FEA-based soft robotic modeling: Simulating a soft-actuator in SOFA

1st Pasquale Ferrentino
Brubotics
Vrije Universiteit Brussel and Imec
Pleinlaan 2, 1050 Elsene, Belgium
pasquale.ferrentino@vub.be

2nd Ellen Roels
Brubotics
Vrije Universiteit Brussel and Imec
Pleinlaan 2, 1050 Elsene, Belgium
ellen.roels@vub.be

3rd Joost Brancart
Physical Chemistry and Polymer Science
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Belgium
joost.brancart@vub.be

4th Seppe Terryn
Brubotics
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Belgium
seppe.terryn@vub.be

5th Guy Van Assche
Physical Chemistry and Polymer Science
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Elsene, Belgium
guy.van.assche@vub.be

6th Bram Vanderborght
Brubotics
Vrije Universiteit Brussel and Imec
Pleinlaan 2, 1050 Elsene, Belgium
bram.vanderborght@vub.be

***Abstract*—Soft robotics modeling is a research topic that is evolving fast. Many techniques are present in literature but most of them require analytical models with a lot of equations that are time consuming, hard to resolve and not so easy to handle. For this reason, the help of a soft mechanics simulator is essential in this field. This paper presents a tutorial on how to build a soft-robot model using an open-source Finite Element Analysis (FEA) simulator, called SOFA. This software is able to generate a simulation scene from a code written in Python or XML, so it can be used by people that with different fields of competence like mechanical knowledge, knowledge of material properties and programming skills. As a case study, a Python simulation of a cable-driven soft actuator that makes contact with a rigid object is considered. The basic working principles of SOFA required to make a scene are explained step by step. In particular, it shows how to simulate the mechanics and animate the bending behavior of the actuator, and the importance of the knowledge of the constitutive material properties for a good modelling of the mechanical system. Furthermore, it will be shown also how to retrieve and save data from simulation, demonstrating that SOFA can easily adapt to a multi-disciplinary subject as the research in soft-robotics, but also be useful for teaching simulation and programming language principles to engineering students.**

***Index Terms*—Soft Robotics, Finite Element Analysis, Modeling, Tutorial.**

I. Introduction

Soft robotics is a research field that involves different competencies resulting in a multi-disciplinary subject that embrace a large number of scientific communities. Among the several applications, one of the main focus of soft robotic community is the creation of bio-inspired robotic structures [1]. As discussed in [2], bio-inspired soft robotics try to mimic the embodied intelligence present in biological tissues of animals and plants. For this reason, one of the main topic in soft robotics research is the development of new materials [3] that improve the performance of current soft robotic applications or enable new applications. Many of these new material types replicate, aside from a mechanical performance, additional functionalities present in living tissues and are called *smart materials* [4]. One of the striking examples of smart materials are the *self-healing polymers* [5], which are materials that can recover from macroscopic damages by chemical re-bonding at the fracture interfaces.
The progress in materials research for soft robotics has led to new bio-inspired robotic structures and new actuation types, including dielectric [6] and shape memory actuation [7]. However, most soft robots are still tendon-driven or pneumatically actuated [8], as these actuation types provide the highest force output and fastest actuation. Inspired by nature, multiple examples of elephant trunks [9], [10] and octopus tentacles [11] are present in literature, mainly used for efficient grasping [12] and manipulation [13] of objects. The main advantages of bio-inspired soft robotic grippers and manipulators are their adaptability to the shape of the object being grasped, and their increased dexterity [14]. However, this creates complicated contact interactions with the object and/or the environment, which is challenging to model and control as discussed in [15]. To enhance the performance of soft grippers, researchers have started to create multi-material structures in which the anisotropic deformation that produces the bending motion in the actuators is enhanced [16], [17]
The use of new materials and (multi-material) designs for soft robotic structures reveal the need of more accurate models and control strategies. Consequently, in scientific literature, model-based techniques have appeared [18]. One of the most common approaches is geometry-based modeling, as presented in [19] and [20]. This modeling approach is based on *modified Denavit-Hartenberg parameters* and the

⋆corresponding author email : pasquale.ferrentino@vub.be

assumption of *Constant curvature* (CC) or *Piecewise Constant Curvature*. This technique is characterized by a large number of equations and matrix calculations, leading to a fast increase in mathematical complexity for precise modeling of larger systems. Furthermore, the geometry-based modeling technique highly depends on the soft robotic design, on its degrees of freedom (DOF) and on the actuation principle, as demonstrated in [21]. As a result, the equations involved and the assumption change for each geometry are impossible to resume in a uniform tutorial or in a lesson for researchers in soft robotics and engineering students.

Alternatively, new techniques are developed for soft robotic modeling, using *Finite Element Analysis* (FEA) simulators, able to recreate the mechanical behavior of soft actuators, grippers and manipulators. One on the most common FEA simulator used in soft robotics research is SOFA [22], an open-source framework that allow to built an interactive simulation. In addition SOFA allows to control a soft robot by inserting the FEA simulation in a control scheme, as demonstrated in [23]. The simulation scene in SOFA can be written in XML programming language or in Python. Using the FEA technique, the model or the controller is based on material characteristics, geometrical constraints and loads. In fact FEA simulation principle can be used for any geometry, actuation principle and geometrical/mechanical constraints. As described in our previous paper [24], it is possible to create a model and controller for soft robots, starting from basic material characterization using uni-axial mechanical testing and fitting of hyper-elastic constitutive laws that are used in a FEA model/controller. This approach truly forms a bridge between material science and soft robotics.

Building further on this past work, this paper presents a tutorial on building a soft robotic actuator model in Python using the SOFA simulator. The development of the simulator in this tutorial, provides new insight in programming skills for the creation of the scene in Python, basic simulation competence for meshing of the design parts, the integration schemes involved etc., mechanical and material science backgrounds for integrating in the simulation the loads, constraints and material constitutive models.

SOFA's multi-disciplinary approach makes this FEA platform usable for a wide range of researchers and students with different backgrounds. As a case study for this tutorial, a tendon driven soft actuator is considered, made out of self-healing material. Its design is based on a paper by Roels et. al [25] and it can be used as a finger in a bionic hand or soft gripper. Aside from simulating the deformation during non-contact actuation, we will model the contact of this actuator with a cube, simulating the resulting collision forces.

The paper is structured as follows: in Section II the working principle of the SOFA simulator is explained, stressing its hierarchical structure. Next, in Section III and IV, the functionalities of the most important components in SOFA are discussed, the animation loop is explained, and we show how data is exchanges with other components in the scene. Furthermore, in Section V the principal SOFA function is illustrated. This includes how to create the soft actuator model, selection of the mesh size, the DOF and the force fields (Section V-A), discussing on the importance of the material properties characterization and why they are important to have a good representation of the mechanical system. Next, resulting actuation simulation and how to interact with it are discussed (Section VI). Finally, we introduce a method to simulate collisions between two objects in SOFA in Section VII.

II. Simulation graph

SOFA is an open-source simulator developed by the *Institut National de Recherche en Informatique et en Automatique* (INRIA) in 2006. It is sustained by a large community of universities, research institutes and companies from 52 countries. The software is composed of a C++ core, but the simulation scene can be developed using XML or Python. All the functionalities of SOFA are split in *plugins*, which are packages of functions, useful for a specific application. The user can choose to download only the plugins that are suitable for his or her simulation. Consequently, SOFA is a compliant environment, in which the user can personalize their current version. Hence, SOFA is an optimal solution to decrease costs of licenses for multiphysics simulation plugin/libraries.

This paper introduces a tutorial on how to make a simulation in SOFA coded in Python, considering as a case study a tendon-driven soft finger that makes contact with an object.

The simulation in the SOFA framework are characterized by a so called *Simulation Graph*, visible in Fig.1. The Simulation Graph structure is composed of different parts called *Nodes*. The different nodes have to be presented in the simulation according a *hierarchical structure* and they are linked to each other by a *parent-child* relationship. As can be noticed from Fig.1, the simulation graph is always initialized by the *rootNode*, which is the parent of all the other nodes present in the simulation graph.

Inside the rootNode, all the other nodes can be declared. Each of these nodes will represent a simulated mechanical object. In Fig.1 a), three different simulated objects are shown, declared with the name of Finger, Floor and Cube. The "Finger" represent the tendon-driven actuator that is used as case-study in this paper, visible in Fig.1 b). The "Cube" is an external object that goes in contact with the robotic finger, and the "Floor" represents the surface on which the cube lies. Furthermore, to have a complete description of a simulated mechanical object in SOFA, each of these nodes can be split in four principal sub-nodes. The *Mechanical Node* all the mechanical properties, mass/density, geometrical constraints and material properties are declared. In Section.V is discussed the importance of the material properties, that in FEA simulations play a key-role for the modelling of the mechanical system.

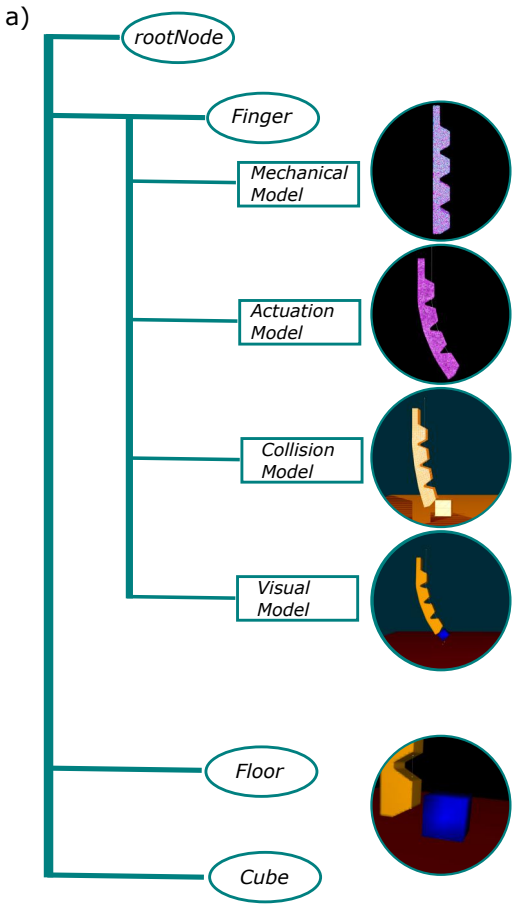

Fig. 1. a) A typical simulation graph of a SOFA simulation scene. The rootNode starts the scene and is the parent of all the other nodes in the scene. In the graph, three different simulated objects can be recognized for this case study, declared as Finger, Floor and Cube. Then especially for the finger mechanical object, the four sub-nodes are shown that are important in SOFA to recreate a model of a tendon-driven soft finger: the mechanical model, actuation model, collision model and visual model. The images are associated to the results that produce each node. b) Realized cable-driven soft finger during actuation.

The *Actuation Node*, needed in soft robotic simulations to model deformation by actuation, derives specific functions from the *SoftRobots plugin* [26]. In this node the type of actuation of the object's motion is declared and animated in the scene by building an internal controller that handles the events in the simulation. The choice of this events is remanded to the user. Next, the sequence contains a *Collision Model* node if the object goes in contact with another object or obstacle. In this node, the user defines the locations at which the collision forces will be calculated, in particular, on which point, surface or line. Finally, there is the *Visual Model* sub-node, where the user chooses the visual style of each object. All these nodes are listed in Fig.1 showing how the simulation appear for each of them. These nodes will be described in detail in the next sections.

III. Animation Loop

The rootNode initializes the simulation graph in SOFA, where the following global variables of the scene are declared: the time step, the gravity field of the simulation environment and the type of contacts that have to be considered in the collision model of the objects. The most important SOFA function that has to be present in the rootNode is the so called *Animation Loop*. The Animation loop updates the scene results at each time step. This loop contains multiple functions:

- It builds and solves the linear systems of differential equations that are present in the simulation.
- It solves collisions and constraints.
- It updates the scene.

In SOFA there are different Animation Loop available and these differ mostly in their method to solve collisions and constraints:

- The *default Animation Loop* solves the constraints and the collision together, triggering an integration scheme.
- The *free Animation Loop* resolves the linear system of differential equation, calculating the "free" solution without taking in account collision and constraints. Next, the constraints and the collisions are added to the outcomes of the free solution via a correction factor. This Animation Loop is indicated for SOFA simulations that involves the functions of the soft robot plugin.
- The *multi step Animation Loop* allows to compute several collisions and integration in one single simulation time step (dt). The user has to define the number of collision steps (C) and the integration time steps (I). Each collision computation will require a time $dt' = dt/(CI)$.

IV. Visitors and Mappings

The hierarchical structure of the SOFA simulation graph implies that all the nodes have to communicate with the Animation Loop present in the rootNode, and the other child nodes present in the simulation. For this reason, SOFA has an internal system of communication which uses specific

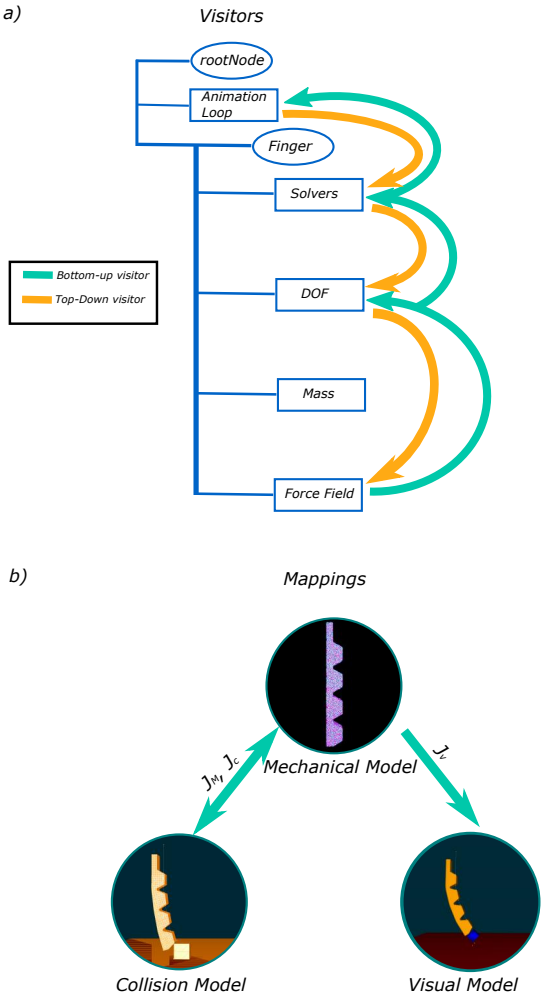

Fig. 2. The internal communication systems used by a SOFA simulation. a) shows the working principle of the visitors between the animation loop in the rootNode and the other child nodes. In particular the communication between the animation loop and a mechanical model is shown. In aquamarine green the bottom-up visitor is shown. In orange, a top-down visitor starting from the animation Loop that triggers all the components in the mechanical objects is shown. b) depicts the working principle of the Mappings and their interaction to map the DOF between different sub-nodes of a particular simulated object. In this case, the robotic finger that will be present in the tutorial.

messengers, called *visitors* that go through the whole SOFA scene and perform several operations in a single time step (t'). The working principle of the visitors is illustrated in Fig.2 a), taking as example the communication between the rootNode and the mechanical model of the robotic finger present in the tutorial. In aquamarine green, a so called *Bottom-up* visitor is shown that collects the loads applied to the mesh from the force field and the deformation state (positions, velocities etc.) from the DOF and sends all this data to the the ODE (Ordinary Differential Equation) equations solver, to calculate the solution. Next, the solution from the solver is communicated to the Animation loop that collects all the equations solved in the other objects present in the scene. In this way, a linear system of equations is made and ready to be solved. After the Animation loop finishes to retrieve the solution of the linear system, it sends a *top-down* visitor, visible in orange in Fig.2 a), which triggers all the components of the mechanical model to prepare the computation explained in the next time step (t'+ Δt).

Furthermore, the description of a simulated object is also hierarchical and split in different nodes that have specific functionalities, as already explained in Section II. Hence, also the multiple sub-nodes of a specific simulated object have to communicate between each other. For this purpose, in SOFA there is another communication system, called *Mappings* that has to be declared by the user in the scene. Their role is to map the DOF and the deformation state between different sub-nodes of a particular simulated object, through *Jacobian matrices*. Fig.2 b) illustrates the communications between the mechanical, collision and a visual model of the robotic finger present in the tutorial. When the Mechanical model is actuated, its updates internally its deformation state according its DOF. This deformation state is propagated to the collision model through the Jacobian J_M, so also the collision model will result in actuation. Whenever the collision model detects a contact with an external object/surface, it internally compute the collision forces for each node involved in the contact. But these contact forces will influence the deformation state of the robotic finger, so this information is transmitted towards the mechanical model, through the Jacobian matrix J_C, that recomputes the deformation state of the object considering the contact forces detected by the collision model. Furthermore, if the user add a visual model to the object, all the changes of the deformation state, will be transmitted from the mechanical model to the visual one, with the Jacobian matrix J_v.

V. Mechanical Model

The simulation of an object in SOFA starts from the definition of the mechanical characteristics of the object, including the mass/density and mechanical properties. The first step is to initialize the node of the mechanical properties with an *ODE solver* and an *integration scheme*. For SOFA simulations for soft robots, these are usually respectively an *Euler Implicit Solver* and a *Sparse LDL linear Solver*. These two components allow to retrieve the solution for the mechanical equations involved in the simulation at each time step. In the following subsections, other important components of the mechanical model in sofa will be described in detail.

A. Mesh size

The SOFA framework is a finite element analysis (FEA) software. This means that the object geometry, produced by the user in any commercial CAD software, can be uploaded in the SOFA simulation. The surface or the volume

of the object's geometry can be discretized respecting the Delaunay triangulation theorem in 2D [27] and in 3D [28], which explains how to divide whatever surface/volume in triangles/tetrahedrons.
The software supports different mesh formats, including STL or OBJ for surface meshes and VTK, GMSH for volume meshes. In the presented tutorial, the cable-driven soft finger is constituted by a tetrahedral mesh, while the cube object and the floor are surface meshes. Hence the two latter are declared as obstacle surface.
In SOFA there is a plugin, called *CGAL plugin* [29], in which it is possible to generate a tetrahedral mesh starting from a triangular one and to regulate the mesh size, through the fine-tuning of four parameters that are typical for the Delaunay theorem. As defined in the Soft Robotics Toolkit [30] they are:

- The *facetSize* provides an upper limitation for the Delaunay ball's surface. A larger value leads to larger tetrahedrons in the mesh.
- The *facetApproximation* provides an upper limit for the distance between the circumcenter of a facet's surface and the center of Delaunay ball's that belongs to this facet.
- The *cellRatio* controls the shape of mesh cells. Also this parameters defines an upper bound for the ratio between the circumradius of a mesh tetrahedron and its shortest edge. Theoretically, for values larger than 2 the Delaunay refinement process converges.
- The *cellSize* refines the size of tetrahedrons in the mesh.

In FEA simulators it is important to choose the right mesh size in order to have accurate results that correspond to the real mechanical behaviour, as demonstrated also in [31]. For this purpose, a typical technique used is to simulate in the same conditions the object with different mesh sizes. For each mesh size, the accuracy of the result is studied. If the meshes start to converge towards the same value, then the largest mesh size (with less tetrahedrons) that leads to these results has to be chosen.
This optimization technique is illustrated in the tutorial for the tendon driven finger simulation. In Fig.3 a) four different mesh sizes are shown. In particular we have meshes of 4500, 9300, 16200 and 36200 tetrahedrons. Each of this meshes is simulated in the same condition, applying a sinusoidal displacement on the cable. From Fig.3 b) it is clear that refining the mesh always lead to more accurate results. In fact the results relative of the meshes with 4500 and 9300 tetrahedrons, visible respectively in blue and red in Fig.3 b), are not converging at the same results as the meshes with 16200 and 36200 tetrahedrons (shown in orange and violet in the same figure). Hence, from the results of Fig.3 b), the mesh size with 16200 tetrahedrons is the most suitable candidate for the cable-driven soft finger simulation.
In SOFA after uploading the mesh, the user has to define another function, called *MechanicalObject*, to store all the information like position, velocity and forces of the uploaded geometry. This information is updated at each time step through the Animation Loop presented in the Section.III.

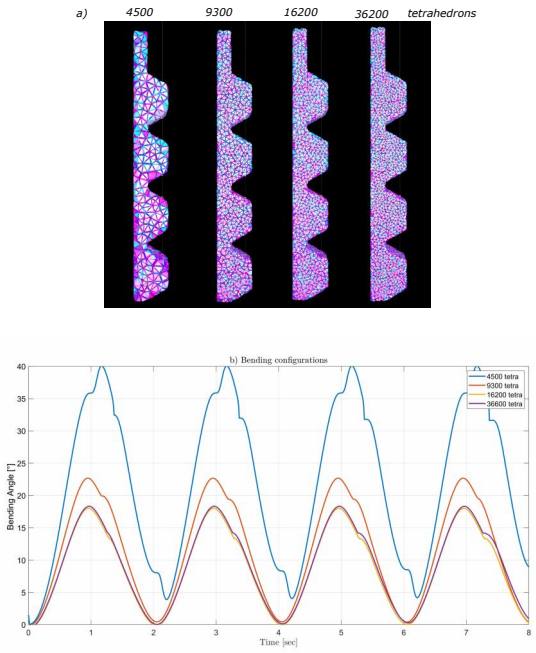

Fig. 3. Convergence in the results at different mesh sizes. a) The image of the cable-driven finger present in the tutorial at different mesh size. To investigate the influence of the mesh, four mesh size are made, with 4500, 9300, 16200 and 36600 tetrahedrons. b) The results of bending angle vs. time for each mesh. The four meshes are simulated in the same condition; a sinusoidal displacement is applied on the cable during the simulation. It is clear that an increase in the number of tetrahedrons in the mesh gives a more accurate result. In fact the mesh with 16200 (orange) and 36600 (violet) tetrahedrons converge at the same bending angle trend, while the one with less tetrahedrons like 4500 (in blue) and 9300 (red) do not have accurate results. In the end, the mesh with 16200 tetrahedrons can be chosen as candidate for the simulation. In the tutorial it is shown that this mesh it is also a good trade-off between results accuracy and computation time.

B. Mechanical properties

Another important part of the FEA simulation, is to define and declare the material properties of the simulated object. The mass/density and the *Constitutive Model* equations of the materials that compose the mesh domains of the object in simulation have to be declared. In SOFA it is possible to declare the density or the mass of the object in three different ways, as explained in SOFA website [32]:

- The *UniformMass* is the simplest way to declare the mass of the simulated object. This function does not consider the volume of the object but it equally distributes the total mass M_{total} over the number of nodes in the mesh. The mass of the single node is: $M_{node} = M_{total}/N$. Where N is the total number of the nodes in the mesh.

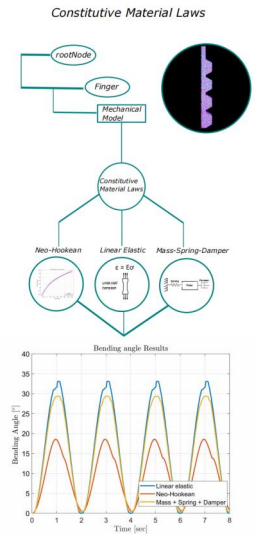

Fig. 4. The choice of the right constitutive material model is crucial to have a good representation of the mechanical system. In the plot at end of the figure are shown the bending angle results for the tendon-driven soft actuator simulations with different constitutive material laws, obtained changing the parameters and the force fields in the mechanical model of the simulation. The bending angle responses are retrieved for the same sinusoidal actuation input signal. The constitutive material laws tried in simulation are a linear elastic model, visible in blue, an hyper-elastic Neo-Hookean model, shown in red, and a mass-spring-damper model in orange. From the plot it is clear that different constitutive models applied at the same design for the same actuation input leads to different results. Hence, the knowledge of the material properties are important to describe the mechanical system.

- The *MeshMatrixMass* takes as input the density of the material that composes the simulated object and calculate the integral over its volume. This function works for any kind on mesh topology. After the integration of the mass density for each element, the function composes the mass matrix ($\vec{M}$) that will be summed in the system matrix ($\vec{A}$). The mass $\vec{M}$ is an element of the linear system of equation $\vec{A}x = b$ that arises from the dynamic equation : $\vec{M}\Delta v = \Delta t(f(x,t))$, where x are the DOF and t is the time.
- the *DiagonalMass* computes the integral of the mass density over the volume of the object geometry in the same way of the MeshMatrixMass function. But the function considers the mass matrix $\vec{M}$ as diagonal.

Constitutive material laws for simulating the mechanical behavior of the materials can be integrated in SOFA via *Force Fields*. Depending on the material, the user can select from a large variety of material laws and import the characterizing parameters, including linear spring models, as discussed in [33] and Hooke's law of elasticity with *singular values decomposition* (SVD) present in [34] and in its corotational formulation like reported in [35]. Furthermore, many hyper-elastic constitutive models, like the Neo-Hookean law, are present as well, all listed in [24] and in [36]. In addition, viscoelasticity can be modelled as well using mass-spring-damper models like the Rayleigh model, as discussed in [37]. From the extensive list above it can be concluded that SOFA allows to model the mechanical properties of the majority of the materials used in soft robotics.

The knowledge of the constitutive material laws is essential to have a good representation of kinematic and dynamic behavior of the soft robot. In fact, different material characterizations simulated at the same actuation conditions, can lead at different kinematic behaviors. To make clear this point, in Fig.4 are visible the bending angle responses of the tendon-driven soft actuator for different constitutive material laws at the same actuation input sinusoidal signal. The constitutive material laws tried in simulation are a linear elastic model, an hyper-elastic Neo-Hookean model, and a mass-spring-damper model. The material constants for the linear elastic model and the Neo-Hookean model are relative to the characterization of the self-healing material called F5000-R0.5. In particular, the linear elastic parameters (Young Modulus and Poisson Ratio) are taken from [38], while the Neo-Hookean parameters are taken from [24], instead the parameters for the mass-spring-damper model are dummy values. From the plot it is clear that different constitutive models applied at the same design for the same actuation input lead to different results. Hence, the knowledge of the constitutive material properties is crucial to understand the behavior of the mechanical system. As demonstrated in [24] and [36], the material modeling choice can be obtained by fitting the experimental data of mechanical test. The material constants obtained from the fitting algorithm have to be inserted in the simulation of the actuator, then the simulation has to be validated experimentally with the real behavior of the actuator.

Anyway, the mechanical tests on material specimen requires expensive equipment and not all the laboratories can access to them. For this reason, in [39] is presented another way to have information on the material properties, creating a inverse FEA approach to retrieve a simulation-ready characterization of soft materials under tensile test.

In the tutorial of tendon-driven finger simulation, the integration of the density based on the MeshMatrixMass function and the constitutive material laws are explained step by step.

VI. Actuation Model

After the definition of the mechanical properties of a soft finger, in SOFA the actuation typology have to be specified. In particular, for soft robots it can be pneumatic or cable-driven. In SOFA the actuation type is declared as constraint. Two functions are important:

- The *SurfacePressureConstraint* is used for pneumatic actuation and takes as input the surface mesh of the air chamber of the robot, visible in Fig.VI a). In this way, it is possible to simulate the pressure in the air chamber or its volume growth.

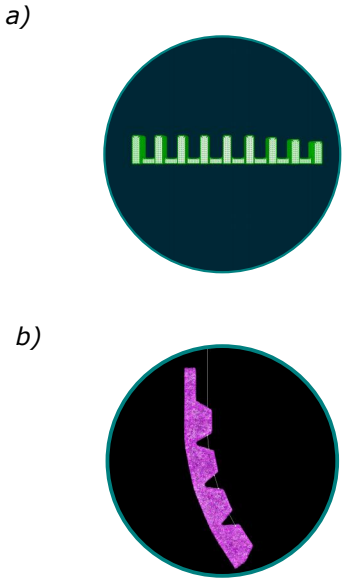

Fig. 5. Actuation models in SOFA. a) Surface mesh of an air chamber of a pneumatic soft actuator. b) cable-driven actuation.

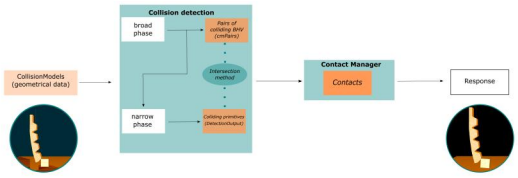

Fig. 6. Collision Pipeline in SOFA. the simulator takes as input the so called collision models. In this tutorial, the triangular mesh of the external surfaces of the object that are going in contact. The collision pipeline has two main steps, the collision detection and the collision response. The collision detection is split in two phases, the broad phase that identify the couple of mesh nodes that are going in contact for the two objects and the narrow phase that calculates the primitives (points, edges, triangles). The two phases rely on an intersection method to assess if the model will collide. After that, there is the part of collision response that will compute the contact forces/deformations and map the DOF to the mechanical model.

- The *CableConstraint* is instead used for the cable/tendon-driven actuation. The user needs to declare the positions in the three dimensional space where the cable/tendon is passing, as illustrated in Fig.VI b). The cable/tendon is assumed as *inextensible* and can be controlled in displacement or in pulling force. The user can indicate also the pulling point.

Furthermore, SOFA allows to interact a lot with the simulation. In fact the user can use a controller, coded in Python, that handles the event in the simulation, for example when the simulation is initialized, at beginning of each time step or at the end of it. Another possibility, is to trigger an event trough this controller, for example pressing a keyboard button or when a collision/contact with an object is happening.

VII. Collisions

In SOFA simulations the collision phase is separated from the physics simulations, and it is triggered before the call to the solvers in the mechanical model. The collision simulation respects a *Collision Pipeline*, shown in Fig.6, that has two main phases, the collision detection and the collision response.

A. Collision detection

As explained in the SOFA framework website [32], the collision detection has the scope of determining if multiple objects collide. In SOFA, the collision detection takes as input a surface mesh, which is called *Collision Model*, and returns pairs of contact points that establish the so called geometric primitives. The collision detection is usually split into two steps, called *broad phase* and *narrow phase*. The broad phase aims at removing objects pairs that are not in collision. It define bounding boxes to each object in the scene, controlling if these boxes collide or not. The narrow phase of detection is based on collision models to detect a contact. Different collision models are available to detect a contact:

- geometric primitives: point, line, triangle, sphere, cube, cylinder or bounding boxes.
- distance grid associated to each object in the scene.
- ray casting, that is faster in computation but less accurate than primitives.

All these collision detection methods are based on intersection methods in order to certify if the models do collide.
The most used intersection method in SOFA is the so called *proximity method* because it will anticipate the contacts guaranteeing its stability. The two main functions used in SOFA for the proximity method are: *MinProximityIntersection* and *LocalMinDistance*.

B. Collision response

The colliding models returned by the narrow phase are finally given to the *ContactManager*, which creates contact interactions. The user can specify the kind of contact that wants to use in the simulation, with the function *DefaultContactManager*. There are three fundamental methods to implement the collision responce in SOFA:

- The penalty method, efficient but need to be fine tuned to avoid instabilities.
- The persistent method.
- Lagrange multipliers, in this case the contact is seen as a constraint and is processed by the solvers together with the other quantity of the mechanical model (forces, geometrical constraints etc..).

VIII. Conclusion

In this paper, we presented a tutorial on how to build and interact with a model of a self-healing cable-driven soft finger in SOFA, one of the most used FEA simulators in

research, being applied in fields such as soft robotics, medical simulations, haptics etc. The tutorial combines programming skills, involving scene-encoding in Python, with simulation, mechanical and material science competences. This combination of multiple disciplines, typically required in many of today's engineering domains, and certainly in soft robotics where robotics and mechatronics meet materials sciences and development, makes learning the principles of SOFA useful for researchers having very different backgrounds (mechanical and electrical engineering, materials or computer sciences, ...) , as well as for engineering students.

In the tutorial that follows this paper, it is shown how to install SOFA in Ubuntu. It is discussed how to select the right mesh size for the model and how to open and interact with the GUI of SOFA, illustrating on how to get access to the simulation graph, access to the simulation results and save them. Furthermore, it also presents how to code a scene in Python for SOFA, explaining step by step each building block of code, and some results are displayed showing also how some important parameters like gravity, stiffness, material constants affect the simulation. At the end of the tutorial how to control the animation of the scene in SOFA, how to handle the events through a controller and how to display the contact forces for a collision simulation between two objects are explained.

IX. Acknowledgment

This tutorial was validated during the 1st International Winter School on Smart Materials for Soft Robots, held December 12-17 2021 in the University of Cambridge, UK. The authors would like to thank the organizational team and participants of this event. This project was funded by Fonds Wetenschappelijk Onderzoek via the personal grants of Ellen Roels (1S84120N), and Seppe Terryn (1100416N), EU FET Open RIA Project SHERO (828818) and euRobin (101070596).

References

[1] S. Coyle, C. Majidi, P. LeDuc, and K. J. Hsia, "Bio-inspired soft robotics: Material selection, actuation, and design," *Extreme Mechanics Letters*, vol. 22, pp. 51–59, 2018.

[2] F. Iida and C. Laschi, "Soft robotics: Challenges and perspectives," *Procedia Computer Science*, vol. 7, pp. 99–102, 2011.

[3] L. Marechal, P. Balland, L. Lindenroth, F. Petrou, C. Kontovounisios, and F. Bello, "Toward a common framework and database of materials for soft robotics," *Soft robotics*, vol. 8, no. 3, pp. 284–297, 2021.

[4] J. M. Rossiter, "Robotics, smart materials, and their future impact for humans," in *The Next Step: Exponential Life*. BBVA, 2017.

[5] S. Terryn, J. Langenbach, E. Roels, J. Brancart, C. Bakkali-Hassani, Q.-A. Poutrel, A. Georgopoulou, T. G. Thuruthel, A. Safaei, P. Ferrentino *et al.*, "A review on self-healing polymers for soft robotics," *Materials Today*, vol. 47, pp. 187–205, 2021.

[6] J.-H. Youn, S. M. Jeong, G. Hwang, H. Kim, K. Hyeon, J. Park, and K.-U. Kyung, "Dielectric elastomer actuator for soft robotics applications and challenges," *Applied Sciences*, vol. 10, no. 2, p. 640, 2020.

[7] X. Huang, M. Ford, Z. J. Patterson, M. Zarepoor, C. Pan, and C. Majidi, "Shape memory materials for electrically-powered soft machines," *Journal of Materials Chemistry B*, vol. 8, no. 21, pp. 4539–4551, 2020.

[8] J. Walker, T. Zidek, C. Harbel, S. Yoon, F. S. Strickland, S. Kumar, and M. Shin, "Soft robotics: A review of recent developments of pneumatic soft actuators," in *Actuators*, vol. 9, no. 1. Multidisciplinary Digital Publishing Institute, 2020, p. 3.

[9] M. Hannan and I. Walker, "The'elephant trunk'manipulator, design and implementation," in *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No. 01TH8556)*, vol. 1. IEEE, 2001, pp. 14–19.

[10] M. Rolf and J. J. Steil, "Efficient exploratory learning of inverse kinematics on a bionic elephant trunk," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 6, pp. 1147–1160, 2013.

[11] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced robotics*, vol. 26, no. 7, pp. 709–727, 2012.

[12] M. Cianchetti, M. Calisti, L. Margheri, M. Kuba, and C. Laschi, "Bioinspired locomotion and grasping in water: the soft eight-arm octopus robot," *Bioinspiration & biomimetics*, vol. 10, no. 3, p. 035003, 2015.

[13] N. Elango and A. Faudzi, "A review article: investigations on soft materials for soft robot manipulations," *The International Journal of Advanced Manufacturing Technology*, vol. 80, no. 5, pp. 1027–1037, 2015.

[14] J. Zhou, X. Chen, J. Li, Y. Tian, and Z. Wang, "A soft robotic approach to robust and dexterous grasping," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2018, pp. 412–417.

[15] M. Thieffry, A. Kruszewski, O. Goury, T.-M. Guerra, and C. Duriez, "Dynamic control of soft robots," in *IFAC World congress*, 2017.

[16] A. A. Calderón, J. C. Ugalde, J. C. Zagal, and N. O. Pérez-Arancibia, "Design, fabrication and control of a multi-material-multi-actuator soft robot inspired by burrowing worms," in *2016 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2016, pp. 31–38.

[17] S. Terryn, E. Roels, J. Brancart, G. Van Assche, and B. Vanderborght, "Self-healing and high interfacial strength in multi-material soft pneumatic robots via reversible diels–alder bonds," in *Actuators*, vol. 9, no. 2. Multidisciplinary Digital Publishing Institute, 2020, p. 34.

[18] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018.

[19] D. Trivedi, A. Lotfi, and C. D. Rahn, "Geometrically exact models for soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 773–780, 2008.

[20] C. Della Santina, A. Bicchi, and D. Rus, "On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1001–1008, 2020.

[21] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, 2006.

[22] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik *et al.*, "Sofa: A multi-model framework for interactive physical simulation," in *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012, pp. 283–321.

[23] C. Duriez, E. Coevoet, F. Largilliere, T. Morales-Bieze, Z. Zhang, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, and J. Dequidt, "Framework for online simulation of soft robots with optimization-based inverse model," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2016, pp. 111–118.

[24] P. Ferrentino, S. K. Tabrizian, J. Brancart, G. Van Assche, B. Vanderborght, and S. Terryn, "Fea based inverse kinematic control on hyperelastic material characterisation of self healing soft robots," *IEEE Robotics and Automation Magazine*, 2021.

[25] E. Roels, S. Terryn, P. Ferrentino, and B. Vanderborght, "An interdisciplinary tutorial:
building your own self-healing soft finger with embedded sensor," *IEEE Robotics and Automation Magazine*, 2022.

[26] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt *et al.*,

"Software toolkit for modeling, simulation, and control of soft robots," *Advanced Robotics*, vol. 31, no. 22, pp. 1208–1224, 2017.

[27] L. Rognant, J.-M. Chassery, S. Goze, and J. Planes, "The delaunay constrained triangulation: the delaunay stable algorithms," in *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*. IEEE, 1999, pp. 147–152.

[28] P. Maur, "Delaunay triangulation in 3d," *Technical Report, Departmen. of Computer Science and Engineering*, 2002.

[29] C. Jamin, P. Alliez, M. Yvinec, and J.-D. Boissonnat, "Cgalmesh: a generic framework for delaunay mesh generation," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 4, pp. 1–24, 2015.

[30] D. P. Holland, E. J. Park, P. Polygerinos, G. J. Bennett, and C. J. Walsh, "The soft robotics toolkit: Shared resources for research and design," *Soft Robotics*, vol. 1, no. 3, pp. 224–230, 2014.

[31] M. Dubied, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, "Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation," *IEEE Robotics and Automation Letters*, pp. 1–1, 2022.

[32] Sofa framework website. INRIA. [Online]. Available: https://www.sofa-framework.org/community/doc/

[33] M. Tymkovych, O. Gryshkov, O. Avrunin, K. Selivanova, Y. Nosova, V. Mutsenko, N. Shushliapina, and B. Glasmacher, "Application of sofa framework for physics-based simulation of deformable human anatomy of nasal cavity," in *European Medical and Biological Engineering Conference*. Springer, 2020, pp. 112–120.

[34] Z. Liu, Z. Lu, and K. Karydis, "Sorx: A soft pneumatic hexapedal robot to traverse rough, steep, and unstable terrain," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 420–426.

[35] Y. Berranen, M. Hayashibe, B. Gilles, and D. Guiraud, "3d volumetric muscle modeling for real-time deformation analysis with fem," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 4863–4866.

[36] P. Ferrentino, A. López-Díaz, S. Terryn, J. Legrand, J. Brancart, G. Van Assche, E. Vázquez, A. Vázquez, and B. Vanderborght, "Quasi-static fea model for a multi-material soft pneumatic actuator in sofa," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7391–7398, 2022.

[37] O. Comas, Z. A. Taylor, J. Allard, S. Ourselin, S. Cotin, and J. Passenger, "Efficient nonlinear fem for soft tissue modelling and its gpu implementation within the open source framework sofa," in *International Symposium on Biomedical Simulation*. Springer, 2008, pp. 28–39.

[38] S. Terryn, J. Brancart, E. Roels, G. Van Assche, and B. Vanderborght, "Room temperature self-healing in soft pneumatic robotics: Autonomous self-healing in a diels-alder polymer network," *IEEE Robotics & Automation Magazine*, vol. 27, no. 4, pp. 44–55, 2020.

[39] C. Schumacher, E. Knoop, and M. Bächer, "Simulation-ready characterization of soft robotic materials," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3775–3782, 2020.