SoRoSim: a MATLAB® Toolbox for Hybrid Rigid-Soft Robots Based on the Geometric Variable-Strain Approach

Anup Teejo Mathew[1†*], Ikhlas Ben Hmida[1†], Costanza Armanini[1], Frederic Boyer[2], Federico Renda[1,3]

***Abstract*—Soft robotics has been a trending topic within the robotics community for almost two decades. However, available tools for the modeling and analysis of soft robots are still limited. This paper introduces a user-friendly MATLAB® toolbox, SoRoSim, that integrates the Geometric Variable Strain (GVS) model of Cosserat rods to facilitate the static and dynamic analysis of soft, rigid, or hybrid robotic systems. We present a brief overview of the design and structure of the toolbox and validate it by comparing its results with those published in the literature. To highlight the toolbox's potential to efficiently model, simulate, optimize, and control various robotic systems, we demonstrate four sample applications. The demonstrated applications explore different actuator and external loading conditions of single-, branched-, open-, and closed-chain robotic systems. We think that the soft-robotics research community will significantly benefit from the SoRoSim toolbox for a wide variety of applications.**

***Index Terms*—Soft Robot Modeling, Continuum Manipulators, Hybrid mechanisms, Geometric Variable Strain approach, MATLAB Toolbox.**

I. Introduction

One of the most trending topics in the robotics community is the development and design of soft robots that can tackle challenges otherwise hard or even impossible to solve using their traditional, rigid counterparts [1]. Soft robots are lightweight, cheap, and adaptable to different environments and scenarios, as demonstrated by the vast number of applications where they have been employed. On the other side, their compliance and infinite number of degrees of freedom (DoF) intrinsically increase the complexity of their modeling.

Different modeling approaches have been proposed previously, varying in their simplifying assumptions and applicability. Some of the most commonly used approaches in soft robotics include the Lumped Mass Model (LMM), FEM-based models, Discrete Elastic Rod (DER) model, and the Piecewise Constant Curvature (PCC) model. The LMM assumes soft links to be repeated segments of point masses connected by springs and dampers corresponding to their geometry, expected motion, and DoFs. FEM-based models provide a way of numerically approximating partial differential equations governing the motion and deformation of the soft body. The PCC model constructs kinematic relations based on the robot's geometry and behavior under loading by discretizing its links into a finite number of circular arcs characterized by constant curvature [2]. Rod models, such as the Euler-Bernoulli Beam, Timoshenko Beam, or the Cosserat Rod, model the material deformation of the robot or manipulator by assuming it to be a 1D continuum mechanics object. The development of precise theoretical models is crucial, but theory alone may not be enough to satisfy the demand for computational tools in the soft robotics field. Knowledge-sharing initiatives become essential to allowing the growth of this relatively new research field [3]. Generalized modeling platforms eliminate the need to write robot-specific scripts for the simulation, design optimization, or model-based control of specific manipulators by providing user-friendly, accurate, fast, and reliable algorithms.

Popular examples of such platforms include, SoMo [4], a Python-based toolbox to simulate continuum manipulators as approximated spring-mass systems and SOFA [5], a simulation tool that employs simplified Finite Element Methods. Toolboxes such as ChainQueen [6] and SimSOFT [7] also employ FEM-based modeling techniques to simulate soft robots. Titan [8], which is a GPU-accelerated C++ library, is another example of a simulator that models soft robots as a spring-mass system. Toolboxes based on DERs include, Elastica [9], which employs the Cosserat rod theory to model and control slender bodies with a finite number of lumped degrees of freedom and VIPER [10], which uses a Volume Invariant Position-based Elastic Rods model to simulate the behavior of muscular hydrostats (muscle-like). Finally, the MATLAB package, TMTDyn [11] uses discretized lumped systems and reduced-order models to control and analyze hybrid rigid-soft robots.

The majority of the available toolboxes for soft robotics modeling are based on FEM or lumped mass systems. These are theoretically simple but computationally heavy approaches, with a lot of nodes and DOFs, designed for general purpose simulation, instead of analysis and control. Most of the DER based simulators are oriented towards computer graphics rather than real mechanical systems. TMTDyn is a geometrically exact package based on the parametrization of positions and orientations rather than strains.

Within this scenario, we present SoRoSim, which is a MATLAB toolbox based on the GVS approach, that directly extends rigid robot modeling techniques to soft and hybrid systems, while maintaining a high level of accuracy. The toolbox takes advantage of the high fidelity of the Cosserat Rod approach and the simplifying assumptions of the GVS approach that minimize the number of DoFs required to represent the system and provide a geometrically exact framework leading to accurate, fast, and computationally less expensive results. We implemented a new computational approach based on a nested Gaussian quadrature scheme to solve the GVS formulation.

The SoRoSim toolbox allows the user to define and manip-

† Anup Teejo Mathew and Ikhlas Ben Hmida are co-first authors.
* Corresponding author anup.mathew@ku.ac.ae
[1] Department of Mechanical Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE.
[2] LS2N Laboratory, Institut Mines Telecom Atlantique, Nantes 44307, France
[3] Khalifa University Center for Autonomous Robotic Systems (KUCARS), Khalifa University, Abu Dhabi, UAE.

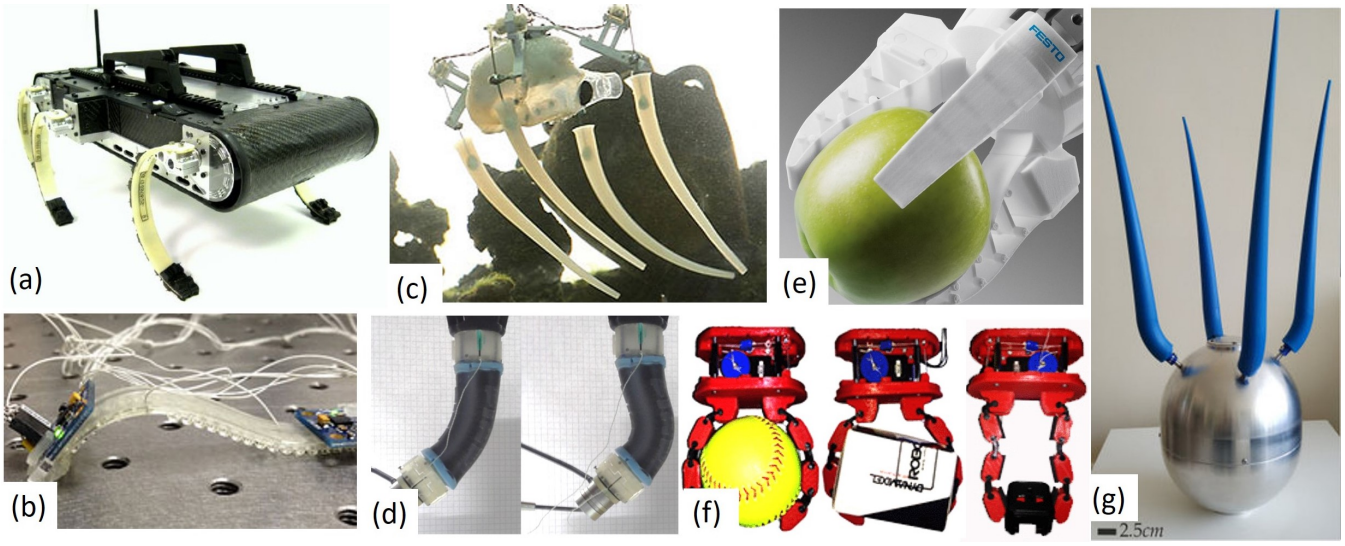

Fig. 1. A visual representation of the various robotic systems the toolbox can model. These systems comprise rigid-soft hybrid robots, where the soft links can be approximated as cosserat rods. (a) X-RHex, hybrid robot with compliant legs, (b) Tuft Softworm, a soft robot that relies on Shape Memory Alloy (SMA) actuators and frictional forces for locomotion, (c) PoseiDRONE, hybrid robot that uses tendon-based soft appendages to swim underwater and move on the seafloor [1], (d) STIFF-FLOP, A pneumatic soft manipulator designed for use in surgical procedures [11], (e) FinRay, a closed-chain soft gripper with rigid connectors [12], (f) A two-finger tendon-driven hybrid gripper consisting of modular segments [13], (g) Bio-inspired hybrid flagellate robot for underwater applications [14]. The SoRoSim toolbox can model all these classes of robots by including contact, friction, or fluidic interactions as custom external forces similar to the examples in section V

.

ulate links (rigid or soft) and robotic systems (linkages) using user-friendly GUIs and the MATLAB workspace. The GUIs assist the definition of links, their assembly, the definition of their degrees of freedom, assignment of constraints as closed-loop joints, and the application of external and actuation forces. A variety of typical external forces and actuation inputs are handled by GUIs, which provides a black-box experience, allowing users from all backgrounds to easily use the toolbox to perform static and dynamic analyses. The toolbox provides specific MATLAB files that the user can edit to input customized values of external and actuation forces as functions of joint coordinates, their derivatives, positions, velocities, etc. Moreover, as a MATLAB toolbox, it can be used alongside built-in functions, add-ons such as the Optimization Toolbox, and user-written codes to further facilitate the analysis and control of robotic systems. The intrinsic limitation of the toolbox is that it can only model soft links as cosserat rods. However, rigid links can be modelled with no limitation on geometry or DoFs. Figure 1 shows a small subset of robotic systems the toolbox can analyze. The SoRoSim toolbox package and user manual are available for free on [15]. Section III provides details on various aspects of SoRoSim toolbox including its design, structure, and workflow.

We performed several toolbox sanity tests by comparing its analysis results with published data and commercial software output. We compared the static equilibrium results with published solutions and ANSYS workbench results. We studied the dynamic simulation of a flexible flying rod and compared it with existing literature. We also investigated the energy transfer between the kinetic, gravitational potential, and elastic potential energy of a cantilever beam under gravity. We discuss these validation studies in Section IV. In addition to validation, in Sections V and VI, we demonstrate four innovative applications where the soft robotics community can use the toolbox. The examples include the analysis of static equilibrium and contact dynamics of hybrid robotic arms, an underwater locomotor, a design optimization problem, and a study of two cases of inverse dynamic control problems. Finally, in Section VII we discuss the computational performance of the toolbox and draw conclusions and future directions of SoRoSim.

II. Governing Equations

The Geometric Variable Strain (GVS) approach was recently introduced by Renda et al.[16] in statics and Boyer et al.[17] in dynamics. It is based on a variable strain parametrization of soft links represented by Cosserat rods, a 1D, slender rod that can bend, twist, stretch, and shear. Cosserat's model is the most general rod model as it accounts for the orientation as well as the position of beam elements and allows for all 6 modes of deformation to be considered during analysis [2]. Since the strain is parameterized in the GVS approach, it is easy to disable any deformation modes. Thus, other beam theories can be kinematically reproduced. For instance, SoRoSim users can enable the rotational modes along with shear along y and z to create a Timoshenko beam. The GVS model is also geometrically-exact and generalizes the geometric theory of rigid robotics to hybrid systems of soft and rigid links with multidimensional joints, externally applied point and distributed forces, as well as distributed actuation forces [18]. In this section we give a summary of the model and an overview of the efficient computational techniques implemented in SoRoSim toolbox.

Consider a floating hybrid kinematic chain, composed of interconnected rigid and soft bodies, represented by Cosserat rods. The configuration of a soft body i (respectively a rigid body) with respect to its predecessor in the chain is defined as

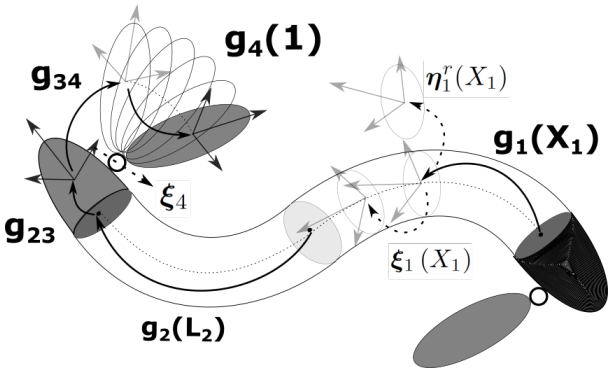

Fig. 2. Schematics of the proposed kinematics for a floating hybrid soft/rigid chain.

a curve:

$$\begin{aligned} &\boldsymbol{g}_i(\cdot) : X_i \in [0, L_i] \mapsto \\ &\boldsymbol{g}_i(X_i) = \begin{pmatrix} \mathbf{R}_i & \mathbf{r}_i \\ \mathbf{0} & 1 \end{pmatrix} \in SE(3). \end{aligned} \tag{1}$$

(respectively a point $\boldsymbol{g}_i \in SE(3)$), mapping the body-frame at X_i to the body-frame of the previous body at the reference configuration, as shown in Figure 2.

To study the exponential representation of $\boldsymbol{g}_i(X_i)$, we introduce its partial derivative with respect to space; $\boldsymbol{g}_i'(X_i) = \boldsymbol{g}_i\widehat{\boldsymbol{\xi}}_i$, and with respect to time; $\dot{\boldsymbol{g}}_i(X_i) = \boldsymbol{g}_i\widehat{\boldsymbol{\eta}}_i^r$. Where, $\boldsymbol{\xi}_i(X_i) \in \mathbb{R}^6$ defines the strain twist in body-frame, $\boldsymbol{\eta}_i^r(X_i) \in \mathbb{R}^6$ is the velocity twist relative to the predecessor in body-frame, and $\widehat{(.)}$ is the isomorphism from $\mathbb{R}^6$ to $se(3)$. The space derivative of $\boldsymbol{g}_i(X_i)$ is a matrix differential equation that can be integrated in space according to: $\boldsymbol{g}_i(X_i) = \exp\left(\widehat{\boldsymbol{\Omega}}_i(X)\right)$, where $\widehat{\boldsymbol{\Omega}}$ is the Magnus expansion of the field $\widehat{\boldsymbol{\xi}}$. For the rigid link case, $\widehat{\boldsymbol{\xi}}_i(X_i)$ is constant in X_i and equal to the body-frame representation of the joint twist attached to i, while for the soft link case, $\widehat{\boldsymbol{\xi}}_i(X_i)$ is variable. The equality of the mixed partial derivative of $\boldsymbol{g}_i$ provides the relation between the time derivative of the strain twists and the link's relative velocity and acceleration twists [19]:

$$\begin{aligned} \boldsymbol{\eta}_i^{r\prime}(X_i) &= \dot{\boldsymbol{\xi}}_i - \mathrm{ad}_{\boldsymbol{\xi}_i}\boldsymbol{\eta}_i^r \ , \\ \dot{\boldsymbol{\eta}}_i^{r\prime}(X_i) &= \ddot{\boldsymbol{\xi}}_i - \mathrm{ad}_{\dot{\boldsymbol{\xi}}_i}\boldsymbol{\eta}_i^r - \mathrm{ad}_{\boldsymbol{\xi}_i}\dot{\boldsymbol{\eta}}_i^r \ , \end{aligned} \tag{2}$$

where $\mathrm{ad}_{(\cdot)} \in \mathbb{R}^{6\times6}$ is the adjoint operator of $\mathfrak{se}(3)$ [20].

It is time now to discretize the system and introduce the generalized coordinates. The continuous strain fields $\boldsymbol{\xi}_i(X_i)$ are parametrized by a finite functional bases of strain modes [16].

$$\boldsymbol{\xi}_i(X_i) = \boldsymbol{\Phi}_{\boldsymbol{\xi}_i}(X_i)\boldsymbol{q}_i + \boldsymbol{\xi}_i^*(X_i) \ , \tag{3}$$

where $\boldsymbol{\Phi}_{\boldsymbol{\xi}_i}(X_i) \in \mathbb{R}^{6\times n_i}$ (n_i being the number of degrees of freedom (DoF) of link i) is a matrix function whose columns form the basis for the strain field, $\boldsymbol{q}_i \in \mathbb{R}^{n_i}$ is the vector of coordinates in that basis, and $\boldsymbol{\xi}_i^*(X_i) \in \mathbb{R}^6$ is a reference strain whose primary function is to model non-zero yet constrained strains such as inextensibility. Note that the matrix $\boldsymbol{\Phi}_{\boldsymbol{\xi}_i}(X_i)$ is constant for rigid joints.

The integration of equation (2) using (3) for all the bodies leads to the definition of the geometric Jacobian ($\boldsymbol{J}_i(\boldsymbol{q}, X) \in \mathbb{R}^{6\times n}$) and its derivative $\dot{\boldsymbol{J}}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, X) \in \mathbb{R}^{6\times n}$ ($n = \sum n_i$). Once $\boldsymbol{J}_i$ and $\dot{\boldsymbol{J}}_i$ is found, we project the free dynamics of the floating hybrid chain onto the space of generalized coordinates to yield the generalized dynamics of the system:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + (\boldsymbol{C} + \boldsymbol{D})\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} = \boldsymbol{B}\boldsymbol{u} + \boldsymbol{F} \ , \tag{4}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the mass matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is the damping matrix, $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is the stiffness matrix, $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n\times n_a}$ (n_a being the total number of actuators) is the actuation matrix, $\boldsymbol{F}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ is the vector of generalized external forces and $\boldsymbol{u} \in \mathbb{R}^{n_a}$ is the vector of applied actuation forces.

We developed a recursive two-level nested quadrature scheme to estimate the coefficients of equation 4. For soft links, the SoRoSim toolbox uses the Gauss-Quadrature numerical integration method (the order is chosen by the user) to evaluate these coefficients. Stiffness ($\boldsymbol{K}$) and damping ($\boldsymbol{D}$) coefficients, which are associated with linear elastic models, are pre-computed offline. However, the framework also allows the computation of non-linear, strain-dependent constitutive laws. Apart from ($\boldsymbol{K}$) and ($\boldsymbol{D}$), all the other coefficients ($\boldsymbol{M}$, $\boldsymbol{C}$, $\boldsymbol{B}$, and $\boldsymbol{F}$) of equation 4 are computed as functions of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$.

Estimating these coefficients involves assessing $\boldsymbol{g}_i$, $\boldsymbol{J}_i$, and $\dot{\boldsymbol{J}}_i$ at every evaluation point, such as Gaussian points of the soft divisions and the center of mass of rigid links. In our previous paper, we used a 4^{th} order Zannah collocation approximation to estimate the value of $\boldsymbol{g}_i$ recursively: $\boldsymbol{g}_i(X_i+h_k) = \boldsymbol{g}_i(X_i)\exp\left(\widehat{\boldsymbol{\Omega}}_i^k(h_k)\right)$, where h_k is the material length in the k^{th} interval between two consecutive Gauss-Quadrature points and $\widehat{\boldsymbol{\Omega}}_i^k(h_k)$ is the approximation of the Magnus expansion [16]. Inspired by this, we derived recursive formulations for the computation of $\boldsymbol{J}_i$ and $\dot{\boldsymbol{J}}_i$. The complete theory and description of the computational strategy behind the toolbox are available at the arXiv link provided in [15].

Equation (4) is an ODE that could be solved using explicit time-integrators such as the 'ode45' or 'ode15s' in MATLAB. The static equilibrium equation of the system can be derived from equation (4) by equating the time derivatives of $\boldsymbol{q}$ ($\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$) to zero. The resulting static equation could be solved numerically using root-finder functions such as the 'fsolve' in MATLAB. For the case of closed-chain robots, additional terms corresponding to the constraint forces due to the closed-loop joints will be present in equation (4) [12].

III. Toolbox Design and Structure

We developed the Sorosim toolbox in MATLAB, which provides users with a vast library of functions for mathematical computations, and gives access to various built-in functions, add-ons and toolboxes to analyze different aspects of robotic systems. We also employ an Object-Oriented Programming (OOP) approach, which entails program design around data, or objects, rather than functions and logic. OOP allows the developer to group 'Objects' with similar attributes under a 'Class',providing a a well-structured map of the program and allowing easy access and adjustment to object-specific data, 'Properties', with the help of class specific 'Methods'.

The SoRoSim toolbox consists of three MATLAB class elements: SorosimLink, Twist, and SorosimLinkage. These

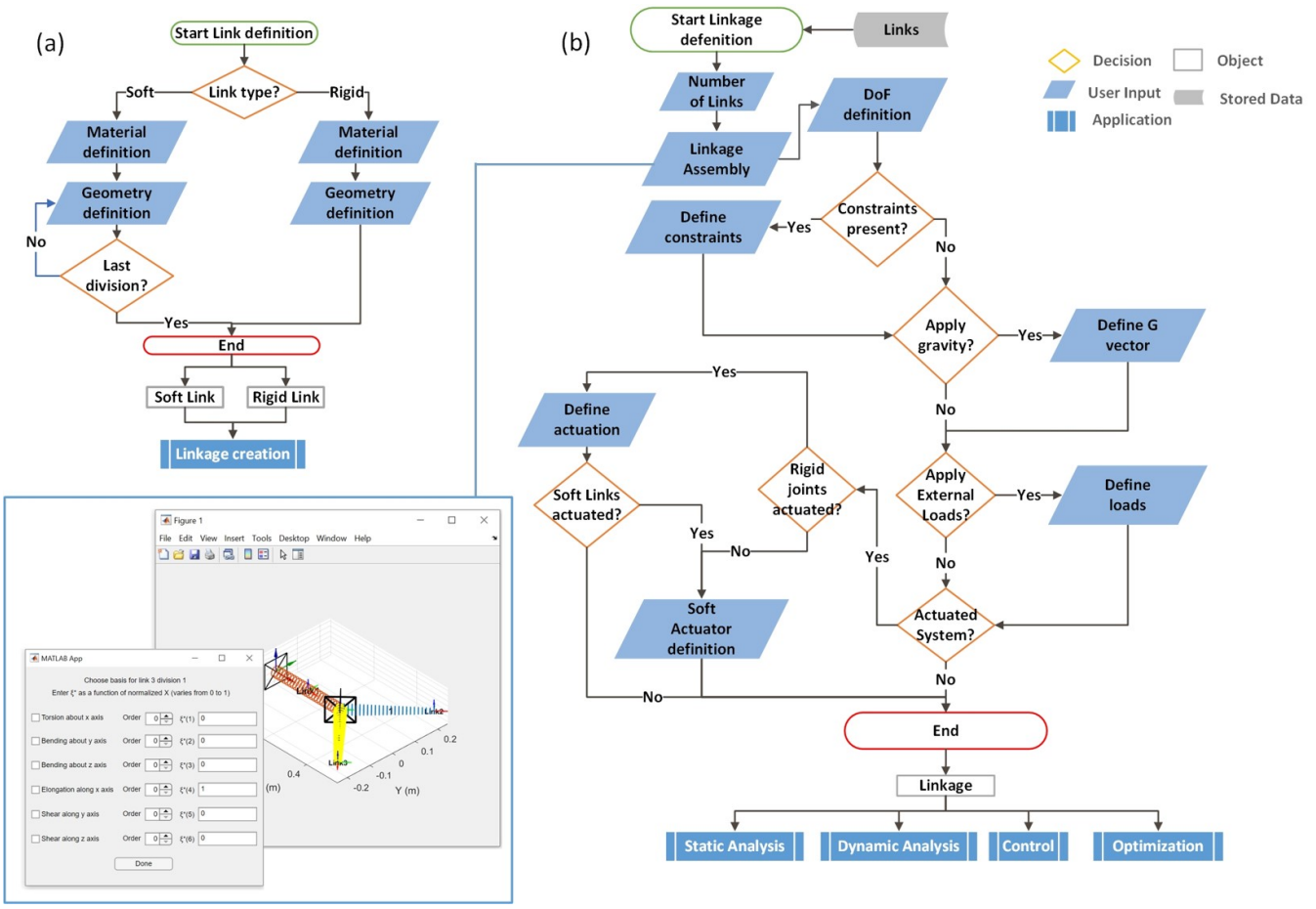

Fig. 3. Toolbox overview: (a) Flowchart of SorosimLink creation (b) Flowchart of SorosimLinkage creation and applications. A preview of the GUI is shown as an inset.

classes work together to facilitate linkage creation and simulation through a sequence of user-friendly GUIs.

The SorosimLink class allows the user to construct soft or rigid links with various joint types and geometry. The user can choose from nine different types of lumped joints (fixed, revolute, prismatic, helical, cylindrical, universal, planar, spherical, and free joint) and three default cross-sectional shapes (circular, rectangular, and ellipsoidal). However, analysis performed within the toolbox is not limited to the link's default cross-sectional shapes. Once a link is defined, the user can update its properties such as screw inertia matrix and stiffness matrix to account for any arbitrary cross-sectional shape or non-homogeneous mass distribution. The shape may also vary as a function of curvilinear abscissa along the link axis, X. The default material model used to compute the cross-sectional screw stiffness matrix is a linear elastic model. This material model provides an accurate representation of the material behaviour when it is subjected to strains that do not exceed 100%, as this is the case for most soft robotics applications this is an appropriate material model to use. However the toolbox allows users to use a custom material model by modifying the elasticity tensor matrix in the SorosimLink Class. An overview of the SorosimLink creation is shown in Figure 3a.

The Twist class specifies the active DoF(s) of lumped joints, and the deformation modes and corresponding strain orders of soft link divisions. For a soft link division, the class allows the user to enable the required modes among six deformation modes: torsion about the x-axis, bending about the y-axis, bending about the z-axis, elongation along the x-axis, shear along the y-axis, and shear along the z axis. The order of a particular mode corresponds to the polynomial that is used to estimate their strain values. The Twist class also allows the user to define a reference strain value for the soft division corresponding to its rest configuration. The inset in Figure 3b shows a sample GUI where the user creates the Twist class for a soft link division. The GVS model allows the definition of the strain base of the soft links as a continuous or discontinuous function of X. The user may change the default polynomial basis of soft links once the SorosimLinkage is defined.

The SorosimLinkage class allows the user to assemble previously defined links into various single-, branched-, open-, and closed-chain systems. SorosimLinkage calls the Twist class to determine each link's DoF(s). The user can also add closed-loop joints by selecting appropriate joint types. The class allows the definition of various external forces and actuation inputs. The class automatically pre-computes and saves constant properties of the linkage such as the generalized stiffness matrix ($\boldsymbol{K}$) and the generalized damping matrix ($\boldsymbol{D}$). It also allows the users to program custom external and actuation forces. An overview of the SorosimLinkage creation is shown in Figure

3b.

We pack the SorosimLinkage class with 'Methods' that facilitate the analysis of multi-body systems and the post-processing of results (static equilibrium configuration and dynamic video output). The methods for analysis include functions to solve the GVS model for static and dynamic analyses. The SorosimLinkage methods can also compute values of system parameters such as the Jacobian ($\boldsymbol{J}$), generalized mass ($\boldsymbol{M}$) and Coriolis ($\boldsymbol{C}$) matrices for a given value of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. Users can utilize these methods for problem-specific analyses. The reader may refer to the toolbox manual [15] for a detailed descriptions of all the properties and methods of SoRoSim classes.

IV. Toolbox Validation

To validate the toolbox, we conduct several numerical tests and comparisons with verified and published data. We present these tests in this section.

A. Test 1: Fixed-free Beam with a Follower Tip Force

We first simulate the bending behaviour of a cantilever beam with a follower force applied at the tip. Many authors have considered this problem and solved using numerical approaches such as the finite strain rod method [21] and other geometrically exact models [17].

Using the toolbox, we constructed a 100 m long cylindrical beam with a Young's Modulus, $E = 6.75$ GPa and a diameter of 57 cm. We set a 4^{th} order bending strain about the y-axis to model the deflection of the rod when subjected to the follower tip force. The force was varied between 0 N to 130 kN. We used 15 Gauss quadrature points, specified during the SorosimLink creation process, for the computation of integrals (in this case, $\boldsymbol{K}$). Figure 4(a)(i) shows the deflection of the link as well as the horizontal and vertical displacement of the tip (ii) as modeled by the toolbox. The results obtained match with those obtained in [21] as shown in Figure. 4(b).

The total DoFs of the rod modeled using SoRoSim was 5 DoFs, while Simo et al. [21] used a 1D finite element mesh consisting of five elements with quadratic shape functions corresponding to 10 DoFs. This demonstrates the GVS approach's ability to recreate accurate results by using a fewer number of DoFs. Additionally, the reported computation time is 16.4 s per loading step in [17], while it took 42 ms for the same in the SoRoSim toolbox.

We also use this example to highlight the scaling process the toolbox uses. The toolbox performs internal computations on a soft division after normalizing its length into 1 unit (here 100 m to 1 unit). Consequently, physical quantities with length dimensions are scaled using the original length of the division. Once the simulation is complete, the toolbox scales back the resulting values of joint coordinates into their actual dimensions. The normalization of soft divisions avoids poorly scaled matrices such as the generalized stiffness matrix. This allows faster static solutions and more stable dynamic simulations.

B. Test 2: FEM study of a Fixed-fixed L-shaped Beam

To test the performance of the toolbox with respect to commonly used methods of modeling, we compare the results

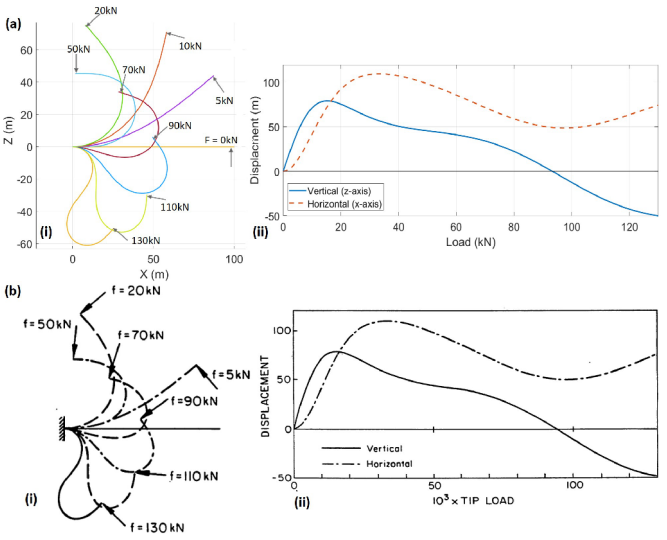

Fig. 4. (a) shows the toolbox simulation output while (b) shows the results obtained in [21]. (i) Is the clamped beam profile under varying follower tip loads and (ii) shows the horizontal and vertical tip displacement at different loads.

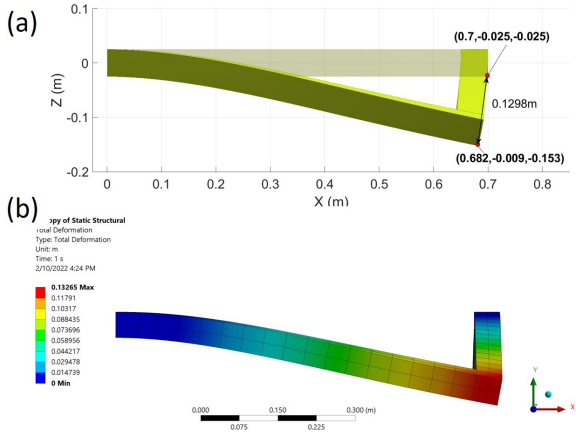

Fig. 5. (a) SoRoSim results showing the reference and deformed shape of the linkage. (b) Total deformation and deformed shape obtained from ANSYS.

obtained when a soft linkage is subjected to a distributed load (gravity) with those obtained through FEM simulation. Two 0.7 m long soft links with a 5cmx5cm square cross-section (aspect ratio 14:1) are connected to form an L-shaped linkage clamped at each end. We used a fixed closed-loop joint to fix the rear end of the second link with the ground. All six deformation modes are enabled for this simulation, quadratic and linear polynomials were used to estimate the rotational and translational modes respectively. Hence, there are 15 DoFs for a link (30 in total). For the material, $E = 10$ MPa, Poisson's ratio, $\nu = 0.5$, and density, $\rho = 1200$ kg/m^3 were used.

We compared the static equilibrium results obtained from the toolbox with those of ANSYS Workbench. A linkage with the same geometry and material properties is created in ANSYS. A total of 152 quadratic 3D elements with 1062 nodes (~ 3000 DoFs) were used for the simulation. Figure 5(a) shows the toolbox result, while 5(b) shows that of ANSYS. The maximum deformation obtained from the toolbox result was 12.98 cm, whereas the FEM simulation gives a deformation of 13.27

cm. Hence, with fewer DoFs ($1:100$), the GVS method can estimate the deformed shape of the L-shaped beam. The simulation results also suggest that the toolbox can effectively handle closed-loop problems.

C. Test 3: Dynamics of a Flexible Flying Rod

In this comparison, we look at the dynamics of a freely flying flexible rod (also known as the flying spaghetti problem) which is a problem introduced by Simo and Vu-Quoc [22] and replicated in [17]. We modeled a 10 m long soft rod, with a free lumped joint and initially at rest in the position shown in Figure 6(a)(i). The position and orientation of the base of the soft rod are parameterized by lumped DoFs of the free joint. We used the in-extensible Kirchhoff model with a quadratic polynomial basis to define the basis of the rod. Hence, including the six DoFs of the free joint, there are 15 DoFs in this system. Time dependent point force $F1$ and moments $M2$ and $M3$, are applied at the tip of the rod as shown in figure 6(a)(i). The magnitude of $M2$ is defined as a triangular pulse function that starts at time, $t = 0s$, peaks at 200 Nm in 2.5s, and goes back to 0 at 5s. The numerical values of the magnitude of $F1$ and $M3$ are 1/10$^{\text{th}}$, and half of $M2$ respectively. The user can define such dynamic inputs as a function of time (t) in the GUI.

We performed the dynamic analysis of the system for the first 7 s. Figures 6(a)(ii) and (a)(iii) show 2 views of superimposed snapshots of the rod mid-flight at different times as solved by the SoRoSim Toolbox. The rod's position, orientation, and deformation match exactly with the published results in [17], shown in Figures 6(b)(ii) and (b)(iii). This example also demonstrates the capabilities of the toolbox in modelling lumped and distributed joints (soft body) within the same framework.

We use this example to highlight the computational efficiency of the toolbox. The system uses 15 DoFs to simulate a complex dynamic motion in 3D. Boyer et al. report a computational time of 4 hours and 30 minutes for a 30-second simulation of the same problem [17]. Hence, on average, the reported computational time for a second of the simulation was 9 minutes. Using SoRoSim, We were able to solve a 7 s simulation, in less than 1 s of computational time, which is three orders ($> 3000X$) faster than previously reported computational time. Faster than real-time computation ($7X$ faster in this case) will allow the use of SoRoSim for real-world applications. To better appreciate the dynamic simulations presented in this paper, the reader may refer to the supplementary video.

D. Test 4: Energy Balance in SoRoSim Dynamics

Here we investigate the energy balance in damped and undamped cantilever beams released from rest with an initial strain under gravity. We create a soft link with a length (L) of 0.5 m and a radius that linearly decreases from 2 cm to 1 cm. The material properties of the link are: $E = 1$ MPa, $\nu = 0.5$, and $\rho = 1000$ kg/m^3. For the damped rod, we used an elastic damping of 11.2 KPas. We enabled all three angular modes of deformation (torsion and rotations about y and z-axis) with a cubic polynomial approximation for the strains. To obtain a complex dynamic motion, the soft link is released from rest with an initial bending of 1 rad/m about the y-axis. We run the

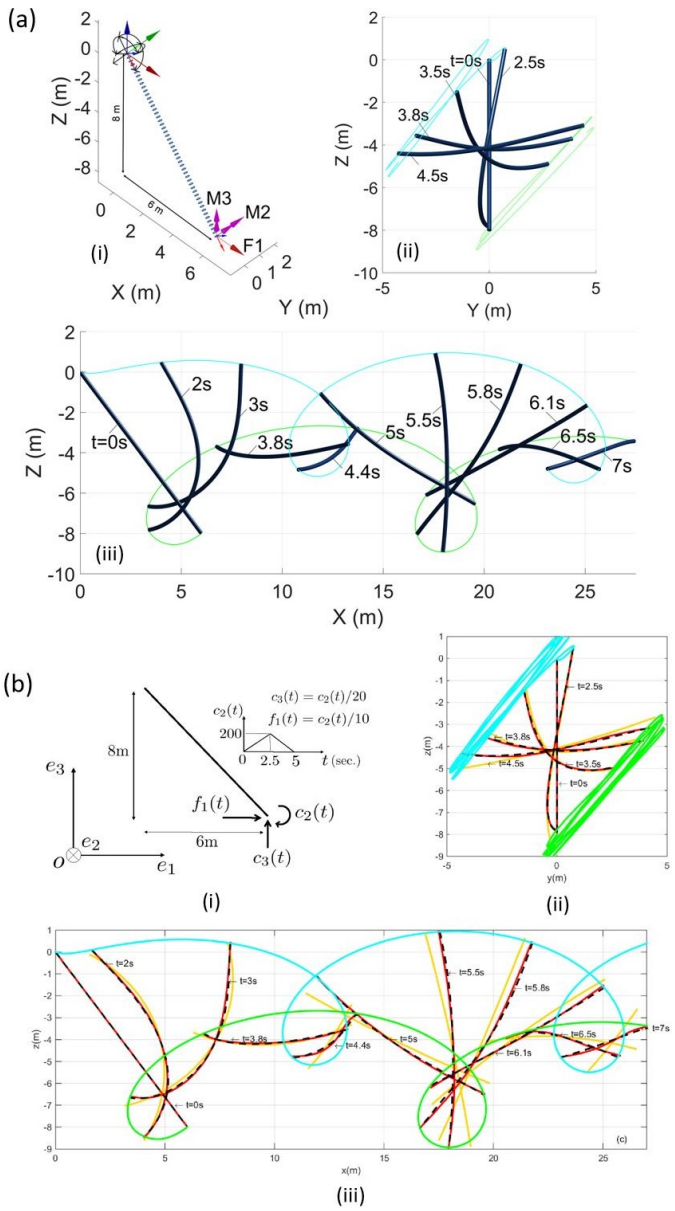

Fig. 6. Comparison of the simulation results obtained using SoRoSim (a) with those obtained in [17] (b), (i) shows the reference configuration of the flexible rod with external force and moments. Snapshots of mid-flight dynamics of the rod in yz (ii) and xz (iii) plane. The comparison is with the dashed black line which models an inextensible rod with three modes/strain.

dynamic simulation for 5 seconds. Figure. 7 shows the motion of the damped (b) and the undamped rod (c) between time, $t = 0$ s to 0.75 s.

Total energy of the beam at a given time is given by the sum of kinetic, gravitational potential and elastic potential energies:

$$\begin{aligned} E_{tot} = \frac{1}{2}\int_0^L \boldsymbol{\eta}^T \bar{\mathcal{M}} \boldsymbol{\eta}\, dX + \int_0^L \rho.A.g.h\, dX + \\ \frac{1}{2}\int_0^L (\boldsymbol{\xi} - \boldsymbol{\xi}^*)^T \boldsymbol{\Sigma} (\boldsymbol{\xi} - \boldsymbol{\xi}^*)\, dX \\ = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{M} \dot{\boldsymbol{q}} + U_g(\boldsymbol{q}) + \frac{1}{2}\boldsymbol{q}^T \boldsymbol{K} \boldsymbol{q}, \end{aligned} \tag{5}$$

where, $\boldsymbol{\eta}$ is the screw velocity, $\bar{\mathcal{M}}$ is the cross-sectional

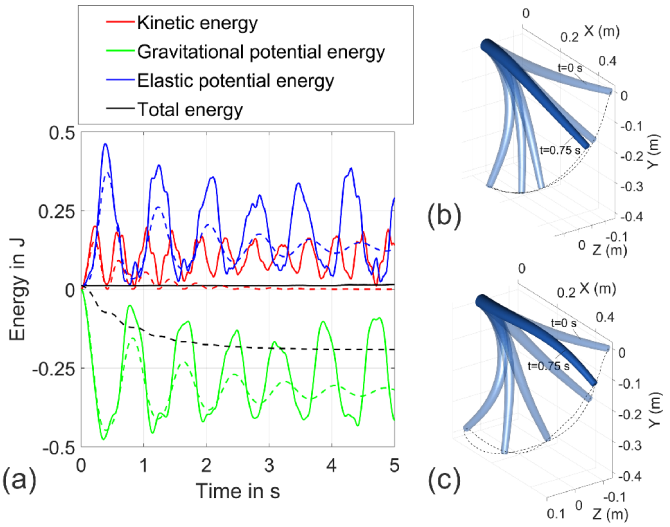

Fig. 7. Free fall dynamics: (a) Energy change for the undamped (solid) and damped (dashed) beam. Superimposed images of the damped case (b) and the undamped case (c). Dashed lines denote the tip trajectory.

screw inertial matrix, A is the area of cross section, g is acceleration due to gravity, h is the y coordinate of a cross section of the rod (in the opposite direction of gravity), $\boldsymbol{\xi}$ is the screw strain vector, $\boldsymbol{\xi}^*$ is the reference screw strain, and $\boldsymbol{\Sigma}$ is the cross-sectional screw stiffness matrix. $\boldsymbol{M}$ and $\boldsymbol{K}$ are defined as in equation (4).

The first, second, and third terms on the right-hand side of the equation represent kinetic, gravitational potential, and elastic potential energies respectively. We plot these for the damped and the undamped cases in Figure 7a. The total energy of the damped system is decaying, and that of the undamped system remains a non-zero positive constant corresponding to the initial strain energy. The results highlight the robustness of the SoRoSim dynamics as any deviation in the energy conservation is solely attributed to errors in numerical integration (in time and space).

For the undamped case, the system's total energy should remain the same as the initial elastic potential energy. While for the damped case, the total energy should decrease monotonically. Figure. 7(a) shows the change in the values of various forms of energies of the system. The total energy of the damped system is decaying, and that of the undamped system remains a non-zero positive constant as expected. This example also emphasizes that apart from dynamic and static simulations, we can use the toolbox for post-processing. The Methods of SorosimLinkage class allow the user to compute quantities such as configuration (position and orientations of cross sections), velocities, and generalized mass matrices as functions of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. The user can compute these values in a separate MATLAB code for post-processing, such as the energy computations in this example.

V. Modelling Applications

This section presents examples of the static equilibrium and dynamic simulations relevant to the soft robotics field, which the toolbox can solve. We demonstrate the applicability of the toolbox to different systems and the use of custom functions to model various external forces, including contact and fluid interactions.

A. Hybrid Manipulators

Serial robotic arms are widely used to automate manufacturing processes or any task requiring object gripping and manipulation. We show the toolbox's ability to model and analyze hybrid manipulators for two different cases.

1) Case 1: Static equilibrium

This example demonstrates the static equilibrium analysis of a hybrid (rigid-continuum) robotic arm with soft grippers under a wide range of loading conditions, including gravity (distributed load), a point force (F), rigid joint actuation as well as cable actuation of soft links for gripping. We use the toolbox to investigate the static equilibrium analysis of the manipulator and demonstrate its ability in modeling systems with multiple types of links, joints, and forces within the same framework.

The system consists of 11 links that form open, branched, and closed-chain components, as shown in Figure 8(a). The first link is a rigid link that is connected to the ground via a planar joint, which allows displacement in the xy plane and rotation about the z-axis. The second and third links are rigid links connected using revolute joints, which rotate about the y- and z-axis respectively. These three rigid links are controlled by their joint coordinate values. The toolbox allows the user to enter values of their joint coordinates as inputs to the simulation as shown in Figures 8b(ii) to (iv). Following the third link, there is a passive buffer consisting of three parallel soft links connected to two rigid disks at each end as shown in Figure 8(c). These soft links are attached to the base disk via spherical joints and connected to the end disk via closed-loop fixed joints. The buffer, which adds to the system's complexity, is an example of a multi-DoF joint, parallel link, and closed-loop component, which the SoRoSim toolbox can model. We assigned two divisions for these soft links, enabled all rotational DoFs, and assigned constant strain bases. Using appropriate reference strain values, we set the pre-curvatures of the soft buffer links. An angle-controlled rigid disk attached at the end of the buffer assists further orientation of the grippers by rotating about its local x-axis as shown in figure 8b(v).

Finally, we have a soft gripper with two fingers that are cable-actuated. The thin sections of the finger act as flexible links, while the thick sections act as rigid links. We assign a strain basis with constant bending about the local y-axis to the thin sections. This design of the gripper, where the cable is positioned outside the thin, soft section, is widely used in the soft robotic community [13]. This particular mode of actuation, where the actuators goes in and out of the system, is an important class of prototype that SoRoSim is able to simulate.

We computed the static equilibrium configuration of the system for the actuation inputs shown in Figure 8(b). The system is subject to gravity and a follower point force. The direction of the gravity and the point force (F) are shown in Figure 8(a). The value of the point force is 2 N. The corresponding equilibrium configurations are shown in figures 8b(ii) to (v). Figure 8b(vi) shows the actuation of the grippers

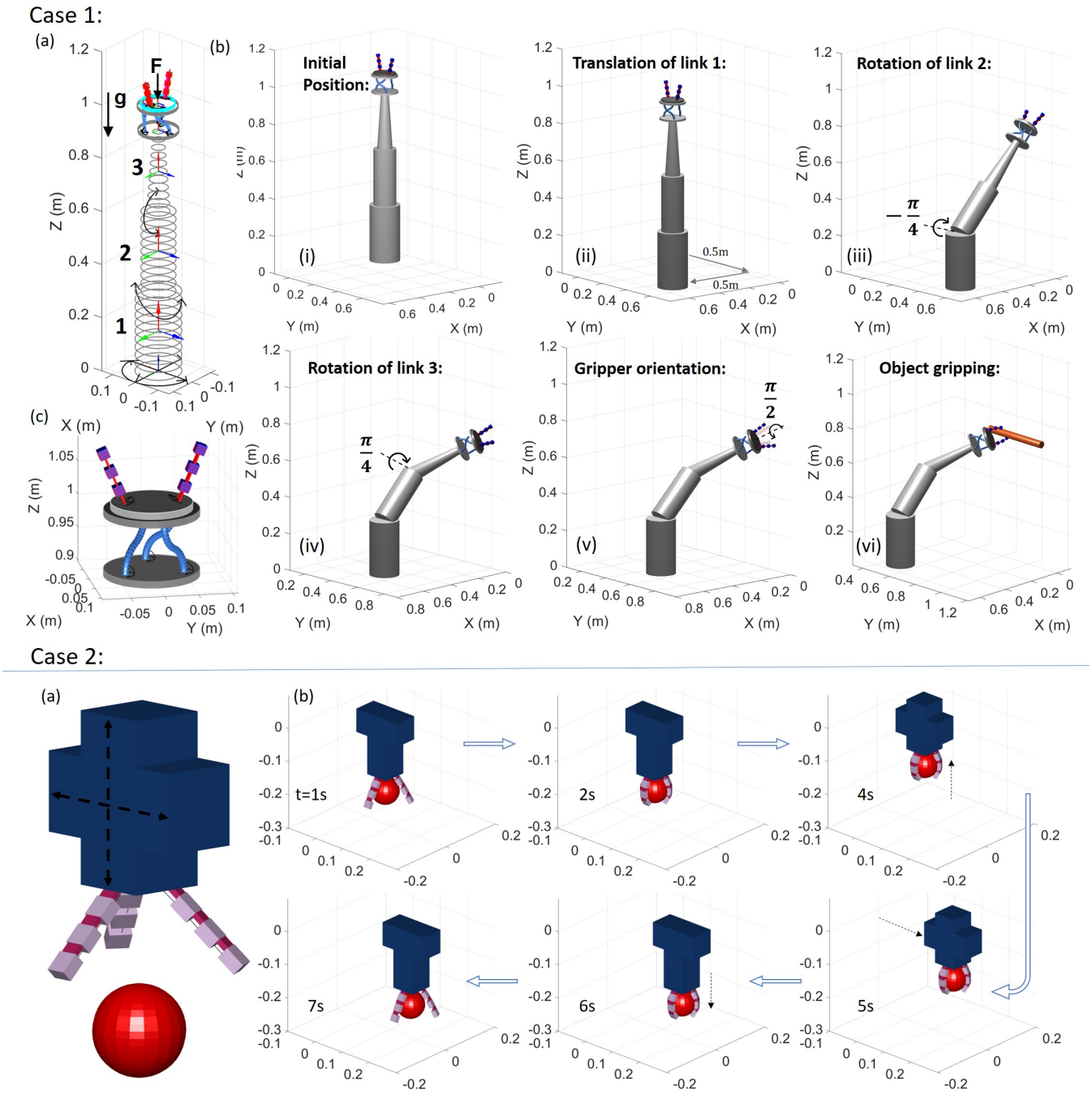

Fig. 8. Case 1: (a) System schematic of the hybrid robotic arm, showing some loads the system is subjected to.(b) Static equilibrium configurations of the arm at different stages of actuation input. (c) Close up view of the manipulator tip shows the buffer with three parallel, closed loop soft links and the cable actuated gripper fingers. Case 2: Contact modelling: (a) Schematic of the hybrid robot and the grip target. (b) Snapshots of object manipulation procedure.

when a cable tension of 2 N is applied. We show a rod as a reference grip target. There are 39 degrees of freedom and seven actuation inputs for the system. On average the static equilibrium simulations (shown in Figures 7(b)(i) to (vi)) took 1.2 s.

This example analyzed a system with passive and actuated rigid joints controlled by joint coordinate values. The toolbox can also define joints controlled by joint force or moment values. The user can also assign stiffness values for rigid joints. For instance, a prismatic joint with a positive stiffness value is equivalent to a linear spring.

2) Case 2: Contact dynamics

Analysis of mechanical systems involving contact-impact events is demanding for many real-world applications. Contact dynamics involves determining potential contact points between colliding bodies, evaluating contact-impact forces, and establishing the transition between different contact scenarios. Implementation of contact mechanics is one of the most challenging and complex problems in modeling and computation [23].

Contact between two points involves the application of equal and opposite normal and tangential forces on each other. The

current SoRoSim toolbox does not have the built-in capability for contact dynamics. However, the user can apply various kinds of external loads by enabling a property of the SorosimLinkage class, namely, 'CEFP,' (an acronym of 'Custom External Force Present) and then editing a MATLAB function 'CustomExtForce.m,' to model the external force. Quantities such as Jacobian $\boldsymbol{J}$, screw velocity $\boldsymbol{\eta}$, and transformation matrices $\boldsymbol{g}$ are passed as inputs into the 'CustomExtForce' function. The user can apply a custom external force as a function of these quantities.

To demonstrate contact dynamics using SoRoSim, we analyzed a three-fingered gripper system attempting to grip and manipulate a spherical rigid body (Figure 8(a)). The system consists of two rigid links with prismatic joints and three symmetrically arranged fingers similar to the previous case. We choose the dimensions of links arbitrarily. The inputs to the dynamic simulations are the kinematic inputs of the prismatic joints (vertical and horizontal translation) and the cable tension which actuates the fingers.

We made the following approximations to simplify the estimation of contact points and the evaluation of contact forces. (i) The three thick rigid sections of each finger are approximated as spheres with diameters ($2r_i$) equal to the section heights and centered at the center of mass of the links. (ii) We neglect the tangential (frictional) component of the contact forces. The normal force due to 9 different contact points (three on each finger) is sufficient to provide a force closure grasp. (iii) We used a linear spring with a linear damper to represent the normal contact force ($\boldsymbol{f}$). The contact force acting on the i^{th} link is given by:

$$\boldsymbol{f_i} = H(\delta_i) \cdot (k\delta_i + b\dot{\delta_i}) \frac{\boldsymbol{p}_{si}}{\|\boldsymbol{p}_{si}\|} \tag{6}$$

Where $\boldsymbol{p}_{si}$ is the vector from the center of sphere to the center of the i^{th} link, $\delta_i = r_s + r_i - \|\boldsymbol{p}_{si}\|$, r_s is the radius of the spherical target, k is the contact stiffness constant, b is a damping constant, and $H(\delta_i)$ is the Heaviside function, which ensures that the force is applied only if there is contact ($\delta_i > 0$). While $\boldsymbol{f_i}$ acts on the i^{th} link -$\sum \boldsymbol{f_i}$ acts on the grip target. Figure 8(b) shows the snapshots of the grip-manipulation maneuver.

While the gripper in Figure 1(f). is a planar equivalent of this example, the robots shown in (a), (b), and (e) can be modeled as rigid-soft hybrid robots with contact dynamics. This example demonstrates that we can use the SoRoSim toolbox to analyze robotic systems with collision or contact events.

B. Underwater Robotics

Due to their compliant nature, soft robots pose no risk to underwater habitats or organisms, making them an excellent choice to employ in underwater exploration. Bacteria-inspired flagellate propellers could be employed for the locomotion of robots underwater. The idea of soft propellers is based on the observation that an inclined rotation of a soft filament in a liquid environment generates a helical filament shape, which produces a positive thrust for propulsion. An advanced version of an underwater robot propelled by four soft flagellate modules is presented in [14], with experimental validations of the dynamic simulation. For these simulations, the authors used

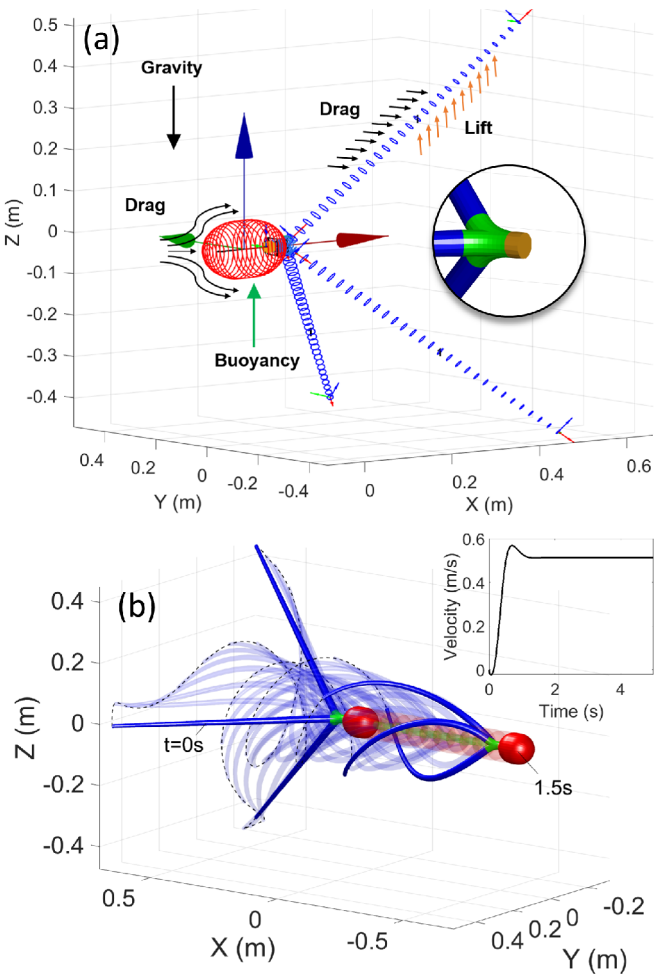

Fig. 9. (a) Underwater robotics: (i) System schematic shows the forces acting on the robot. The inset shows the hook geometry. (ii) Motion of the flagellate robot (superimposed images) due to the thrust generated by the soft propeller. The dotted lines show the tip trajectory of each soft filament. The inset shows the robot's speed vs time.

custom-made MATLAB codes with constant strain (order 0 strain polynomials) approximations. Here, using the SoRoSim toolbox, we demonstrate the variable strain dynamics of a similar system with a propeller consisting of three flagella (Figure 9a).

To rotate the filaments in an inclined fashion, we attach them to the terminals of a rigid body, namely the hook (inset in Figure 9a). All hook terminals have an inclination of 45° with the local x-axis. They are also separated by a rotation of 120° with respect to each other. The hook is connected to a motor shaft that provides the required torque. Additional components such as, motors, electronics, battery, and balancing weights are kept inside the spheroidal shell of the robot. We create a SorosimLinkage consisting of six links: the shell, the shaft, the hook, and three filaments. The shell is modeled as a rigid link with an adjusted inertia matrix to account for the internal components and spheroidal (non-default) shape. Its joint is a passive prismatic joint that only allows translation along the x-axis. The shaft is modeled as a cylindrical rigid link with an angle-controlled revolute joint to simulate the motor's input to

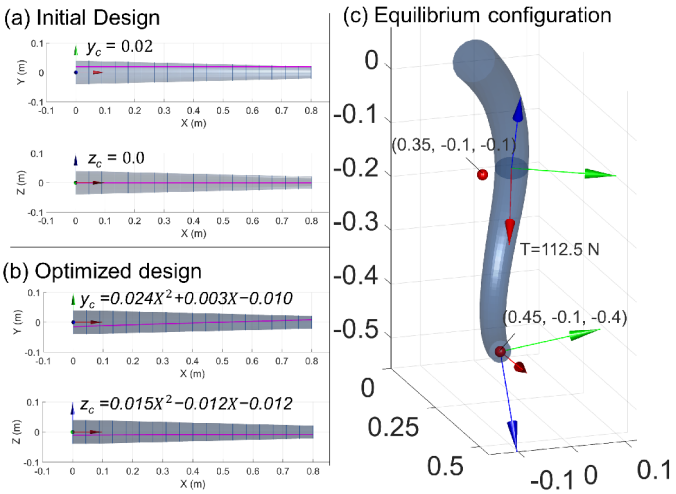

Fig. 10. Initial (a) and optimized (b) cable paths. (c) Final configuration where the optimized cable path and tension are used. Red points indicate the desired points.

the propeller. The hook is modeled as a fixed joint rigid link with an adjusted inertia matrix to account for its shape. Lastly, we define the three 0.7 m long soft filaments as fixed joint soft links with linear angular strains (torsion about the x-axis and bending about y and z-axis). The dimensions and material properties of these components are arbitrary but inspired by the proposed design in [14].

To simulate the fluid-robot interaction forces described in Figure 9(a), we need to include the external forces due to the presence of the fluid (buoyancy, drag and lift forces) and the forces due to volume of fluid moved by the propeller (added mass), described in [14], that are not part of the default force inputs during the SorosimLinkage creation. However, this does not prevent the user from implementing them using the 'CustomExtForce.m' function.

By providing an actuation input of $\theta = 2\pi t$, we run the simulation to capture the system's dynamic response for the first 5 seconds. Figure 9b shows superimposed images of the motion of the robot starting from rest to $t = 1.5$ s. In the inset, we plot the robot's forward velocity as a function of time. The combination of actuation and environmental interaction leads to the emergence of intelligent helical deformations of the filaments, which in turn, causes a thrust that propels the robot forward. This example emphasizes the usefulness of the SoRoSim toolbox in modeling rigid-soft hybrid underwater robots. The toolbox can model the robots mentioned Figure. 1(c) and (g) in a similar way.

VI. Design Analysis and Control Applications

In addition to static equilibrium analysis and dynamic simulations, we can use the toolbox for design optimization, inverse kinematics, or control applications. In this section, we demonstrate examples of an optimization problem and two control problems that can be solved using the SoRoSim package and custom-made MATLAB codes.

A. Design Optimization

Here, we demonstrate the use of the toolbox in optimizing an objective function based on the static equilibrium of a single cable actuated soft manipulator. We begin by defining a soft link with two division, with a total length of 80 cm and a radius linearly varying from 4 cm to 2 cm. We used a Young's modulus of 10 MPa and a Poisson's ratio of 0.5. We enabled all the rotational DoFs (torsion about the x-axis and bending about the y- and z-axis) and approximated the corresponding strains using a first-order polynomial. No external forces are taken into account for the static equilibrium. We set up an optimization problem to find an appropriate cable path and cable tension, minimizing the error in the manipulator end and middle point coordinates with desired positions at static equilibrium. The objective function is given by

$$\Omega = \|\bar{\boldsymbol{r}}_{mid} - \boldsymbol{r}_{mid}\| + \|\bar{\boldsymbol{r}}_{end} - \boldsymbol{r}_{end}\|, \tag{7}$$

where, $\boldsymbol{r}_{mid}$ and $\boldsymbol{r}_{end}$ are the equilibrium positions of the mid point and the end point of the manipulator and $\bar{\boldsymbol{r}}_{mid}$ and $\bar{\boldsymbol{r}}_{end}$ are the desired mid and end point positions.

We set the desired midpoint, $\bar{\boldsymbol{r}}_{mid}$ to $(0.35, -0.1, -0.1)$ and the desired tip point, $\bar{\boldsymbol{r}}_{end}$ to $(0.45, -0.1, -0.4)$. As the cable position is constrained to be inside the soft manipulator, the problem is a bounded optimization problem with an inequality condition. We parametrize the cable path's y and z coordinates using quadratic polynomials. Hence, we need to optimize seven variables: the coefficients of the quadratic polynomials defining y_c and z_c (y and z coordinates of the cable as a function of X) and the cable tension applied.

We use the MATLAB 'patternsearch' algorithm to find the optimal values of variables at the local minimum of the objective function. The initial condition is obtained from $y_c = 0X^2 + 0X + 0.02$, $z_c = 0X^2 + 0X + 0$, and $T = 0$ N. The 'patternsearch' algorithm, with parallel computing enabled, took about 1.5 minutes (93 s) to converge at the optimized parameters. There were 2,887 objective function evaluations (> 30 evaluations per second), including static equilibrium and forward kinematics evaluation to estimate the mid and endpoint coordinates. The optimal solution is given by, $y_c = 0.024X^2 + 0.003X - 0.010$, $z_c = 0.015X^2 - 0.012X - 0.012$, and $T = 112.5$ N. The initial and optimized cable path is shown in Figure 10a and b. The static equilibrium state of the manipulator at the optimized parameters is shown in Figure 10c. The manipulator tip position matches perfectly with the desired tip position. However, there is an error of about 14% in the midpoint position. The error could be due to geometric constrains such as the length and radius of the manipulator. It may be decreased by testing different initial values for the optimization.

This example stresses that the user can combine SoRoSim with well-established MATLAB packages like the global optimization toolbox to deliver analysis results outside the package of SoRoSim toolbox. The user may also define an optimization problem based on a dynamic simulation to estimate the design parameters of a multi-body system that optimizes its dynamic performance.

B. Inverse Dynamic Control

Finally, we use the toolbox to solve two inverse dynamics control problems. We begin by creating a soft link that is 1 m long, consisting of two divisions (0.5 m each). The radius

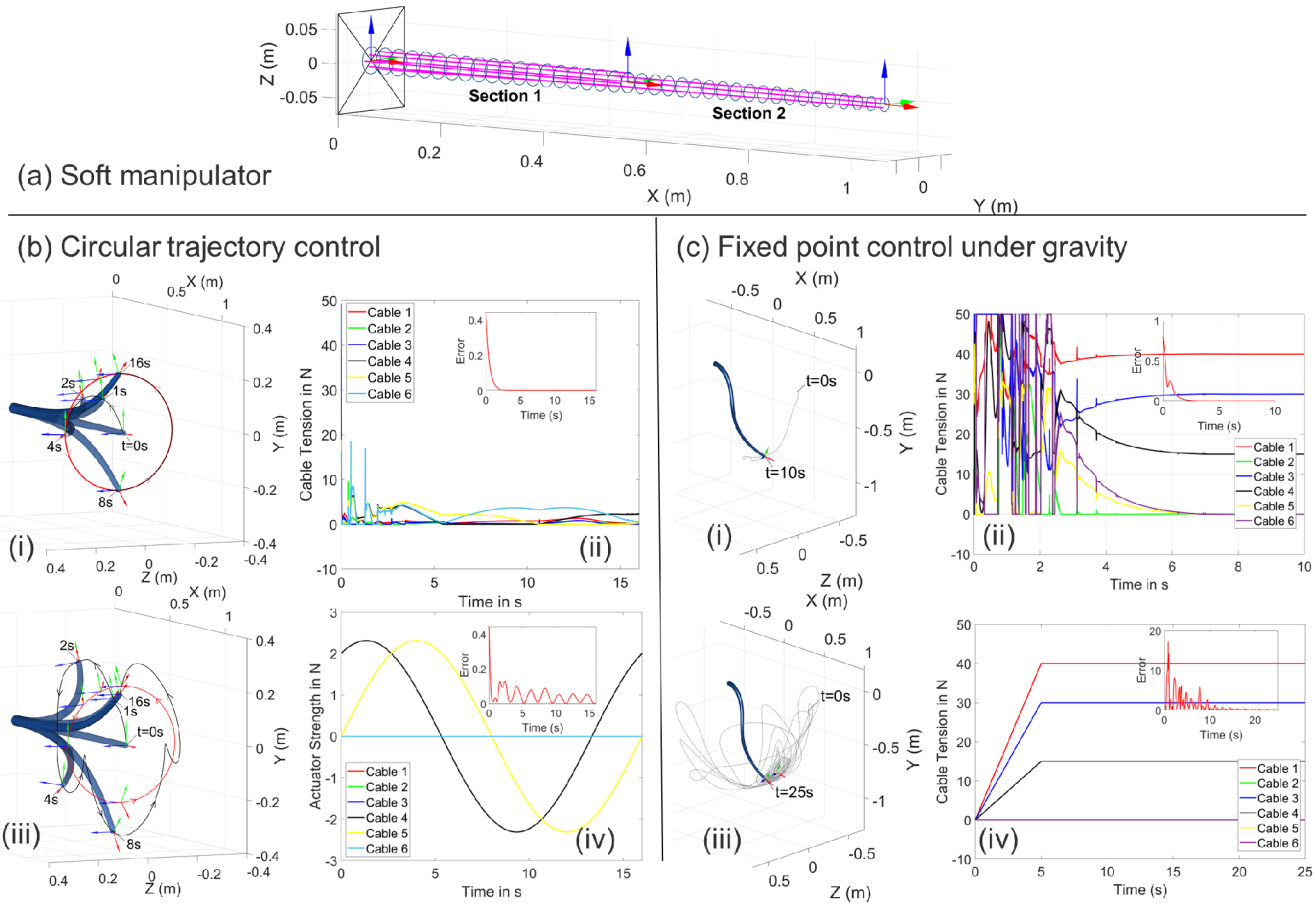

Fig. 11. Control of a soft manipulator. (a) Definition of the manipulator with six actuators. (b) Circular trajectory control (c) Fixed position and orientation control under gravity. (i) and (ii) are the outputs of the closed-loop PD controller, while (iii) and (iv) are that of an open-loop input.

TABLE I
CABLE COORDINATES. $d_1 = 1.5cm$ AND $d_2 = 1cm$.

Cable Number	Cable Length (m)	Y-coordinate	Z-coordinate
1	0.5	$\frac{1}{2}d_1$	$\frac{\sqrt{3}}{2}d_1$
2	0.5	$-d_1$	0
3	0.5	$\frac{1}{2}d_1(1-X)$	$-\frac{\sqrt{3}}{2}d_1(1-X)$
4	1	d_2	0
5	1	$-\frac{1}{2}d_2$	$\frac{\sqrt{3}}{2}d_2$
6	1	$-\frac{1}{2}d_2(1-\frac{X}{2})$	$\frac{\sqrt{3}}{2}d_2(1-\frac{X}{2})$

of the link varies linearly from 2 cm to 1 cm from the base to the tip (Figure 11(a)). The material properties used are the same as those used for the optimization example. For the first division, we define a linear bending about the y-axis and the z-axis. In the second section, the same deformation modes are defined using a constant strain basis. The soft manipulator is actuated using six linearly independent cables (Table I).

Velocity of the manipulator tip is given by, $\boldsymbol{\eta}_t = \boldsymbol{J}_t\dot{\boldsymbol{q}}$, where $\boldsymbol{\eta}_t$ is the tip velocity and $\boldsymbol{J}_t$ is the Jacobian at the tip. Taking a time derivative of the equation and substituting $\ddot{\boldsymbol{q}}$ from equation 4 we get the dynamics equation of the manipulator tip. From this equation, if we solve for the actuator strength, $\boldsymbol{u}$ and apply a PD controller [24], we get the following task space control law:

$$\boldsymbol{u} = \left(\boldsymbol{J}_t\boldsymbol{M}^{-1}\boldsymbol{B}\right)^{\dagger}\Big[\dot{\bar{\boldsymbol{\eta}}}_t + K_d\left(\bar{\boldsymbol{\eta}}_t - \boldsymbol{\eta}_t\right) + K_p(\log(\boldsymbol{g}_t^{-1}\bar{\boldsymbol{g}}_t))^{\vee} + \boldsymbol{J}_t\boldsymbol{M}^{-1}\left((\boldsymbol{C}+\boldsymbol{D})\dot{\boldsymbol{q}} + \boldsymbol{K}\boldsymbol{q} + \boldsymbol{Q}\right) - \dot{\boldsymbol{J}}_t\dot{\boldsymbol{q}}\Big], \tag{8}$$

where K_p and K_d are the proportional and derivative gains, $\dot{\bar{\boldsymbol{\eta}}}_t$ is the desired tip acceleration, $\bar{\boldsymbol{\eta}}_t$ is the desired tip velocity, $\boldsymbol{g}_t$ and $\bar{\boldsymbol{g}}_t$ the actual and desired tip transformation matrices, and log is the logarithmic operator in $SE(3)$. Note that the proposed control law may not ensure a full state ($\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$) convergence of the manipulator at steady state for an underactuated system [24]. Addressing issues of task space-based controllers is out of this paper's scope. Here we demonstrate the toolbox's ability to incorporate the control law.

The user can define a custom actuator strength by enabling the 'CAS' property of the SorosimLinkage and by editing the 'CustomActuatorStrength.m' file. The quantities such as Jacobian $\boldsymbol{J}$, derivatives of Jacobian $\dot{\boldsymbol{J}}$, and generalized mass matrix $\boldsymbol{M}$ are passed as inputs into the 'CustomActuatorStrength' function. The user can directly use these to compute the actuator strength given by equation 8.

1) Case 1: Tip Pose Trajectory Tracking

Here we attempt to control the position and orientation of the manipulator tip based on that of a reference frame moving in a circular trajectory. The circular trajectory we defined has a radius of 0.22 m, parallel to the yz plane, and its center is located at 0.96 m on the x-axis. Moreover, the reference frame of the trajectory is at 35.96° with respect to the x-axis. The angular velocity of the reference frame is set to 3.75 rpm. We run the dynamic simulation for 16 s using the actuator strength computed using equation 8. Superimposed images of dynamic results, corresponding actuator strengths, and errors (inset) are shown in Figure 11(b)(i) and (ii). The dynamics of the system is perfectly cancelled out by the controller and the error between the tip and the reference frames converges to zero in less than 4 seconds. The plot in figure 11(b)(ii) shows the process of nullifying the system dynamics at the beginning and the oscillation of the actuator strength as time elapses. We used the 'lsqlin' function of MATLAB to compute actuator strength given by equation 8. Using this function, we ensured the cable tension to be a positive value less than 50 N.

To compare the controller's performance, we input actuator strength corresponding to the quasi-static solution that matches the position and orientation of the manipulator tip and the moving frame. The dynamic of the simulation are shown in Figure 11(b)(iii) and (iv). In this case the error decreases initially but continues to oscillate indefinitely due to the dynamic response of the system.

2) Case 2: Tip Pose Regulation Under Gravity

For the second control case, we attempt to control the position and orientation of the manipulator tip based on a fixed reference frame under the influence of gravity (external force). The fixed reference frame is the same as the manipulator tip frame at the static equilibrium when the cable tensions are, 40 N, 0 N, 30 N, 15 N, 0 N, and 0 N respectively, for each cable.

Using the PD controller given by equation 8 we get a dynamic response as shown in Figure 11(c)(i) and (ii). We can see a steady approach to the desired tip position and orientation, denoted by the reference frame in the figure. The controller overcomes the dynamic response due to gravity and the actuation force and the error is quickly reduced to zero in less than 5 s. Over time the cable tensions converge to constant values corresponding to the static equilibrium case. To compare the controller performance, we input cable tensions as ramp functions that increase to reach the static equilibrium cable tension values at 5 s. This manual actuator strength input was unable to quickly cancel the system's oscillatory response Figure 11(c)(iii). The error is observed to decrease slowly allowing the tip to approach the desired position and orientation.

VII. Discussion and Conclusion

The computational performance of the toolbox depends on problem parameters such as the number of links, the degrees of freedom of the system, the strain order used, the number of Gauss Quadrature points on soft links, number of actuators, values of actuator strengths, external forces, material, and geometric properties. Here we report the simulation speed of the current version of the SoRoSim toolbox. Table II shows a summary of parameters that characterize the computational performance of the toolbox. For each analysis discussed in sections IV and V, the table shows information such as the total number of links (N), total degrees of freedom (N_{dof}), maximum strain order ($\boldsymbol{\xi}(O)$), order of Zannah collocation approximation ($Z(O)$), and the total number of points at which static or dynamic (4) equations are evaluated (N_{eva}). For static analyses (highlighted in gray), the seventh column of the table indicates the total static equilibrium simulations performed (N_s). For dynamic simulations, the column shows the real-time duration (0 to t_{max}) of the dynamics problem. MATLAB solvers used to solve the static or dynamic equations are shown in the eighth column. The last column shows the total simulation run-time (t_s). We used a PC with the following specifications for these analyses: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, 16.0 GB RAM. The MATLAB version used for these analyses is R2021a. The comparison of t_{max} and t_s indicates that the toolbox performs in real-time or faster for most of the examples presented in this paper. This is the result of the geometrically exact formulation, which requires the least DoFs to estimate the state of a robot. Fast (2.5 times faster than real-time) computation of actuator strengths could be used for model-based inverse dynamic control of a real-world soft manipulators. We will implement an implicit time integrator to speed up the simulation in a future version of the toolbox. Further code optimization and implementation of parallel computing can also increase the toolbox performance.

It is important to emphasize that, unlike FEM packages, the toolbox is only suitable for systems whose soft links can be modeled as Cosserat rods. Despite this limitation, the geometrically exact approach used in the toolbox makes it a fast (due to a smaller number of DoFs) and accurate tool for systems involving large deformations, which is the case for the majority of soft robotic applications. The toolbox allows the modelling and analysis of systems with open-, branched- or closed-chain and interconnected structures. Additionally, the toolbox provides the user with plenty of ways to use the output data with existing MATLAB packages and user-written MATLAB codes.

In summary, this paper presented SoRoSim, an intuitive MATLAB toolbox that uses the Geometric Variable Strain (GVS) approach to provide a unified framework for the modeling, analysis, and control of soft, rigid, and hybrid robots. It provides users with a level of freedom that will enable them to tailor and adjust material models, external forces, actuation paths, and functions to fit their applications and intended systems. We validated the toolbox simulation results by comparing the analysis results with existing literature and numerical studies. We provided four examples to highlight the application of the toolbox in soft robotics. SoRoSim successfully bridges the gap between soft and traditional robotics modeling and analysis. It is an ongoing effort and will be under constant improvement in terms of performance, features, and GUI enhancement. We hope that the users of the SoRoSim find it beneficial and user-friendly and we look forward to it being widely used in the research and engineering community.

TABLE II
SUMMARY OF TOOLBOX PERFORMANCE. STATIC SIMULATIONS ARE HIGHLIGHTED IN GRAY. * UNDAMPED ($\boldsymbol{D} = \boldsymbol{0}$) DYNAMICS. † WITH A TIME STEP OF 0.01 S.

Example	N	N_{dof}	$\boldsymbol{\xi}(O)$	$Z(O)$	N_{eva}	N_s (#) / t_{max} (s)	Solver	t_s (s)
IV-A	1	5	4	4	16	130	$fsolve$	5.5
IV-B	2	30	2	4	20	1	$fsolve$	0.5
IV-C	1	15	2	4	13	7	$ode45$	1
IV-D	1	12	3	4	15	5	$ode15s$	1.5
IV-D*	1	12	3	2	15	5	$ode113$	6
V-A	11	39	0	4	143	5	$fsolve$	6
V-A2	6	21	0	4	96	9	$ode15s$	27
V-B	6	20	1	2	30	5	$ode113$	5
VI-A	1	12	1	4	15	2,887	$fsolve$, $patternsearch$	93
VI-B1	1	6	1	4	15	16	$ode1^\dagger$	5
VI-B2	1	6	1	4	15	10	$ode1^\dagger$	4

ACKNOWLEDGEMENT

This work was supported in part by US Office of Naval Research Global under Grant N62909-21-1-2033, and in part by the the Khalifa University of Science and Technology under Grants CIRA-2020-074, RC1-2018-KUCARS.

REFERENCES

[1] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, 2016. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aah3690

[2] C. Armanini, C. Messer, A. T. Mathew, F. Boyer, C. Duriez, and F. Renda, "Soft robots modeling: a literature unwinding," 2021. [Online]. Available: https://arxiv.org/abs/2112.03645

[3] D. Holland, S. Berndt, M. Herman, and C. Walsh, "Growing the soft robotics community through knowledge-sharing initiatives," *Soft Robotics*, vol. 5, no. 2, pp. 119–121, 2018. [Online]. Available: https://doi.org/10.1089/soro.2018.29013.dph

[4] M. A. Graule, C. B. Teeple, T. P. McCarthy, R. C. St. Louis, G. R. Kim, and R. J. Wood, "Somo: Fast and accurate simulations of continuum robots in complex environments," in *2021 IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, p. In Review. [Online]. Available: https://ieeexplore.ieee.org/document/9636059

[5] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321. [Online]. Available: https://hal.inria.fr/hal-00681539

[6] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International Conference on Robotics and Automation*, 2019, pp. 6265–6271. [Online]. Available: https://ieeexplore.ieee.org/document/8794333

[7] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A geometrically exact model for soft continuum robots: The finite element deformation space formulation," *Soft Robotics*, vol. 6, no. 6, pp. 790–811, 2019, pMID: 30481112. [Online]. Available: https://doi.org/10.1089/soro.2018.0047

[8] J. Austin, R. Corrales-Fatou, S. Wyetzner, and H. Lipson, "Titan: A parallel asynchronous library for multi-agent and soft-body robotics using nvidia cuda," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7754–7760.

[9] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9369003

[10] B. Angles, D. Rebain, M. Macklin, B. Wyvill, L. Barthe, J. Lewis, J. Von Der Pahlen, S. Izadi, J. Valentin, S. Bouaziz *et al.*, "Viper: Volume invariant position-based elastic rods," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 1–26, 2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3340260

[11] S. H. Sadati, S. E. Naghibi, A. Shiva, B. Michael, L. Renson, M. Howard, C. D. Rucker, K. Althoefer, T. Nanayakkara, S. Zschaler *et al.*, "Tmtdyn: A matlab package for modeling and control of hybrid rigid–continuum robots based on discretized lumped systems and reduced-order models," *The International Journal of Robotics Research*, vol. 40, no. 1, pp. 296–347, 2021. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/0278364919881685

[12] 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*, pp. 1–10, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9442856

[13] I. Hussain, F. Renda, Z. Iqbal, M. Malvezzi, G. Salvietti, L. Seneviratne, D. Gan, and D. Prattichizzo, "Modeling and prototyping of an underactuated gripper exploiting joint compliance and modularity," *IEEE Robotics and automation letters*, vol. 3, no. 4, pp. 2854–2861, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8378053

[14] C. Armanini, M. Farman, M. Calisti, F. Giorgio-Serchi, C. Stefanini, and F. Renda, "Flagellate underwater robotics at macro-scale: Design, modeling and characterization," *IEEE Transaction on Robotics*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9486942

[15] "Sorosim: A unified framework for soft and rigid robotics simulation, control and optimization," https://github.com/Ikhlas-Ben-Hmida/SoRoSim, version 2.32, Updated 11 Jul 2021.

[16] 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. [Online]. Available: https://ieeexplore.ieee.org/document/9057619

[17] 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, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9196808

[18] F. Renda and L. Seneviratne, "A geometric and unified approach for modeling soft-rigid multi-body systems with

lumped and distributed degrees of freedom," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1567–1574. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8461186

[19] F. Boyer and F. Renda, "Poincare's equations for cosserat media: Application to shells," *Journal of Nonlinear Science*, vol. 27, no. 1, pp. 1–44, 2017.

[20] J. Selig, *Geometric Fundamentals of Robotics*, ser. Monographs in Computer Science. Springer New York, 2007. [Online]. Available: https://link.springer.com/book/10.1007/b138859

[21] J. C. Simo and L. Vu-Quoc, "A three-dimensional finite-strain rod model. part ii: Computational aspects," *Computer methods in applied mechanics and engineering*, vol. 58, no. 1, pp. 79–116, 1986. [Online]. Available: https://doi.org/10.1016/0045-7825(86)90079-4

[22] J. Simo and L. Vu-Quoc, "On the dynamics in space of rods undergoing large motions - A geometrically exact approach," *Computer Methods in Applied Mechanics and Engineering*, vol. 66, no. 2, pp. 125–161, 1988. [Online]. Available: https://doi.org/10.1016/0045-7825(88)90073-4

[23] P. Flores, "Contact mechanics for dynamical systems: a comprehensive review," *Multibody System Dynamics*, vol. 54, no. 2, pp. 127–177, 2022. [Online]. Available: http://dx.doi.org/10.1007/s11044-021-09803-y

[24] C. D. Santina, C. Duriez, and D. Rus, "Model based control of soft robots: A survey of the state of the art and open challenges," 2021. [Online]. Available: https://arxiv.org/abs/2110.01358