Andrew Choi
Department of Computer Science,
University of California Los Angeles,
Los Angeles, CA 90095
e-mail: asjchoi@cs.ucla.edu

Dezhong Tong
Department of Mechanical and
Aerospace Engineering,
University of California Los Angeles,
Los Angeles, CA 90095
e-mail: tltl960308@ucla.edu

Mohammad K. Jawed[1]
Assistant Professor
Department of Mechanical and
Aerospace Engineering,
University of California Los Angeles,
Los Angeles, CA 90095
e-mail: khalidjm@seas.ucla.edu

Jungseock Joo[1]
Assistant Professor
Department of Communication,
University of California Los Angeles,
Los Angeles, CA 90095
e-mail: jjoo@comm.ucla.edu

Implicit Contact Model for Discrete Elastic Rods in Knot Tying

Rod–rod contact is critical in simulating knots and tangles. To simulate contact, typically a contact force is applied to enforce nonpenetration condition. This force is often applied explicitly (Euler forward). At every time-step in a dynamic simulation, the equations of motions are solved over and over again until the right amount of contact force successfully imposes the nonpenetration condition. There are two drawbacks: (1) Explicit implementation brings numerical convergence issues. (2) Solving equations of motion iteratively to find this right contact force slows down the simulation. In this article, we propose a simple, efficient, and fully implicit contact model with high convergence properties. This model is shown to be capable of taking large time-steps without forfeiting accuracy during knot tying simulations when compared to previous methods. We introduce a new contact potential, based on a smoothed segment–segment distance function, that is an analytic function of the four endpoints of the two contacting edges. Since this contact potential is differentiable, we can incorporate its force (negative gradient of the energy) and Jacobian (negative Hessian of the energy) into the elastic rod simulation. [DOI: 10.1115/1.4050238]

Keywords: computational mechanics, elasticity

1 Introduction

Discrete differential geometry-based simulations have shown surprisingly successful performance in simulating slender structures, e.g., rods [1–4], viscous threads [2,5], ribbons [6], and plates/shells [7–9]. Discrete elastic rod (DER) algorithm [2,4]—originally developed in the computer graphics community to simulate hair, fur, and other filamentary structures in movies—has been borrowed by the engineering community to solve a variety of problems involving deployment of rods [10–12], propulsion of bacterial flagella [13–15], elastic gridshells [16–18], and self-assembly of carbon nanotubes [19]. One of the most interesting and challenging problems in these tasks is the simulation of knots, which introduces complex and intricate patterns of self-contact within an elastic rod. The mechanics of knots [20–26] tied in elastic rods is an intricate interplay of the elastic forces and friction. Therefore, when simulating knots, a fast, stable, and physically accurate contact model is desired.

Contact handling methods can generally be divided into three categories: impulse methods [27], constraint-based methods [28], and penalty methods [22,29]. DER, which implements a variation of Kirchhoff's rod model [30], has been used to handle contact during knot tying using a contact method proposed by Spillmann and Teschner [27]. This model resolves contact by computing the contact forces that will exactly lead to the desired, collision-free state. Although computationally efficient, unrealistic visual jittering during knot tying occurs for sufficiently large time-steps due to its explicit nature. Kaufman et al. [28] proposed a method capable of simulating large assemblies of elastic rods by adaptively adjusting its degree of nonlinearity. This method formulates frictional contact using discrete Signorini–Fischera conditions [31] and the maximal dissipation principle [32]. By adaptively incorporating sufficient nonlinearity into the collision response, more physically realistic results are produced compared to impulse methods. More recently, Li et al. [29] proposed an implicit time-stepping method that utilizes smooth barrier functions to simulate contact between deformable objects and induce deformations. Here, approximated functions are introduced to smooth the contact energy and frictional responses between two elements. Finally, Patil et al. [22] formulated contact forces in knots by using strain potential, while friction was formulated as a damping force. Although robust, many of these state-of-the-art methods [28,29] are difficult to implement, computationally expensive, and can be often overkill for simulating knots. This can be attributed to the fact that they are formulated in a way that robustly handles difficult contact scenarios such as high velocity impacts, which are often absent in knot tying. To combat this, we formulate an algorithm that is efficient, intuitive, easy to implement, and is specially catered toward knot tying. We then compare simulation results with the contact method proposed by Spillmann and Teschner (SPT) [27] as it is most comparable in implementation complexity and computation cost.

In this article, we introduce implicit contact model (IMC), a contact handling method for DER simulations [1,2] where the contact forces are handled implicitly. DER discretizes the elastic rod into a number of "nodes." Two consecutive nodes are connected by an "edge" To deal with contact, we define a contact energy, which will be used to derive the normal contact forces (responsible for enforcing nonpenetration) and Coulombic frictional forces. Instead of formulating this contact energy as a function of the distance between nodes, we formulate it as a function of the minimum distance between two edges, which results in more visually and physically realistic results. Figure 1 presents snapshots from our simulations of the knot tying process, where the two free ends of the "tails" of the knots are pulled. Throughout this article, we consider open overhand knots [20]; such knots can be described by the unknotting number n, related to the number of turns in the "braid"

[1]Corresponding authors.

Contributed by the Applied Mechanics Division of ASME for publication in the Journal of Applied Mechanics. Manuscript received October 15, 2020; final manuscript received February 12, 2021; published online March 10, 2021. Assoc. Editor: Pedro Reis.

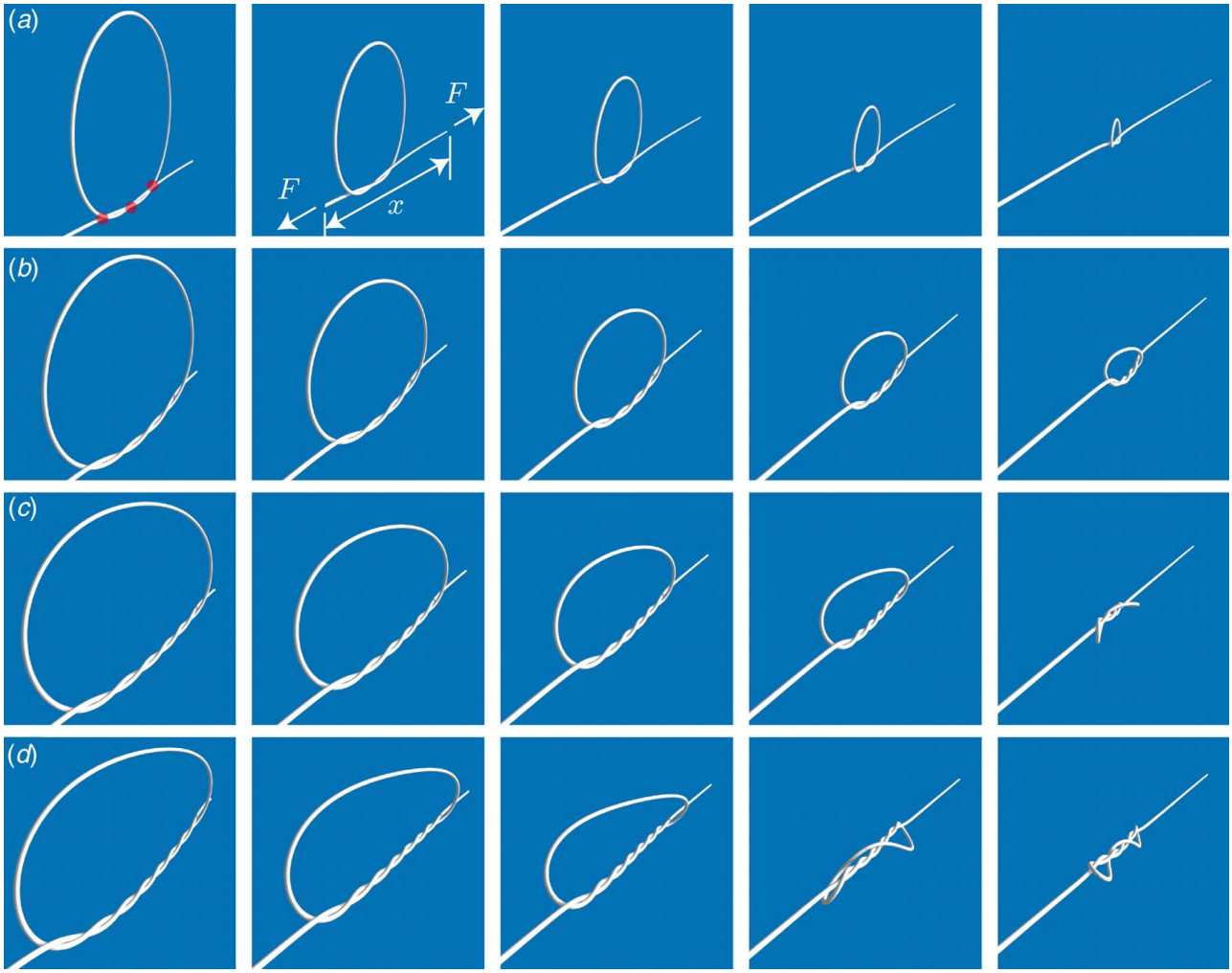

Fig. 1 Knot tying process using DER and IMC for knots of (*a*) $n = 1$, (*b*) $n = 2$, (c) $n = 3$, and (*d*) $n = 4$. The dots in the leftmost frame in (*a*) represent the crossing points of the braid. Unknotting number n is equal to $\frac{1}{2}\times$ (number of crossing points - 1). The end-to-end distance of a knot is $e = L - x$, where L is the total length of the rod and x is the distance between the two ends of the knot. The knot starts off in the configuration denoted by the leftmost column and is gradually pulled tight from both ends leading to the configuration shown in the rightmost column. Physical parameters are detailed in Sec. 3.2.

of the knot. Figures 1(*a*)–1(*d*) show knots with $n = 1, \ldots, 4$. Interestingly, when the knots with $n = 3$ and $n = 4$ are sufficiently tight, they undergo snap-through buckling and the "loop" of the knot suddenly transitions from a near-circular shape to a distorted configuration. The simulation can reliably capture this behavior.

This article is organized as follows. In Sec. 2, we discuss the methodology of the proposed version of DER with IMC as the contact model. Next, in Sec. 3, we undertake both theoretical validation of the IMC contact model and compare it to a well-established preexisting contact model in terms of pull force accuracy and runtime. Finally, conclusive remarks as well as potential future research directions are discussed in Sec. 4.

2 Methodology

In this section, first, we very briefly discuss the DER model (for a more in-depth introduction, please refer to Ref. [4]). Then, we formulate the IMC contact model and its intuitive integration into the existing DER algorithm. Finally, hyperparameters are explained in terms of their effect on algorithmic performance.

2.1 Discrete Elastic Rods. Under the DER framework, the centerline of a rod is discretized into N nodes, $\mathbf{x}_i$ $(1 \le i \le N)$ which corresponds to $N-1$ edges, $\mathbf{e}_i$ $(1 \le i \le N-1)$, where $\mathbf{e}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$. Each edge has an orthonormal adapted reference frame, $\{\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{t}^i\}$, and a material frame, $\{\mathbf{m}_1^i, \mathbf{m}_2^i, \mathbf{t}^i\}$ as shown in Fig. 2(*a*). The reference frame is updated through parallel transport in time [1,2], while the material frame can be obtained from the scalar twist angle θ^i with respect to the reference frame. Following this, for a given rod, the Cartesian coordinates of each node as well as the twist angle for each edge results in $4N-1$ total degrees-of-freedom (DOF), which can be represented by the following vector $\mathbf{q} = [\mathbf{x}_1, \theta_1, \mathbf{x}_2, \theta_2, \ldots, \mathbf{x}_{N-1}, \theta_{N-1}, \mathbf{x}_N]^T$; here, ^{T} denotes transposition operator. In this representation, the location of the ith node, $\mathbf{x}_i$, corresponds to the $4i-3$, $4i-2$, and $4i-1$th entries of $\mathbf{q}$.

The total elastic energy of an elastic rod, E^{elastic}, is composed of the stretching energy E^s, bending energy E^b, and twisting energy E^t as follows:

$$E^{\text{elastic}} = \sum_{i=1}^{N-1} E_i^s + \sum_{i=2}^{N-1} E_i^b + \sum_{i=2}^{N-1} E_i^t \tag{1}$$

where E_i^s is the stretching energy associated with the edge $\mathbf{e}_i$, E_i^b is the bending energy at node $\mathbf{x}_i$, and E_i^t is the twisting energy at node $\mathbf{x}_i$. For a rod with Young's modulus E, shear modulus G, area moment of inertia I, polar moment of inertia J, and cross-sectional

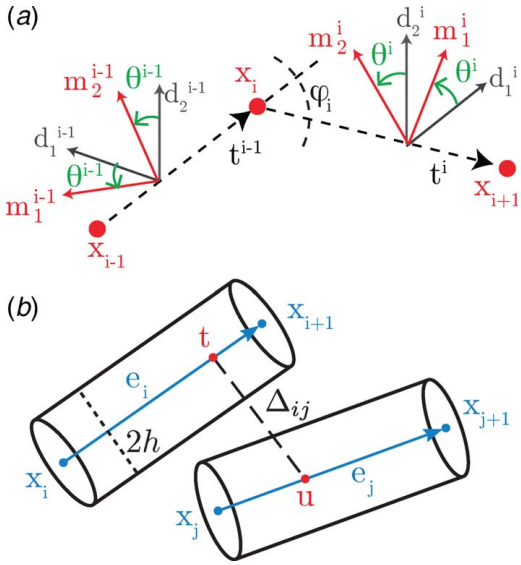

Fig. 2 *(a)* **Schematic diagram of a discrete rod and (*b*) two edges (and four nodes) involved in a contact**

area A, the energies are formulated as follows:

$$E_i^s = \frac{1}{2} EA \left(\frac{|\mathbf{x}_{i+1} - \mathbf{x}_i|}{|\bar{\mathbf{e}}_i|} - 1 \right)^2 |\bar{\mathbf{e}}_i|$$
$$E_i^b = \frac{1}{2} EI(|\kappa_i - \kappa_i^0|)^2 \frac{1}{dL_i} \tag{2}$$
$$E_i^t = \frac{1}{2} GJ\tau_i^2 \frac{1}{dL_i}$$

where $|\bar{\mathbf{e}}_i|$ is the length of edge $\mathbf{e}_i$ in undeformed state, κ_i is the curvature vector at node $\mathbf{x}_i$, κ_i^0 is the undeformed curvature for the same node, τ_i is the integrated twist at node $\mathbf{x}_i$, and $dL_i = (|\bar{\mathbf{e}}_{i-1}| + |\bar{\mathbf{e}}_i|)/2$ is the Voronoi length in the undeformed state. Typically, the rod is uniformly discretized and the length of each edge dL is the same throughout the rod. All these energies are functions of the configuration of the rod represented by $\mathbf{q}$ [4].

For each DOF q_i (i.e., ith element of $\mathbf{q}$), the elastic forces (for nodal positions) and elastic moments (for twist angles) can be written as follows:

$$F_i^{\text{int}} = -\frac{\partial}{\partial q_i}(E^{\text{elastic}}) \tag{3}$$

where F_i^{int} is the ith element of $(4N-1)$ sized elastic force vector, $\mathbf{F}^{\text{int}}$.

With this definition for the internal forces, the system of equations of motion is given as follows:

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{F}^{\text{int}} + \mathbf{F}^{\text{ext}} \tag{4}$$

where $\mathbf{F}^{\text{ext}}$ is the external force vector (e.g., contact forces, gravity), $\mathbf{M}$ is the diagonal mass matrix, and $\ddot{\mathbf{q}}$ is the second derivative of the DOFs with respect to time. In DER, backward Euler method is used to solve the $4N-1$ equations of motion to update the DOF vector $\mathbf{q}$. For the march from time t_i to $t_{i+1} = t_i + dt$, where dt is the time-step, Eq. (4) can be rewritten as follows:

$$\frac{\mathbf{M}}{dt}\left[\frac{\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)}{dt} - \dot{\mathbf{q}}(t_i)\right] - \mathbf{F}^{\text{int}}(t_{i+1}) - \mathbf{F}^{\text{ext}}(t_{i+1}) = 0 \tag{5}$$

where $\mathbf{q}(t_i)$ are the DOFs at time-step t_i and $\dot{\mathbf{q}}(t_i)$ are the velocities at time-step t_i. In the setup studied in this article, the three external forces are (1) the contact force $\mathbf{F}_c$, (2) the friction force $\mathbf{F}_{fr}$, and (3) the viscous force $\mathbf{F}_v$; and therefore, we have $\mathbf{F}^{\text{ext}} = \mathbf{F}_c + \mathbf{F}_{fr} + \mathbf{F}_v$. The formulation of these three forces will be discussed throughout the remainder of this article.

The old DOFs and velocities ($\mathbf{q}(t_i)$, $\dot{\mathbf{q}}(t_i)$) are known, and the task at hand is to compute the new DOFs and velocities ($\mathbf{q}(t_{i+1})$, $\dot{\mathbf{q}}(t_{i+1})$). As the Jacobian for Eq. (5) can be computed, the Newton–Raphson method is then used to solve for $\mathbf{q}$ (t_{i+1}) iteratively. Each element of the Jacobian matrix $\mathbf{J}$ at row i and column j is expressed as follows:

$$\mathbf{J}_{ij} = \frac{m_i}{dt^2}\delta_{ij} + \frac{\partial^2 E_{\text{elastic}}}{\partial q_i \partial q_j} - (\mathbf{J}_c)_{ij} - (\mathbf{J}_{fr})_{ij} - (\mathbf{J}_v)_{ij} \tag{6}$$

where $\mathbf{J}_c$, $\mathbf{J}_{fr}$, and $\mathbf{J}_v$ are square matrices representing the gradient of the three external forces—contact, friction, and viscous, respectively—with respect to the DOFs. Once $\mathbf{q}(t_{i+1})$ is known, the new velocity is simply $\dot{\mathbf{q}}(t_{i+1}) = \big(\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)\big)/dt$.

Normally, if the gradient of an external force, $\partial F_i^{\text{ext}}/\partial q_j$, cannot be analytically evaluated, this term is omitted during Newton iterations and that external force is considered "explicitly" (Euler forward). This generally requires a smaller time-step size dt, leading to larger computation time. Later, we will show that for IMC, the contact and friction Jacobians are analytically obtainable. Implicit treatment of contact and friction is a key contribution of this article.

2.2 Contact Model. Referring to Fig. 2(*b*), denote $\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1} \in \mathbb{R}^3$ as the Cartesian nodal coordinates of the ith and jth edges in a rod configuration. Next, denote an edge "combination" as the following vector concatenation: $\mathbf{x}_{ij} := (\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1})$. We denote the set of all valid edge combinations as $\mathcal{X}$; two consecutive edges are always in contact, and those combinations are not included in this *valid* combination $\mathcal{X}$.

From here, an arbitrary edge combination is denoted as simply $\mathbf{x}$ and all edge combinations are assumed to be valid: $\mathbf{x} \in \mathcal{X}$. The contact energy E is then expressed as a differentiable analytical expression, which takes the four nodes of the two contacting edges as inputs, $E(\mathbf{x}) : \mathbb{R}^{12} \to \mathbb{R}^1$. Under this formulation, we can see that the proposed contact energy is only dependent on the nodal coordinates of the discretized rod and not on the twist angles of the edges, θ, as the contact forces are computed based on the minimum distance, Δ, between two contacting edges.

Following this, to calculate Δ, we utilize an efficient and accurate algorithm for computing the minimum distance between two finite line segments in N dimensions proposed by Lumelsky [33]. Originally a piecewise function, the algorithm is then modified to become a twice differentiable smooth approximation. By using this completed expression, we can then obtain the negative gradient of the energy $-\nabla E(\mathbf{x}_{ij})$ as well as the negative Hessian $-\nabla^2 E(\mathbf{x}_{ij})$, which are then used to evaluate $\mathbf{F}_c$ and $\mathbf{J}_c$ in Eqs. (5) and (6). The gradient of $E(\mathbf{x})$ produces contact forces that act in the direction of the contact normal and whose magnitude varies with Δ, which results in physically realistic forces when dealing with rod–rod contact. Finally, as this method is essentially a penalty method, a stiffness parameter k_c is then used to scale $\mathbf{F}_c$ and $\mathbf{J}_c$ appropriately.

By producing the contact forces in this way, dynamic friction can be calculated according to Coulomb's friction law. In the past, previous methods [27,34] have been unable to simulate Coulomb friction due to the inability of obtaining its Jacobian. As we have access to the normal contact force Jacobian, we can calculate Coulombic friction forces as well as the friction Jacobian. By having access to the Jacobians of the normal and friction force, our contact model reliably converges even in complex contact states. As the cost of producing the Jacobian is relatively high and leads to having to solve a nonbanded matrix, we introduce a hybrid approach that ensures computational speed by only calculating the Jacobian when necessary.

To model contact energy, we compute the minimum distance Δ between two edges and feed this value into a smooth inverse ReLU function whose origin is based on the contact distance $2h$ as shown below where h is the cross-sectional radius of the rod and ce_k is a stiffness term that determines how sharp the curve is.

$$E = \frac{\log\left(1 + \exp\left(ce_k \cdot (2h - \Delta)\right)\right)}{ce_k} \tag{7}$$

Intuitively, this function starts to gradually increase the contact energy between two edges as Δ decreases while $\Delta > 2h$ and then sharply increases as Δ approaches $2h$. Although the gradients in the region $\Delta > 2h$ are nonzero, the inclusion of these "cushioning" forces greatly aid convergence and reduce any unwanted oscillating behavior that can often occur in penalty methods. The effect of these nonzero gradients are explained in detail in Secs. 3.3 and 4. Moving on, the key component of the contact model involves obtaining a differentiable analytical expression for Δ, which is difficult as computing the minimum distance between two line segments is a highly nonlinear and noncontinuous process. Next, we briefly describe Lumelsky's min-distance algorithm and a new modified smooth approximation that will be used as $\Delta(\mathbf{x})$.

Lumelsky's algorithm produces the minimum distance between two line segments in $\mathbb{R}^N$ and contains three noncontinuous components; we can eliminate one of them by simply taking the assumption that no edge can be reduced to a point. In other words, each edge must have a finite length greater than zero. With this condition eliminated, we now briefly layout the simplified min-distance algorithm for an arbitrary edge combination $\mathbf{x}_{ij}$. Note that here, we simply go over the steps of the algorithm. For further intuition on how exactly the algorithm is computing the min-distance, please refer to the original paper [33]. For notation, $\cdot$ is scalar multiplication and $\bullet$ is the dot product.

To start off the min-distance algorithm, first, we add another "edge" vector $\mathbf{e}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ to the ones already previously formulated: $\mathbf{e}_i$ and $\mathbf{e}_j$. With these vectors, we can then calculate the necessary intermediary values as follows, where i and j subscripts are left out for clarity:

$$\begin{aligned} D_1 &= \mathbf{e}_i \bullet \mathbf{e}_i \\ D_2 &= \mathbf{e}_j \bullet \mathbf{e}_j \\ S_1 &= \mathbf{e}_i \bullet \mathbf{e}_{ij} \\ S_2 &= \mathbf{e}_j \bullet \mathbf{e}_{ij} \\ R &= \mathbf{e}_i \bullet \mathbf{e}_j \\ \lambda &= D_1 \cdot D_2 - R^2 \end{aligned} \tag{8}$$

Next, denote $F(x)$ as a fix bound function, where all values above 1 are 1 and all values below 0 are 0. Anything between is outputted identically. This function is the piecewise function shown below and is first of the two remaining noncontinuous components.

$$F(x) = \begin{cases} 0 & x < 0 \\ x & 0 \le x \le 1 \\ 1 & x > 1 \end{cases} \tag{9}$$

The rest of the algorithm is as follows, where t, $u \in [0, 1]$ are the ratios that determine at which point along the length of each edge the connecting min-distance vector lies as shown in Fig. 2(*b*). With this in mind, the fix bound function $F(x)$ is used to ensure that these values do not go outside the appropriate range. As two edges become increasingly closer to parallel, λ approaches 0 and becomes 0 when perfectly parallel. To prevent division by zero, a piecewise function is used to describe the assignments of t.

$$\begin{aligned} t &:= \begin{cases} (S_1 \cdot D_2 - S_2 \cdot R)/\lambda & \lambda \neq 0 \\ 0 & \lambda = 0 \end{cases} \\ t &:= F(t) \\ u &:= (t \cdot R - S_2)/D_2 \\ u_f &:= F(u) \end{aligned} \tag{10}$$

The last noncontinuous component is a conditional assignment where if $u_f \neq u$ (i.e. $u < 0$ or $u > 1$), the t value is reassigned as follows.

$$\begin{aligned} t &:= (u_f \cdot R + S_1)/D_1 \\ t &:= F(t) \\ u &:= u_f \end{aligned} \tag{11}$$

Finally, Δ can be computed as follows:

$$\Delta_{ij} = \left\| \mathbf{e}_i \cdot t - \mathbf{e}_j \cdot u - \mathbf{e}_{ij} \right\| \tag{12}$$

2.3 Contact Energy, Its Gradient, and Hessian. To obtain the gradient and Hessian of the contact energy $E(\Delta(\mathbf{x}))$, $\Delta(\mathbf{x})$ must be differentiable. In Sec. 2.2, we introduced an algorithm that can compute Δ. Now, we modify the min-distance algorithm into a differentiable analytical expression. As Eq. (8) is analytical, the only necessary modifications lie in Eqs. (10) and (11). First, the fixbound function $F(x)$ can be modeled with the following smooth approximation, which is denoted by $H(x)$.

$$H(x) = \frac{\log\left(1 + \exp\left(k \cdot x\right)\right) - \log\left(1 + \exp\left(k \cdot (x - 1)\right)\right)}{k} \tag{13}$$

Here, k is a hyperparameter, which determines how stiff the curves are. A larger k value will result in a more accurate approximation but will result in "stiff" first and second derivatives leading to reduced convergence; thus, this value should be determined empirically. Next is the conditional reassignment in Eq. (11). As the reassignment only depends on whether $u_f \neq u$, this is equivalent to the reassignment only occurring when $u < 0$ or $u > 1$. To model this, we can use a boxcar function denoted $B(x)$, which consists of two compounded logistic functions.

$$B(x) = \frac{1}{1 + \exp\left(-k \cdot x\right)} - \frac{1}{1 + \exp\left(-k \cdot (x - 1)\right)} \tag{14}$$

Both functions $H(x)$ and $B(x)$ are plotted in Fig. 3 for a value of $k = 50$, which is the value used to produce the simulation results. This k value is a good tradeoff between accuracy and reliable convergence.

The last noncontinuous component of the algorithm lies in the piecewise function in Eq. (10), which is actually left noncontinuous. Although this introduces a piecewise function into the expression, this does not hurt convergence for the following reasons. First, it should be noted that λ will almost never equal exactly 0 due to floating point arithmetic. Therefore, the piecewise function is only required to prevent simulation crashes during simulation starts with perfectly parallel rod configurations. Furthermore, the numeric stability of this algorithm is maintained as whenever λ approaches zero, the numerator $S_1 \cdot D_2 - S_2 \cdot R$ also approaches zero by a similar magnitude, effectively avoiding numeric overflow problems [33]. In terms of performance, we have found that the produced Hessian is an excellent indicator of gradient direction when approaching or passing through parallel configurations when validating against finite difference for a wide variety of edge configurations.

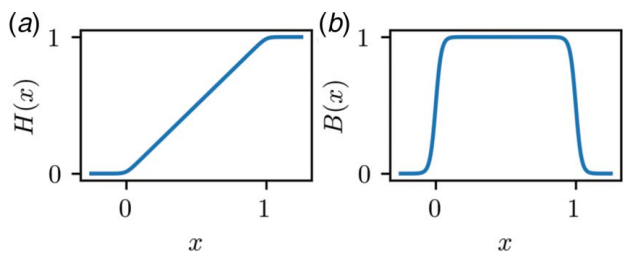

Fig. 3 (*a*) Smooth approximated fixbound function $H(x)$ which models the piecewise function in Eq. (9). (*b*) Boxcar function $B(x)$ which allows for an analytical conditional reassignment. Both functions are plotted with a stiffness parameter of $k = 50$.

With these two functions and the aforementioned prevention of division by zero, we can then replace Eqs. (10) and (11) with the following expression:

$$
\begin{aligned}
t_1 &= \begin{cases} (S_1 \cdot D_2 - S_2 \cdot R)/\lambda & \lambda \neq 0 \\ 0 & \lambda = 0 \end{cases} \\
t_2 &= H(t_1) \\
u_1 &= (t_2 \cdot R - S_2)/D_2 \\
u_2 &= H(u_1) \\
t_3 &= (1 - B(u_1)) \cdot (u_2 \cdot R + S_1)/D_1 + B(u_1) \cdot t_2 \\
\Delta_{ij} &= \left\| \mathbf{e}_i \cdot t_3 - \mathbf{e}_j \cdot u_2 - \mathbf{e}_{ij} \right\| \\
E_{ij} &= \frac{\log\left(1 + \exp\left(ce_k \cdot (2h - \Delta_{ij})\right)\right)}{ce_k}
\end{aligned}
\tag{15}
$$

As shown earlier, the min-distance Δ is fed into Eq. (7), which then leads to a fully differentiable analytical expression $E(\mathbf{x})$. It should be noted that an end-to-end differentiation of $E(\mathbf{x})$ ends up with an extremely large and complex equation when using symbolic differentiation [35]. Therefore, to greatly simplify the expression and improve computational efficiency, the gradient and Hessian for several of the intermediary algorithmic values are taken and then chain ruled together. Effectively, we can define $E(\mathbf{x})$ by the following equivalent functional:

$$
E(\mathbf{x}) := f(\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_{ij}, D_1, D_2, S_1, S_2, R, t_2). \tag{16}
$$

Since the inputs for $f(\cdot)$ are all functions of $\mathbf{x}$, chain rule tells us that we can obtain the gradient of the contact energy by the following:

$$
\begin{aligned}
\nabla E(\mathbf{x}) = &\frac{\partial f}{\partial \mathbf{e}_i} \cdot \frac{\partial \mathbf{e}_i}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{e}_j} \cdot \frac{\partial \mathbf{e}_j}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{e}_{ij}} \cdot \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{x}} \\
&+ \frac{\partial f}{\partial D_1} \cdot \frac{\partial D_1}{\partial \mathbf{x}} + \frac{\partial f}{\partial D_2} \cdot \frac{\partial D_2}{\partial \mathbf{x}} + \frac{\partial f}{\partial R} \cdot \frac{\partial R}{\partial \mathbf{x}} \\
&+ \frac{\partial f}{\partial S_1} \cdot \frac{\partial S_1}{\partial \mathbf{x}} + \frac{\partial f}{\partial S_2} \cdot \frac{\partial S_2}{\partial \mathbf{x}} + \frac{\partial f}{\partial t} \cdot \frac{\partial t}{\partial \mathbf{x}}
\end{aligned}
\tag{17}
$$

Here, we see that for any arbitrary edge combination $\mathbf{x}$, the produced force $-\nabla E(\mathbf{x})$ will be a vector of size 12 consisting of four concatenated three-dimensional contact force vectors for every node making up $\mathbf{x}$. These 12 elements contribute to the 12 entries of the $(4N-1)$-sized $\mathbf{F}_c$ vector located at the following positions: $4i-3$, $4i-2$, $4i-1$, $4(i+1)-3$, $4(i+1)-2$, $4(i+1)-1$, $4j-3$, $4j-2$, $4j-1$, $4(j+1)-3$, $4(j+1)-2$, and $4(j+1)-1$. Once the contact forces are computed for every contacting edge combination during a time-step, the force values are added to $\mathbf{F}_c$ and then incorporated into DER.

To obtain the Hessian, we simply take Eq. (17) and differentiate once again to obtain $-\nabla^2 E(\mathbf{x})$. Once obtained, the Hessian is added to the $(4N-1 \times 4N-1)$-sized Jacobian matrix $\mathbf{J}_c$ in a similar manner. The derivation of the Hessian can be done by using the product rule and its derivation is left out for brevity. See Sec. 5 for source code implementing these expressions.

2.4 Adding Friction. Just as we obtained contact force vectors of size $\mathbb{R}^{12}$, we produce frictional forces in a similar manner where given an edge combination $\mathbf{x}_{ij}$, we compute a friction force vector $\mathbf{F}_{fr}^{ij} \in \mathbb{R}^{12}$ according to Coulomb's dynamic friction law. For each edge combination, this 12-sized vector is added to the appropriate entries of $\mathbf{F}_{fr}$ (Eq. (5)).

Coulomb's friction law states the following:

(1) Frictional force is independent of velocity, and
(2) $\|\mathbf{F}_{fr}\| = \mu_k \cdot F_n$ during sliding,

where μ_k is the dynamic friction coefficient and F_n is the normal force (more details later in this section). From the contact model, we were able to derive $-\nabla E(\mathbf{x})$, which is equivalently the normal contact forces $\mathbf{F}_c$. As mentioned earlier, the gradient of the contact energy will always produce forces that are along the contact normal. Therefore, we can obtain F_n at the ith edge of the contact pair $\mathbf{x}_{ij}$ by simply summing up the contact forces on the ith and $i+1$th nodes: $F_n = \|\mathbf{F}_c^i + \mathbf{F}_c^{i+1}\|$. We can also use these contact forces to obtain the contact norm $\mathbf{n} = (\mathbf{F}_c^i + \mathbf{F}_c^{i+1})/F_n$. The direction of friction is then determined by the tangential relative velocity $\mathbf{v}_{\text{rel}}^{\text{T}}$ of edge i with respect to edge j. This can be obtained using the nodal velocities $\mathbf{v}_i$, $\mathbf{v}_{i+1}$, $\mathbf{v}_j$, and $\mathbf{v}_{j+1}$ as shown.

$$
\begin{aligned}
\mathbf{v}_i^e &= 0.5 \cdot (\mathbf{v}_i + \mathbf{v}_{i+1}) \\
\mathbf{v}_j^e &= 0.5 \cdot (\mathbf{v}_j + \mathbf{v}_{j+1}) \\
\mathbf{v}_{\text{rel}} &= \mathbf{v}_i^e - \mathbf{v}_j^e \\
\mathbf{v}_{\text{rel}}^{\text{T}} &= \mathbf{v}_{\text{rel}} - (\mathbf{v}_{\text{rel}} \bullet \mathbf{n}) \cdot \mathbf{n} \\
\hat{\mathbf{v}}_{\text{rel}}^{\text{T}} &= \mathbf{v}_{\text{rel}}^{\text{T}} / \|\mathbf{v}_{\text{rel}}^{\text{T}}\|
\end{aligned}
\tag{18}
$$

Finally, we can then formulate the friction force on the ith edge using Coulomb's friction equation as shown below. Here, we add a weight $\gamma \in [0, 1]$ to get rid of frictional forces between edges with extremely small relative velocities as this can cause unwanted behavior. To maintain differentiability, this weight is obtained using a smooth Heaviside step function with the tangential relative velocity as input and the stiffness k set to 50. Here, c determines the limit for the step transition and was set to 0.15 for our experiments. Note that this limit c must take into consideration the scaling of the model, which is explained in Sec. 2.5.

$$
\begin{aligned}
\gamma &= \frac{1}{1 + \exp\left(-k \cdot (\mathbf{v}_{\text{rel}}^{\text{T}} - c)\right)} \\
\mathbf{F}_{fr}^{ie} &= -\mu_k \cdot \gamma \cdot \hat{\mathbf{v}}_{\text{rel}}^{\text{T}} \cdot F_n
\end{aligned}
\tag{19}
$$

After obtaining the friction force on the ith edge $\mathbf{F}_{fr}^{ie}$, we can do the same for the jth edge as well to obtain $\mathbf{F}_{fr}^{je}$. The computed frictional forces are then equally distributed to each node and then concatenated four times to form the final friction vector $\mathbf{F}_{fr}^{ij} \in \mathbb{R}^{12}$.

$$
\mathbf{F}_{fr}^{ij} = (0.5 \cdot \mathbf{F}_{fr}^{ie}, \quad 0.5 \cdot \mathbf{F}_{fr}^{ie}, \quad 0.5 \cdot \mathbf{F}_{fr}^{je}, \quad 0.5 \cdot \mathbf{F}_{fr}^{je}) \tag{20}
$$

Once these friction force vectors are computed for every contacting edge combination during a time-step, we can then compute the friction force Jacobian matrix $\mathbf{J}_{fr}^{ij}$ as well. These are then added to $\mathbf{F}_{fr}$ and $\mathbf{J}_{fr}$ in exactly the same way as the contact energy gradient and Hessian as described in Sec. 2.3.

It should be noted that several simplifications were made for the friction model. First, the relative velocities were computed using the midpoint of the edges rather than the contact points. Likewise, the friction forces were evenly distributed rather than being dependent on the contact points. This was done as it greatly simplifies the friction force Jacobian, leads to improved convergence, and does not have any noticeable effects so long as the rod is sufficiently discretized.

Furthermore, we treat friction semi-explicitly by using the known velocities from the previous time-step. This allows the friction direction to remain constant during Newton iterations, which improves convergence considerably. Although a fully implicit scheme is possible, computing the necessary contact Hessian on every iteration is costly and the overall speed of the algorithm greatly benefits from this formulation.

In terms of limitations, this method clearly does not enforce static friction and so should only be used for continuous sliding scenarios such as knot tying. Furthermore, friction occurring due to the rod twist θ is not modeled. When an edge undergoes enough twist and is in contact with a receiving edge, friction forces occur slightly off the centerline of this receiving edge. As our model assumes that all friction occurs precisely on the centerline and formulates all contact only using $\mathbf{x}$, such friction–twist coupling is neglected.

Finally, the produced friction forces can possibly overtake the pull forces when the pull speed is very low leading to unrealistic sliding in the opposite direction. This must be remedied by pulling at a sufficiently high speed.

Algorithm 1 Implicit contact method.

Parameter: k_c, δ, ω, S
Input: $\mathbf{x}, \mathbf{v}, n$; // from DER
Output: $\mathbf{F}_c, \mathbf{J}_c, \mathbf{F}_{fr}, \mathbf{J}_{fr}, \alpha$
Function: IMC($\mathbf{x}, \mathbf{v}, n$):
  scale $\mathbf{x}$ and $\mathbf{v}$ by S
  if $n == 0$ **then** // run only on first iter
    $\mathcal{C}, md \leftarrow$ collisionDetection($\mathbf{x}, \delta$)
    $k_c \leftarrow$ updateConStiffness(k_c, md)
  end
  if $n < \omega$ **then** // compute only forces
    $\mathbf{F}_c \leftarrow$ genContact($\mathcal{C}$)
    $\mathbf{F}_{fr} \leftarrow$ genFriction($\mathcal{C}, \mathbf{v}, \mathbf{F}_c$)
    $\mathbf{J}_c \leftarrow$ zero square matrix
    $\mathbf{J}_{fr} \leftarrow$ zero square matrix
  end
  else // compute Jacobian for convergence
    $\mathbf{F}_c, \mathbf{J}_c \leftarrow$ genContact($\mathcal{C}$)
    $\mathbf{F}_{fr}, \mathbf{J}_{fr} \leftarrow$ genFriction($\mathcal{C}, \mathbf{v}, \mathbf{F}_c, \mathbf{J}_c$)
  end
  $\mathbf{F} \leftarrow k_c \cdot (\mathbf{F}_c + \mathbf{F}_{fr})$
  $\mathbf{J}_c \leftarrow k_c \cdot \mathbf{J}_c$
  $\mathbf{J}_{fr} \leftarrow k_c \cdot \mathbf{J}_{fr}$
  $\alpha \leftarrow$ newtonDamper(n)
  return $\mathbf{F}_c, \mathbf{J}_c, \mathbf{F}_{fr}, \mathbf{J}_{fr}, \alpha$

Algorithm 2 Discrete elastic rods.

Input: $\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$
Output: $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$
Require: boundary conditions → free
Function: DER($\mathbf{q}(t_i), \dot{\mathbf{q}}(t_i)$)
  Guess: $\mathbf{q}^{(1)} \leftarrow \mathbf{q}(t_i)$
  $n \leftarrow 0,\ \epsilon \leftarrow \infty$
  while $\epsilon >$ *tolerance* **do**
    $\mathbf{F}^{\text{int}} \leftarrow$ genForces($\cdot$)
    $\mathbf{J}^{\text{int}} \leftarrow$ genJacobian($\cdot$) ; // $\frac{\partial^2 E_{\text{elastic}}}{\partial q_i \partial q_j}$ in Eq. (6)
    $\mathbf{F}_c, \mathbf{J}_c, \mathbf{F}_{fr}, \mathbf{J}_{fr}, \alpha \leftarrow$ IMC($\mathbf{x}^{(n)}, \dot{\mathbf{q}}(t_i), n$); // Alg 1
    $\mathbf{F}_{\text{der}} \leftarrow$ left side of Eq. (5)
    $\mathbf{J}_{\text{der}} \leftarrow$ left side of Eq. (6)
    $\mathbf{F}_{\text{free}} \leftarrow \mathbf{F}_{\text{der}}$(free) ; // Downsize to only include free DOFs
    $\mathbf{J}_{\text{free}} \leftarrow \mathbf{J}_{\text{der}}$(free, free)
    $\Delta\mathbf{q}_{\text{free}} \leftarrow \mathbf{F}_{\text{free}}/\mathbf{J}_{\text{free}}$; // Solve $\mathbf{J}_{\text{free}}\Delta\mathbf{q}_{\text{free}} = \mathbf{F}_{\text{free}}$
    $\mathbf{q}^{(n+1)}$(free) $\leftarrow \mathbf{q}^{(n)}$(free) $- \alpha \cdot \Delta\mathbf{q}_{\text{free}}$
    $\epsilon \leftarrow \|\mathbf{F}_{free}\|$; // update error
    $n \leftarrow n + 1$
  end
  $\mathbf{q}(t_{i+1}) \leftarrow \mathbf{x}^{(n)}$
  $\dot{\mathbf{q}}(t_{i+1}) \leftarrow (\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i))/\Delta t$
  return $\mathbf{q}(t_{i+1}), \dot{\mathbf{q}}(t_{i+1})$

2.5 Full Algorithm. In addition to the force and Jacobian generation, there are several additional steps to the IMC algorithm that are explained in this section as well as several hyperparameters that must be properly tuned for optimal performance and convergence which are listed as follows:

(1) δ, the collision limit,
(2) k_c, the contact stiffness,
(3) ce_k, the contact energy stiffness, and
(4) ω, the number of iterations before the hybrid algorithm computes the Jacobian.

First, the nodal coordinates are scaled by a scaling factor $S = 1/h$, so that the adjusted rod radius equals a unit value of 1. To ensure that the distance at which two edges experience a force is very close to the rod surface, the following energy function is used.

$$E = \frac{\log\left(1 + \exp\left(ce_k \cdot (2 - \Delta/h)\right)\right)}{ce_k} \tag{21}$$

Following this, the collision limit δ is the threshold value used to determine when two edges are "in contact." This value is fed into a collision detection algorithm, which returns all edge combinations falling into this threshold, which is denoted by the following set:

$$\mathcal{C} = \left\{\mathbf{x}_{ij} \in \mathcal{X} \mid \Delta_{ij}/h < 2 + \delta\right\} \tag{22}$$

The minimum distance from this set is denoted by $md = \min_{\mathbf{x}\in\mathcal{C}} \Delta(\mathbf{x})$, which will be used later to adjust the contact stiffness k_c accordingly.

The collision limit δ must be chosen carefully as a higher δ value results in additional computation due to more qualifying edge combinations, whereas a δ value that is too low will produce nonsmooth gradients that hamper convergence.

A good way to determine a proper δ value is to observe the plotted contact energy function from Eq. (21) for a chosen ce_k value. By observing the contact energy curve, the point at which the generated gradients are ≈ 0 can be found when the slope of the curve is nearly flattened out. Choosing a δ value that encompasses this region ensures that the generated gradients are sufficiently smooth. An example of this process can be found in Fig. 4.

The stiffness of the contact forces is determined by the contact energy stiffness value ce_k. As this value becomes higher, the region above the contact surface at which two edges experience a force decreases. This leads to stiff contact, which is more physically accurate as realistically the contact energy should be zero whenever $\Delta > 2h$ and shoot to ∞ as soon as $\Delta = 2h$. Conversely, a ce_k value that is too large will result in convergence issues, while a ce_k value that is too low will have excessive oscillations between the contact bodies. A value that was found to be a good compromise between physical accuracy and convergence was $ce_k = 50$.

The next hyperparameter that must be specified is the contact stiffness k_c. Not to be confused with the approximation stiffness k in Eqs. (13) and (14), the contact stiffness k_c is a scalar value that

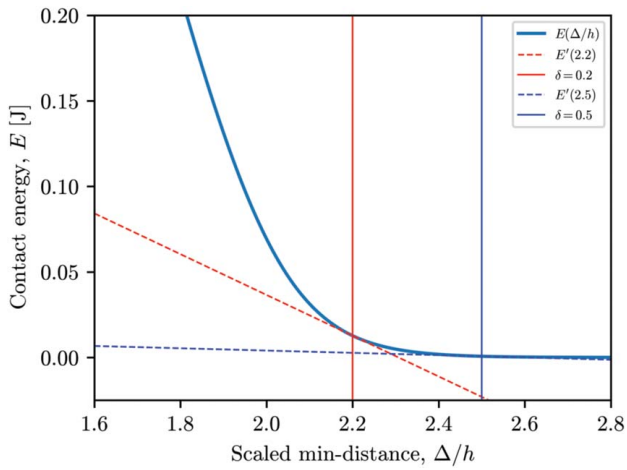

Fig. 4 Contact energy curve of Eq. (21) for $ce_k = 10$. Scaling by $S = 1/h$ results in the curve being centered at a collision length of $2h_s = 2.0$. As denoted by the vertical solid line, the curve starts to flatten out to zero at $\Delta/h = 2.5$, which indicates that generated gradients are ≈ 0. Therefore, a collision limit $\delta = 0.5$ would be suitable. Conversely, $\delta = 0.2$ as denoted by the vertical solid line would result in the Newton solver potentially failing to converge due to nonsmooth force generation.

is used to scale the contact force and Jacobian. This value is adaptively readjusted every time-step to ensure that excessive hovering or penetration is minimized, and so, only the initial value must be specified, which can be found empirically. This initial k_c should be reasonably close to the value that would result in $md = 2h$. In our experiments, we employed a simple algorithm that decreased or increased k_c by a fraction of a percent depending on whether md was $< 2h - \epsilon_1$ or $> 2h + \epsilon_2$, where ϵ_1 and ϵ_2 are limits indicating an acceptable contact range. This algorithm updates k_c only when md is deviating from the region defined by $[2h - \epsilon_1, 2h + \epsilon_2]$ and is otherwise left constant to prevent overshooting.

As mentioned earlier, this algorithm applies a hybrid approach in which the Jacobian of the contact and friction forces are only computed once the number of iterations passes a limit ω. This is done because often convergence can be quickly obtained even without the contact Jacobian, and with the absence of the contact Jacobian, the overall Jacobian matrix remains a banded matrix, which can be solved significantly faster. Although the Jacobian results in a decrease in iteration count, the consequent increase in computational time outweighs this benefit as IMC is able to reliably converge rapidly without it for a majority of time-steps. With this in mind, the Jacobian is crucial for completing volatile contact states with high velocities and impacts that could otherwise end the simulation prematurely. Therefore, this hybrid approach maximizes computational speed while ensuring that the simulation can consistently reach the next time-step during especially difficult contact scenarios such as inversion and the initialization phase where the rod rapidly reverts to its lowest energy state. The limit ω should be chosen empirically, so that the Jacobian is only generated when necessary. In our experiments, we used an ω of 20.

Finally, although not a hyperparameter, a damping coefficient α is used to reduce the step size of the Newton solver as the number of iterations increase for a particular time-step. For this, a simple decaying algorithm is used which reduces α by a factor of 2 every other iteration. Overall, aside from the collision detection algorithm (*which is only performed on the first iteration of every time-step*), the time complexity of IMC with and without Jacobian generation is $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$, respectively, where n is the number of collisions detected. The full contact algorithm and its implementation in DER can be seen in Algorithms 1 and 2, respectively. In Algorithm 2, the term "free" are the indices that correspond to the free degrees-of-freedom of the elastic rod. The remaining degrees-of-freedom are "fixed" and depends on the user defined boundary conditions.

3 Results

In this section, we first validate the correctness of our contact model against theory. Afterward, to observe the benefits of using IMC, we compare simulation results with the contact model (SPT) proposed by Spillmann and Teschner [27] for knots of unknotting numbers $n \in [1, 4]$, which are shown in Fig. 1. For both methods, the contact model is used to simulate knot tying until a mutual termination state, which is shown in the rightmost column of Fig. 1. Finally, we compare the computational efficiency and convergence properties between the two methods. All simulations for IMC used a contact energy stiffness $ce_k = 50$ and a collision limit $\delta = 0.15$, which was obtained using the method mentioned in Fig. 4.

3.1 Damping for Stability. Since IMC is inherently a penalty method, the inclusion of damping forces greatly aid the stability of the contact model. A simple viscous damping force is applied to the elastic rod by applying a force $\mathbf{F}_v$ on a node as denoted in Eq. (23), where η(Pa · s) is a viscosity coefficient and dL is the node's Voronoi length.

$$\mathbf{F}_v = -\eta \cdot \left(\frac{\mathbf{q}(t_{i+1}) - \mathbf{q}(t_i)}{dt} \right) \cdot dL \tag{23}$$

The Jacobian of this damping force is represented as follows:

$$\mathbf{J}_v = -\frac{\eta \cdot dL}{dt} \mathbb{I} \tag{24}$$

where $\mathbb{I}$ is the square identity matrix of size $(4N - 1)$. This damping force was also added to SPT for fair comparison. For all simulations, a viscosity value of $\eta = 0.01$ Pa · s was used unless otherwise specified.

3.2 Theoretical Validation. Coulomb's friction law states that friction is independent of velocity. To show that our model abides by this, we plot the pull forces F in Newton (N) when tightening a knot with unknotting number $n = 2$ with friction coefficient $\mu_k = 0.10$ for pull speeds u of 3, 6, and 9 mm/s (*pulled from both ends*). A 1 m long rod with cross-sectional radius $h = 1.6$ mm, density $\rho = 1180$ kg/m^3, and Young's Modulus $E = 1.8e5$ Pa is used and is discretized into 301 nodes. We plot a regularized pull force Fh^2/EI against $\sqrt{h/R}$, where $EI = \pi E h^4/4$ is the flexural modulus and R is the radius of the knot loop. The radius R can be computed using the knot circumference from Fig. 1 as $R = e/(2\pi)$. As Fig. 5(*a*) shows, the magnitude of the $n = 2$ pull forces is approximately the same for all pull speeds. Dynamic friction force in Coulomb's model is independent of velocity, and therefore, this observation supports the physical correctness of the generated friction forces.

Next, the pull forces for a trefoil knot ($n = 1$) are compared with the predictive model by Audoly et al. [23]. This model states the following theoretical equivalence, where σ is a numerical constant ($\sigma = 0.492$ for trefoil knots) and $\epsilon = \sqrt{h/R}$. The $\pm$ term is the frictional component from tightening (+) and loosening (−).

$$\frac{Fh^2}{EI} = \frac{\epsilon^4}{2} \pm \mu_k \sigma \epsilon^3 \tag{25}$$

Using the same rod properties as mentioned earlier, a trefoil knot is tightened and then loosened at 3 mm/s using IMC. Here, we reduce η to 0.0005 as loosening can be sensitive to small forces. As shown in Fig. 5(*a*), the recorded pull forces roughly follow the curves of the predictive model albeit with some displacements when tightening increases sufficiently. This can be attributed to imperfections in the predictive model as Eq. (25) is an elegant lightweight solution that does not perfectly model friction when the knot becomes sufficiently tight, i.e., $\sqrt{h/R}$ becomes large. Still, aside from the displacement, the pull forces follow the rate of increase/decrease of the predictive model well, which is a good indicator of correctness.

Furthermore, in Fig. 5(*a*), during the loosening, the trefoil knot can be seen being locked by friction at a point $\epsilon_0 = \sqrt{h/R_0} \approx 0.1285$. From the right-hand side of Eq. (25), we can rearrange the terms to obtain the following equivalence.

$$\mu = \frac{1}{2\sigma} \epsilon_0 = 1.02 \sqrt{h/R_0} \tag{26}$$

When plugging in our obtained ϵ_0 value, we obtain $\mu_{\text{theory}} = 0.1306$, which is reasonably close to the friction coefficient used in simulation, $\mu_k = 0.10$.

Finally, we show that the pull forces monotonically increase with μ_k in Fig. 5(*b*). Here, we consider the $n = 4$ case. When recording the pull forces for μ_k of 0.1, 0.2, and 0.3, we can see that the rate of increase is held constant, while the magnitudes monotonically increase. Furthermore, we can see that as friction increases, the inversion point occurs sooner, which indicates that the point of inversion is highly dependent on μ_k.

3.3 Pull Force Accuracy. Simulations of knot pulling are performed for both IMC and SPT using a pull speed $u = 6$ mm/s, time-step $dt = 0.5$ ms, friction coefficient $\mu_k = 0.10$, and the same rod properties from Sec. 3.2. With these settings, knot tying was simulated for each method with the experienced pull forces being recorded at each

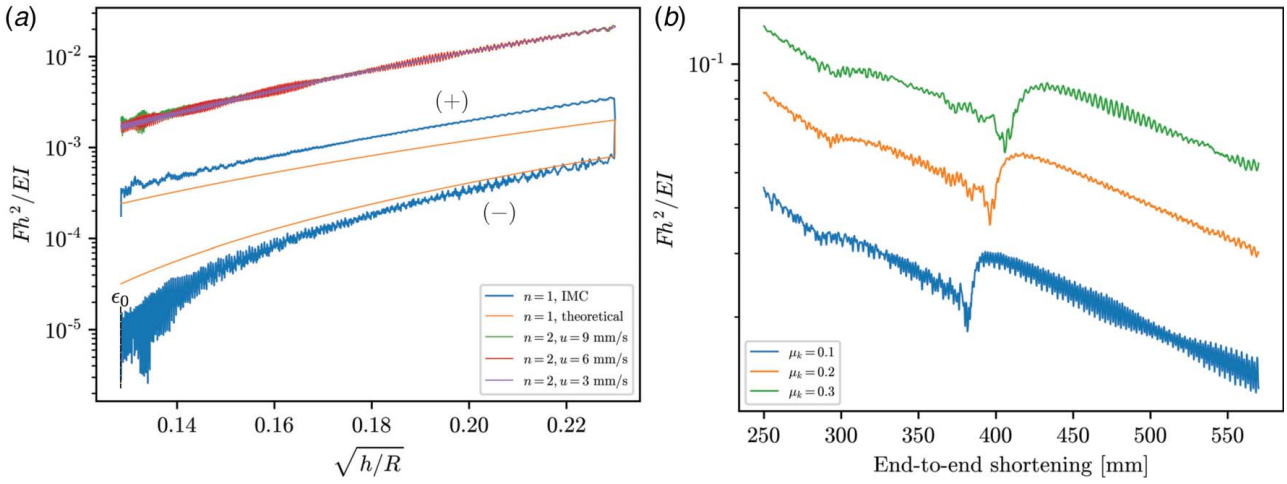

Fig. 5 Model validation and comparison with theory. (*a*) In the above half is the pull force comparison for $n=2$ with different pull speeds u and friction $\mu_k=0.10$. As shown, the pull forces are identical in magnitude indicating that friction is independent of velocity. In the bottom half is the $n=1$ simulation data comparison with Audoly's predictive model shown in Eq. (25). Starting with an open trefoil knot, the knot is tightened and then loosened with $\mu_k=0.10$, which is indicated by (+) and (−), respectively. A moving average of 200 steps is used for the $n=1$ pull forces to minimize visually large variations caused by the lower end of the log scale. (*b*) Pull force comparison for $n=4$ and different μ_k values. There is a clear monotonically increasing relationship between pull forces and μ_k. In addition, inversion (indicated by the sudden drop in pull force) occurs earlier for higher μ_k values as expected. A pull speed of 6, 9, and 12 mm/s was used for $\mu_k=0.1$, 0.2, and 0.3, respectively. This was necessary as higher friction coefficients induced the limitation mentioned in Sec. 2.4 for higher pull speeds.

time-step. Figure 6(*a*) plots these pull forces with respect to the end-to-end shortening. Here, we see that for both IMC and SPT, as the end-to-end shortening decreases (*knot is pulled tight*), the pull forces increase identically as expected. They also increase in magnitude as the unknotting number n goes up, and for $n=3$ and $n=4$, inversion occurs, which is indicated by the sudden drop in pull force. This is shown visually in Fig. 6(*b*) for $n=4$.

The largest difference between the methods can be seen in the considerable amount of force jittering by SPT, which occurs due to the time-step being too large. This leads to visually unrealistic results, where the knot continuously "trembles" while being tied. Conversely, IMC produces much smoother pull forces, which directly translate to visually smooth simulations for the same time-step size. One caveat that should be noted is that SPT ensures exact nonpenetration during contact, whereas IMC allows small penetrations and hovering to occur, which is physically unrealistic. Still, through the adaptive contact stiffness and the inclusion of damping, IMC contains any hovering to stay within 20 μm above the contact surface, while penetrations rarely exceed 5 μm (for comparison, cross-sectional radius is $h=1.6$ mm). Thus, this minor error is largely indiscernible both visually and physically.

Overall, the enforcement of nonpenetration at every time-step limits the maximal time-step that SPT can take without experiencing significant force jittering, whereas IMC produces smooth results in exchange for contact varying within a small region.

3.4 Runtime. Next, we discuss runtime comparisons and convergence characteristics. Here, we show that in addition to IMC producing smoother results than SPT at sufficiently large time-steps, IMC is also computationally competitive. Both models use the same DER implementation, which is written in C++. One

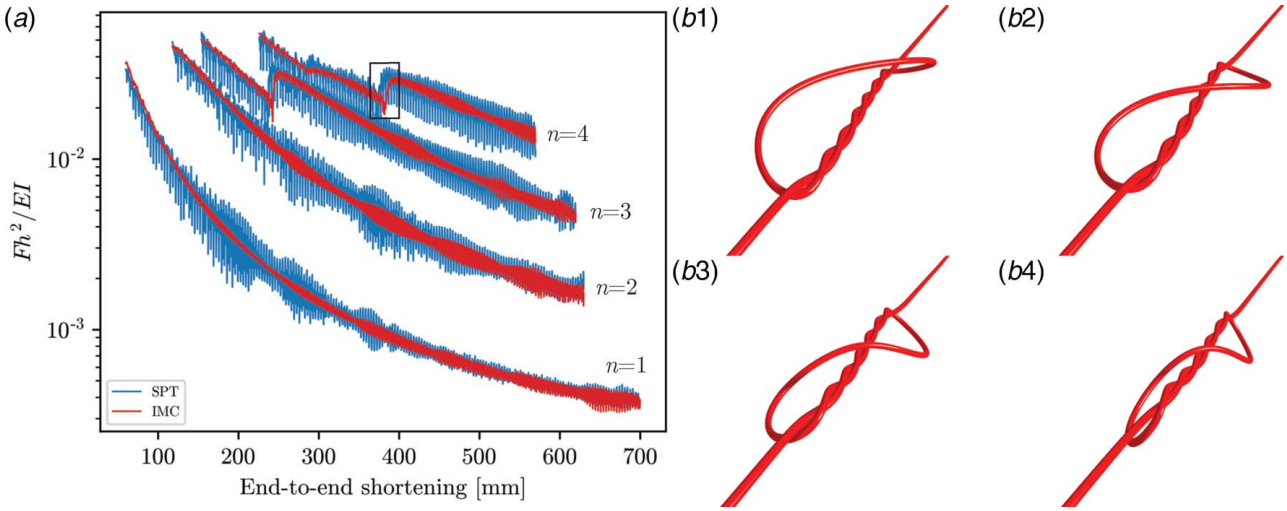

Fig. 6 Simulation results comparison: (*a*) pull force comparison for unknotting numbers 1 through 4 using SPT [27] and IMC with $u=6$ mm/s, $dt=0.5$ ms, and $\mu_k=0.10$ and (*b*) inversion occurring for $n=4$ and the corresponding drop in pull forces shown by the border box in (*a*).

Table 1 IMC versus SPT [27] runtime data, $\mu_k = 0.10$, $dt = 0.5$ ms, pull speed = 6 mm/s

Model	n	AIPT	ATPI (ms)	Iters	Time (s)
IMC	1	2.23	2.24	245,824	551
	2	2.67	2.20	245,830	542
	3	3.04	2.18	261,707	570
	4	3.24	2.12	239,987	509
SPT	1	8.25	2.30	907,541	2085
	2	9.28	2.26	853,935	1931
	3	10.25	2.26	881,907	1990
	4	10.94	2.21	810,160	1790

Note: AIPT, average iterations per time-step; ATPI, average time per iteration; Iters, the total number of Newton's iterations that were necessary to complete the simulation. The time is the total computational time to completion.

discrepancy is that while SPT is directly implemented into DER in C++, IMC is written entirely in PYTHON for ease of prototyping. To minimize any performance differences arising from using different languages, we employed an LLVM-based PYTHON JIT compiler [36] for certain computational intensive portions of the code such as the chain ruling procedure from Eq. (17). The computational time for each iteration for both contact methods is recorded using the ctime library. This timing was done so that any computational time arising from IO usage recording the data and rendering the rod graphically were excluded. One thing to note is that the timings for IMC include all of the shared memory overhead between the C++ and PYTHON programs. Therefore, a significant performance increase for IMC can be expected when fully implemented in C++ and compiled without this overhead. Finally, both methods used identical Newton tolerances for all simulations, and all simulations were run on a single thread on an Intel Core i7-9700K 3.60GHz CPU.

In Table 1, the runtime, total number of iterations, average iterations per time-step (AIPT), and average time per iteration (ATPI) are reported. For all knots, we can immediately see that IMC converges with a noticeably smaller amount of iterations than SPT. While the ATPI between both methods are about equal, the simulations for IMC finish approximately 4 × faster than SPT for all knots. For both methods, AIPT increases as the knot complexity n increases as expected.

One observation was that the number of iterations to complete a time-step increased for IMC as the knot loop became extremely small and/or tightly inverted. Dynamically reducing the time-step solves this issue, but this was left out so that all reported results were for a constant time-step. For the same time-step size, SPT can more reliably converge when the knot loop becomes very small albeit with the large force jittering still present. Therefore, it may be worth investigating performance differences between the two methods when SPT uses a time-step size that is small enough to compete with the smoothness of IMC, while IMC starts at a larger time-step and dynamically reduces as convergence becomes an issue.

Still, even for a constant time-step size of 0.5 ms, IMC is seen to be more computationally efficient when pulling the knots close to taut and for the majority of the knot tying procedure, takes far less iterations to converge. This difference should only increase with IMC being implemented directly into DER in C++.

4 Concluding Remarks

In this article, we introduced a novel contact model for DER simulations in which the contact forces are handled implicitly using a smooth penalty force formulation. This model was shown to be able to model dynamic friction and simulate knot tying accurately. We showcased comparisons with previous methods [27] and concluded that our method was capable of producing more visually smooth realistic results, while also producing physically accurate data. Furthermore, our method was stable, computationally competitive, and took minimal iterations to converge.

Although this method has shown promising results for knot tying simulations, it is not without its drawbacks. First, IMC is largely unsuitable for contact scenarios that frequently involve sudden excessively large velocities and impacts as these will result in excessive penetration and possible overshoot of the contacting body without appropriate damping. Where IMC shines is scenarios with constant sliding contact such as knot tying where contact forces may or may not gradually rise.

In addition, to be absolutely physically realistic, contact forces should equal zero when $\Delta > 2h$, whereas the usage of Eq. (7) produces a force $F_c \neq 0$ when $\Delta > 2h$. It has been shown that employing a smooth approximation such as this can greatly improve convergence in penalty methods [25], which is one of the goals for this algorithm. As the contact forces approach zero as ce_k becomes sufficiently large while $\Delta > 2h$, this is not a significant problem as discussed in Sec. 3.3. Still, the fact that F_c is not exactly zero is something to consider. Finally, the large amount of hyperparameters leaves something to be desired as tuning may be necessary when switching between rod properties, which can be time consuming.

Some possible future research directions involve modifications that further improve the realism of the contact model. One of these pertains to the stiffness parameter k_c. A simplification that was employed is the usage of a global stiffness parameter. More realistic contact can be simulated using local stiffness parameters as shown previously in Ref. [25] in exchange for more computation. This addition may also fix the problem where friction forces overtake the pull force of the knot if the pull speed is too low as mentioned in Sec. 2.4.

Another research direction is to extend IMC to noncircular cross sections. Currently, IMC is only capable of handling circular cross sections due to computing the minimum distance between edges using purely the centerlines. Extending IMC to noncircular cross sections while maintaining differentiability is nontrivial but will be necessary to simulate more complexly shaped rods. Furthermore, handling more complex knots will require IMC to take into consideration cross section deformation as IMC currently assumes that the cross-sectional shape remains constant.

Finally, another challenging problem that remains is the proper modeling of static friction. Although dynamic friction is adequately modeled, ultimately, subtle frictional threshold events such as the transition from sticking to sliding and vice-versa are necessary to simulate realistic contact outside the realm of constant sliding.

5 Source Code

The source code for IMC as well as the initial knot configurations for all conducted simulation tests can be found at https://github.com/QuantuMope/imc-der.

Funding Data

- National Science Foundation (Grant No. IIS-1925360).

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available at https://github.com/QuantuMope/imc-der.

References

[1] Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E., 2008, "Discrete Elastic Rods," ACM Trans. Graphics (TOG), **27**(3), p. 63.

[2] Bergou, M., Audoly, B., Vouga, E., Wardetzky, M., and Grinspun, E., 2010, "Discrete Viscous Threads," ACM Trans. Graphics (TOG), **29**(4), p. 116.

[3] Audoly, B., and Pomeau, Y., 2010, *Elasticity and Geometry: From Hair Curls to the Non-Linear Response of Shells*, Oxford University Press, Oxford, UK

[4] Jawed, M. K., Novelia, A., and O'Reilly, O. M., 2018, *A Primer on the Kinematics of Discrete Elastic Rods*, Springer, New York.

[5] Audoly, B., Clauvelin, N., Brun, P.-T., Bergou, M., Grinspun, E., and Wardetzky, M., 2013, "A Discrete Geometric Approach for Simulating the Dynamics of Thin Viscous Threads," J. Comput. Phys., **253**, pp. 18–49.

[6] Shen, Z., Huang, J., Chen, W., and Bao, H., 2015, "Geometrically Exact Simulation of Inextensible Ribbon," Computer Graphics Forum, **34**(7), pp. 145–154.

[7] Baraff, D., and Witkin, A., 1998, "Large Steps in Cloth Simulation," Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, Orlando, FL, July, ACM, pp. 43–54.

[8] Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P., 2003, "Discrete Shells," Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, CA, July, pp. 62–67.

[9] Batty, C., Uribe, A., Audoly, B., and Grinspun, E., 2012, "Discrete Viscous Sheets," ACM Trans. Graphics (TOG), **31**(4), p. 113.

[10] Jawed, M. K., Da, F., Joo, J., Grinspun, E., and Reis, P. M., 2014, "Coiling of Elastic Rods on Rigid Substrates," Proc. Natl. Acad. Sci. USA, **111**(41), pp. 14663–14668.

[11] Jawed, M. K., and Reis, P. M., 2014, "Pattern Morphology in the Elastic Sewing Machine," Extreme Mech. Lett., **1**, pp. 76–82.

[12] Jawed, M. K., Brun, P. -T., and Reis, P. M., 2015, "A Geometric Model for the Coiling of an Elastic Rod Deployed Onto a Moving Substrate," ASME J. Appl. Mech., **82**(12), p. 121007.

[13] Jawed, M. K., Khouri, N. K., Da, F., Grinspun, E., and Reis, P. M., 2015, "Propulsion and Instability of a Flexible Helical Rod Rotating in a Viscous Fluid," Phys. Rev. Lett., **115**(16), p. 168101.

[14] Jawed, M. K., and Reis, P. M., 2016, "Deformation of a Soft Helical Filament in an Axial Flow At Low Reynolds Number," Soft. Matter., **12**(6), pp. 1898–1905.

[15] Jawed, M., and Reis, P. M., 2017, "Dynamics of a Flexible Helical Filament Rotating in a Viscous Fluid Near a Rigid Boundary," Phys. Rev. Fluids, **2**(3), p. 034101.

[16] Panetta, J., Konaković-Luković, M., Isvoranu, F., Bouleau, E., and Pauly, M., 2019, "X-shells: A New Class of Deployable Beam Structures," ACM Trans. Graphics (TOG), **38**(4), p. 83.

[17] Baek, C., Sageman-Furnas, A. O., Jawed, M. K., and Reis, P. M., 2018, "Form Finding in Elastic Gridshells," Proc. Natl. Acad. Sci. USA, **115**(1), pp. 75–80.

[18] Baek, C., and Reis, P. M., 2019, "Rigidity of Hemispherical Elastic Gridshells Under Point Load Indentation," J. Mech. Phys. Solids., **124**, pp. 411–426.

[19] Jawed, M. K., Hadjiconstantinou, N., Parks, D., and Reis, P., 2018, "Patterns of Carbon Nanotubes by Flow-Directed Deposition on Substrates With Architectured Topographies," Nano Lett., **18**(3), pp. 1660–1667.

[20] Jawed, M., Dieleman, P., Audoly, B., and Reis, P. M., 2015, "Untangling the Mechanics and Topology in the Frictional Response of Long Overhand Elastic Knots," Phys. Rev. Lett., **115**(11), p. 118302.

[21] Moulton, D. E., Grandgeorge, P., and Neukirch, S., 2018, "Stable Elastic Knots With No Self-Contact," J. Mech. Phys. Solids., **116**, pp. 33–53.

[22] Patil, V. P., Sandt, J. D., Kolle, M., and Dunkel, J., 2020, "Topological Mechanics of Knots and Tangles," Science, **367**(6473), pp. 71–75.

[23] Audoly, B., Clauvelin, N., and Neukirch, S., 2007, "Elastic Knots," Phys. Rev. Lett., **99**(16), p. 164301.

[24] Clauvelin, N., Audoly, B., and Neukirch, S., 2009, "Matched Asymptotic Expansions for Twisted Elastic Knots: A Self-Contact Problem With Non-Trivial Contact Topology," J. Mech. Phys. Solids., **57**(9), pp. 1623–1656.

[25] Durville, D., 2012, "Contact-Friction Modeling Within Elastic Beam Assemblies: An Application to Knot Tightening," Computat. Mech., **49**(6), pp. 687–707.

[26] Przybyl, S., and Pieranski, P., 2009, "Tightening of the Elastic Overhand Knot," Phys. Rev. E, **79**(3), p. 031801.

[27] Spillmann, J., and Teschner, M., 2008, "An Adaptive Contact Model for the Robust Simulation of Knots," Computer Graphics Forum, **27**(2), pp. 497–506.

[28] Kaufman, D. M., Tamstorf, R., Smith, B., Aubry, J. -M., and Grinspun, E., 2014, "Adaptive Nonlinearity for Collisions in Complex Rod Assemblies," ACM Trans. Graphics (TOG), **33**(4), pp. 1–12.

[29] Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., and Kaufman, D. M., 2020, "Incremental Potential Contact: Intersection- and Inversion-free, Large-Deformation Dynamics," ACM Trans. Graphics (TOG), **39**(4), p. 3392425.

[30] Kirchhoff, G., 1859, "Uber Das Gleichgewicht Und Die Bewegung Eines Unendlich Dunnen Elastischen Stabes," J. Reine Angew. Math., **56**, pp. 285–313.

[31] Kikuchi, N., and Oden, J. T., 1988, *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM, University City, Philadelphia.

[32] Goyal, S., Ruina, A., and Papadopoulos, J., 1991, "Planar Sliding With Dry Friction Part 2. Dynamics of Motion," Wear, **143**(2), pp. 331–352.

[33] Lumelsky, V. J., 1985, "On Fast Computation of Distance Between Line Segments," Inf. Process. Lett., **21**(2), pp. 55–61.

[34] Choe, B., Choi, M. G., and Ko, H.-S., 2005, "Simulating Complex Hair With Robust Collision Handling," Proceedings of the 2005 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation, Los Angeles, CA, July, pp. 153–160.

[35] Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Fernando, I., Kulal, S., Cimrman, R., and Scopatz, A., 2017, "Sympy: Symbolic Computing in Python," Peer J. Computer Sci., **3**, pp. e103.

[36] Lam, S. K., Pitrou, A., and Seibert, S., 2015, "Numba: A llvm-Based Python jit Compiler," Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, Austin, TX, November, Association for Computing Machinery, pp. 1–6.