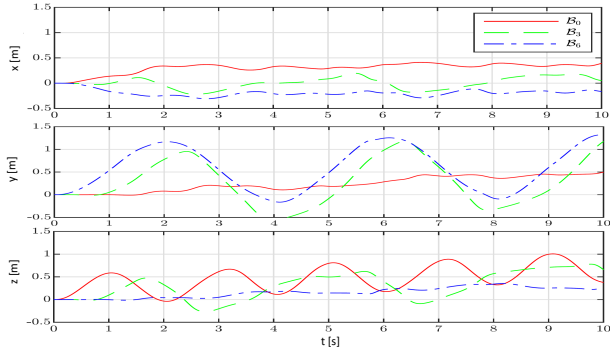


Fig. 9: Displacements of body 0, 3 and 6 vs time in the inertial frame, projected onto the  $x$  (top),  $y$  (middle) and  $z$ -axis (bottom).

### C. Example 2: A continuum parallel robot (CPR)

This second example is a planar continuum parallel manipulator with two identical legs connected at both ends by friction-less revolute joints, the proximal ones are actuated, while the distal one connecting the two legs together, is passive. As imposed by the modelling approach, the closed loop of the CPR is first opened by virtually cutting the passive joint, and the resulting tree-like system is segmented as indicated in figure 10 i.e., it consists of two identical branches, each with two bodies (rod's end cross-sections)  $\mathcal{B}_{1,2}$  and  $\mathcal{B}_{3,4}$ , and four joints  $\mathcal{J}_{1,2,3,4}$ ,  $\mathcal{J}_{1,3}$  being revolute,  $\mathcal{J}_{2,4}$  being soft. The ground defines the fixed reference body  $\mathcal{B}_0$ . The design parameters of this CPR are as follows. The legs are two identical 1-meter Kirchhoff rods with section such that  $I = 1.5708 \times 10^{-8} \text{ m}^4$  and  $A = 3.1416 \times 10^{-6} \text{ m}^2$ . They are made of an ordinary steel, of Young modulus  $E = 2.1 \times 10^5 \text{ MPa}$  and volume mass  $\rho = 7800 \text{ kg.m}^{-3}$ . In this example, the closure loop introduces 2 constraints with two Lagrange multipliers. As for all closed loop systems, any dynamics simulation needs to be preceded with a static mounting phase which consists in closing the loop in statics, starting from an open configuration of the two branches. In this case, this reference configuration is such that the two

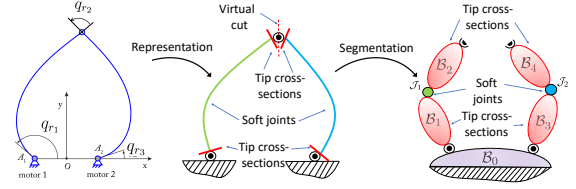


Fig. 10: Segmentation of a planar CPR. The two Cosserat rods are declared as joints  $\mathcal{J}_2$  and  $\mathcal{J}_4$ , respectively. Their two tip-cross sections are declared as rigid bodies ( $\mathcal{B}_1, \mathcal{B}_2$ ) and ( $\mathcal{B}_3, \mathcal{B}_4$ ), respectively.

$t(\text{s}) \in$	$[1, 2]$	$[2, 4]$	$[4, 7]$	$[7, 10]$
$\tau_{d,1}$	$+kt$	$+\tau_{\text{hold}}$	$\tau_{\text{hold}} + k_s(1 - \cos(\theta(t)))$	$+\tau_{\text{hold}}$
$\tau_{d,2}$	$-kt$	$-\tau_{\text{hold}}$	$-\tau_{\text{hold}}$	$-\tau_{\text{hold}}$

TABLE II: Loading time law of the CPR.

links form the sides of a triangle symmetrical about the horizontal axis ( $x$ ) (see figure 13 (a)). Note that this initial configuration is a resting one i.e., it is free of internal stress.

Once this static phasis achieved, the dynamics is started by imposing a prescribed time-evolution of the joint torques  $t \mapsto \tau_d(t) = (\tau_{d1}, \tau_{d2})^T(t)$  defined as indicated in Table II, and plotted in figure 11 (a). The two torques start with zero values and then increase linearly with time, before reaching a constant value that is maintained until one of the two torques oscillates and finally recovers its constant value. In Table II,  $k = 0.5 \text{ Nm/s}$  is the ramp coefficient,  $\tau_{\text{hold}}$  is the maximum torque reached by the ramp i.e., when  $t = 2$  s. The harmonic phase is defined by  $k_s = 7 \text{ Nm}$  and  $\theta(t) = \omega(t - t_{\text{hold}})$  with  $\omega = \frac{2c\pi}{t_s - t_{\text{hold}}}$ ,  $c = 3$  the number of cycles, and  $t_{\text{hold}} = 4$  s and  $t_s = 7$  s, the ending time of the holding and harmonic phase respectively. Note that the sinusoidal addendum to the torque is not symmetric but ranges from 0 to 2. Figures 11 (b) and 12 (a,b) respectively show the time evolution of the joint angles, the Lagrange multipliers in the inertial frame, and the inertial position of the tip (the passive joint) for a  $\Delta t = 0.01$  s and 5 Chebyshev polynomials for each rod curvature. Figure 13 (a,b) display a sequence of snapshots over  $[1, 2]$  s and  $[4, 7]$  s. As observed in these snapshots and the attached video, with the torque law of Table II, the CPR first bends symmetrically with its tip getting closer to the base, then because of the oscillating component added to  $\tau_{d,1}$ , the CPR tilts upwards. While the torque makes three cycles, the up and down movement of the robot makes only two. This is due to inertia effects. When we control the robot by imposing torques, the inertia of the robot can affect the joint angles, as shown in figures 12 (a) and (b). Note that as expected, one can observe structural (harmonic) oscillations superimposed with overall deformations imposed by the torque time variations. This simulation is representative of many others we tested. In any case, we observed that using the constitutive law (33) of Remark 9 applied to  $\mathcal{J}_2$  and  $\mathcal{J}_4$ , a slight internal damping (here  $\mu_2 = \mu_4 = 10^{-4}$ ), is required to stabilize the algorithm. This observation is fully consistent with the



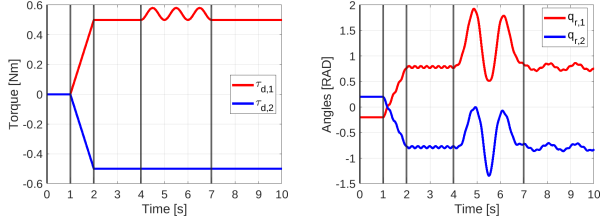


Fig. 11: (a) Actuation torques  $\tau_{d,1}$  and  $\tau_{d,2}$  vs time. (b) Revolute joint angles  $q_{r,1}$  and  $q_{r,2}$  vs time.

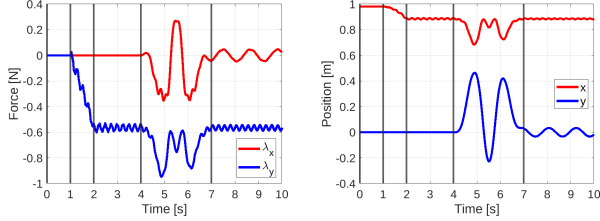


Fig. 12: (a) Two components of  $\lambda$  in the inertial frame vs time. (b) Inertial position  $(x, y)$  of the tip of the robot (passive revolute joint) vs time.

knowledge about stiff systems of DAEs, for which it is well known that the geometric constraints of the residual balance (8) introduce high-frequency parasitic (numerical) modes, that can compromise the stability of simulations [32].

#### D. Example 3: A bio-inspired swimmer actuated with tendons

In this last example, we consider a bio-inspired locomotor, namely an undulatory swimmer of 1m length and non-uniform elliptical cross-sections along its rostro-caudal axis. Its geometry is that of [48] with maximum lateral width of 0.1 m. However, in contrast to [48], we here consider it as being constituted of three segments serially fixed to each other from the head to the tail (see figure 14). The first segment (the head) is a rigid body, the second is a rod internally actuated by a pair of antagonistic tendons as those currently used for TACRs [5], the third is a passive elastic rod. The swimmer is neutrally buoyant i.e.,  $\rho = 10^3 \text{ kg.m}^{-3}$ , and the first and second rod are Kirchhoff with a bending stiffness  $EI = 4.9 \text{ Pa.m}^4$  and  $EI = 4.9 \times 10^{-2} \text{ Pa.m}^4$  respectively. The tendons are parallel to the robot axis and located at

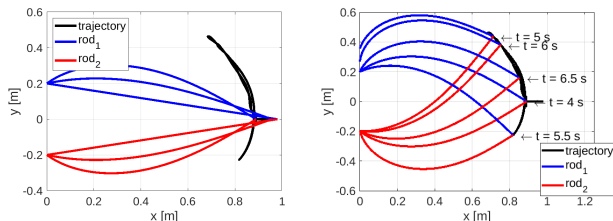


Fig. 13: (a) Snapshot every 0.5 s of the robot between: (a) 1 and 2 s, and (b) 4 and 7 s.

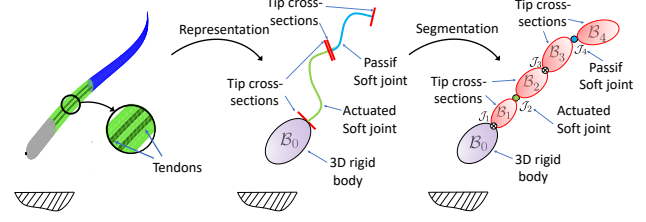


Fig. 14: Segmentation of a bio-inspired swimmer actuated with tendons. The two Cosserat rods are declared as joints  $\mathcal{J}_2$  and  $\mathcal{J}_4$ , respectively. Their two tip-cross sections are declared as rigid bodies  $(\mathcal{B}_1, \mathcal{B}_2)$  and  $(\mathcal{B}_3, \mathcal{B}_4)$ , respectively.

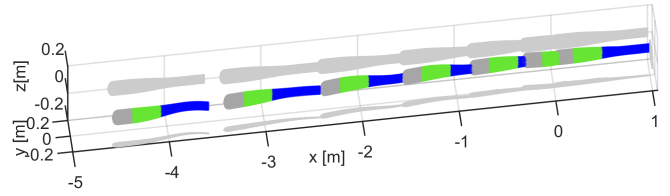


Fig. 15: Snapshots every 1.65 s of a continuum undulatory swimmer internally controlled with a single pair of tendons.

a distance  $D = 0.05 \text{ m}$  from it. According to our general setting, this system is segmented in 5 bodies  $\mathcal{B}_{0,1,2,3,4}$ , with  $\mathcal{B}_0$ , a 3D rigid body modelling the head, and all the others standing for the rod's end cross-sections. All these bodies are connected by four joints  $\mathcal{J}_{1,2,3,4}$ , with  $\mathcal{J}_{1,3}$  two lumped fixed joints, and  $\mathcal{J}_{2,4}$  two distributed soft ones. The hydrodynamic forces are modelled with Lighthill's Large Amplitude Elongated Body Theory (LAEBT), here coupled to the Cosserat model as proposed in [48]. The planar swimming of this bio-inspired system can be achieved by applying to the two tendons, the tensions:

$$T_{\pm}(t) = T_0 + T_1 \sin\left(\omega t \pm \frac{\pi}{2}\right), \quad (70)$$

where indexes  $+$  and  $-$  denote the left and right tendons respectively,  $T_0 > 0$  is a constant antagonistic component ensuring that  $T_{\pm}(t) > 0$  for all  $t$ , and  $T_1 > 0$  is the amplitude of an harmonic component of pulsation  $\omega$ , responsible of the body undulations. In figures 15 and 16, we reported some simulation results carried out over 10 s with  $\Delta t = 0.01 \text{ s}$ , and 5 Chebychev polynomials per rod curvature. The results are obtained by using the control law (70) in the model (32), with  $T_0 = T_1 = 20 \text{ N}$ , and  $\omega = 5\pi \text{ rad/s}$  (see also video). In figure 15, snapshots of the robot are drawn every

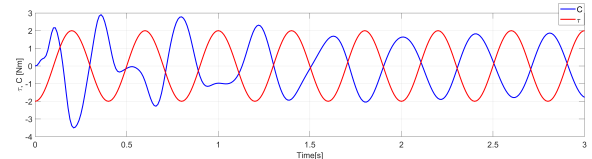


Fig. 16: Time evolutions of  $\tau(t) = D(T_+ - T_-)(t)$ , and  $C = (EI_3 K_{Z,3})(0)$  along the first 3 s of swim.

1.65 s. Figure 16 shows the time evolution (over the first 3 s) of the control torque  $\tau(t) = D(T_+ - T_-)(t)$ , and of the elastic torque  $C = (EI_3 K_{Z,3})(0)$  at the root of the third (elastic) body  $\mathcal{B}_3$ , where the subscript  $Z$ , indicates a component taken along the normal to the plane of the swim.

## XI. CONCLUSION AND PERSPECTIVES

In this paper, we have proposed a new algorithm for solving the forward dynamics of soft and continuum robots consisting of rigid bodies connected in arbitrary topologies by localised joints and/or soft links, possibly actuated or not. The soft joints are modelled as Cosserat rods parameterized with strain modes. Unlike the literature approaches for simulating these models, our approach is designed to work with an implicit time integration scheme. We choose the Newmark scheme which is known to be among the best suited for stiff problems of non-linear structural dynamics. Using such a scheme allows changing the DAEs of the robot into a system of nonlinear algebraic equations. At each time-step, we solve this system with a Newton-Raphson scheme, requiring the residual and its Jacobian. To this aim we proposed a new recursive Newton-Euler algorithm. Although the results are encouraging, in its current state the approach has been implemented and tested as a proof of concept. Therefore, we are currently optimising it with the short-term objective of building a toolbox<sup>7</sup>, dedicated to simulation (and control) of soft and continuum robots. We strive for improvements in terms of robustness, computational efficiency and programming simplicity.

### A. Perspectives for numerical robustness

As expected, the implicit integrator makes the algorithm robust and stable even for large time steps. However, the geometrical constraints introduce high frequency parasitic modes. As a result, in (2) some physical damping has been introduced into the constitutive laws (33). This artificial damping could in general affect the low frequency content of the response of an SMMS. Another solution consists in replacing the Newmark integrator by the HHT method or the  $\alpha$ -method which are designed to solve this problem [32],[49].

### B. Perspectives for numerical efficiency

Due to the intrinsic predictive-corrective structure of the algorithm, the simulation time is directly proportional to the number of iterations of the Newton loop. The computational efficiency of the algorithm can therefore be improved in two ways. Firstly, by reducing the number of Newton iterations (i.e. by improving the convergence of the loop), and secondly, by optimising the calculation of the

residual and, above all, of its Jacobian and its inversion. Regarding the last point, thanks to the strong reduction power of the strain-based parametrization [26], the Jacobian matrix remains moderate in size for typical continuum and soft robotics designs, and its inversion remains reasonable in terms of computation time.

1) *Reducing the number of Newton iterations:* In the case of closed-loop systems, we can add penalties to the augmented Lagrangian in order to increase the convexity of the original variational formulation and improve the convergence of the Newton loop. Technically, these penalties simply add restoring forces and torques to the Lagrange multipliers that initialise the *IDM*. Finally, several other additional improvements based on the scaling of the original DAE system and its tangent linearization, will be applied in order to optimize the conditioning of the residual vector and its Jacobian matrix [39], [50].

2) *Improving spatial integration:* A detailed analysis on the computation times showed that the spatial integration of the forward and backward ODEs of the Cosserat rods (41,50), and (44,54) take more than 90% of the total computation time, of which 70% is spent on the forward kinematics, and 30% on the backward dynamics. Although numerical integration by spectral methods reduces the time required, compared to the usual finite difference integration [26], we are considering other options to further reduce this bottleneck. A first step will be to implement the integrations in a compiled language like C++. A second front of improvements will be the investigation of other integration methods like those based on quadrature and Magnus expansions [46].

3) *Parallelization:* Newton-Euler algorithms are naturally suited to parallelization [33]. As a first example, we can define the relative kinematics of every joint in a parallel loop before entering the forward kinematics routine. In addition, the updates of Remark 13, which are necessary before entering the backward dynamics, can be executed in a parallel loop too. Moreover, depending on the structure of the robot, if we proceed with a breadth-first algorithm to move along the robot tree, we can also parallelise the backward dynamics of every layer of the tree. Beyond this parallelization based on NE formulation, the computation of the Jacobian matrix is the main source of slowdown, since it requires the *TIDM* to be called several times. As this matrix is calculated column-wise, the parallelization of its calculation should considerably reduce the overall calculation time.

### C. Programming simplicity

As any Newton-Euler algorithm, the computational complexity of our algorithm is  $O(n)$ , meaning that it increases linearly with the number  $n$  of joints [33]. This scalability in design and implementation allows it to be applied to complex systems with a large number of joints or complicated

<sup>7</sup>So far, a first version of the Matlab toolbox (for SMMS without closed loops) can be found at the following address: <https://gitlab.univ-nantes.fr/cosseroots/library/matlab>. Likewise, the code used to generate the results of section X.B can be found at: <https://gitlab.univ-nantes.fr/cosseroots/example/t-ro-2023>.

architectures. Among other advantages, it allows for scalable software architectures, being well suited to object-oriented programming.

## XII. APPENDIX 1: DETAILED EXPRESSIONS FOR THE NEWMARK SCHEME

The implicit Newmark scheme takes the generic form (10), where  $a$ ,  $b$  and the two functions  $f$  and  $h$  are defined for any pair of vectors of same dimension  $(x, y)$ , by:

$$a = \gamma/(\beta\Delta t) \quad , \quad b = 1/(\beta\Delta t^2), \quad (71)$$

$$f(x, y) = (1 - (\gamma/\beta))x + \Delta t(1 - (\gamma/(2\beta)))y, \quad (72)$$

$$h(x, y) = -(1/(\beta\Delta t))x + (1 - (1/(2\beta)))y. \quad (73)$$

Where  $(\beta, \gamma)$  are two constant parameters such that  $(\beta, \gamma) = (1/4, 1/2)$  ensures second order accuracy with no damping, a choice that is systematically adopted in the numerical examples. Taking  $(x, y) = (\dot{q}^{(n)}, \ddot{q}^{(n)})$ ,  $(\dot{r}_0^{(n)}, \ddot{r}_0^{(n)})$ ,  $(\dot{\Omega}_0^{(n)}, \ddot{\Omega}_0^{(n)})$  in (72) and (73) defines  $f_q^{(n)}$ ,  $f_r^{(n)}$ ,  $f_\theta^{(n)}$  and  $h_q^{(n)}$ ,  $h_r^{(n)}$ ,  $h_\theta^{(n)}$  of (10) and (11) respectively. With these notations, the matrices  $C$ ,  $A$ ,  $B$  of (13), can be detailed as [11]:

$$C(\nu_0) = \begin{pmatrix} R_0^{(n)} \exp(\hat{\Theta}_0) & r_0^{(n)} + d_0 \\ 0_{1 \times 3} & 1 \end{pmatrix}, \quad (74)$$

$$A(\nu_0) = \begin{pmatrix} \Omega_0 \\ V_0 \end{pmatrix} = \begin{pmatrix} a\Theta_0 + f_\theta^{(n)} \\ R_0^T(a d_0 + f_r^{(n)}) \end{pmatrix}, \quad (75)$$

$$B(\nu_0) = \begin{pmatrix} \dot{\Omega}_0 \\ \dot{V}_0 \end{pmatrix} = \begin{pmatrix} b\Theta_0 + h_\theta^{(n)} \\ R_0^T(b d_0 + h_r^{(n)}) + V_0 \times \Omega_0 \end{pmatrix}.$$

Differentiating the above expressions, one can show that:

$$\frac{\partial C}{\partial \nu_0} = \begin{pmatrix} T(\Theta_0) & 0_{3 \times 3} \\ 0_{3 \times 3} & R_0^T \end{pmatrix}, \quad \frac{\partial A}{\partial \nu_0} = \begin{pmatrix} a1_{3 \times 3} & 0_{3 \times 3} \\ \hat{V}_0 T(\Theta_0) & aR_0^T \end{pmatrix},$$

$$\frac{\partial B}{\partial \nu_0} = \begin{pmatrix} b1_{3 \times 3} & 0_{3 \times 3} \\ E_0 T(\Theta_0) + a\hat{V}_0 & L_0 R_0^T \end{pmatrix}, \quad (76)$$

where  $E_0 = (\hat{A}_0 - \hat{\Omega}_0 \hat{V}_0)$ ,  $L_0 = (b1_{3 \times 3} - a\hat{\Omega}_0)$ ,  $A_0 = \hat{V}_0 + \Omega_0 \times V_0$ , and  $(R_0^T \Delta R_0)^\vee = T(\Theta_0) \Delta \Theta_0$ , is the differential of the exponential of  $SO(3)$  [16].

## XIII. APPENDIX 2: APPLICATION OF THE ALGORITHM TO AN ILLUSTRATIVE EXAMPLE

To illustrate the simulation algorithm, we apply it to the simple case of the cantilevered rod of section X (see figure 17). The rod has a length  $l$ , and a circular cross-section of area and axial moment of inertia  $A$  and  $I$  respectively. Its Young's modulus is  $E$ , and its density is  $\rho$ . According to the segmentation of section VI.A, the rod corresponds to two cross-sections  $\mathcal{B}_1$  and  $\mathcal{B}_2$  connected together through a soft joint  $\mathcal{J}_2$ . This soft joint is modeled as a Kirchhoff rod moving in the  $(e_1, e_2)$  plane and straight at rest, i.e.  $B_2 = (0, 0, 1, 0, 0, 0)^T$  and  $\xi_2^o = (0, 0, 0, 1, 0, 0)^T$ , where  $B_2$  selects the curvature field  $K_{2,Z}(\cdot)$  as the system's unique

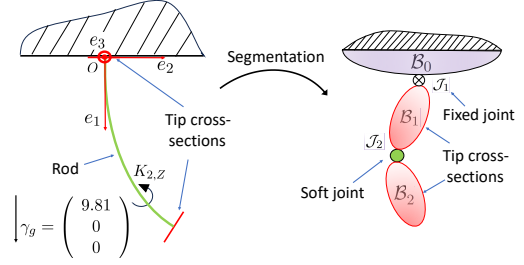


Fig. 17: Segmentation and parametrization of a cantilevered rod. The rod is fixed to the ground through  $\mathcal{J}_1$ . Its tip-cross sections define rigid bodies  $(\mathcal{B}_1, \mathcal{B}_2)$ , its internal domain is a soft joint  $\mathcal{J}_2$ . The basis  $\mathcal{B}_0$  is fixed.  $\mathcal{F}_e = \mathcal{F}_0 = (O, e_1, e_2, e_3)$  is the inertial frame.  $K_{2,Z}(\cdot)$  is the curvature field of the rod deformed in the plane  $(e_1, e_2)$ .  $\gamma_g = (9.81, 0, 0)^T$  is the gravity acceleration field in  $\mathcal{F}_e$ .

degrees of freedom. Once reduced on the basis  $\Phi_2$  of the first  $n_\epsilon$  Chebyshev polynomials, this field is replaced by the vector  $q_{\epsilon,2} = q_2$  of the  $n_\epsilon$  modal curvature coordinates of the rod. Referring to the general context of section IV, we thus have  $\chi = (q_2, \dot{q}_2, \ddot{q}_2)$  and  $\bar{\chi} = q_2$ , while the residual vector (constrained by the scheme) and its Jacobian are  $\bar{\mathcal{R}} = \bar{\mathcal{R}}_2$  and  $(\partial \bar{\mathcal{R}} / \partial \bar{\chi}) = (\partial \bar{\mathcal{R}}_2 / \partial q_2)$  respectively. According to the flowchart 2, the predictor-corrector algorithm consists of an inner Newton correction loop included in an outer time-loop. At each new time step  $t_{n+1}$ , the state  $(q_2, \dot{q}_2)$  is first predicted by forcing  $\ddot{q}_2 = 0_{n_\epsilon \times 1}$  in the Newmark scheme (10) (inertial predictor), before entering the Newton loop, whose correction equation (18) requires computing both  $\bar{\mathcal{R}}_2$  and  $(\partial \bar{\mathcal{R}}_2 / \partial q_2)$ .

The computation of  $\bar{\mathcal{R}}_2$  is performed with the  $\overline{IDM}_*$  algorithm of section VIII.C. Since  $\mathcal{J}_2$  is soft,  $\bar{\mathcal{R}}_2$  is defined by (64), where  $j = 2$ ,  $\Lambda_{d,2} = 0_{6 \times 1}$  (no internal actuation),  $\int_0^l \Phi_2^T B_2^T \mathcal{H}_2 B_2 \Phi_2 dX = EI 1_{n_\epsilon \times n_\epsilon}$  ( $\Phi_2$  is orthonormal), and  $\Lambda_2(\cdot)$  is calculated by backward integrating (44) from  $\Lambda_2(l) = 0_{6 \times 1}$  (no tip load), with  $\mathcal{M}_2 = \rho \text{diag}(2I, I, I, A, A, A)$  and  $\bar{F}_{\text{ext},2} = (0_{1 \times 3}, (R_0^T \gamma_g)^T)^T$  (gravity). The integration of (44) requires first to compute the inertial kinematic fields along the rod  $(g_2, \eta_2, \dot{\eta}_2)(\cdot)$ . Since  $\mathcal{J}_1$  is fixed, one can take  $\mathcal{F}_1 = \mathcal{F}_e$ , and  $(g_2, \eta_2, \dot{\eta}_2)(\cdot) = ({}^1g_2, \eta_{2/1}, \dot{\eta}_{2/1})(\cdot)$  is obtained by forward integrating (41), starting from  $({}^1g_2, \eta_{2/1}, \dot{\eta}_{2/1})(0) = (1_{4 \times 4}, 0_{6 \times 1}, 0_{6 \times 1})$ , with  $(\xi_2, \dot{\xi}_2, \ddot{\xi}_2) = (\xi_2^o + B_2 \Phi_2 q_2, B_2 \Phi_2 \dot{q}_2, B_2 \Phi_2 \ddot{q}_2)$  and  $(\dot{q}_2, \ddot{q}_2)$  function of  $q_2$ , and  $(q_2^{(n)}, \dot{q}_2^{(n)}, \ddot{q}_2^{(n)})$  through the Newmark scheme (10). The computation of  $(\partial \bar{\mathcal{R}}_2 / \partial q_2)$  is performed with the  $\overline{TIDM}_*$  algorithm of section VIII.C. This algorithm calculates the variation  $\Delta \bar{\mathcal{R}}_2$  of the outputs of the algorithm  $\overline{IDM}_*$ , due to a variation  $\Delta q_2$  of its inputs. Exploiting the identity  $\Delta \bar{\mathcal{R}}_2 = (\partial \bar{\mathcal{R}}_2 / \partial q_2) \Delta q_2$  shows that by imposing  $\Delta q_2 = \delta_\alpha$ ,  $\alpha = 1, 2, \dots, n_\epsilon$ , we can calculate all the columns of  $(\partial \bar{\mathcal{R}}_2 / \partial q_2)$  one after the other. This algorithm proceeds as the previous one except that (66) replaces (64) and the back-

---

**Algorithm 1:** Simulation of the cantilevered rod.
 

---

```

1 Initial conditions:
2  $q_2 = q_{2\text{init}}, \dot{q}_2 = \ddot{q}_2 = 0_{n_\epsilon \times 1};$ 
3 Boundary conditions:  ${}^1q_2(0) = 1_{4 \times 4},$ 
    $\eta_{2/1}(0) = \dot{\eta}_{2/1}(0) = \Lambda_2(l) = 0_{6 \times 1};$ 
4  $\Delta\zeta_{2/1}(0) = \Delta\eta_{2/1}(0) = \Delta\dot{\eta}_{2/1}(0) = \Delta\Lambda_2(l) = 0_{6 \times 1};$ 
5 for  $n = 0 \dots (t_f - t_0)/\Delta t$  do
6    $q_2^{(n)} := q_2, \dot{q}_2^{(n)} := \dot{q}_2, \ddot{q}_2^{(n)} := \ddot{q}_2;$ 
7   Prediction (Newmark scheme s.t.  $\ddot{q}_2 = 0_{n_\epsilon \times 1}$ ):
8    $q_2 = q_2^{(n)} - (1/b)h_q^{(n)}, \dot{q}_2 = \dot{q}_2^{(n)} - (a/b)h_{\dot{q}}^{(n)}, \ddot{q}_2 = 0_{n_\epsilon \times 1};$ 
9   Correction:  $\bar{\mathcal{R}}_2 = 1;$ 
10  while  $\bar{\mathcal{R}}_2 > \epsilon$  do
11    Computation of  $\bar{\mathcal{R}}_2$  with  $\overline{TDM}_*$ :
12     $\xi_2 := \xi_2^0 + B_2\Phi_2q_2, \xi_{\dot{2}} := B_2\Phi_2\dot{q}_2, \xi_{\ddot{2}} := B_2\Phi_2\ddot{q}_2$ 
13    Integrate forward from  $X = 0$  to  $l$  Eq. (41) to get:
14     $g_2 := {}^1g_2, \eta_2 := \eta_{2/1}, \dot{\eta}_2 := \dot{\eta}_{2/1};$ 
15    Integrate backward from  $X = l$  to  $0$  Eq. (44) to get:  $\Lambda_2;$ 
16    Compute  $\bar{\mathcal{R}}_2$  using (64);
17    Computation of  $(\partial\bar{\mathcal{R}}_2/\partial q_2)$  with  $\overline{TIDM}_*$ :
18    for  $\alpha = 1 \dots n_\epsilon$  do
19       $\Delta\xi_2 := B_2\Phi_2\delta_\alpha, \Delta\xi_{\dot{2}} := a\Delta\xi_2, \Delta\xi_{\ddot{2}} := b\Delta\xi_2$ 
20      Integrate forward from  $X = 0$  to  $l$  Eq. (50) to get:
21       $\Delta\zeta_2 := \Delta\zeta_{2/1}, \Delta\eta_2 := \Delta\eta_{2/1}, \Delta\dot{\eta}_2 := \Delta\dot{\eta}_{2/1};$ 
22      Integrate backward from  $X = l$  to  $0$  Eq. (54) to get  $\Delta\Lambda_2;$ 
23      Compute  $\Delta\bar{\mathcal{R}}_2$  using (66);
24       $\alpha^{\text{th}}$  column of  $(\partial\bar{\mathcal{R}}_2/\partial q_2) := \Delta\bar{\mathcal{R}}_2;$ 
25    end
26  end
27 end
  
```

---

ward and forward ODEs (44) and (41) are replaced by (54) and (50), the first being initialized with  $\Delta\Lambda_2(l) = 0_{6 \times 1}$ , and the second by  $\Delta\zeta_{2/1}(0) = \Delta\eta_{2/1}(0) = \Delta\dot{\eta}_{2/1}(0) = 0_{6 \times 1}$ . In (54)  $\Delta\bar{F}_{\text{ext},2} = (0_{1 \times 3}, (\Delta R_2^T \gamma_g)^T)^T$  with  $\Delta R_2^T \gamma_g = ((R_2^T \gamma_g)^\wedge, 0_{3 \times 3})\Delta\zeta_2$ . In (50),  $(\Delta\dot{q}_2, \Delta\ddot{q}_2) = (a\Delta q_2, b\Delta\dot{q}_2)$ . This simulation algorithm is summarized by the pseudocode Alg. 1, where all space-integrations of ODEs are achieved in cascade with a standard spectral method as indicated in section IX. Note also that in this particular case,  $\mathcal{J}_1$  being fixed and the cross-sections of the rod ends  $\mathcal{B}_1$  and  $\mathcal{B}_2$  being massless, their model, used in the boundary conditions, introduces trivial twist and wrench transfers.

#### XIV. ACKNOWLEDGMENT

We thank Frédéric Jourdan for the instructive discussions on the geometric content of this article, and dedicate this work to the memory of our colleague and friend Wisama Khalil, Newton-Euler dynamics specialist, who died accidentally on 15 November 2017. This work was supported by the French ANR COSSEROOTS (ANR-20-CE33-0001), the French project CominLabs Mambo, the Office of Naval Research Global under Grant N62909-21-1-2033, and by Khalifa University under Grant RC1-2018-KUCARS.

#### REFERENCES

- [1] W. Guanlun and S. Guanglin, “Experimental statics calibration of a multi-constraint parallel continuum robot,” *Mechanism and Machine Theory*, vol. 136, pp. 72–85, 2019.
- [2] G. Bottcher, S. Lilje, and J. Burgner-Kahrs, “Design of a reconfigurable parallel continuum robot with tendon-actuated kinematic chains,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1272–1279, 2021.
- [3] F. Boyer, M. Porez, and W. Khalil, “Macro-continuous computed torque algorithm for a three-dimensional eel-like robot,” *Robotics, IEEE Transactions on*, vol. 22, no. 4, pp. 763–775, Aug 2006.

- [4] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, “Design and control of concentric-tube robots,” *IEEE Trans. Robot.*, vol. 26, no. 2, pp. 209–225, apr 2010.
- [5] D. Rucker and R. Webster, “Statics and dynamics of continuum robots with general tendon routing and external loading,” *Robotics, IEEE Transactions on*, vol. 27, no. 6, pp. 1033–1044, Dec 2011.
- [6] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, “Dynamic model of a multibending soft robot arm driven by cables,” *IEEE Transactions on Robotics*, vol. 30 (5), pp. 1109–1122, 2014.
- [7] F. Boyer and F. Renda, “Poincaré’s equations for cosserat media: Application to shells,” *Journal of Nonlinear Science*, 2016.
- [8] S. S. Antman, “Ordinary differential equations of nonlinear elasticity i: Foundations of the theories of non-linearly elastic rods and shell,” *Arch. Rat. Mech. Anal.*, vol. 61, no. 4, pp. 307–351, 1976.
- [9] M. E. Gurtin, *Topics in Finite Elasticity*. Society for Industrial and Applied Mathematics, 1981.
- [10] J. Till, V. Aloil, and C. Rucker, “Real-time dynamics of soft and continuum robots based on cosserat-rod models,” *Inter. Journal of Robotics Research*, vol. 38, no. 6, pp. 723–746, April 2019.
- [11] F. Boyer, V. Lebastard, F. Candelier, F. Renda, and M. Alamir, “Statics and dynamics of continuum robots based on cosserat rods and optimal control theories,” *IEEE Transactions on Robotics*, pp. 1–19, 2022.
- [12] H. Goldstein, C. P. Poole, and J. L. Safko, *Classical Mechanics*, 3<sup>rd</sup> ed. Pearson, 2001.
- [13] L. Meirovitch, *Methods of Analytical Dynamics*. McGraw-Hill, New York, 1970.
- [14] —, *Dynamics and Control of structures*. Wiley, New York, 1989.
- [15] J. Allard, S. Cotin, F. Faure, P. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni, “Sofa—an open source framework for medical simulation.” in *MMVR 15-Medicine Meets Virtual Reality*, vol. 125, 2007, pp. 13–18.
- [16] J. Simo, “A finite strain beam formulation. the three-dimensional dynamic problem. part i,” *Computer Methods in Applied Mechanics and Engineering*, vol. 49, no. 1, pp. 55 – 70, 1985.
- [17] F. Boyer and D. Primault, “Finite element of slender beams in finite transformations: A geometrically exact approach,” *International Journal for Numerical Methods in Engineering*, vol. 59, no. 5, pp. 669–702, 2004.
- [18] V. Sonneville, A. Cardona, and O. Bruls, “Geometrically exact beam finite element formulated on the special euclidean group se(3),” *Comput. Methods Appl. Mech. Engrg.*, vol. 268, pp. 451–474, 2014.
- [19] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, “Discrete elastic rods,” in *ACM transactions on graphics (SIGGRAPH)*. ACM, 2008, pp. 63:1–63:12.
- [20] B. Miklos, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun, “Discrete viscous threads,” in *ACM Transactions on Graphics 29(116)*, pp. 1–10.
- [21] E. Grinspun and A. Secord, *Discret Differential Geometry: An Applied Introduction*. SIGGRAPH, 2006, ch. 1: Introduction to Discrete Differential Geometry: The geometry of plane curves, pp. 1–4.
- [22] R. Goldenthal, D. Harmon, F. Raanan, M. Bercovier, and E. Grinspun, “Efficient simulation of inextensible cloth,” in *July 2007ACM Transactions on Graphics 26(3)*, pp. 49:1–49:7.
- [23] M. Gazzola, L. H. Dudte, A. G. McCormick, and L. Mahadevan, “Forward and inverse problems in the mechanics of soft filaments,” *Royal Society Open Science*, vol. 5, no. 171628, 2017.
- [24] M. Gazzola, “Pyelastica,” <https://www.cosseratrods.org/>.
- [25] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, “Discrete cosserat approach for multisection soft manipulator dynamics,” *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1518–1533, Dec 2018.
- [26] F. Boyer, V. Lebastard, F. Candelier, and F. Renda, “Dynamics of continuum and soft robots: A strain parameterization based approach,” *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 847–863, 2021.
- [27] A. T. Mathew, I. M. B. Hmida, C. Armanini, F. Boyer, and F. Renda, “Sorosim: A matlab toolbox for hybrid rigid-soft robots based on the geometric variable-strain approach,” *Robotics & Automation Magazine, IEEE*, pp. 2–18, 2022.
- [28] F. Renda, C. Armanini, V. Lebastard, F. Candelier, and F. Boyer, “A geometric variable-strain approach for static modeling of soft manipulators with tendon and fluidic actuation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4006–4013, 2020.
- [29] T. R. Kane and D. A. Levinson, “The use of kane’s dynamical equations in robotics,” *The International Journal of Robotics Research*, vol. 2, no. 3, pp. 3–21, sep 1983.

- [30] J. Y. S. Luh, M. W. Walker, and P. R. C. P. I., "On-line computational scheme for mechanical manipulator," *Trans. ASME, J. Dyn. Syst.*, vol. 102, pp. 69–76, 1980.
- [31] A. Cardona, M. Geradin, and D. Doan, "Rigid and flexible joint modelling in multibody dynamics using finite elements," *Computer Methods in Applied Mechanics and Engineering*, vol. 89, no. 1-3, pp. 395 – 418, 1991.
- [32] A. Cardona and M. Geradin, "Time-integration of the equations of motion in mechanism analysis," *Computers and Structures*, vol. 33, no. 3, pp. 801–820, 1989.
- [33] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer New York, 2008.
- [34] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A lie group formulation of robot dynamics," vol. 14, no. 6, pp. 609–618.
- [35] J. Kim, "Lie group formulation of articulated rigid body dynamics," School of Computer Science, Carnegie Mellon University, Tech. Rep. [Online]. Available: <https://www.cs.cmu.edu/~junggon/tools/liegroupdynamics.pdf>
- [36] D. Pogorelov, "Differential-algebraic equations in multibody system modeling," *Numerical Algorithms*, vol. 19, pp. 183–194, 1998.
- [37] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1 – 16, 1972.
- [38] C. Armanini, I. Hussain, M. Z. Iqbal, D. Gan, D. Prattichizzo, and F. Renda, "Discrete cosserat approach for closed-chain soft robots: Application to the fin-ray finger," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2083–2098, 2021.
- [39] A. Cardona and M. Geradin, "Numerical integration of second order differential-algebraic systems in flexible mechanism dynamics," in *Computer Aided Analysis of Rigid and Flexible Mechanical Systems*, M. S. P. E. J.A.C. Ambrosio, Ed., vol. 268. Kluwer Academic Publishers, Dordrecht, 1994, pp. 501–529.
- [40] S. Briot and F. Boyer, "A geometrically exact assumed strain modes approach for the geometrico-and kinemato-static modelings of continuum parallel robots," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1527–1543, 2022.
- [41] M. Tummers, V. Lebastard, F. Boyer, J. Troccaz, B. Rosa, and M. T. Chikhaoui, "Cosserat rod modeling of continuum robots from newtonian and lagrangian perspectives," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2360–2378, 2023.
- [42] L. N. Trefethen, *Spectral methods in MATLAB*. Siam, 2000, vol. 10.
- [43] F. Boyer, M. Porez, and J. Mauny, "Reduced dynamics of the non-holonomic whipple bicycle," *Journal of Nonlinear Science*, vol. 28, pp. 943–983, 2018.
- [44] T. J. R. Hughes, *Finite Element Method-Linear Static and Dynamic Finite Element Analysis*, prentice-hall ed. Englewood Cliffs, 1987.
- [45] W. Khalil and E. Dombre, *Modeling Identification and Control of Robots*. Hermes, Penton-Sciences, London, 2002.
- [46] A. L. Orekhov and N. Simaan, "Solving cosserat rod models via collocation and the magnus expansion," in *IEEE/RSJ International Conference on Robots and Systems (IROS)*. IEEE, 2020.
- [47] K. Subbaraj and M. Dokainish, "A survey of direct time-integration methods in computational structural dynamics-ii. implicit methods," *Computers and Structures*, vol. 32, no. 6, pp. 1387–1401, 1989.
- [48] F. Boyer, M. Porez, A. Leroyer, and M. Visonneau, "Fast dynamics of an eel-like robot-comparisons with navier-stokes simulations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1274–1288, Dec. 2008.
- [49] M. Arnold and O. Bruls, "Convergence of the generalized- $\alpha$  scheme for constrained mechanical systems," *Multibody System Dynamics*, vol. 18, no. 2, pp. 185 – 202, 2007.
- [50] C. Bottasso, D. Dopico, and L. Trainelli, "On the optimal scaling of index three daes in multibody dynamics," *Multibody Syst Dyn*, vol. 19, pp. 3–20, 2008.



include structural dynamics, geometric mechanics, and biorobotics.



**Andrea Gotelli** was born in Italy in 1997. Received the Bachelor Degree in Mechanical Engineering from the University of Genoa, Genoa, Italy in 2019, and the Master Degree in Robotics Engineering with a double degree from École Centrale Nantes, Nante, France and University of Genoa, Genoa, Italy in 2021. Since 2021 he has been in a PhD with the École Centrale Nantes, Nante, France and IMT-Atlantique, Nantes, France.



interests include design and control of continuum parallel robots and co-manipulation of deformable objects.



**Vincent Lebastard** (Member, IEEE) received a Ph.D degree from the University of Nantes, France, in 2007. He is currently a Assitant Professor with the Department of Automatic Control, IMT-Atlantique, Nantes, France, where he is a member of the Robotics Team, Laboratoire des Sciences du Numérique de Nantes (LS2N). His current research interests include soft robotics and electric sense.



and underwater robotics

**Dr. Federico Renda** (Member, IEEE) received the B.Sc. and M.Sc. degrees in biomedical engineering from the University of Pisa, Pisa, Italy, in 2007 and 2009, respectively, and the Ph.D. degree in Biorobotics from the Biorobotics Institute, Scuola Superiore Sant'Anna, Pisa, in 2014. He is currently an Associate Professor with the Department of Mechanical and Nuclear Engineering at Khalifa University, Abu Dhabi, UAE. His research interests include dynamic modeling and control of soft and continuum robots for medical applications.



and the analysis of their performance, especially their singularities.

**Sébastien Briot** received the B.S. and M.S. degrees in mechanical engineering in 2004 and the Ph.D. degree in robotics in 2007, from the Institut National des Sciences Appliquées de Rennes, Rennes, France. In 2009, he has been recruited at CNRS as a researcher in the Laboratoire des Sciences du Numérique de Nantes, Nantes, France, where he has been the Head of the ARMEN Research Team since 2017. Since 2022, he is CNRS Director of Research in the same lab. His research interests include the design optimization of robots