

Real-time dynamics of soft and continuum robots based on Cosserat rod models

The International Journal of
Robotics Research
2019, Vol. 38(6) 723–746
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364919842269
journals.sagepub.com/home/ijr
 SAGE

John Till¹, Vincent Aloï and Caleb Rucker¹

Abstract

The dynamic equations of many continuum and soft robot designs can be succinctly formulated as a set of partial differential equations (PDEs) based on classical Cosserat rod theory, which includes bending, torsion, shear, and extension. In this work we present a numerical approach for forward dynamics simulation of Cosserat-based robot models in real time. The approach implicitly discretizes the time derivatives in the PDEs and then solves the resulting ordinary differential equation (ODE) boundary value problem (BVP) in arc length at each timestep. We show that this strategy can encompass a wide variety of robot models and numerical schemes in both time and space, with minimal symbolic manipulation required. Computational efficiency is gained owing to the stability of implicit methods at large timesteps, and implementation is relatively simple, which we demonstrate by providing a short MATLAB-coded example. We investigate and quantify the tradeoffs associated with several numerical subroutines, and we validate accuracy compared with dynamic rod data gathered with a high-speed camera system. To demonstrate the method's application to continuum and soft robots, we derive several Cosserat-based dynamic models for robots using various actuation schemes (extensible rods, tendons, and fluidic chambers) and apply our approach to achieve real-time simulation in each case, with additional experimental validation on a tendon robot. Results show that these models capture several important phenomena, such as stability transitions and the effect of a compressible working fluid.

Keywords

Continuum robot, soft robot, forward dynamics

1. Introduction

1.1. Motivation

Slender elastic objects are becoming increasingly prevalent in robotics, e.g. in the study of soft (Rus and Tolley, 2015) and continuum (Burgner-Kahrs et al., 2015) manipulators, and in interactions with objects such as ropes, sutures, needles, and catheters (Tang et al., 2010). Forward dynamic models are essential in these applications as they provide a way to simulate and develop robot designs and control schemes and enable the study of dynamic events. Even for tasks where desired task accelerations are slow compared with robot dynamics, dynamic models are sometimes valuable. In some robots, elastic instability due to robot structure (Webster et al., 2009) or loading conditions (Till and Rucker, 2017b) can result in a sudden, dynamic movement of the robot even when actuator commands are slowly changing. Further, discontinuous changes in loading associated with tasks such as cutting, grabbing, and releasing objects may result in dynamic motion even for robots that behave quasi-statically in free space. Forward dynamics simulation allows one to study and anticipate these

phenomena. For such simulation-based research, design, and control, we believe that the competing goals of approximation accuracy and computational speed are the most important considerations of a potential forward dynamics approach for soft/continuum robots and slender elastic objects.

1.2. Related work

Modeling and simulating the dynamics of continuum/soft robots is an ongoing research problem. In Table 1 we give a non-exhaustive, representative sampling of journal publications on the topic of dynamic simulation of elastic rods and/or continuum/soft robots in three dimensions. For each method, we have listed the principle or method upon which

University of Tennessee, Knoxville, TN, USA

Corresponding author:

John Till, REACH Laboratory, The University of Tennessee, Knoxville, TN 37996, USA.

Email: JTill@vols.utk.edu

Table 1. Representative papers on forward dynamics simulation of spatial elastic rods and continuum/soft manipulators.

Reference	Dynamics Derivation	Geometric Representation	Time Integration	Application / Robot Type
Hadap (2006)	Newton–Euler	Discrete Rigid Bodies	DASPK DAE Solver	Graphics
Bergou et al. (2008)	Euler–Lagrange, Quasistatic Torsion	Discrete Framed Curve	Symplectic Euler, Constraint Projection	Graphics
Spillmann and Teschner (2009)	Energy Minimization with Penalty Forces	Discrete Framed Curve	Semi-Implicit Euler	Graphics
Bertails (2009)	Euler–Lagrange	Helical Segments	—	Graphics
Lang et al. (2011)	Lagrangian DAE	Staggered-Grid	DAE Solvers	Rods
Rucker and Webster III (2011)	Cosserat PDEs	Finite-Difference	e.g. RADAU5	Cable-driven
Rone and Ben-Tzvi (2014)	Virtual Power / Kane’s Method	Lax–Wendroff	Lax–Wendroff	Cable-driven
Umetani et al. (2014)	Constant Curvature Segments with Discrete Torsional Joints	—	—	—
Renda et al. (2014)	Gauss’s Principle of Least Constraint	Discrete Points and Point-Based Orientation	Pos.-Based Dynamics with Gauss–Siedel	Graphics
Falkenhahn et al. (2015)	Cosserat PDEs	Upwind Finite-Difference	ODE23	Cable-driven
Godage et al. (2015)	Euler–Lagrange	Constant Curvature Segments	—	Pneumatic
Godage et al. (2016)	Euler–Lagrange with CoG Lumped Mass	Modal Constant Curvature	ODE15s	Pneumatic
Marchese et al. (2016)	Euler–Lagrange	Constant Curvature Segments	—	Pneumatic
Kugelstadt and Schömer (2016)	Newton–Euler	Discrete Rigid Bodies	Semi-Implicit Euler, Constraint Solver	Graphics
Sadati et al. (2018)	Virtual Work / Ritz–Galerkin	Lagrange Polynomials	ODE113	Pneumatic
Ours	Cosserat/Kirchhoff PDEs	Any ODE Method (e.g. RK4), Solve BVP by Shooting	Any Implicit Method (e.g. BDF2)	Various Examples

the derivation of the dynamics was based, how the shape is represented or discretized in space, and the method of integration in time if specified in the paper. Robot-focused works such as Rone and Ben-Tzvi (2014), Falkenhahn et al. (2015), Godage et al. (2016, 2015), and Marchese et al. (2016) have predominantly started with a piecewise-constant curvature representation of robot shape, which is naturally suited to the serial segment structure and actuation of continuum robots. Application of variational principles then leads to the governing equations for the generalized coordinates (segment curvatures or arc parameters) analogous to conventional rigid-link dynamics equations. The coordinate accelerations are then obtained and integrated using standard explicit ODE solver packages. However, when external loads or inertial dynamics are considered, the constant-curvature representation may no longer be appropriate owing to the presence of torsion, shear, extension, or variations in curvature over an actuated segment. These limitations can be somewhat overcome by simply using a finer constant-curvature discretization as in Rone and Ben-Tzvi (2014), and torsion can be incorporated by adding a discrete torsional degree of freedom (DOF) at each segment Rone and Ben-Tzvi (2014) or by generalizing the approach to piecewise helical segments (Bertails, 2009). However, as the number of segments grows, the computation can scale badly (typically $O(n^3)$ (Godage et al., 2016)) because the mass matrix is dense owing to the use of a *minimal coordinate* representation. Some $O(n^3)$ algorithms are tractable for small n , which is the case for

the real-time modal dynamics of a three-segment, variable-length, piecewise-constant-curvature arm demonstrated by Godage et al. (2016), but scalability becomes important if one wishes to accurately capture higher-resolution curvature dynamics. Bertails (2009) adapted the $O(n)$ recursive articulated-body algorithm of Featherstone and Orin (2008) to helical segments.

Alternative *maximal coordinates* approaches (Bergou et al., 2008; Kugelstadt and Schömer, 2016; Lang et al., 2011; Spillmann and Teschner, 2009; Umetani et al., 2014) are primarily pursued in the computer graphics literature and typically represent the shape with a discrete geometry convention (e.g. a chain of points with adapted frames on the line segments between them). This requires resolving extensibility and shearability constraints numerically during the integration process, which has been done by projection (Bergou et al., 2008), sequential enforcement (Kugelstadt and Schömer, 2016; Umetani et al., 2014), methods for differential algebraic equations (Hadap, 2006), and penalty forces (Spillmann and Teschner, 2009; Tang et al., 2010).

Using penalty forces to approximately enforce extensibility and shearability constraints is equivalent to adopting a full Cosserat rod model instead of a Kirchhoff rod model. In this vein, Lang et al. (2011), Rucker and Webster III (2011), and Renda et al. (2014) used various numerical methods for solving the partial differential equations (PDEs) of a Cosserat rod, and Renda et al. (2016) has recently proposed a piecewise-constant strain representation that generalizes the constant-curvature framework to

include the Cosserat strains of torsion, shear, and extension. While such Cosserat models are essentially unconstrained and thus efficiently solvable (typically in $O(n)$ time), the dynamics are *stiff* owing to the high-frequency vibrational modes of the shear and extension DOF. This creates difficulties for efficient simulation of the slower modes of interest (bending–torsion) because the Courant–Friedrichs–Lewy stability condition puts a significantly restrictive upper bound on the timestep length that can be stably used for any explicit time integration method.

Thus, a main theme of the existing literature is the set of computational difficulties and tradeoffs associated with minimal coordinates versus constraint enforcement versus stiff dynamics, which are all different sides of the same problem. We aim to address these issues with a new implicit dynamics framework for Kirchhoff and Cosserat rod models.

1.3. Contributions

Our contribution in this paper is to derive, validate, and demonstrate a new approach for the numerical solution of the Cosserat and Kirchhoff PDEs with application to continuum and soft robots. We eschew the predominant approach of first establishing a geometric representation followed by dynamics derivation and explicit time integration. Instead, we start with the continuous set of Cosserat or Kirchhoff PDEs and then discretize the time derivatives using an implicit differentiation formula. This creates a continuous ordinary differential equation (ODE) in the spatial dimension and a boundary value problem (BVP) that we can solve using any numerical integration scheme in the spatial dimension and a shooting method to obtain the solution for the rod/robot state at each timestep. Variants of this approach have been explored in Gatti-Bono and Perkins (2002) and Lan and Lee (2006); Lan et al. (2009) for *planar* dynamics of fly-fishing lines and compliant mechanisms, but not rods or continuum robots in three dimensions. It is a commonly held view that Cosserat-rod-based models for continuum and soft robots are overly complex and too costly for real-time implementation. In contrast to this, we hope to demonstrate that our implicit approach to the Cosserat/Kirchhoff PDE models has the following advantages.

- *Adaptability*: Various model assumptions and numerical schemes are handled with minimal reformulation.
- *Real time*: Typical simulations can be computed faster than the corresponding physical systems evolve.
- *Scalability*: Computation time increases linearly with spatial resolution, which can be adapted online.
- *Accuracy*: High-order methods provide a good error versus effort tradeoff and exhibit low numerical damping.
- *Stability*: Large timesteps can be taken stably. Static solutions are simply obtained with an infinite timestep.

- *Consistency*: The continuous theory is approached as the resolution is increased, and bulk moduli are used.

Our preliminary work on this topic was presented by Till and Rucker (2017a), where we proposed and validated an implicit discretization approach for a dynamic Kirchhoff rod with negligible cross-section rotational inertia. Beyond this, our additional advances here include (1) derivation and application to the full Cosserat rod model including shear, extension, and rotational cross-section inertia, (2) a performance characterization of several variants of the method on a benchmark test problem, (3) additional experimental validation and speed characterization of the full Cosserat model and a tendon robot model, and (4) derivation and application of our method to three different continuum and soft robot designs, including a novel soft robot model, with real-time simulation demonstration in each case.

The development of the rod model in Sections 2.1–2.4 is foundational to the rest of the paper. A working MATLAB implementation of a dynamic cantilever problem is provided in Appendix B for the reader to confirm their understanding of the basic model before proceeding. After the theoretical rod model development, the remaining sections are largely independent, including each robot model in Section 5.

2. Cosserat models and methods

In this section, we provide details of our specific numerical approach to solving continuum robot PDEs accurately and stably in real time. In order to make the discussion of the discretization more concrete, we begin by presenting the PDE system describing a single Cosserat rod. Then we discuss the general implicit time discretization strategy for PDE problems, and demonstrate how this method may be applied to the dynamic Cosserat rod to obtain numerical solutions.

Toward providing a real-time simulation tool for the continuum and soft robotics community, we have emphasized the presentation of self-contained model equations and examples that can be implemented directly. Sample MATLAB code that implements our approach for a cantilevered elastic rod is provided in Appendix B, also included as a .m file in Extension 2. This code was tested for the MATLAB 2017a release version.

2.1. Cosserat rod PDEs

A related statement of Kirchhoff rod dynamics was developed by Till and Rucker (2017a). The derivation here is more general: we consider the effects of shear, extension, and rotational inertia of the rod cross-section. Our choice of notation is summarized in Table 2.

2.1.1. Dimensions. A Cosserat rod is approximated as a one-dimensional slender object so that all state variables are parametrized by a reference arc length parameter along

Table 2. Notation and definitions.

Symbol	Unit	Definition
s	m	Reference arclength
t	s	Time
\mathbf{p}	m	Global position in Cartesian coordinates
\mathbf{R}	—	Rotation matrix of material orientation
\mathbf{h}	—	Quaternion for the material orientation
\mathbf{n}	N	Internal force in the global frame
\mathbf{m}	Nm	Internal moment in the global frame
\mathbf{f}	N/m	Distributed force in the global frame
\mathbf{l}	Nm/m	Distributed moment in the global frame
\mathbf{v}	—	Rate of change of position with respect to arclength in the local frame
\mathbf{u}	1/m	Curvature vector in the local frame
\mathbf{q}	m/s	Velocity in the local frame
$\boldsymbol{\omega}$	1/s	Angular velocity in the local frame
A	m ²	Cross-sectional area
ρ	kg/m ³	Material density
\mathbf{J}	m ⁴	Second mass moment of inertia tensor
\mathbf{v}^*	—	Value of \mathbf{v} when $\mathbf{n} = \mathbf{v}_t = \mathbf{0}$. For a straight rod $\mathbf{v}^* = \mathbf{e}_3$.
\mathbf{u}^*	1/m	Local curvature when $\mathbf{m} = \mathbf{u}_t = \mathbf{0}$. For a straight rod $\mathbf{u}^* = \mathbf{0}$.
\mathbf{K}_{se}	N	Stiffness matrix for shear and extension $\mathbf{K}_{se} = \begin{bmatrix} GA & 0 & 0 \\ 0 & GA & 0 \\ 0 & 0 & EA \end{bmatrix}$
\mathbf{K}_{bt}	Nm ²	Stiffness matrix for bending and twisting $\mathbf{K}_{bt} = \begin{bmatrix} EI_{xx} & 0 & 0 \\ 0 & EI_{yy} & 0 \\ 0 & 0 & GI_{zz} \end{bmatrix}$
E	Pa	Young's modulus
G	Pa	Shear modulus
\mathbf{B}_{se}	N s	Damping matrix for shear and extension
\mathbf{B}_{bt}	Nm ² s	Damping matrix for bending and twisting
\mathbf{C}	kg/m ²	Square law drag coefficient matrix
\mathbf{g}	m/s ²	Gravitational acceleration vector
\mathbf{e}_3	—	Unit vector; $\mathbf{e}_3 = [0 \ 0 \ 1]^T$
α	—	Coefficient of the BDF- α method
c_i	1/t	Implicit difference coefficient for a state at $t - i\delta t$
d_i	none	Implicit difference coefficient for a time derivative at $t - i\delta t$
\mathbf{y}	misc.	General ODE state vector
\mathbf{z}	misc.	General vector with relevant time derivatives, but elements not in \mathbf{y}
$(\cdot)^b$		Local-frame representation of variable, e.g. $\mathbf{n}^b = \mathbf{R}^T \mathbf{n}$
$\overset{h}{(\cdot)}$		Historical part of time derivative, e.g. $\mathbf{q}_t \approx c_0 \mathbf{q} + \overset{h}{\mathbf{q}}$
$\hat{\cdot}$ or $(\cdot)^\wedge$		Mapping from \mathbb{R}^3 to $\mathfrak{se}(3)$, e.g. $\hat{\mathbf{u}} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$
$(\cdot)^\vee$		Mapping from $\mathfrak{se}(3)$ to \mathbb{R}^3 , $\hat{\mathbf{u}}^\vee = \mathbf{u}$

the undeformed rod centerline $s \in [0 \ L] \subset \mathbb{R}$ and by the time $t \in \mathbb{R}$. We will use the convention of denoting partial

derivatives by subscripting s or t , for example $\partial \mathbf{y} / \partial s \equiv \mathbf{y}_s$. Note that s is merely a coordinate used to identify material points, and in general, L does not correspond to the member's deformed arc length since axial elongation and compression modes are included in the model. The actual deformed length of the member would be obtained by integrating the axial strain over the interval $[0 \ L]$ and adding the result to L . This makes the method suitable for describing variable-length robots.

2.1.2. State variables. The rod has a centerline $\mathbf{p}(t, s) \in \mathbb{R}^3$ and orientation represented by a rotation matrix $\mathbf{R}(t, s) \in \text{SO}(3)$. The partial derivative of \mathbf{p} with respect to arclength in the local frame is denoted by \mathbf{v} , that is $\mathbf{v} := \mathbf{R}^T \mathbf{p}_s$. Likewise there is a term for the curvature in the local frame $\mathbf{u}(t, s) := (\mathbf{R}^T \mathbf{R}_s)^\vee$, where the $(\cdot)^\vee$ operator maps $\mathfrak{so}(3)$ to \mathbb{R}^3 (Murray et al., 1994). There are also analogous terms for partial derivatives in time $\mathbf{q} := \mathbf{R}^T \mathbf{p}_t$ and $\boldsymbol{\omega} := (\mathbf{R}^T \mathbf{R}_t)^\vee$. The internal force in the global frame is represented by $\mathbf{n}(t, s) \in \mathbb{R}^3$, with the sign convention that $\mathbf{n}(t, s)$ represents the force which the material at $\mathbf{p}(t, s + ds)$ exerts on the material at $\mathbf{p}(t, s - ds)$ for infinitesimal ds . The internal moment $\mathbf{m} \in \mathbb{R}^3$ is defined similarly.

2.1.3. General parameters and inputs. The material density is represented by $\rho \in \mathbb{R}^+$ and the cross-sectional area is $A \in \mathbb{R}^+$. The rotational inertia matrix is $\mathbf{J} \in \mathbb{R}^{3 \times 3}$. We consider cases where \mathbf{p} coincides with the cross-sectional centroid and with constant density over the cross-section, in which case \mathbf{J} is a diagonal matrix

$$\mathbf{J} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

For a circular rod $I_{xx} = I_{yy} = \pi r^4 / 4$ with radius $r \in \mathbb{R}^+$, and $I_{zz} = I_{xx} + I_{yy}$.

There is also a general distributed force $\mathbf{f}(t, s) \in \mathbb{R}^3$ acting on the rod. This may be independent of the rod state, for example a rod in free space at rest has $\mathbf{f} = \rho \mathbf{A} \mathbf{g}$, where $\mathbf{g} \in \mathbb{R}^3$ is the gravitational acceleration vector. Other scenarios dictate that \mathbf{f} is dependent on the state, e.g. including viscous damping results in $\mathbf{f} = -\mathbf{R} \mathbf{C} \mathbf{q}$ with some damping coefficient matrix \mathbf{C} . In addition to the distributed force, there is a distributed moment $\mathbf{l}(t, s) \in \mathbb{R}^3$.

2.1.4. PDEs. We rely on equations (23) and (25)–(27) from Rucker and Webster III (2011) as the initial statement of the PDE system. These equations are consistent with the derivation in chapter eight of Antman's classic text (Antman, 2005). We express the equations in the global frame here for compactness, but equivalent local-frame formulations are often used, and our method applies to either local or global PDE expressions. The dynamics of an elastic rod are then given by the PDE set

$$\begin{aligned}
p_s &= Rv, & p_t &= Rq \\
R_s &= R\hat{u}, & R_t &= R\hat{\omega} \\
n_s &= \rho A R(\hat{\omega}q + q_t) - f \\
m_s &= \partial_t(R\rho J\omega) - \hat{p}_s n - l \\
q_s &= v_t - \hat{u}q + \hat{\omega}v \\
\omega_s &= u_t - \hat{u}\omega
\end{aligned} \tag{1}$$

where $\hat{\cdot}$ represents the mapping from \mathbb{R}^3 to $\mathfrak{so}(3)$. This is defined by

$$\hat{a} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2}$$

and satisfies $(\hat{a})^\vee = a$.

We can expand the distributed force term f to explicitly consider various forces. In subsequent sections, we will apply actuation forces within f and l (as well as in the boundary conditions) in order to model various robots, but here as an example we can simply consider the member's self-weight and square law drag air resistance so that

$$f = \rho A g - RCq \odot |q| + \bar{f} \tag{3}$$

where the Hadamard product \odot performs element-wise multiplication, so that

$$q \odot |q| = [q_1^2 \operatorname{sgn}(q_1) \quad q_2^2 \operatorname{sgn}(q_2) \quad q_3^2 \operatorname{sgn}(q_3)]^T$$

Here \bar{f} contains any remaining forces not explicitly considered yet. In many cases the rotational inertia term may reasonably be neglected, but we include it for generality as its influence will be more significant for less-slender soft robots. The rotational momentum derivative term is expanded by

$$\begin{aligned}
\partial_t(R\rho J\omega) &= R_t\rho J\omega + R\rho J\omega_t \\
&= \rho R(\hat{\omega}J\omega + J\omega_t)
\end{aligned}$$

The system (1) has independent variables v and u ; an appropriate material constitutive law must be chosen to relate these to internal loading. Here we use a linear elastic law with material damping

$$\begin{aligned}
n &= R[K_{se}(v - v^*) + B_{se}v_t] \\
m &= R[K_{bt}(u - u^*) + B_{bt}u_t]
\end{aligned} \tag{4}$$

where the “se” subscript refers to shear and extension and “bt” refers to bending and torsion. The matrices $B_{se}, B_{bt} \in \mathbb{R}^{3 \times 3}$ contain coefficients for Kelvin–Voigt-type viscous damping as described by Linn et al. (2012). The $K_{se}, K_{bt} \in \mathbb{R}^{3 \times 3}$ matrices are stiffness coefficient matrices, which are determined by the material properties and cross-sectional geometry. Note that a rod cross-section does not need to be circular; it is only required to be slender. For a symmetric rod with isotropic material properties, the stiffness matrices are

$$K_{se} = \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & E \end{bmatrix} A, \quad K_{bt} = \begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & G \end{bmatrix} J$$

where $E \in \mathbb{R}^+$ and $G \in \mathbb{R}^+$ are the Young's and shear moduli. The variables $v^* \in \mathbb{R}^3$ and $u^* \in \mathbb{R}^3$ represent the undeformed reference shape the rod will take when it is unstressed and resting. If we ignore the possibility of plastic deformation, then these parameters are constant with respect to time, although they may vary with arc length. An initially straight rod has $v^* = e_3 := [0 \ 0 \ 1]^T$ and $u^* = 0$. For ease of reference we summarize the notation in Table 2.

2.2. Semi-discretization in time for general PDE solution

An ODE in the spatial dimension can be obtained by replacing all the time derivative terms with an implicit differentiation formula for them. For discretized variables we denote the time index with a left superscript, e.g. for any subset of state variables $y \in \mathbb{R}^n$, $y(t_i, s) = {}^{(i)}y(s)$. Any general implicit differentiation formula for a first derivative can be written in the form

$$\begin{aligned}
{}^{(i)}y_t &\approx c_0 {}^{(i)}y + \sum_{k=1}^{\infty} [c_k {}^{(i-k)}y + d_k {}^{(i-k)}y_t] \\
&:= c_0 {}^{(i)}y + {}^{(i)}y^h
\end{aligned} \tag{5}$$

The only term in ${}^{(i)}y_t$ corresponding to time t_i is $c_0 {}^{(i)}y$. Here ${}^{(i)}y^h$ is defined as the sum of all remaining terms that rely on the past history of y . In general, we use this notation for the history-dependent part of a variable's derivative approximation. Lumping all the history-dependent terms together is useful for the shooting method we employ later, because (1) the separate terms for current and previous state allow one to decouple the details of the implicit time discretization from the spatial shooting method, and (2) each shot has a new value for the current state y , but the history term y^h is common for all shots taken at a single timestep.

In applying (5), one could choose from a variety of implicit differentiation formulas, such as backward-Euler (which is first order and exhibits high numerical damping), the trapezoidal rule (which is second order with low damping and error, but is only marginally stable for conservative systems), or any of the family of backward differentiation formulas (denoted BDF#, e.g. BDF2 denotes the second-order formula). In this paper, we explore and compare several different choices for the particular implicit differentiation formula used. In addition to the methods mentioned above, we will evaluate a less well-known approach called the BDF- α method (Celaya and Jos, 2013), which for a timestep δt is $O(\delta t^2)$ accurate. This is described by

$${}^{(i)}y_t = c_0 {}^{(i)}y + c_1 {}^{(i-1)}y + c_2 {}^{(i-2)}y + d_1 {}^{(i-1)}y_t$$

$$:= c_0 {}^{(i)}\mathbf{y} + {}^{(i)}\mathbf{y}^{\frac{h}{\delta t}}$$

where

$$c_0 = (1.5 + \alpha) / [\delta t(1 + \alpha)]$$

$$c_1 = -2 / \delta t$$

$$c_2 = (0.5 + \alpha) / [\delta t(1 + \alpha)]$$

$$d_1 = \alpha / (1 + \alpha)$$

The variable $\alpha \in [-0.5, 0] \subset \mathbb{R}$ is a parameter that may be assigned and that essentially interpolates between the trapezoidal method and BDF2. The trapezoidal method is obtained for $\alpha = -0.5$ and the second-order backward differentiation formula BDF2 is obtained for $\alpha = 0$. Using this approximation, the solution to the ODE in s at t_i is only dependent on the solutions to previous ODEs at times t_{i-1} and t_{i-2} .

2.3. Application to a rod PDE system

Following discretization of the time derivative we have $\mathbf{v}_t \approx c_0 \mathbf{v} + \mathbf{v}^{\frac{h}{\delta t}}$ and $\mathbf{u}_t \approx c_0 \mathbf{u} + \mathbf{u}^{\frac{h}{\delta t}}$, so the linear constitutive relationship can be solved as

$$\begin{aligned} \mathbf{v} &= (\mathbf{K}_{se} + c_0 \mathbf{B}_{se})^{-1} \left(\mathbf{R}^T \mathbf{n} + \mathbf{K}_{se} \mathbf{v}^* - \mathbf{B}_{se} \mathbf{v}^{\frac{h}{\delta t}} \right) \\ \mathbf{u} &= (\mathbf{K}_{bt} + c_0 \mathbf{B}_{bt})^{-1} \left(\mathbf{R}^T \mathbf{m} + \mathbf{K}_{bt} \mathbf{u}^* - \mathbf{B}_{bt} \mathbf{u}^{\frac{h}{\delta t}} \right) \end{aligned} \quad (6)$$

Our implicit time discretization is particularly convenient since the step from (4) to (6) is the only symbolic manipulation required after discretizing, whereas spatially discretized rod models tend to have an involved process of solving for node accelerations (Bergou et al., 2008; Spillmann and Teschner, 2009). For this reason, our method is especially useful for applications where the underlying equations are subject to change (e.g. continuum robotics).

Substituting the distributed force and rotational inertia terms into the system, the PDE set is reduced to the following system of ODEs in arc length:

$$\begin{aligned} \mathbf{p}_s &= \mathbf{R} \mathbf{v} \\ \mathbf{R}_s &= \mathbf{R} \hat{\mathbf{u}} \\ \mathbf{n}_s &= \mathbf{R} [\rho A (\hat{\boldsymbol{\omega}} \mathbf{q} + \mathbf{q}_t) + \mathbf{C} \mathbf{q} \odot |\mathbf{q}|] - \rho A \mathbf{g} - \bar{\mathbf{f}} \\ \mathbf{m}_s &= \rho \mathbf{R} (\hat{\boldsymbol{\omega}} \mathbf{J} \boldsymbol{\omega} + \mathbf{J} \boldsymbol{\omega}_t) - \hat{\mathbf{p}}_s \mathbf{n} - \mathbf{l} \\ \mathbf{q}_s &= \mathbf{v}_t - \hat{\mathbf{u}} \mathbf{q} + \hat{\boldsymbol{\omega}} \mathbf{v} \\ \boldsymbol{\omega}_s &= \mathbf{u}_t - \hat{\mathbf{u}} \boldsymbol{\omega} \end{aligned} \quad (7)$$

where \mathbf{u} , \mathbf{v} , and all time derivative terms above are computed algebraically from state variables at the present and/or previous timesteps by the following equations,

$$\begin{aligned} \mathbf{v} &= (\mathbf{K}_{se} + c_0 \mathbf{B}_{se})^{-1} (\mathbf{R}^T \mathbf{n} + \mathbf{K}_{se} \mathbf{v}^* - \mathbf{B}_{se} \mathbf{v}^{\frac{h}{\delta t}}) \\ \mathbf{u} &= (\mathbf{K}_{bt} + c_0 \mathbf{B}_{bt})^{-1} (\mathbf{R}^T \mathbf{m} + \mathbf{K}_{bt} \mathbf{u}^* - \mathbf{B}_{bt} \mathbf{u}^{\frac{h}{\delta t}}) \end{aligned}$$

$$\mathbf{v}_t = c_0 \mathbf{v} + \mathbf{v}^{\frac{h}{\delta t}}$$

$$\mathbf{u}_t = c_0 \mathbf{u} + \mathbf{u}^{\frac{h}{\delta t}}$$

$$\mathbf{q}_t = c_0 \mathbf{q} + \mathbf{q}^{\frac{h}{\delta t}}$$

$$\boldsymbol{\omega}_t = c_0 \boldsymbol{\omega} + \boldsymbol{\omega}^{\frac{h}{\delta t}}$$

The initial conditions for the PDE system will often be given as some steady-state solution to the system, which is defined by the following ODEs in arc length:

$$\begin{aligned} \mathbf{v} &= \mathbf{v}^* + \mathbf{K}_{se}^{-1} \mathbf{R}^T \mathbf{n} \\ \mathbf{u} &= \mathbf{u}^* + \mathbf{K}_{bt}^{-1} \mathbf{R}^T \mathbf{m} \\ \mathbf{p}_s &= \mathbf{R} \mathbf{v} \\ \mathbf{R}_s &= \mathbf{R} \hat{\mathbf{u}} \\ \mathbf{n}_s &= -\rho A \mathbf{g} - \bar{\mathbf{f}} \\ \mathbf{m}_s &= -\hat{\mathbf{p}}_s \mathbf{n} - \mathbf{l} \end{aligned} \quad (8)$$

2.3.1. Orientation as a quaternion. Truncation error from numerically integrating $\mathbf{R}_s = \mathbf{R} \hat{\mathbf{u}}$ can result in $\mathbf{R}(s) \notin \text{SO}(3)$. This rotation matrix degeneration may be acceptably small (since the integration occurs only over arc length and not over time), but the issue can be avoided altogether by integrating the orientation in quaternion form (Rucker, 2018). We may replace the rotation matrix differential equation with the equivalent equation for a quaternion $\mathbf{h} \in \mathbb{H}$. We use the notation $\mathbf{h} = h_1 + h_2 i + h_3 j + h_4 k$ so that

$$\mathbf{h}_s = \frac{1}{2} \begin{bmatrix} 0 & -u_1 & -u_2 & -u_3 \\ u_1 & 0 & u_3 & -u_2 \\ u_2 & -u_3 & 0 & u_1 \\ u_3 & u_2 & -u_1 & 0 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} \quad (9)$$

We can continue to use a rotation matrix in any required calculations, where the orthonormal rotation \mathbf{R} is calculated from the quaternion by

$$\mathbf{R}(\mathbf{h}) = \mathbf{I} + \frac{2}{h^T \mathbf{h}} \begin{bmatrix} -h_3^2 - h_4^2 & h_2 h_3 - h_4 h_1 & h_2 h_4 + h_3 h_1 \\ h_2 h_3 + h_4 h_1 & -h_2^2 - h_4^2 & h_3 h_4 - h_2 h_1 \\ h_2 h_4 - h_3 h_1 & h_3 h_4 + h_2 h_1 & -h_2^2 - h_3^2 \end{bmatrix} \quad (10)$$

Note that in this approach the quaternion magnitude does not need to be constrained to unity. The normalization is already embedded in (10), and (9) is correct and consistent with (10) even for non-unit quaternions.

2.3.2. Kirchhoff case. Oftentimes the shear and extension strains have a negligible effect compared with bending and torsion, particularly for slender robots since the bending stiffness is proportional to r^4 while the shear stiffness is proportional to r^2 . The absence of shear and extension

strains implies that $\mathbf{v} = \mathbf{e}_3$ and $\mathbf{v}_t = \mathbf{0}$. After neglecting these strain modes, the system in (7) reduces to

$$\begin{aligned} \mathbf{p}_s &= R\mathbf{e}_3 \\ \mathbf{R}_s &= R\hat{\mathbf{u}} \\ \mathbf{n}_s &= R[\rho A(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t) + \mathbf{C}\mathbf{q} \odot |\mathbf{q}|] - \rho A\mathbf{g} - \bar{\mathbf{f}} \\ \mathbf{m}_s &= \rho R(\hat{\boldsymbol{\omega}}\mathbf{J}\boldsymbol{\omega} + \mathbf{J}\boldsymbol{\omega}_t) - \hat{\mathbf{p}}_s\mathbf{n} - \mathbf{l} \\ \mathbf{q}_s &= -\hat{\mathbf{u}}\mathbf{q} + \hat{\boldsymbol{\omega}}\mathbf{e}_3 \\ \boldsymbol{\omega}_s &= \mathbf{u}_t - \hat{\mathbf{u}}\boldsymbol{\omega} \end{aligned} \quad (11)$$

where

$$\begin{aligned} \mathbf{u} &= (\mathbf{K}_{bt} + c_0\mathbf{B}_{bt})^{-1} \left(\mathbf{R}^T\mathbf{m} + \mathbf{K}_{bt}\mathbf{u}^* - \mathbf{B}_{bt}\hat{\mathbf{u}} \right) \\ \mathbf{u}_t &= c_0\mathbf{u} + \hat{\mathbf{u}} \\ \mathbf{q}_t &= c_0\mathbf{q} + \hat{\mathbf{q}} \\ \boldsymbol{\omega}_t &= c_0\boldsymbol{\omega} + \hat{\boldsymbol{\omega}} \end{aligned}$$

In addition, for the slower modes of interest, the rotational inertia of the cross-section $\rho\mathbf{J}$ often has a negligible effect compared with that of the linear mass distribution ρA , which can reduce complexity further (Till and Rucker, 2017a). We will primarily rely on the full Cosserat rod equations in (7) throughout the paper, but the Kirchhoff assumption can be a useful step to reduce the problem's complexity and improve computational efficiency in some scenarios.

2.4. ODE solution with the shooting method

When considering a specific problem, the ODE system in (7) is accompanied by a set of boundary conditions. For instance, a cantilevered rod has known values of $\mathbf{p}(t_i, 0) = \mathbf{p}_0$, $\mathbf{h}(t_i, 0) = \mathbf{h}_0$, and $\mathbf{q}(t_i, 0) = \boldsymbol{\omega}(t_i, 0) = \mathbf{0}$. If the rod has a length L and no forces at the free end, the distal constraints are $\mathbf{n}(t_i, L) = \mathbf{m}(t_i, L) = \mathbf{0}$. Such BVPs can be solved iteratively by guessing the unknown initial values, for example $\mathbf{n}(t_i, 0)$ and $\mathbf{m}(t_i, 0)$ for the cantilever problem. The guessed values are then iteratively updated by a chosen nonlinear optimization routine in order to reduce the residual error of the distal boundary conditions to zero. This aspect of the shooting method can be implemented by a Levenberg–Marquardt algorithm with an adaptive damping coefficient as described by Till et al. (2015) or by a trust region dogleg method (Nocedal and Wright, 2006). The shooting method process is depicted in Figure 1.

We hope to decrease the burden on the reader by including an example code to solve this problem, which is presented in Appendix B. The program begins with declarations of the various physical and numerical independent parameters, then sets up boundary conditions for the cantilever problem and calculates dependent parameters. The rod starts out in a straight configuration and then falls under the influence of its own weight. Gravitational acceleration is increased by a factor of 10 to achieve an

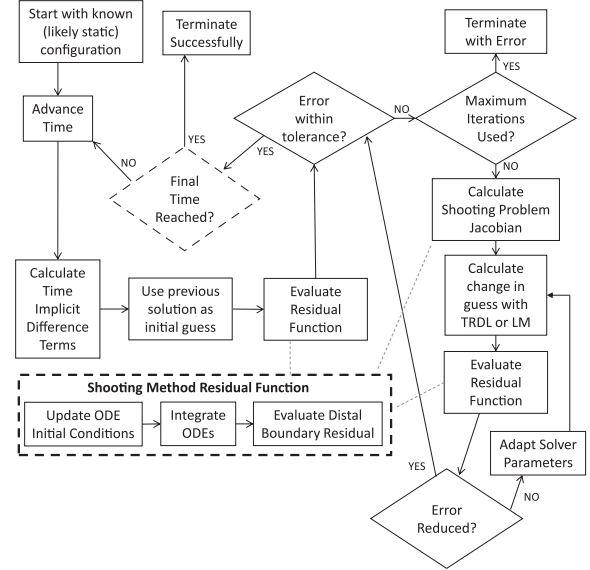


Fig. 1. The simulation loop is illustrated. The leftmost loop advances the simulation time. Many of the details of the shooting method on the right may be handled by a prepackaged solver.

exaggerated movement. After setting up initial conditions for the dynamic problem, the program reaches the main simulation loop, where it iteratively solves the dynamic problem and visualizes the result. The discretization uses BDF2 for the implicit approximation of time derivatives, MATLAB's "fsolve" for the shooting method solver, and either forward Euler or the classical fourth-order Runge–Kutta (RK4) method for spatial integration.

As described by Till and Rucker (2017a), at very small timesteps implicit discretization inherently has a “catastrophic cancelation” loss of floating-point precision caused by subtracting nearly equal numbers. For example, the backward Euler method relies on the calculation

$$\mathbf{y}_t(t) \approx [\mathbf{y}(t) - \mathbf{y}(t - \delta t)] / \delta t$$

but for continuous \mathbf{y} we have

$$\lim_{\delta t \rightarrow 0} \mathbf{y}(t) - \mathbf{y}(t - \delta t) = \mathbf{0}$$

Despite this fundamental issue, we have observed that the problem is well posed for reasonably small timesteps, and of course explicit methods have the opposite problem of instability at large timesteps. The modified shooting method (Holsapple et al., 2003) is used for the simulations in Section 4 to aid solver convergence, particularly for the numerically challenging impulse problem. This method aims to prevent situations where the shooting BVP initial conditions cause the integrated ODE state variables to become unbounded. In all other sections we use standard simple shooting.

There is significant freedom in the choice of spatial numerical integration method used in the ODE shooting

problem. For the simulations herein, we consider and compare both Euler's first-order method and the classic RK4 integration, but any method is able to be used. For instance, we have also used the fourth-/fifth-order Dormand–Prince adaptive stepsize solver implemented by MATLAB's "ode45." Using a fourth-order method in space should provide excellent error scaling and computational efficiency, and it will produce highly accurate steady-state solutions, but it requires some extra effort to evaluate ${}^{(i)}\mathbf{y}(s_{j+0.5})$ since this lies between the grid points of previous ODE solutions. For the purposes of this paper we use simple linear interpolation, but we note that hermite interpolation or a higher-order interpolant could be used if a dense output solver is used.

2.5. Generalized PDE form

With the rod example in mind, we could also consider a wider class of problems that may be solved by this implicit time discretization method. A general form of PDEs that is solvable by our approach and includes the various robot models we consider herein as special cases, is

$$\begin{aligned} \mathbf{A}\mathbf{y}_s + \mathbf{B}\mathbf{y}_{st} &= \mathbf{f}_1 \\ \mathbf{C}\mathbf{z} + \mathbf{D}\mathbf{z}_t &= \mathbf{f}_2 \end{aligned} \quad (12)$$

where \mathbf{y} is a set of variables for which spatial derivatives appear, and \mathbf{z} is a set of variables for which the spatial derivative \mathbf{z}_s does not appear, and all the matrices are square. Here \mathbf{A} , \mathbf{B} , and \mathbf{f}_1 are all functions of $s, t, \mathbf{y}, \mathbf{y}_t, \mathbf{z}$, and \mathbf{z}_t . In the second equation, \mathbf{C} , \mathbf{D} , and \mathbf{f}_2 are all functions of s, t, \mathbf{y} , and \mathbf{y}_t . After our implicit discretization, the solution for \mathbf{y}_s which we will integrate over s is given by

$$\begin{aligned} \mathbf{y}_s &= (\mathbf{A} + c_0\mathbf{B})^{-1}(\mathbf{f}_1 - \mathbf{B}\mathbf{y}_s^h) \\ \mathbf{z} &= (\mathbf{C} + c_0\mathbf{D})^{-1}(\mathbf{f}_2 - \mathbf{D}\mathbf{z}^h) \end{aligned}$$

where the inverted matrices must be non-singular. Although it would be possible to solve some problems where the first equation is nonlinear in \mathbf{y}_s or \mathbf{y}_{st} , or where the second equation is nonlinear in \mathbf{z} or \mathbf{z}_t , it would be difficult to generalize such solutions. The various examples of continuum robots in this paper fit the form of (12), and we assert that the method has a wide range of applications.

2.6. Inverse dynamics

Although this paper focuses on the forward dynamic simulation problem, we would like to note that the Kirchhoff rod equations naturally express a continuous version of the conventional recursive Newton–Euler scheme for rigid-body chains. The curvature vector $\mathbf{u}(s)$ is analogous to the set of joint variables, and the internal force $\mathbf{n}(s)$ and moment $\mathbf{m}(s)$ are analogous to the forces and moments across joints. Thus, if we are given a desired curvature acceleration $\mathbf{u}_t(s)$, we can integrate the kinematic equations from base to tip to

find all spatial velocities and accelerations, then integrate the required internal forces and moments from tip to base. This analogy has been previously made by Boyer et al. (2012) and used to simulate snake-like locomotion gaits for hyperredundant robots. However, it is unclear how to use this continuous Newton–Euler procedure to control most continuum manipulators because the analogy entails the idealization that one has direct control over $\mathbf{m}(s)$ over $s \in [0, L]$. In practice, continuum robots are controlled by a finite set of actuators that can only influence the internal moment function in finite-dimensional ways, and not every desired curvature acceleration will be achievable by some actuator action. So practical approaches for the inverse dynamics control of continuum manipulators will likely need to adapt approaches from the control of underactuated systems.

2.7. Software implementation

Aside from the example in Appendix B, the simulations are written in C++ to achieve our goal of real-time performance. The only model we took the effort to parallelize was the continuum Stewart–Gough robot model. All models could potentially benefit from parallelism because approximating the columns of the shooting problem Jacobian is a major source of computational effort.

Our code relies on the matrix library "eigen" (Guennebaud et al., 2010) to implement the nonlinear solver and implicit time difference. In our experience, the main chokepoint of simulations is the numerical ODE integration routine. We have invested some effort to statically evaluate expressions in the integration routines when possible to complement the efforts of the optimizing compiler. Beyond the MATLAB example in Appendix B, additional code is available online at <https://github.com/JohnDTill/ContinuumRobotExamples>.

3. Numerical analysis of benchmark case

In this section we compare the properties and performance of various implicit discretization schemes and numerical integration methods on a benchmark problem for a cantilevered rod. Our comparison table includes spatial integration using Euler's method and RK4, and implicit time discretization with backward Euler, BDF2, BDF3, and the BDF- α method with various choices of α . This section is largely independent of the other sections, although understanding the implications of choosing a numerical scheme is useful when implementing and analyzing simulations.

Artificial numerical damping is a primary factor affecting accuracy for implicit methods. We compare the schemes listed above in terms of their stability, runtime, and numerical damping by simulating a cantilevered rod with no physical damping ($\mathbf{B}_{se} = \mathbf{B}_{bt} = \mathbf{C} = 0$). At $t = 0^-$, the rod has a load attached at the tip and is in static equilibrium. At $t = 0^+$ this load is removed from the rod.

Table 3. Comparison of numerical methods for Cosserat equations.

Spatial Integration Method	Time Discretization Method	Numerical Damping Time Constant: $\Delta t = 5$ ms	Runtime: $\Delta t = 5$ ms Timeframe = 1 s Spatial Steps = 150	Minimum Stable Time Step
Euler	Backward Euler	0.13 s	0.35 s	0.8 ms
	BDF2	3.86 s	0.46 s	1.3 ms
	BDF- α , $\alpha = -0.2$	7.57 s	0.48 s	1.5 ms
	BDF- α , $\alpha = -0.48$	161.8 s	0.56 s	3.8 ms
	Trapezoidal Method	*NA	0.62 s	4.2 ms
	BDF3	Unstable	Unstable	48.2 ms
Fourth-Order Runge Kutta	Backward Euler	0.13 s	1.31 s	0.8 ms
	BDF2	3.88 s	1.83 s	1.4 ms
	BDF- α , $\alpha = -0.2$	7.63 s	1.88 s	1.6 ms
	BDF- α , $\alpha = -0.48$	201.6 s	2.23 s	4.1 ms
	Trapezoidal Method	NA	2.46 s	4.4 ms
	BDF3	Unstable	Unstable	48.4 ms

*No discernible numerical damping.

The numerical damping exhibited by a method can be measured by calculating the total amount of energy present in the system. Since there is no physical damping in the model, the system is conservative, and any energy decay will be solely due to the numerical damping. We can then use the decay rate to quantify the amount of numerical damping. The energy in this benchmark test problem is composed of kinetic, elastic potential, and gravitational potential terms so that

$$\begin{aligned}
 E(t) = & \int_0^L \left[\frac{1}{2} (\mathbf{u} - \mathbf{u}^*)^T \mathbf{K}_{bt} (\mathbf{u} - \mathbf{u}^*) \right. \\
 & + \frac{1}{2} (\mathbf{v} - \mathbf{v}^*)^T \mathbf{K}_{se} (\mathbf{v} - \mathbf{v}^*) + \frac{1}{2} \rho A \mathbf{q}^T \mathbf{q} \\
 & \left. + \frac{1}{2} \boldsymbol{\omega}^T \rho \mathbf{J} \boldsymbol{\omega} - \rho A \mathbf{g}^T \mathbf{p} \right] ds
 \end{aligned}$$

Over the long term, the energy decay is well approximated as a first-order exponential function $E(t) = E_0 e^{-2t/\tau}$. (By analogy with a simple harmonic oscillator, the energy decay time constant is twice the time constant τ associated with the position oscillation decay). We thus use the time constant τ to quantify the amount of numerical damping of a method. A large, positive time constant is desirable in this test. In this benchmark example, 1 second of simulated time involves 25 oscillations of the lowest frequency mode in the rod. Thus, if a method exhibits a numerical damping time constant of $\tau = 32$ seconds in this example, we would only see an energy loss of 0.25% per low-frequency cycle. Depending on the desired accuracy, an engineer can choose an appropriate timestep to reduce artificial energy loss to an acceptable level. The exponential fit was found using MATLAB's "fit" function. The methods are also classified by simulation runtime and "minimum stable timestep," which is defined as the minimum timestep for which the solution converges with a positive time constant.

The rod parameters were $E = 207$ GPa, $G = 79$ GPa, $L = 0.4$ m, $r = 1$ mm, and $\rho = 8,000$ kg/m³, which are consistent with spring steel. The force applied at the tip was 0.5 N. The total time frame simulated was 1 second (which corresponds to 25 cycles of vibration in the rod). A total of 150 discrete points were used for the spatial integration. We used a common timestep of 5 ms to compare the amounts of numerical damping in each method. The comparison was performed for both the Cosserat and Kirchhoff rod models outlined in the previous section. The results can be seen in Tables 3 and 4, respectively.

There are two apparent differences between the Cosserat and Kirchhoff rod model results in the tables. For the runtime, using the Kirchhoff assumption sped up the simulations significantly, especially for the Runge-Kutta spatial integration method. The Kirchhoff assumption also allowed the simulations to use smaller timesteps. The effect was minimal for discretization methods with larger amounts of damping, but for methods with little to no damping, the Cosserat theory has a minimum timestep about 2.5 ms greater than the Kirchhoff theory. Interestingly, both the Kirchhoff assumptions and the method of spatial integration had minimal effects on the amount of numerical damping. The major factors affecting numerical damping were the timestep and the method used.

The log-log plot in Figure 2 shows that there is an approximate power-law relationship between timestep length and the numerical damping time constant. Each time discretization method maintained this relationship but with different rates of decrease. The damping time constants in Figure 2 were computed using simulations for the benchmark rod over a 10 second time interval using the Cosserat model and RK4.

The BDF3 method exhibits very little damping but has a relatively large minimum timestep for stability. The trapezoidal method appears more stable than BDF3; however, its minimum timestep is 4.4 ms for Cosserat rods with RK4 integration in space. For this benchmark simulation,

Table 4. Comparison of numerical methods for Kirchhoff equations.

Spatial Integration Method	Time Discretization Method	Numerical Damping Time Constant: $\Delta t = 5$ ms	Runtime: $\Delta t = 5$ ms Timeframe = 1 s Spatial Steps = 150	Minimum Stable Time Step
Euler	Backward Euler	0.13 s	0.20 s	0.5 ms
	BDF2	3.87 s	0.26 s	1.0 ms
	BDF- α , $\alpha = -0.2$	7.58 s	0.27 s	1.1 ms
	BDF- α , $\alpha = -0.48$	161.8 s	0.29 s	1.7 ms
	Trapezoidal Method	*NA	0.30 s	1.8 ms
	BDF3	Unstable	Unstable	32.8 ms
Fourth-Order Runge Kutta	Backward Euler	0.13 s	0.36 s	0.5 ms
	BDF2	3.84 s	0.50 s	0.9 ms
	BDF- α , $\alpha = -0.2$	7.54 s	0.51 s	1.1 ms
	BDF- α , $\alpha = -0.48$	186.7 s	0.54 s	1.4 ms
	Trapezoidal Method	*NA	0.57 s	1.5 ms
	BDF3	Unstable	Unstable	40.3 ms

*No discernible numerical damping

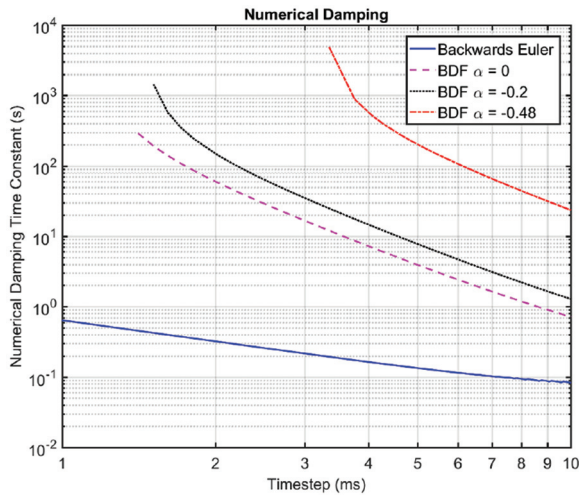


Fig. 2. Numerical damping time constant plotted versus timestep length for various methods using the benchmark case over 10 seconds with the Cosserat model and RK4. Larger time constants imply lower numerical damping and higher accuracy.

BDF2 and BDF- α , where $\alpha = -0.2$, can both achieve a good balance of stability and accuracy as they allow for small timesteps while maintaining energy decay less than 1% per cycle. The spatial methods of integration typically had no significant effect on the numerical damping. Thus, we advocate for using higher-order methods such as RK4 owing to the $O(n^4)$ scaling behavior of accuracy versus spatial resolution.

We performed an additional set of simulations with varying numbers of spatial integration steps (nodes) from 5 to 500 in increments of 5. Using the same simulation parameters and BDF2 with a timestep of 5 ms, the computation time per timestep is plotted versus the number of spatial nodes n in Figure 3 for both Kirchhoff and Cosserat models with Euler and RK4 methods in the spatial dimension. In this figure, computation times below 5 ms represent

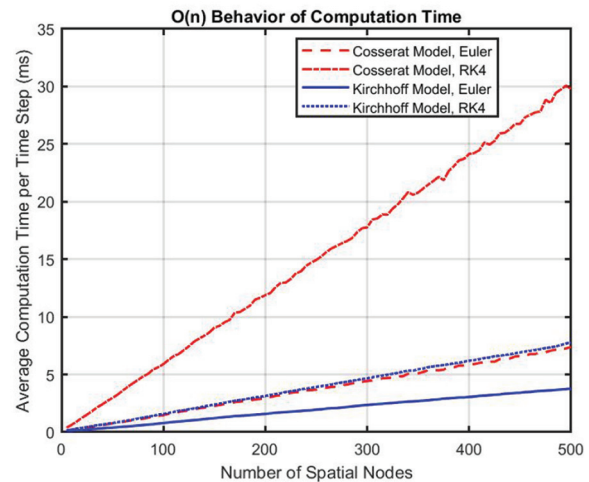


Fig. 3. Average computation time per timestep as a function of the number of spatial integration steps n for a 1 second simulation with $\delta t = 5$ ms using BDF2. The linearity of the data substantiates our claim of $O(n)$ scaling for computation time. Any value below 5 ms is faster than real time for this simulation.

simulation solutions computed faster than real time, and the linearity of all the data series indicates that computation time using our proposed approach is $O(n)$. In the experimental validation in the following section, we further compare the runtime performance of these methods in a physical experiment.

4. Experimental validation

In this section, we review the experimental procedure and data from Till and Rucker (2017a) (which therein validated the Kirchhoff model with $J = 0$), and we provide a new validation of the full Cosserat model on the same dataset. Further, beyond the performance characterization of Till and Rucker (2017a), we also provide an updated analysis

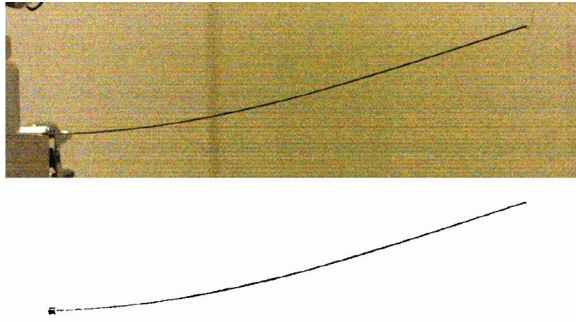


Fig. 4. The experimental rod shape was quantified by obtaining binary data based on darkness. The rightmost pixel is taken as the tip position, specifically the top rightmost pixel when multiple rightmost pixels exist.

of execution speeds for both models as a function of time-step length for different spatial resolutions and spatial integration methods. This section shows that the theoretical developments in Section 2 are indeed useful to describe the behavior of real, physical rods. However, the developments in this section are not necessary to understand the rest of the paper.

We experimentally validated the Cosserat (7) and Kirchhoff (11) rod models by comparing simulation results to high-speed footage of a cantilevered rod clamped to a table. The simulations implement a full three-dimensional model, but the experimental data was taken from planar cases to simplify the process of reconstructing the scene and evaluating the results. The rod was spring steel with a 1.42 mm diameter. There were two scenarios. First, a 20 g weight was hung by a string at the tip of a rod with a cantilevered length of 0.408 m, and after equilibrium was reached, the string was cut. Second, the cantilevered length was increased to 0.517 m to obtain larger vibrations, and the rod was hit with a rigid object near its base to excite high-frequency vibration modes. The BDF- α coefficient was $\alpha = -0.4$ for all simulations, which is close to the trapezoidal method and, thus, exhibits very little numerical damping.

The camera was placed about 3 m from the rod with the viewing plane parallel to the rod's plane of motion. The camera recorded a frame every millisecond. The rod was darker than the background so that the experimental rod position could be easily extracted by comparing pixel brightness values, as shown in Figure 4.

4.1. Weight release

The weight release trial was used to calibrate the rod parameters EI , ρ , and C . This calibration is implemented in MATLAB using “fsolve” to minimize error between experimental data and model prediction. The weight release response is very nearly a decaying sine wave, as shown in Figure 5. The calibration objective function was evaluated

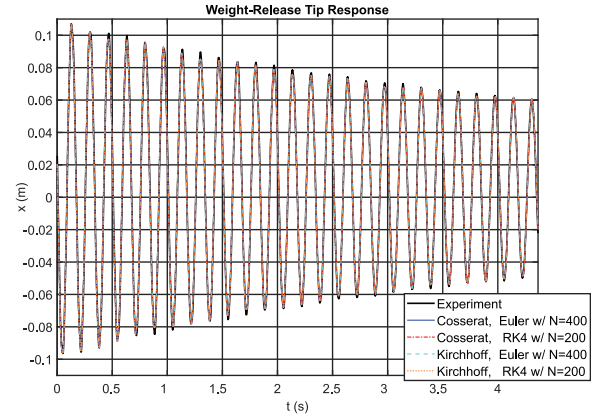


Fig. 5. A weight was attached to the free end of a cantilevered rod by a string. After the weighted rod reached equilibrium, the string was cut. This scenario was simulated, and the simulation parameters were calibrated so that the simulation response matches the experimental response. These calibrated values were used while validating the other impulse experiment.

Table 5. Calibrated parameters.

Parameter	Cosserat		Kirchhoff	
	Euler, $N = 400$	RK4, $N = 100$	Euler, $N = 400$	RK4, $N = 100$
EI (Nm ²)	0.0380	0.0380	0.0380	0.0380
ρ (kg/m ³)	7602	7602	7602	7603
C (g/m ²)	2.09	2.12	2.08	2.12

by running the simulation for a set of parameters and evaluating the characteristics of the simulated response versus the experimental data. The magnitude of the first peak, magnitude of the first valley, magnitude of the final peak, and frequency are compared and combined to form the objective function residual. MATLAB's “findpeaks” command can easily detect peaks in the smooth simulation data. The experimental data has some noise, but since the experimental response only needs to be analyzed once, this was done manually. The calibrated values are shown in Table 5. For steel, ρ is typically around 7,800 kg/m³. With an assumed Young's modulus of 200 GPa, the 1.42 mm diameter rod would have a bending stiffness EI of 0.03992 Nm². Thus, the calibrated values are within reason.

4.2. Impulse near the base

After calibration of the model parameters using the weight release dataset, we evaluated the model prediction versus data taken from the impulse response experiment. The impulse point force was modeled as a hat function in time with

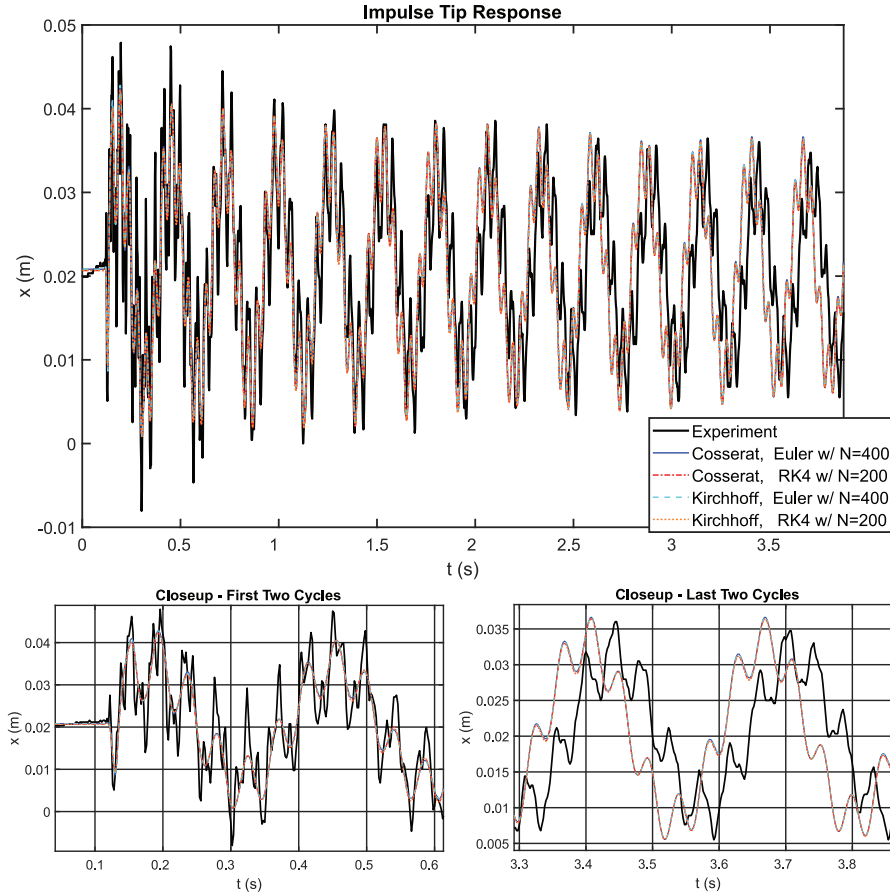


Fig. 6. An impulse was applied near the base of a cantilevered rod. The experimental impulse response is compared with two simulations that used the proposed method. Extension 1 includes a video of this trial.

$$F(t) = \begin{cases} M \frac{t}{0.5d}, & t < 0.5d \\ M(2 - \frac{t}{0.5d}), & 0.5d \leq t \leq d \\ 0, & t > d \end{cases}$$

Appropriate values for the impulse's peak magnitude and duration were found: $M = 5$ N and $d = 0.016$ s. The impulse point force is included in the simulation by performing piecewise integration of the ODEs in space and applying the point force at the transition. The impulse response is shown in Figure 6. Extension 1 shows this simulation, and a still frame is shown in Figure 8. Figure 9 shows the error between the experimental and simulated tip positions. The simulation responses are similar for the Cosserat and Kirchhoff models, so we conclude that the shear and extension strains are indeed negligible for the experimental rod.

4.3. Real-time performance

To evaluate computational speed, we ran many simulations of both the weight release and impulse response scenarios using increasing values of δt (logarithmically spaced). Results are shown in the log-log plots of Figure 7. The

real-time performance ratio is the amount of time simulated divided by the wall-clock time spent running the simulation, and the green regions indicate real-time performance. The plot confirms that most of the simulations ran in real time. At higher timesteps, the dependence of the runtime on the timestep is nearly linear. The runtime also appears linear in the number of spatial steps, again confirming $O(N)$ computation time as we showed in the previous section.

Not surprisingly, the impulse response case requires higher computational times owing to the increased presence of faster dynamic modes that require more solver iterations per timestep. A timestep of 2 ms captured the high-frequency dynamics very accurately, as shown in Figure 6, but this simulation required more computation time than it simulated, and the speed is further reduced when the solver begins to encounter numerical ill-conditioning at smaller timesteps.

Thus, we have shown that the model presented in Section 2 accurately describes elastic rod behavior, and that the mathematical model can be implemented in software to achieve real-time simulation.

Performance with Intel Core i7-4790K CPU @ 4.00GHz

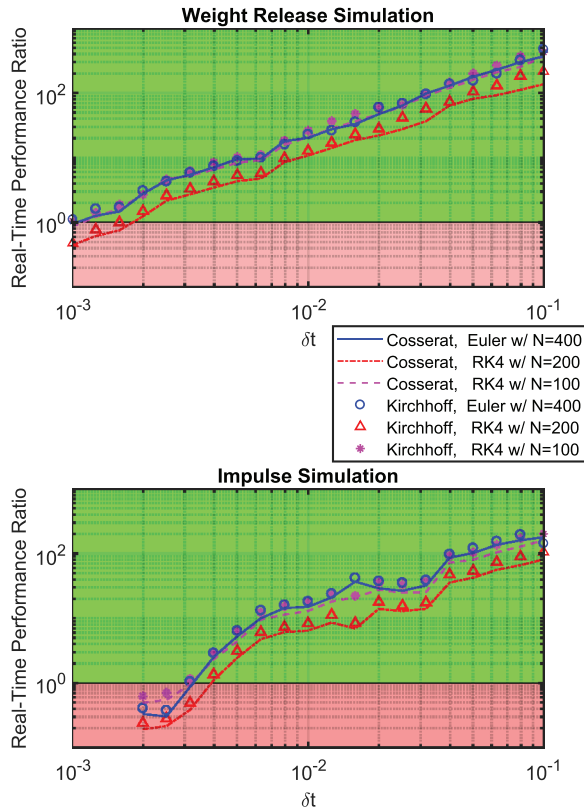


Fig. 7. We plot the real-time performance ratio versus the timestep on a log-log scale for various simulation datasets. The real-time performance ratio is the amount of time simulated divided by the wall-clock time spent running the simulation. A ratio greater than or equal to one indicates soft real-time performance, as shown by the green regions. The weight release scenario requires significantly less effort to solve than the impulse scenario. For the difficult impulse scenario, the simulation can run in soft real-time with $\delta t = 4$ ms. The smallest timestep for the impulse simulations is 2 ms because of convergence issues with small timesteps.

5. Application to continuum robots

As we discussed in Section 2.4 our proposed solution approach is directly applicable to various PDE models of a certain form. Beyond simple Cosserat rods, this PDE form encompasses many different continuum and soft robot models with actuation forces and/or constraints. In this section we give the relevant PDEs for three example dynamic robots and use our approach to seamlessly simulate their dynamics in real-time. We consider (1) a 6-DOF continuum Stewart–Gough platform, (2) a tendon-driven robot with a continuum backbone, (3) a pneumatic or hydraulic soft robot finger. The real-time simulations are also visualized in Extension 1. Each robot design may be understood independently; the only prerequisite section is the development of the rod model in Sections 2.1–2.4. Note that in each case, the equations are developed based on the continuous PDEs, independent of any specific discretization strategy.

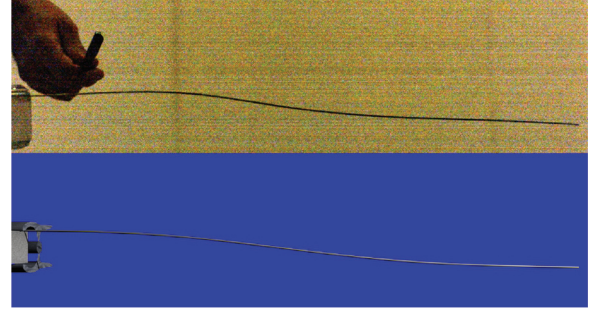


Fig. 8. An impulse point force is applied near the base of a cantilevered rod, resulting in a variety of vibration modes. The scenario is simulated using our proposed method and visualized with Blender.

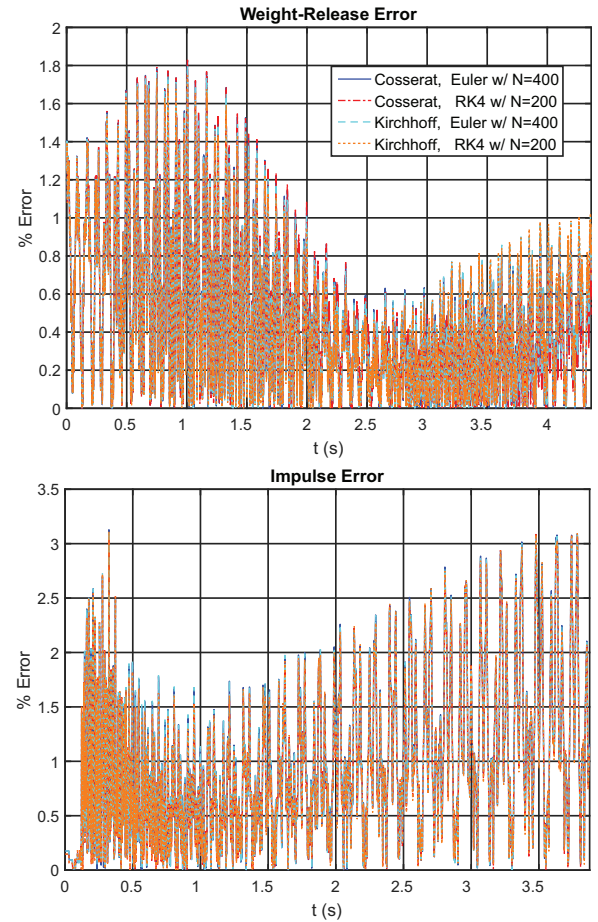


Fig. 9. The percentage error between measured and simulated transverse tip position is shown for the two experiments. The error is defined as the absolute difference in experimental and simulated transverse tip displacements normalized by the cantilever length, that is $\%Error = |p_{x,exp} - p_{x,sim}|/L * 100\%$.

A more conventional approach based on a priori spatial discretization of the robot structure would involve significantly more low-level symbolic manipulation effort before

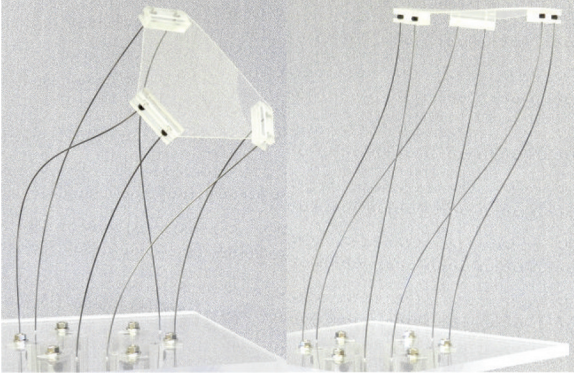


Fig. 10. A PCR is actuated to two different poses in its 6-DOF workspace

implementation, and would likely entail a worse speed/accuracy tradeoff than our high-order approach as discussed in the introduction.

5.1. Parallel continuum robots: capturing dynamic stability transitions

As shown in Figure 10, a parallel continuum robot (PCR) consists of multiple compliant continuum links connected to an end-effector platform in parallel, which can be actuated by translating the bases of the compliant links independently. In prior work, we have established the quasi-static modeling (Black et al., 2018) and explored the elastic stability (Till and Rucker, 2017b) of static solutions for PCRs, but PCR dynamics can also be significant, especially for larger-scale applications with potential human interaction, and dynamic modeling has not yet been explored. A particular feature that we would like our model to capture is the dynamic transition from one stable static state to another when the robot configuration becomes unstable owing to actuation or external loading. In this section, we outline the equations necessary to solve this problem via the numerical approach detailed above and demonstrate real-time simulation of a six-DOF parallel Stewart–Gough platform undergoing a dynamic stability transition.

A continuum Stewart–Gough platform is a PCR with six flexible links arranged in the traditional Stewart–Gough configuration (Bryson and Rucker, 2014). The rods are numbered from 1 to 6, and the rod state variables are denoted by a subscript so that \mathbf{p}_i refers to the position of the i th rod. The system is composed of six dynamic rod ODEs as in (7) subject to coupled boundary conditions. The distal ends of the rods are attached to the end-effector plate with a constant local-frame offset $\mathbf{r}_i \in \mathbb{R}^3$ from the plate center of mass $\mathbf{p}_e \in \mathbb{R}^3$ so that

$$\mathbf{p}_e(t) + \mathbf{R}_e(t)\mathbf{r}_i = \mathbf{p}_i(t, L_i) \quad \text{for } i = 1 \dots 6 \quad (13)$$

where $\mathbf{R}_e \in \text{SO}(3)$ is the end-effector orientation. The distal orientation of the rods is constrained to the plate so that

$$\mathbf{R}_e(t) = \mathbf{R}_i(t, L_i) \quad \text{for } i = 1 \dots 6 \quad (14)$$

The end-effector plate is governed by rigid-body dynamic equilibrium equations. The sum of forces has contributions from the rod attachments, gravity, and external loads as described by

$$\mathbf{F}_e(t) + m_e \mathbf{g} - \sum_{i=1}^6 \mathbf{n}_i(t, L_i) = m_e \mathbf{a}_e(t) \quad (15)$$

where $m_e \in \mathbb{R}^+$ is the end-effector mass, $\mathbf{a}_e \in \mathbb{R}^3$ is the end-effector acceleration in the global frame, and $\mathbf{F}_e(t) \in \mathbb{R}^3$ is an external force acting on the end-effector center of mass. The moment balance equation, with end-effector angular velocity $\boldsymbol{\omega}_e \in \mathbb{R}^3$ defined in the global frame, is

$$\begin{aligned} \mathbf{M}_e(t) - \sum_{i=1}^6 \mathbf{m}_i(t, L_i) + [\mathbf{R}_e(t)\mathbf{r}_i] \times \mathbf{n}_i(t, L_i) \\ = \mathbf{R}_e(t)\mathbf{J}_e\mathbf{R}_e^T(t)\boldsymbol{\omega}_{te}(t) + \hat{\boldsymbol{\omega}}_e(t)\mathbf{R}_e(t)\mathbf{J}_e\mathbf{R}_e^T(t)\boldsymbol{\omega}_e(t) \end{aligned} \quad (16)$$

The constraint equations (13), (14), (15), and (16) define a complete set of distal boundary conditions. Proximal boundary conditions include the known pose of each link where it originates from a fixed location in a base platform. The robot is actuated by extending or retracting the individual links out of holes in the baseplate so that the effective link length $L_i(t)$ of each leg is a control input. Similarly to the single cantilevered rod problem, after performing our implicit time integration, we are left with multiple sets of ODEs in the arc length of each link. We can solve this coupled boundary problem with a shooting method, simultaneously solving for the unknown proximal boundary conditions (reaction forces and moments at the base of each rod) as well as other unknowns (end-effector pose) which will satisfy the distal boundary conditions. We define the residual equations which are zero when the geometric and equilibrium constraints are satisfied as

$$\mathbf{E} = \{\mathbf{E}^F \quad \mathbf{E}^M \quad \mathbf{E}_1^p \quad \mathbf{E}_1^R \quad \dots \quad \mathbf{E}_6^p \quad \mathbf{E}_6^R\}$$

where

$$\mathbf{E}_i^p = \mathbf{p}_e + \mathbf{R}_e \mathbf{r}_i - \mathbf{p}_i$$

$$\mathbf{E}_i^R = [\mathbf{R}_e^T \mathbf{R}_i - \mathbf{R}_e \mathbf{R}_i^T]^V$$

$$\mathbf{E}^F = \mathbf{F}_e + m_e(\mathbf{g} - \mathbf{a}_e) - \sum_{i=1}^6 \mathbf{n}_i$$

$$\begin{aligned} \mathbf{E}^M = \mathbf{M}_e - \sum_{i=1}^6 \mathbf{m}_i + (\mathbf{R}_e \mathbf{r}_i) \times \mathbf{n}_i - \mathbf{R}_e \mathbf{J}_e \mathbf{R}_e^T \boldsymbol{\omega}_{te} \\ - \hat{\boldsymbol{\omega}}_e \mathbf{R}_e \mathbf{J}_e \mathbf{R}_e^T \boldsymbol{\omega}_e \end{aligned}$$

The above metric for rotation error E_i^R , which quantifies the difference between two rotation matrices as a 3×1 vector, is described in Mahony et al. (2012). Function arguments were omitted, but the rod state variables are functions of t and $s = L_i$, while the end-effector state variables are functions of t . The shooting method we implement uses a set of guessed variables

$$\mathbf{G} = \{p_e \quad \mathbf{k} \quad n_1 \quad m_1 \quad \cdots \quad n_6 \quad m_6\}$$

where the rod state variables are functions of t and $s = 0$, while the end-effector state variables are functions of t , and $\mathbf{k}(t) \in \mathbb{R}^3$ is an angular displacement vector used to generate the rotation matrix via Rodrigues' formula,

$$\mathbf{R}_e = \begin{cases} \mathbf{I} + \sin\|\mathbf{k}\| \frac{\hat{\mathbf{k}}}{\|\mathbf{k}\|} + (1 - \cos\|\mathbf{k}\|) \frac{\hat{\mathbf{k}}^2}{\|\mathbf{k}\|^2}, & \|\mathbf{k}\| > 0 \\ \mathbf{I}, & \|\mathbf{k}\| = 0 \end{cases}$$

The guessed set \mathbf{G} and residual \mathbf{E} form a square 42×42 system of nonlinear equations. There is some interesting flexibility in forming the sets \mathbf{G} and \mathbf{E} , but we have found this particular choice to be concise and simple to understand.

There is a potential for instability of PCRs that we have observed experimentally, and we have established a method based on optimal control theory to assess the stability of a particular solution to the static model equations (Till and Rucker, 2017b). However, this method does not provide information about what actually happens to the robot when it reaches a statically unstable configuration. The physical robot will not stay at an unstable static solution but will dynamically transition to a new stable equilibrium elsewhere in the workspace. We demonstrate the ability of our dynamic modeling framework to capture this behavior by simulating the forward dynamics of a robot which is actuated to a statically unstable configuration. The robot is shown in Figure 11. In this scenario, slow changes in the actuator positions translate the end-effector in a straight line, until eventually there is a bifurcation leading to dynamic end-effector behavior.

This simulation had a real-time ratio of about seven to eight using three threads working in parallel for the integration of (7). The rods had identical parameters of $E = 207$ GPa, $r = 1$ mm, and $\rho = 8,000$ kg/m³. The Stewart–Gough leg-spacing pattern used a major angle of 100° and a radius of 87 mm. The end-effector was modeled as a short acrylic ($\rho = 1,180$ kg/m³) cylinder with 3 mm depth and radius of 91 mm, which results in a mass of 92.1 g and a mass moment of inertia $\mathbf{J} = \text{diag}(1.91, 1.91, 3.81) \times 10^{-4}$ kg-m². The damping parameters were all zero. The discretization parameters were $\delta t = 1/120$ s, $\alpha = -0.2$, and 200 points per rod with Euler's method. The shooting method solver was the trust region dogleg scheme.

In Extension 1, we compare two simulations of this same scenario: one using our quasi-static model (Black et al., 2018) and one using our dynamic model. The quasi-static solution reaches the unstable configuration and stays

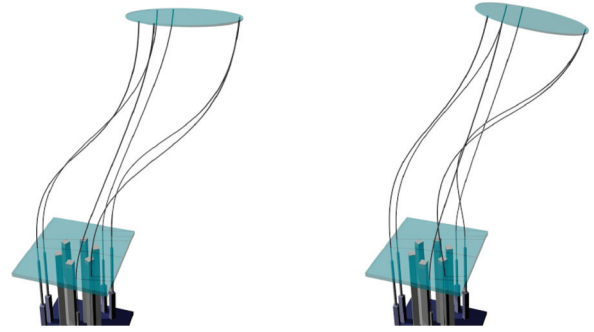


Fig. 11. As the continuum Stewart–Gough robot is translated along a path which satisfies the equilibrium equations, the robot encounters a bifurcation. A moment prior to instability is shown on the left. On the right, the end-effector bends to an angle and begins to sway dynamically. This scenario is shown in Extension 1.

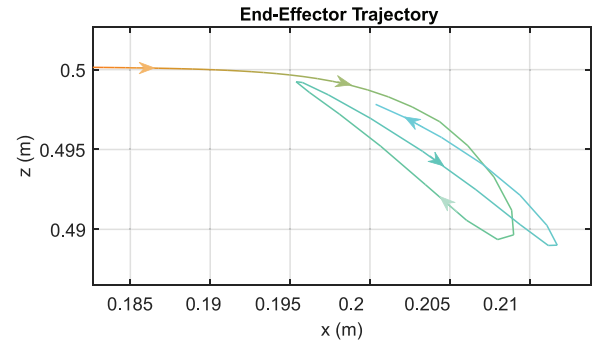


Fig. 12. As the CSG is actuated past a bifurcation, the end-effector sways back and forth dynamically as illustrated by the trajectory plot. This scenario is shown in Extension 1.

on the unstable solution branch. While our optimal control test can predict that this quasi-static model solution is unrealistic because it is elastically unstable, the quasi-static model has no way of reliably finding the true stable configuration or predicting the robot's motion on the way to that state. In contrast, the dynamic model successfully predicts the dynamic transition to the new stable state and the ensuing vibrations caused by the transition.

5.2. Cable-driven robots

The statics and dynamics of cable-driven (tendon-driven) continuum robots (as shown in Figure 13) and steerable catheters were derived from the Cosserat rod framework of Rucker and Webster III (2011). This prior work simulated the robot dynamics with an explicit Lax–Wendroff finite-difference scheme which was severely limited by the Courant–Friedrichs–Lewy numerical stability condition, even for a model with less than 10 spatial segments. In this section, we review and extend the model of Rucker and Webster III (2011), adding internal damping and drag terms and demonstrating how to use our implicit approach to achieve efficient real-time simulation of the dynamics. We

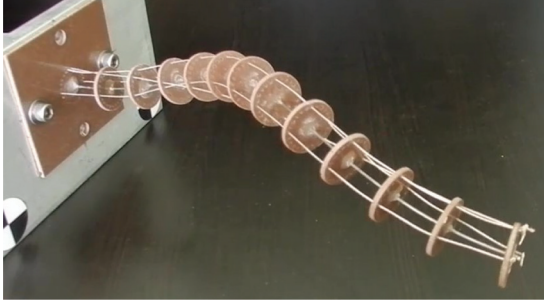


Fig. 13. A tendon-driven continuum robot with a helical tendon is actuated to a variable-curvature shape with significant inertial dynamics.

assume the robot consists of an elastic backbone member (modeled as a Cosserat rod) with continuous channels for actuation cables which apply shape-dependent forces and moments to the backbone when tensioned. This basic design describes many tools and is a reasonable continuous approximation in cases with discrete routing holes created by spacer disks. There are n cables, and each cable experiences a tension τ_i and is offset from the cross-section center of mass by a vector $\mathbf{r}_i(s)$ in the local cross-sectional plane, such that each tendon's position in the global frame is

$$\mathbf{p}_i = \mathbf{p} + \mathbf{R}\mathbf{r}_i$$

The cables cause distributed forces and moments on the rod which are derived under the assumption of negligible tendon friction and inertia as

$$\begin{aligned} \mathbf{f}_c &= - \sum_{i=1}^n \tau_i \frac{\hat{\mathbf{p}}_{si}^2}{\|\mathbf{p}_{si}\|} \mathbf{p}_{ssi} \\ \mathbf{l}_c &= - \sum_{i=1}^n (\hat{\mathbf{p}}_i - \hat{\mathbf{p}}) \tau_i \frac{\hat{\mathbf{p}}_{si}^2}{\|\mathbf{p}_{si}\|} \mathbf{p}_{ssi} \end{aligned}$$

These can be rewritten in terms of the backbone's kinematic variables as

$$\begin{aligned} \mathbf{f}_c &= \mathbf{R}(\mathbf{a} + \mathbf{A}\mathbf{v}_s + \mathbf{G}\mathbf{u}_s) \\ \mathbf{l}_c &= \mathbf{R}(\mathbf{b} + \mathbf{G}^T \mathbf{v}_s + \mathbf{H}\mathbf{u}_s) \end{aligned}$$

where expressions needed to calculate \mathbf{a} , \mathbf{b} , \mathbf{A} , \mathbf{G} , and \mathbf{H} are defined as follows:

$$\begin{aligned} (\mathbf{p}_{si})^b &= \hat{\mathbf{u}}\mathbf{r}_i + \mathbf{r}_{si} + \mathbf{v} \\ \mathbf{A}_i &= - \tau_i \frac{\left(((\mathbf{p}_{si})^b)^\wedge \right)^2}{\|(\mathbf{p}_{si})^b\|^3} \\ \mathbf{G}_i &= - \mathbf{A}_i \hat{\mathbf{r}}_i \\ \mathbf{a}_i &= \mathbf{A}_i [\hat{\mathbf{u}}((\mathbf{p}_{si})^b + \mathbf{r}_{si}) + \mathbf{r}_{ssi}] \end{aligned}$$

$$\mathbf{b}_i = \hat{\mathbf{r}}_i \mathbf{a}_i$$

$$\mathbf{a} = \sum_{i=1}^n \mathbf{a}_i, \quad \mathbf{b} = \sum_{i=1}^n \mathbf{b}_i, \quad \mathbf{A} = \sum_{i=1}^n \mathbf{A}_i,$$

$$\mathbf{G} = \sum_{i=1}^n \mathbf{G}_i, \quad \mathbf{H} = \sum_{i=1}^n \hat{\mathbf{r}}_i \mathbf{G}_i$$

The differential equations for internal loading are then

$$\mathbf{n}_s = - \mathbf{R}(\mathbf{a} + \mathbf{A}\mathbf{v}_s + \mathbf{G}\mathbf{u}_s) - \bar{\mathbf{f}}$$

$$\mathbf{m}_s = - \mathbf{R}(\mathbf{b} + \mathbf{G}^T \mathbf{v}_s + \mathbf{H}\mathbf{u}_s) + \partial_t(\mathbf{R}\rho\mathbf{J}\boldsymbol{\omega}) - \hat{\mathbf{p}}_s \mathbf{n} - \mathbf{l}$$

where $\bar{\mathbf{f}}$ and \mathbf{l} represent any distributed loading components not caused by the tendons. These equations are implicit, that is $\mathbf{v}_s = \mathbf{v}_s(\mathbf{n}_s)$ and $\mathbf{u}_s = \mathbf{u}_s(\mathbf{m}_s)$ because differentiating the constitutive law (4) leads to

$$\begin{aligned} \mathbf{n}_s &= \mathbf{R}_s [\mathbf{K}_{se}(\mathbf{v} - \mathbf{v}^*) + \mathbf{B}_{se}\mathbf{v}_t] + \mathbf{R}[\mathbf{K}_{sse}(\mathbf{v} - \mathbf{v}^*) \\ &\quad + \mathbf{K}_{se}(\mathbf{v}_s - \mathbf{v}_s^*) + \mathbf{B}_{sse}\mathbf{v}_t + \mathbf{B}_{se}\mathbf{v}_{st}] \\ \mathbf{m}_s &= \mathbf{R}_s [\mathbf{K}_{bt}(\mathbf{u} - \mathbf{u}^*) + \mathbf{B}_{bt}\mathbf{u}_t] + \mathbf{R}[\mathbf{K}_{sbt}(\mathbf{u} - \mathbf{u}^*) \\ &\quad + \mathbf{K}_{bt}(\mathbf{u}_s - \mathbf{u}_s^*) + \mathbf{B}_{sbt}\mathbf{u}_t + \mathbf{B}_{bt}\mathbf{u}_{st}] \end{aligned}$$

Convenience motivates us to choose \mathbf{v} and \mathbf{u} as state variables because the resulting equations are simpler than those for \mathbf{m} and \mathbf{n} . We apply the time discretization, rotate the equations, and introduce intermediate variables so that the internal loading differential equations are described by

$$\mathbf{R}^T \mathbf{n}_s = - \mathbf{A}\mathbf{v}_s - \mathbf{G}\mathbf{u}_s + \boldsymbol{\Lambda}_n$$

$$\mathbf{R}^T \mathbf{m}_s = - \mathbf{G}^T \mathbf{v}_s - \mathbf{H}\mathbf{u}_s + \boldsymbol{\Lambda}_m$$

and the constitutive law by

$$\mathbf{R}^T \mathbf{n}_s = (\mathbf{K}_{se} + c_0 \mathbf{B}_{se})\mathbf{v}_s + \boldsymbol{\Gamma}_v$$

$$\mathbf{R}^T \mathbf{m}_s = (\mathbf{K}_{bt} + c_0 \mathbf{B}_{bt})\mathbf{u}_s + \boldsymbol{\Gamma}_u$$

where

$$\boldsymbol{\Lambda}_n = - \mathbf{a} + \rho \mathbf{A}(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t) + \mathbf{C}\mathbf{q} \odot |\mathbf{q}| - \mathbf{R}^T(\rho \mathbf{A}\mathbf{g} + \bar{\mathbf{f}})$$

$$\boldsymbol{\Lambda}_m = - \mathbf{b} + \rho(\hat{\boldsymbol{\omega}}\mathbf{J}\boldsymbol{\omega} + \mathbf{J}\boldsymbol{\omega}_t) - \hat{\mathbf{v}}\mathbf{n}^b - \mathbf{R}^T \mathbf{l}$$

$$\boldsymbol{\Gamma}_v = \hat{\mathbf{u}}\mathbf{n}^b + \mathbf{K}_{sse}(\mathbf{v} - \mathbf{v}^*) - \mathbf{K}_{se}\mathbf{v}_s^* + \mathbf{B}_{sse}\mathbf{v}_t + \mathbf{B}_{se}\mathbf{v}_s^h$$

$$\boldsymbol{\Gamma}_u = \hat{\mathbf{u}}\mathbf{m}^b + \mathbf{K}_{sbt}(\mathbf{u} - \mathbf{u}^*) - \mathbf{K}_{bt}\mathbf{u}_s^* + \mathbf{B}_{sbt}\mathbf{u}_t + \mathbf{B}_{bt}\mathbf{u}_s^h$$

Note that the internal loads are calculated by

$$\mathbf{n}^b = \mathbf{K}_{se}(\mathbf{v} - \mathbf{v}^*) + \mathbf{B}_{se}\mathbf{v}_t$$

$$\mathbf{m}^b = \mathbf{K}_{bt}(\mathbf{u} - \mathbf{u}^*) + \mathbf{B}_{bt}\mathbf{u}_t$$

With the four coupled equations above, we may solve a linear system for \mathbf{v}_s and \mathbf{u}_s . The resulting set of ODEs for the tendon robot is

$$\begin{aligned}
p_s &= Rv \\
R_s &= R\hat{u} \\
\begin{bmatrix} v_s \\ u_s \end{bmatrix} &= \Phi^{-1} \begin{bmatrix} -\Gamma_v + \Lambda_n \\ -\Gamma_u + \Lambda_m \end{bmatrix} \\
q_s &= v_t - \hat{u}q + \hat{\omega}v \\
\omega_s &= u_t - \hat{u}\omega
\end{aligned} \quad (17)$$

where

$$\Phi = \begin{bmatrix} (K_{se} + c_0 B_{se} + A) & G \\ G^T & (K_{bt} + c_0 B_{bt} + H) \end{bmatrix}$$

We note that depending on the scenario and the value of B_{se} , stiff shear and extension dynamics can cause convergence issues for the tendon system. In many situations it is appropriate to neglect the shear and extension strains. If shear and extension are neglected, only u_s is directly dependent on solving a linear system so that

$$\begin{aligned}
u_s &= (K_{bt} + c_0 B_{bt} + H)^{-1} \\
&\quad \left[-\Gamma_u - b + \rho(\hat{\omega}J\omega + J\omega_t) - \hat{e}_3 n^b - R^T l \right] \\
n_s &= R[-Gu_s - a + \rho A(\hat{\omega}q + q_t) + Cq \odot |q|] - \rho Ag - \bar{f}
\end{aligned}$$

The state equation for v_s is replaced by n_s . The other equations are unaffected, except of course that $v = e_3$ and $v_t = 0$. Shear and extension are included in the simulation here for the sake of generality.

Aside from the ODE describing the continuous behavior of the backbone and tendons, the boundary conditions must account for the final rigid attachment of a tendon to the backbone which causes step changes in the backbone internal loading as described by Rucker and Webster III (2011). The termination of tendons at a point s cause point loads

$$\begin{aligned}
F_i^b &= -\tau_i \frac{p_{st}^b(s^-)}{\|p_{st}^b(s^-)\|} \\
L_i^b &= \hat{r}_i F_i^b
\end{aligned}$$

which change the internal force by

$$\begin{aligned}
n^b(s^+) &= n^b(s^-) - F_i^b \\
m^b(s^+) &= m^b(s^-) - L_i^b
\end{aligned}$$

This can be alternatively be written in terms of strains

$$\begin{aligned}
v(s^+) &= v(s^-) - K_{se}^{-1} F_i^b \\
u(s^+) &= u(s^-) - K_{bt}^{-1} L_i^b
\end{aligned}$$

We applied this dynamic tendon robot model and computational approach to simulate a tendon robot performing an object transfer task, as shown in Figure 14. Depending on the design and scale of a tendon robot, the inertial dynamics can give rise to significant vibrations even with slow actuator movements. After the simulated tendon robot picks up the object, its movement is highly

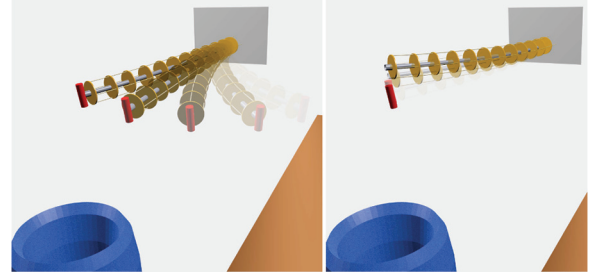


Fig. 14. To demonstrate dynamic tendon–robot motion, a robot with four tendons and a “vacuum gripper” removes a weighted object from a table and drops it into a bin. On the left, the robot moves the object towards the bin, and on the right the arm swings upward after releasing the object. This scenario is shown in Extension 1.

influenced by the inertial dynamics from the added tip mass. In this simulation, the actuation to reach the object as well as the loading and unloading associated with picking it up and dropping it all involve underactuated robot dynamics captured by our model in real time, as can be seen in Extension 1.

The simulated robot contains four tendons offset from the backbone by 9.5 mm with an angular separation of 90° about the backbone. The backbone length was 0.24 m. The backbone had Young’s modulus $E = 207$ GPa, $r = 0.4$ mm, and $\rho = 1.6 \times 10^4$ kg/m³, which is about twice as heavy as steel to account for both the backbone and the support disks. There is some damping with $C = I * 0.03$ kg/m² and $B_{bt} = I \times 10^{-6}$ Nm²s. The object had a mass of 1 g, and the rigid-body dynamics were coupled to the robot system through the distal boundary conditions by

$$\begin{aligned}
F_e(t) + m_e g - n(t, L^+) &= m_e a_e(t) \\
M_e(t) - m(t, L^+) &= R_e(t) J_e R_e^T(t) \omega_{ie}(t) \\
&\quad + \hat{\omega}_e(t) R_e(t) J_e R_e^T(t) \omega_e(t)
\end{aligned}$$

The ODE integration was performed with Euler’s method using 200 points. The time discretization used the BDF- α method with $\delta t = 1/60$ s and $\alpha = -0.03$. The simulation achieved a real-time speed ratio greater than 3.5.

5.2.1. Experimental comparison with approximate tendon damping model. To evaluate the physical realism of the tendon robot dynamic model, a robot was constructed from a spring steel backbone with acrylic spacer disks and a single Kevlar tendon, shown in Figure 15. The tendon displacement is controlled by a geared servo motor (Dynamixel MX-28-AT), which is a prescribed-displacement actuation setup as recently studied by Oliver-Butler et al. (2019). A step input (shown in Figure 15) is applied to pull the robot upward, then after a steady state is reached, another step input is applied to return to the original tendon displacement as shown in Figure 15.

Friction between the robot and the actuation tendons is more significant in bent configurations due to higher normal forces on the tendons. To approximate the effect of tendon friction without significantly altering the structure of our model, we define a simple distributed damping force and moment applied to the backbone. If the tendon is routed in a straight path parallel to the backbone, the damping force is approximately proportional to the tension τ (still assumed constant), the magnitude of the backbone curvature $\|\mathbf{u}\|$, and the relative tangential velocity \mathbf{v}_i between the tendon and its channel as follows:

$$\mathbf{f}_{f,i} = -\beta\tau\|\mathbf{u}\|\mathbf{v}_i\mathbf{Re}_3$$

$$\mathbf{l}_{f,i} = \mathbf{r}_i \times \mathbf{f}_{f,i},$$

where β is a damping coefficient, and we can compute \mathbf{v}_i numerically from configuration variables at previous timesteps.

The backbone has a length of $L = 0.7144$ m and diameter of $d = 0.00135$ m. The tendon is at a constant offset of 0.0151 m from the backbone. The frame convention is such that the tendon offset vector is $\mathbf{r} = 0.0151\mathbf{e}_1$ m and the direction of gravity $\mathbf{g} = -9.81\mathbf{e}_1$ m/s². The whole arm was weighed to have a mass of 0.034 kg. The initial tendon displacement holds the tip tangent to the z -axis, and the tendon is retracted 0.01619 m by the step displacement. The tip position of the robot and actual response of the motor were measured with a stereoscopic camera tracking system (MicronTracker H3-60, Claron Technology Inc.). The actual motor response to the commanded step was very nearly a linear ramp, which occurred over 0.31 s for both the upward and downward motions. The tendon compliance was calibrated to a value of 1.6×10^{-3} m/m, and there is an additional distance of 0.0518 m from the baseplate to the motor. The simulation used BDF2 implicit time discretization with $\delta t = 0.05$ s. The spatial integration used Euler's method with $N = 200$ points. The air drag, material damping, and frictional coefficients are $\mathbf{C} = \mathbf{I} \times 10^{-4}$ kg/m², $\mathbf{B}_{bt} = \mathbf{I} \times 5 \times 10^{-4}$ Nm²s, and $\beta = 5$ s/m.

The model response roughly lines up with the observed behavior. The friction model is simplified since it does not include static friction effects (stiction), but it does reproduce the experimentally observed effect of greater dissipation in the bent state as shown in Figure 15. The magnitudes and steady-state behavior are accurately predicted.

5.3. Fluidic soft robots

Finally, motivated by the recent increase in research activity around soft robotic structures actuated pneumatically or hydraulically, we derive a Cosserat-rod-based model for fluidic soft robots with pressurizable chambers and demonstrate the real-time solution of its forward dynamics for a soft robotic finger, using both air and incompressible water as the transmission fluid.

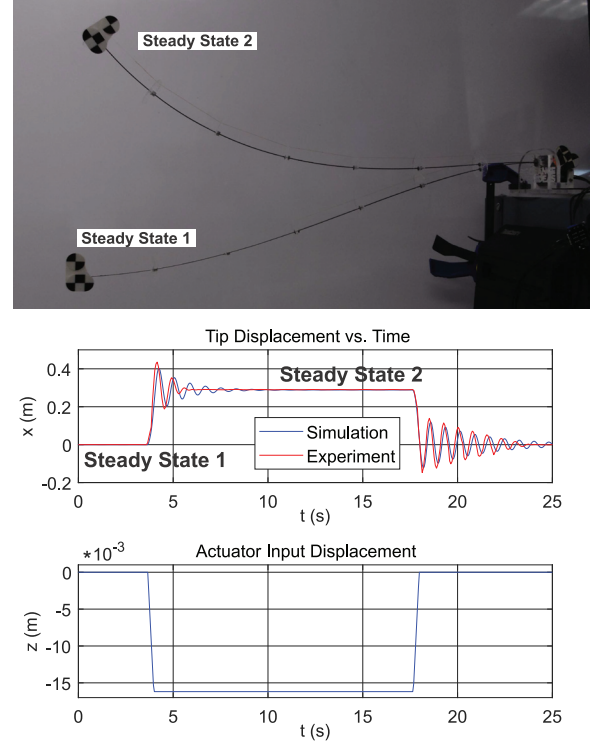


Fig. 15. The dynamics model is validated against this tendon robot. The attached markers allow the tip position to be measured by a stereoscopic camera system. The composite image shows the steady-state configurations before and after the step input. A pair of step responses is applied to the displacement of a tendon, resulting in dynamic tip motion.

We consider a soft robot with one or more hollow actuation chambers offset from the neutral axis. There is a vector \mathbf{r}_i from the cross-section center of mass to the center of the i th chamber, which is similar to the tendon robot variable \mathbf{r}_i . We restrict our attention to cases with $\mathbf{r}_{si} = 0$ so that the channel has a constant offset from the cross-section centroid. Fluid pressure is applied in the chamber, which results in a bending motion of the robot. We assume quasi-static fluid dynamics in the chamber so that there is a single uniform pressure $P_i(t)$. The situation is illustrated in Figure 16. We note that robots with multiple sections connected in serial can be modeled by piecewise integration with breaks at each section transition, but we only consider a single section here.

The distributed loading and point wrenches caused by pressurizing the chamber are similar to the effects of a tendon robot, but the two are not quite equivalent because tendon forces are transmitted along the tendon tangent line, while pressure forces act normal to a cross-sectional plane. We assume all chambers extend to the distal end of the robot, with flat chamber caps of area A_i so that the magnitude of the force on the cap is $P_i A_i$. The force and moment vectors applied to the end of the elastic member are then

$$\mathbf{F}_i^b = P_i A_i \mathbf{e}_3$$

$$\mathbf{L}_i^b = \hat{\mathbf{r}}_i \mathbf{F}_i^b$$

The wrenches at the chamber ends cause a point change in the internal loading by

$$\begin{aligned}\mathbf{n}^b(L) &= \mathbf{n}^b(L^-) - \sum_{i=1}^n \mathbf{F}_i^b \\ \mathbf{m}^b(L) &= \mathbf{m}^b(L^-) - \sum_{i=1}^n \mathbf{L}_i^b\end{aligned}$$

To obtain an expression for the distributed loading, we will consider the force and moment balance for a cut section of the soft robot as shown in Figure 17.

We write the acceleration \mathbf{p}_{tt} in terms of the local frame velocity \mathbf{q} so that

$$\mathbf{p}_{tt} \equiv \mathbf{R}(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t)$$

The dynamic force balance on the section is given by

$$\begin{aligned}\int_c^s \mathbf{f}_e(\sigma) d\sigma + \mathbf{n}(s) - \mathbf{n}(c) - \sum_{i=1}^n P_i A_i [\mathbf{R}(s) - \mathbf{R}(c)] \mathbf{e}_3 \\ = \int_c^s \rho A \mathbf{R}(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t) d\sigma\end{aligned}$$

This can be differentiated and simplified to obtain

$$\mathbf{n}_s = -\mathbf{f}_e + \rho A \mathbf{R}(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t) + \sum_{i=1}^n P_i A_i \mathbf{R}_s \mathbf{e}_3$$

Similarly, we consider the moment balance

$$\begin{aligned}\mathbf{m}(s) - \mathbf{m}(c) + \mathbf{p}(s) \times \mathbf{n}(s) - \mathbf{p}(c) \times \mathbf{n}(c) \\ - \sum_{i=1}^n [\mathbf{p}(s) + \mathbf{R}(s)\mathbf{r}_i] \times P_i A_i \mathbf{R}(s) \mathbf{e}_3 \\ - [\mathbf{p}(c) + \mathbf{R}(c)\mathbf{r}_i] \times P_i A_i \mathbf{R}(c) \mathbf{e}_3 \\ + \int_c^s [\mathbf{l}_e(\sigma) + \mathbf{p}(\sigma) \times \mathbf{f}_e(\sigma)] d\sigma \\ = \int_c^s \partial_t(\mathbf{R}\rho\mathbf{J}\boldsymbol{\omega}) d\sigma\end{aligned}$$

which is differentiated to find

$$\begin{aligned}\mathbf{m}_s = -\mathbf{l}_e - (\mathbf{p} \times \mathbf{n})_s - \mathbf{p} \times \mathbf{f}_e + \partial_t(\mathbf{R}\rho\mathbf{J}\boldsymbol{\omega}) + \sum_{i=1}^n P_i A_i \frac{\partial}{\partial s} \\ [(\mathbf{p} + \mathbf{R}\mathbf{r}_i) \times \mathbf{R}\mathbf{e}_3]\end{aligned}$$

We note that additional terms are present if A_i varies as a function of arc length. Expanding and canceling terms leads to

$$\begin{aligned}\mathbf{m}_s = -\mathbf{l}_e - \mathbf{p}_s \times \mathbf{n} + \partial_t(\mathbf{R}\rho\mathbf{J}\boldsymbol{\omega}) + \sum_{i=1}^n P_i A_i \mathbf{R} \\ [(\mathbf{v} + \hat{\mathbf{u}}\mathbf{r}_i) \times \mathbf{e}_3 + \mathbf{r}_i \times \hat{\mathbf{u}}\mathbf{e}_3]\end{aligned}$$

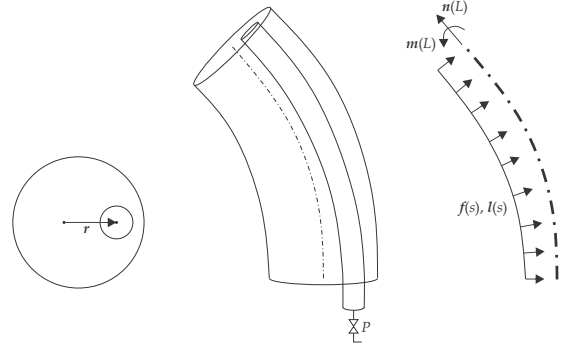


Fig. 16. A soft elastic rod has a hollow chamber offset from the central axis which is subject to some quasi-static and uniform pressure $P(t)$. The chamber is offset from the central axis by a vector $\mathbf{r}(s)$ in the local frame. The pressure results in a force at the cap of the chamber, which also generates a moment due to the offset. The outer curve of the chamber has more surface area than the inner curve, resulting in a net distributed force and moment.

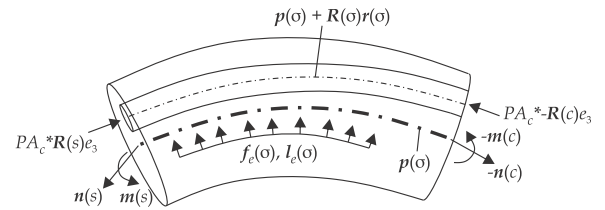


Fig. 17. The forces and moments acting on a segment of the robot are shown for cuts at s and c with $s > c$. The segment is subject to internal forces and moments at the centerline of the cuts, forces maintaining the pressure where the chamber is cut, and external distributed loading.

To more clearly indicate the distributed loads caused by the pressure, we define intermediate variables

$$\begin{aligned}\mathbf{f}_P &:= - \sum_{i=1}^n P_i A_i \mathbf{R}_s \mathbf{e}_3 \\ \mathbf{l}_P &:= - \sum_{i=1}^n P_i A_i \mathbf{R}[(\mathbf{v} + \hat{\mathbf{u}}\mathbf{r}_i) \times \mathbf{e}_3 + \mathbf{r}_i \times \hat{\mathbf{u}}\mathbf{e}_3]\end{aligned}\quad (18)$$

so that the differential equations are

$$\begin{aligned}\mathbf{n}_s &= \rho A \mathbf{R}(\hat{\boldsymbol{\omega}}\mathbf{q} + \mathbf{q}_t) - \mathbf{f}_e - \mathbf{f}_P \\ \mathbf{m}_s &= \partial_t(\mathbf{R}\rho\mathbf{J}\boldsymbol{\omega}) - \mathbf{p}_s \times \mathbf{n} - \mathbf{l}_e - \mathbf{l}_P\end{aligned}$$

With the differential equations solved, we may consider how the fluid properties influence the boundary conditions. The robot configuration defines the fluid chamber volume by integrating

$$V_i^I = \int_0^L A_i \left\| \frac{\partial}{\partial s} (\mathbf{p} + \mathbf{r}_i) \right\| ds$$

where

$$\left\| \frac{\partial}{\partial s} (\mathbf{p} + \mathbf{r}_i) \right\| = \|\hat{\mathbf{u}}\mathbf{r}_i + \mathbf{v}\|$$

Note in this preliminary derivation we consider the cross-sectional geometry constant in time, but future work should consider the dependence of A_i on robot shape and chamber pressure.

The fluid volume is coupled to pressure, temperature, and molar quantity. Modeling approaches vary; it can be adequate to assume direct input control of fluid pressure, but in some cases the fluid response is sufficiently slow that it is more appropriate to use sophisticated fluid dynamics models (Polygerinos et al., 2017). We assume uniform fluid pressure subject to the ideal gas law, which has precedent. Although pump dynamics are potentially significant (Katzschmann et al., 2016), actuation problems are often idealized as independent from the robot system, and we neglect pump dynamics here so that the system input is the mass of fluid in the chamber. We consider two cases: (1) incompressible fluid as in a hydraulic robot, so that fluid volume is effectively controlled; and (2) compressible fluid as in a pneumatic robot, so that molar quantity of fluid is the input to the system.

5.3.1. Incompressible working fluid. In the case of an incompressible working fluid, the volume of fluid inside a chamber could be directly controlled by a positive-displacement pump. We denote the controlled volume by V_i^C . For the shooting problem, we may guess the chamber pressures P_i and obtain a residual equation from comparing the controlled volume to the integrated chamber volume V_i^I . The error in volume may be poorly scaled in the shooting method since it has units of distance cubed. Our simulation code addresses this by normalizing the error by the volume in the straight configuration so that the error is $E_i^V = (V_i^I - V_i^C)/(A_i L)$. The full shooting method residual function has guessed values

$$\mathbf{G} = \{\mathbf{n}(t, 0) \quad \mathbf{m}(t, 0) \quad P_1(t) \quad \cdots \quad P_n(t)\}$$

and the residual terms include the force and moment balance at the tip

$$\mathbf{E}(\mathbf{G}) = \{\mathbf{E}^F \quad \mathbf{E}^M \quad E_1^V \quad \cdots \quad E_n^V\}$$

5.3.2. Compressible working fluid. For compressible working fluids, the boundary conditions must account for a gas law relating the mass of fluid to the pressure. We consider the ideal gas law

$$(P_i + P_{\text{atm}})V_i^G = n_i R_i T_i$$

where n_i is the moles of gas in the chamber, R_i is the gas constant, and T_i the temperature of the fluid. One may form a shooting method by guessing the pressures P_i and comparing the resulting gas law volume $V_i^G = n_i R_i T_i / (P_i + P_{\text{atm}})$ to the integrated volume V_i^I . We

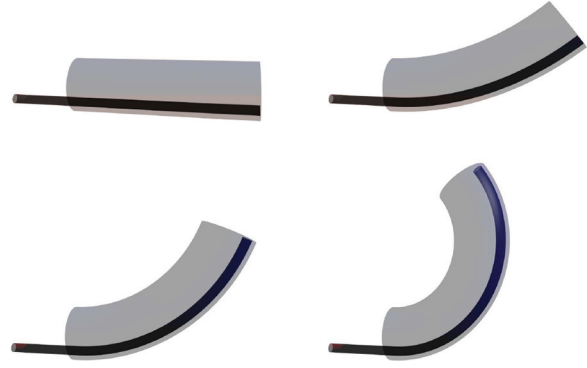


Fig. 18. Still frames convey the simulated soft robot motion. Starting from zero gauge pressure, additional fluid is added to the chamber, resulting in a bending motion.

note that the difference between the compressible and incompressible working fluid models is that the compressible model compares V_i^I to the volume according to the gas law $V_i^G(n_i)$ involving the controlled mass of fluid, whereas the incompressible model simply compares V_i^I with the controlled volume V_i^C .

5.3.3. Simulated comparison. We use the simulation to study the change in natural frequency associated with compressibility of the working fluid in soft robots. We compare two sets of fluid parameters, the first has a controlled amount of air governed by the ideal gas law, and the second has an incompressible fluid of controlled volume. The air has $R = 8.314 \text{ J/(mol K)}$, $P_{\text{atm}} = 101,325 \text{ Pa}$, and $T = 20^\circ\text{C} = 293.15 \text{ K}$. There are no parameters for the incompressible fluid. In addition to the fluids, the robots have identical parameters, which are $L = 0.1 \text{ m}$, $E = 20 \text{ MPa}$, radius = 0.015 m , and $\rho = 300 \text{ kg/m}^3$. There is a single fluid chamber with a radius of 2.5 mm and a constant offset from the center of 0.01 m in the negative x -direction, which leads to an offset $\mathbf{r}_1 \approx -0.0103\mathbf{e}_1 \text{ m}$ from the neutral axis. The discretization parameters were $\delta t = 0.01 \text{ s}$, $\alpha = -0.3$, and $N = 50$ for the number of points in space.

The robots began in the straight configuration and increased either n or V at a constant rate. The first simulation increases n to $2 \times 10^{-4} \text{ mol}$, and the second simulation increases the volume of the chamber linearly up to $2.65 \times 10^{-6} \text{ m}^3$. These two final configurations are equivalent at steady state. For both simulations the changes occur over a period of 2 seconds, and the dynamic response without any additional actuation is simulated for another 2.5 seconds. The results of the simulation are shown in Figure 18. The real-time ratio was in the range of 26–32 for the incompressible fluid and about 28 with air as the fluid (a single-chamber robot is a simple case).

The transverse tip response is shown in Figure 19. The limit cycle amplitude is roughly twice as large for the compressible air compared with an incompressible fluid. This indicates that choosing a fluid with greater compressibility

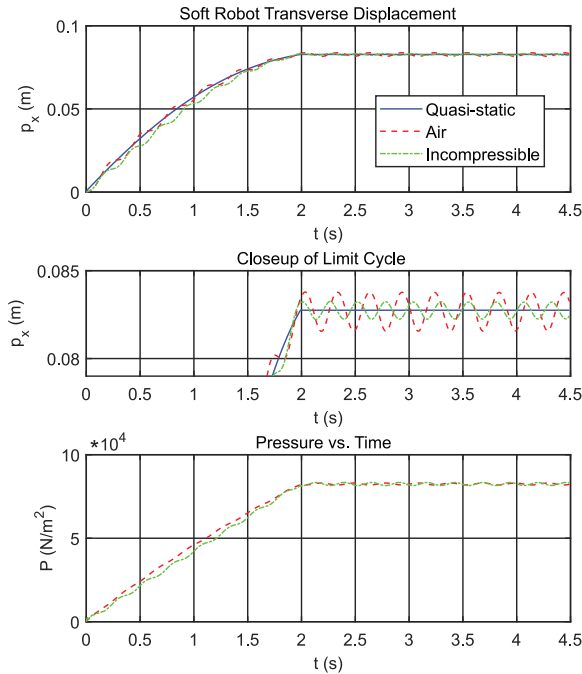


Fig. 19. The dynamic responses for soft robots with compressible and incompressible fluids are compared. The robot with incompressible fluid experiences less vibration when the fluid quantity is held constant at $t = 2$ s. The two fluids have similar fluctuations in pressure.

results in larger vibrations. In addition, we note that the shear and extension strains turn out to be significant for this robot; at the final steady-state solution the elongation strain average over arclength is 0.12.

6. Conclusions

We have presented a numerical framework for solving Cosserat-rod-based dynamic models of soft and continuum robots. The stability provided by the implicit time discretization enables one to solve robot dynamics problems at real-time rates. The computational effort of the method scales linearly with respect to spatial resolution. Our comparison of numerical schemes revealed the BDF- α method can be used to achieve low numerical damping. Experimental trials demonstrated the accuracy of the proposed model. The framework is adaptable to various designs of continuum robots with different operating principles as shown by the examples considered here.

In the future, the process of converting a statement of ODEs in the form of (12) to an efficient IVP computer routine may even be amenable to automatic code generation. We anticipate that our approach can be widely applied across the spectrum of continuum robot designs, and future work will address applications of this approach to the analysis and control of specific prototype robots, including concentric-tube robots.

Acknowledgements


The high-speed camera was provided and operated by Christopher Combs, Phillip Kreth, and John Schmisser of the University of Tennessee Space Institute. Kaitlin Oliver-Butler lent her experience for the design and fabrication of the tendon robot.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This material is based upon work supported by the National Science Foundation (under CMMI-1427122 as part of the NSF/NASA/NIH/USDA/DOD National Robotics Initiative and under NSF CAREER Award IIS-1652588). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

ORCID iDs

John Till  <https://orcid.org/0000-0002-4563-6884>

Caleb Rucker  <https://orcid.org/0000-0001-7181-1933>

References

- Antman SS (2005) *Nonlinear Problems of Elasticity*, Vol. 107. 2nd edn. New York, NY: Springer.
- Bergou M, Wardetzky M, Robinson S, Audoly B and Grinspun E (2008) Discrete elastic rods. *ACM Transactions on Graphics* 27(3): 1.
- Bertails F (2009) Linear time super-helices. *Computer Graphics Forum* 28(2): 417–426.
- Black CB, Till J and Rucker DC (2018) Parallel continuum robots: Modeling, analysis, and actuation-based force sensing. *IEEE Transactions on Robotics* 34(1): 29–47.
- Boyer F, Ali S and Porez M (2012) Macrocontinuous dynamics for hyperredundant robots: application to kinematic locomotion bioinspired by elongated body animals. *IEEE Transactions on Robotics* 28(2): 303–317.
- Bryson CE and Rucker DC (2014) Toward parallel continuum manipulators. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 778–785.
- Burgner-Kahrs J, Rucker DC and Choset H (2015) Continuum robots for medical applications: A survey. *IEEE Transactions on Robotics* 31(6): 1261–1280.
- Celaya EA and Jos J (2013) BDF- α : A multistep method with numerical damping control. *Universal Journal of Computational Mathematics* 1(3): 96–108.
- Falkenhahn V, Mahl T, Hildebrandt A, Neumann R and Sawodny O (2015) Dynamic modeling of bellows-actuated continuum robots using the Euler–Lagrange formalism. *IEEE Transactions on Robotics* 31(6): 1483–1496.
- Featherstone R and Orin DE (2008) Dynamics. In: *Springer Handbook of Robotics*. Berlin: Springer, pp. 35–65.
- Gatti-Bono C and Perkins N (2002) Physical and numerical modelling of the dynamic behavior of a fly line. *Journal of Sound and Vibration* 255(3): 555–577.
- Godage IS, Medrano-Cerda GA, Branson DT, Guglielmino E and Caldwell DG (2016) Dynamics for variable length multisection continuum arms. *The International Journal of Robotics Research* 35(6): 695–722.

- Godage IS, Wirz R, Walker ID and Webster RJ (2015) Accurate and efficient dynamics for variable-length continuum arms: A center of gravity approach. *Soft Robotics* 2(3): 96–106.
- Guennebaud G and Jacob B Others (2010) Eigen v3. <http://eigen.tuxfamily.org>.
- Hadap S (2006) Oriented strands: Dynamics of stiff multi-body system. *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 91–100.
- Holsapple R, Venkataraman R and Doman D (2003) A modified simple shooting method for solving two-point boundary-value problems. *IEEE Aerospace Conference Proceedings* 6: 2783–2790.
- Katzschmann RK, de Maille A, Dorhout DL and Rus D (2016) Cyclic hydraulic actuation for soft robotic devices. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3048–3055.
- Kugelstadt T and Schömer E (2016) Position and orientation based Cosserat rods. In: L Kavan and C Wojtan (eds.) *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, Zurich, Switzerland, pp. 169–178.
- Lan CC and Lee KM (2006) Generalized shooting method for analysing compliant mechanisms with curved members. *Journal of Mechanical Design* 128: 765–775.
- Lan CC, Lee KM and Liou JH (2009) Dynamics of highly elastic mechanisms using the generalized multiple shooting method: Simulations and experiments. *Mechanism and Machine Theory* 44(12): 2164–2178.
- Lang H, Linn J and Arnold M (2011) Multi-body dynamics simulation of geometrically exact Cosserat rods. *Multibody System Dynamics* 25(3): 285–312.
- Linn J, Lang H and Tuganov A (2012) Geometrically exact Cosserat rods with Kelvin–Voigt type viscous damping. *Mechanical Sciences* 4(1): 79–96.
- Mahony R, Kumar V and Corke P (2012) Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation Magazine* 19(3): 20–32.
- Marchese AD, Tedrake R and Rus D (2016) Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. *The International Journal of Robotics Research* 35(8): 1000–1019.
- Murray RM, Li Z and Sastry S (1994) *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press.
- Nocedal J and Wright SJ (2006) *Numerical Optimization*. New York: Springer.
- Oliver-Butler K, Till J and Rucker C (2019) Continuum robot stiffness under external loads and prescribed tendon displacements. *IEEE Transactions on Robotics*, in press.
- Polygerinos P, Correll N, Morin SA, et al. (2017) Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human–robot interaction. *Advanced Engineering Materials* 19(12): 1700016.
- Renda F, Cacucciolo V, Dias J and Seneviratne L (2016) Discrete Cosserat approach for soft robot dynamics: A new piece-wise constant strain model with torsion and shears. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5495–5502.
- Renda F, Giorelli M, Calisti M, Cianchetti M and Laschi C (2014) Dynamic model of a multibending soft robot arm driven by cables. *IEEE Transactions on Robotics* 30(5): 1109–1122.
- Rone WS and Ben-Tzvi P (2014) Continuum robot dynamics utilizing the principle of virtual power. *IEEE Transactions on Robotics* 30(1): 275–287.
- Rucker C (2018) Integrating rotations using non-unit quaternions. *IEEE Robotics and Automation Letters* 3(4): 2979–2986.
- Rucker DC and Webster RJ III (2011) Statics and dynamics of continuum robots with general tendon routing and external loading. *IEEE Transactions on Robotics* 27(6): 1033–1044.
- Rus D and Tolley MT (2015) Design, fabrication and control of soft robots. *Nature* 521(7553): 467–475.
- Sadati SMH, Naghibi SE, Walker ID, Althoefer K and Nanayakkara T (2018) Control space reduction and real-time accurate modeling of continuum manipulators using Ritz and Ritz–Galerkin methods. *IEEE Robotics and Automation Letters* 3(1): 328–335.
- Spillmann J and Teschner M (2009) Cosserat nets. *IEEE Transactions on Visualization and Computer Graphics* 15(2): 325–338.
- Tang W, Lagadee P, Gould D, Wan TR, Zhai J and How T (2010) A realistic elastic rod model for real-time simulation of minimally invasive vascular interventions. *Visual Computer* 26(9): 1157–1165.
- Till J, Bryson CE, Chung S, Orekhov A and Rucker DC (2015) Efficient computation of multiple coupled Cosserat rod models for real-time simulation and control of parallel continuum manipulators. In: *Proceedings IEEE Conference on Robotics and Automation*, Seattle, WA, pp. 5067–5074.
- Till J and Rucker DC (2017 a) Elastic rod dynamics: Validation of a real-time implicit approach. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada. IEEE, pp. 3013–3019.
- Till J and Rucker DC (2017 b) Elastic stability of Cosserat rods and parallel continuum robots. *IEEE Transactions on Robotics* 33(3): 718–733.
- Umetani N, Schmidt R and Stam J (2014) Position-based elastic rods. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Copenhagen, Denmark: Eurographics Association, pp. 21–30.
- Webster R, Romano J and Cowan N (2009) Mechanics of precurved-tube continuum robots. *IEEE Transactions on Robotics* 25(1): 67–78.

Appendix A. Index to multimedia extensions

Archives of IJRR multimedia extensions published prior to 2014 can be found at <http://www.ijrr.org>, after 2014 all videos are available on the IJRR YouTube channel at <http://www.youtube.com/user/ijrrmultimedia>

Table of Multimedia Extension

Extension	Media type	Description
1	Video	Animations of robot simulations and cantilever rod experiments
2	Code	Example cantilever problem .m script

Appendix B. Dynamic cantilever in MATLAB

```

1 function CantileverRod
2 %Parameters - Section 2.1
3 L = 0.4; N = 40; E = 207e9; r = 0.0012;
4 rho = 8000; vstar = [0;0;1];
5 g = 10*[-9.81;0;0]; %10x greater for effect
6 Bse = zeros(3); Bbt = zeros(3); C = zeros(3);
7 del_t = 0.005; STEPS = 50;
8 F_tip = [0;0;0]; M_tip = [0;0;0];
9 %Boundary Conditions - Section 2.4
10 p0 = [0;0;0]; h0 = [1;0;0;0];
11 q0 = [0;0;0]; w0 = [0;0;0];
12 %Dependent Parameter Calculations
13 A = pi*r^2; G = E/(2*(1+0.3)); ds = L/(N-1);
14 J = diag([pi*r^4/4 pi*r^4/4 pi*r^4/2]);
15 Kse = diag([G*A, G*A, E*A]);
16 Kbt = diag([E*J(1,1), E*J(2,2), G*J(3,3)]);
17 %BDF2 Coefficients
18 c0=1.5/del_t; c1 = -2/del_t; c2=0.5/del_t;
19 %Expressions extracted from simulation
    loop
20 Kse_plus_c0_Bse_inv = (Kse+c0*Bse)^-1;
21 Kbt_plus_c0_Bbt_inv = (Kbt+c0*Bbt)^-1;
22 Kse_vstar = Kse*vstar;
23 rhoA = rho*A; rhoAg = rho*A*g; rhoJ = rho*J;
24 %Initialize to straight configuration
25 % y and z are general variables as in Eq (12)
26 % y:=[p;h;n;m;q;w] and z:=[v;u]
27 y = [zeros(2,N); linspace(0,L,N);
        zeros(16,N)];
28 z = [zeros(2,N); ones(1,N); zeros(3,N)];
29 y_prev = y; z_prev = z;
30 %Main Simulation Loop - Section 2.4
31 visualize();
32 G = zeros(6,1); %Shooting method initial
    guess
33 for i = 2 : STEPS
34 %Set history terms - Eq (5)
35 yh = c1*y + c2*y_prev; zh = c1*z + c2*z_prev;
36 y_prev = y; z_prev = z;
37 %Midpoints are linearly interpolated for
    RK4
38 yh_int = 0.5*(yh(:,1:end-1)
    + yh(:,2:end));
39 zh_int = 0.5*(zh(:,1:end-1)
    + zh(:,2:end));
40 %Shooting method solver call
41 G = fsolve(@getResidual, G);
42 visualize();
43 end
44
45 %Function Definitions
46 function visualize()
47 plot(y(3,:),y(1,:)); title
    ('Cantilever Rod');
48 xlabel('z (m)'); ylabel('x (m)');
49 axis([0 1.1*L -0.55*L 0.55*L]);
50 grid on; daspect([1 1 1]); drawnow;
51 end
52
53 function E = getResidual(G)
54 %Reaction force and moment are guessed
55 n0 = G(1:3); m0 = G(4:6);
56 y(:,1) = [p0; h0; n0; m0; q0; w0];
57 %Fourth-Order Runge-Kutta Integration
58 for j = 1 : N-1
59 yj = y(:,j); yhj_int = yh_int(:,j);
60 [k1,z(:,j)] = ODE(yj,yh(:,j),zh(:,j));
61 [k2,~] = ODE(yj + k1*ds/2,yhj_int,
    zh_int(:,j));
62 [k3,~] = ODE(yj + k2*ds/2,yhj_int,
    zh_int(:,j));
63 [k4,~] = ODE(yj + k3*ds,yh(:,j+1),
    zh(:,j+1));
64 y(:,j+1) = yj + ds*(k1 + 2*(k2+k3)
    + k4)/6;
65 %y(:,j+1) = yj + ds*k1; %Euler's Method
66 end
67 %Cantilever boundary conditions
68 nL = y(8:10,N); mL = y(11:13,N);
69 E = [F_tip-nL; M_tip-mL];
70 end
71
72 function [ys,z] = ODE(y,yh,zh)
73 h = y(4:7); n = y(8:10); m = y(11:13);
74 q = y(14:16); w = y(17:19);
75 vh = zh(1:3); uh = zh(4:6);
76 %Quaternion to Rotation - Eq (10)
77 h1=h(1); h2=h(2); h3=h(3); h4=h(4);
78 R = eye(3) + 2/(h'*h)*...
79 [-h3^2-h4^2, h2*h3-h4*h1,
    h2*h4+h3*h1;
80 h2*h3+h4*h1, -h2^2-h4^2, h3*h4-h2*h1;
81 h2*h4-h3*h1, h3*h4+h2*h1,
    -h2^2-h3^2];
82 %Solved Constitutive Law - Eq (6)
83 v=Kse_plus_c0_Bse_inv*(R'*n+Kse_
    vstar-Bse*vh);
84 u=Kbt_plus_c0_Bbt_inv*(R'*m-Bbt*uh);
85 z=[v;u];
86 %Time Derivatives - Eq (5)
87 yt = c0*y + yh; zt = c0*z + zh;
88 vt = zt(1:3);
89 ut = zt(4:6);
90 qt = yt(14:16);
91 wt = yt(17:19);
92 %Weight and Square-Law-Drag - Eq (3)
93 f = rhoAg - R*C*q.*abs(q);
94 %Rod State Derivatives - Eq (7)
95 ps = R*v;
96 ns = rhoA*R*(cross(w,q) + qt) - f;
97 ms = R*(cross(w,rhoJ*w) + rhoJ*wt) -
    cross(ps,n);

```

```
98 qs = vt - cross(u,q) + cross(w,v);
99 ws = ut - cross(u,w);
100 %Quaternion Derivative - Eq (9)
101 hs = [ 0, -u(1), -u(2), -u(3);
102 u(1), 0, u(3), -u(2);
103 u(2), -u(3), 0, u(1);
104 u(3), u(2), -u(1), 0 ] * h/2;
105 ys = [ps;hs;ns;ms;qs;ws];
106 end
107 end
```