Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Efficient and Stable Simulation of Inextensible Cosserat Rods by a Compact Representation

Chongyao Zhao[1], Jinkeng Lin[1], Tianyu Wang[2], Hujun Bao[1], Jin Huang[1†]

[1]State Key Lab of CAD&CG, Zhejiang University
[2] School of Computer Science and Technology, Zhejiang Sci-Tech University

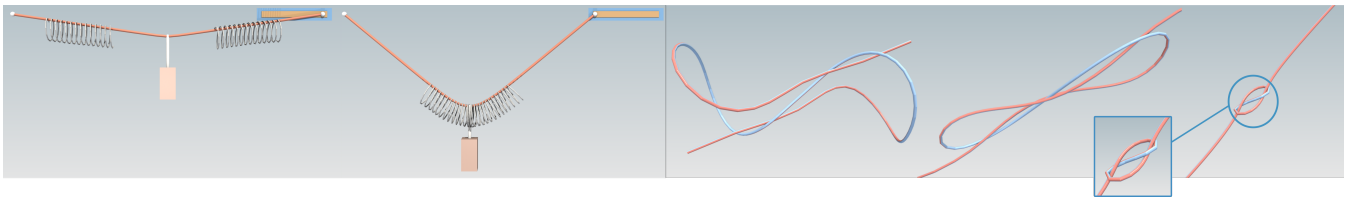


Figure 1: ***Strand through springs and knotting.*** *A strand through two springs and a knotted rod are simulated by our method. Using our compact representation, these complex behaviors can be efficiently and stably simulated with perfect inextensibility.*

Abstract

Piecewise linear inextensible Cosserat rods are usually represented by Cartesian coordinates of vertices and quaternions on the segments. Such representations use excessive degrees of freedom (DOFs), and need many additional constraints, which causes unnecessary numerical difficulties and computational burden for simulation. We propose a simple yet compact representation that exactly matches the intrinsic DOFs and naturally satisfies all such constraints. Specifically, viewing a rod as a chain of rigid segments, we encode its shape as the Cartesian coordinates of its root vertex, and use axis-angle representation for the material frame on each segment. Under our representation, the Hessian of the implicit time-stepping has special non-zero patterns. Exploiting such specialties, we can solve the associated linear equations in nearly linear complexity. Furthermore, we carefully designed a preconditioner, which is proved to be always symmetric positive-definite and accelerates the PCG solver in one or two orders of magnitude compared with the widely used block-diagonal one. Compared with other technical choices including Super-Helices, a specially designed compact representation for inextensible Cosserat rods, our method achieves better performance and stability, and can simulate an inextensible Cosserat rod with hundreds of vertices and tens of collisions in real time under relatively large time steps.

CCS Concepts

• ***Computing methodologies*** → *Physical simulation;* • ***Animation*** → *Animation with Constraints;*

1. Introduction

Compared with the mass-spring model [SLF08], the Cosserat rod model [Pai02, ST07, ST09] attaches material frames along the center line, which helps to reproduce some interesting special phenomena of a real rod, e.g., buckling caused by twisting. Therefore, it is widely used in the simulation of hairs [KTS*14], threads in yarn-level cloths [KJM08] etc., and most of them are inextensible or nearly inextensible. A discrete model of this type can be represented in a variety of ways, each of which leads to different numerical behaviors. This paper proposes a compact representation and uses it in a robust implicit integrator with a carefully designed preconditioner for efficiency.

1.1. Related work

There are many different ways to represent a Cosserat rod. A large number of methods [ST07, ST09, WCU*20] encode the rod shape by Cartesian coordinates of the vertices on the center line, then directly represent the material frame in the world coordinates on each segment between adjacent vertices by a unit quaternion. Though simple and intuitive, for inextensible rods, length constraints on

† Corresponding author: hj@cad.zju.edu.cn

each segment are inevitable. Besides, one needs to constrain a quaternion to be unit. Furthermore, because a thin rod is unshearable, one of the axes in the material frame should align with tangent of the center line, posing additional alignment constraints.

A lot of methods can be used to handle the aforementioned intrinsic (length, unit-quaternion, and axis-alignment) constraints. The most common ones are penalty-based methods, Lagrange-multipliers and their variants. Penalty-based methods are not accurate and may lead to an ill-conditioned problem. Lagrange-multipliers methods [GHF*07, SH10, DKWB18] are good in accuracy but solving KKT systems containing many constraints is costly. The step and projection (SAP) strategy can be used to enforce some decoupled constraints, e.g., explicitly normalizing every quaternion in each time step [ST07]. They generally introduce physical artifacts like excessive dispersion.

As a result, people propose more compact representations that intrinsically satisfy these constraints. As a typical method, [BWR*08, BAV*10] first compute a twisting-free frame (i.e., Bishop frame) by parallel transport along the center line, then encode the material frame as the difference (twisting angles) with respect to the Bishop frame. This representation eliminates the unit-quaternion and alignment constraints naturally. However, the stretch on segments is still left as a degree of freedom. So the fast projection method [GHF*07] is extended to enforce the inextensibility constraints for inextensible rods. However, the length constraint is nonlinear. One needs to iteratively solve KKT systems with the parameter-controlled termination condition. Early termination may lead to a large error while a large number of iterations hurt the performance significantly. The Super-Helices [BAC*06] is a representation exactly matching the intrinsic DOFs of an inextensible Cosserat rod, which discretizes a rod into N inextensible elements whose material curvatures and twist, i.e., derivative of the material frame, are constant in each element. To get a material frame in the world coordinates from such a representation, one needs to integrate the derivative from the beginning of the rod. This integration brings a dense mass matrix and results in an algorithm in the complexity of $\mathcal{O}(N^2)$. It also makes the algorithm complicated and even makes it difficult to use a highly stable implicit time stepping scheme. [Ber09] reduces the complexity to linear, but as shown in Fig. 6 of that paper, both [BAC*06, Ber09] have stability issues when using relatively large time steps. Therefore, many recent rod simulation algorithms [SMSH18, WCU*20] still follow the representation of [ST07] though it comes with a lot of constraints.

Another strategy is to represent piecewise linear inextensible rods as chains of articulated rigid bodies. [Had06] uses a reduced-coordinate formulation to simulate strands. While this formulation is capable of handling high bending and torsion stiffness, interaction requirements such as contact with the environment are difficult to model. For real-time applications, the performance of this method is not satisfactory. Recently, a very interesting articulated object representation, RedMax [WWB*19], along with an efficient numerical method is proposed. This method recursively represents the positions and orientations of each body with respect to its parent one, and the resulting generalized-coordinates representation favors an efficient solver through a block-diagonal preconditioner.

1.2. Contributions

Our method is inspired by RedMax [WWB*19], but does not represent the orientation on each segment recursively. To be more specific, we find that simply replacing the unit quaternion on each segment by the axis-angle representation and then using the chain representation, all the length, unit-quaternion, and alignment constraints in conventional representation [ST07] can be naturally eliminated. This representation is compact and can be easily integrated into a typical optimization-based implicit time integrator [MTGG11] for unconditional stability and a classic active set method [NW99] for collision handling. Besides, structure and nonzero patterns of the Hessian share many similarities to [WWB*19], which leads to nearly linear complexity in solving linear equations about it. However, using the common block-diagonal scheme to construct the preconditioner [TWL*18, WWB*19] cannot fully exploit the advantage of our representation. Indeed, a carefully designed hybrid one can significantly improve the convergence rate. This preconditioner leads to a smaller condition number and can be proved to be always symmetric positive-definite, which lays down a solid base for stability.

In summary, the contributions of this work include:

- a compact representation that eliminates all the intrinsic constraints for inextensible Cosserat rods and is friendly for implicit integrator with large time steps,
- an effective preconditioner of linear complexity for both construction and solving which significantly accelerates the convergence,
- and a proof that the preconditioner is symmetric positive-definite.

2. Representation of Inextensible Cosserat Rods

Before elaborating on our method, we first describe the conventional representation in [ST07] briefly.

In this representation, a non-looped inextensible Cosserat rod is discretized by n vertices on it, which split the rod into $n-1$ segments. The shape of the rod is represented by the Cartesian coordinates $\mathbf{p}_i$ on i-th vertex. On i-th segment $\mathbf{e}_i$ from $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$ with the rest length l_i, a quaternion $\mathbf{q}_i$ is used to indicate its material frame. Then, the variables to represent such a rod are $\mathbf{x} = [\mathbf{p}^\top, \mathbf{q}^\top]^\top$, where $\mathbf{p} \in \mathcal{R}^{3n}$ are all vertex positions in Cartesian coordinates, and $\mathbf{q} \in \mathcal{R}^{4(n-1)}$ is composed of quaternion-represented material frames on all segments.

These $7n-4$ variables are not independent and the simulation should carefully handle the following constraints among them:

- inextensible constraints:

$$\|\mathbf{p}_{i+1} - \mathbf{p}_i\| - l_i = 0, \tag{1}$$

- unit quaternion constraints:

$$\|\mathbf{q}_i\| - 1 = 0, \tag{2}$$

- unshearable constraints or alignment constraints:

$$d(\mathbf{q}_i) - \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|} = 0, \tag{3}$$

where d is a function extracting the segment-aligned axis (e.g., the third axis in [ST07]) from the material frame represented by $\mathbf{q}_i$.

2.1. The compact representation

The material frame is a rotation $R \in SO(3)$, and there exist many possible ways to represent it. The Lie algebra of it is a 3×3 skew-symmetric matrix, and can be compactly represented as a vector of length three, which is also named axis-angle representation. We choose to replace the quaternion $\mathbf{q}_i$ on the i-th segment by such a representation $\omega_i \in \mathcal{R}^3$. The corresponding unit quaternion can be easily computed from ω_i as:

$$\mathbf{q}_i = \mathcal{B}(\omega_i) = \left[\cos\frac{\|\omega_i\|}{2}, \frac{\omega_i}{\|\omega_i\|}\sin\frac{\|\omega_i\|}{2}\right]^\top. \tag{4}$$

When $\|\omega_i\|$ goes to zero, we apply L'Hospital's rule to avoid the division-by-zero problem:

$$\mathbf{q}_i = \lim_{\|\omega_i\|\to 0}\mathcal{B}(\omega_i) = \left[1, \frac{\omega_i}{2}\right]^\top. \tag{5}$$

This choice eliminates the unit quaternion constraints in Eq. 2.

The common 3×3 matrix representation can also be easily computed via exponential map according to Rodrigues' rotation formula:

$$\mathbf{m}_i = \exp(\omega_i). \tag{6}$$

Now taking a chain representation shown in Fig. 2, the non-root vertex positions on an inextensible rod can be recursively represented as below:

$$\mathbf{p}_{i+1} = \mathbf{p}_i + l_i d(\mathbf{m}_i) = \mathbf{p}_i + l_i d(\exp(\omega_i)), i = 1,2,\cdots,n-1. \tag{7}$$

This eliminates the alignment constraints in Eq. 3 and inextensible constraints in Eq. 1 simultaneously. One can also get the segment-aligned axis through quaternion by $d(\mathcal{B}(\omega_i))$, but the cost will be higher.

Finally, we use the Cartesian coordinates $\mathbf{p}_c = \mathbf{p}_0$ to represent the location of the root vertex.

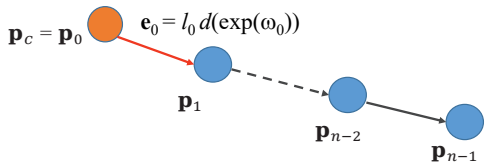


Figure 2: ***Chain structure.*** *We propose a compact chain structure without redundant degrees of freedom to represent an inextensible Cosserat rod, i.e., a Kirchhoff rod.*

Putting all of the above together, we use $\mathbf{p}_c \in \mathcal{R}^3$ at the root vertex and $\omega_i \in \mathcal{R}^3$ on i-th segment $\mathbf{e}_i$ to representation the rod. Look back to the conventional represent using $7n-4$ variables in $\mathbf{p}$ and $\mathbf{q}$, there are $4(n-1)$ linear independent constraints in Eqs. 1 to 3, so there are only $(7n-4)-4(n-1) = 3n$ DOFs indeed. Our representation $\mathbf{s} = [\mathbf{p}_c^\top, \omega^\top]^\top$ exactly has $3n$ DOFs, which perfectly matches the nature of a non-looped inextensible piecewise linear Cosserat rod. All the equality constraints in Eqs. 1 to 3 are unnecessary, which have been naturally satisfied when taking our representation. Actually, our method perfectly models a special kind of unshearable and inextensible Cosserat rod, i.e., the Kirchhoff rod [Dil92].

3. Simulation

Under the conventional representation, implicit Euler time-stepping leads to the following nonlinear constrained optimization problem [MTGG11]:

$$\mathbf{x}_{t+\Delta t} = \arg\min_{\mathbf{x}} E(\mathbf{x}) = V(\mathbf{q}) + T_p(\mathbf{p}) + T_q(\mathbf{q}) + G(\mathbf{p}), \tag{8a}$$

$$\text{s.t.} \begin{cases} \mathbf{C_E}(\mathbf{x}) = \mathbf{0}, \\ \mathbf{C_I}(\mathbf{x}) \geq \mathbf{0}. \end{cases} \tag{8b}$$

In the above equations, $V(\mathbf{q})$ is the discrete bending and twisting potential energy, $T_p(\mathbf{p})$ and $T_q(\mathbf{q})$ are translational and rotational kinetic energies respectively. $G(\mathbf{p})$ is the potential of external force fields (e.g., gravity). The specific equations for these terms are in App. A, and more details about them can be found in [ST07, WCU*20].

A large number of equality constraints $\mathbf{C_E}$ in Eq. 8b come from Eqs. 1 to 3. They have been eliminated by replacing the conventional representation by ours via the transformation $\mathcal{T}$ derived from Eqs. 4, 5 and 7 and $\mathbf{p}_c = \mathbf{p}_0$:

$$\mathbf{x} = \mathcal{T}(\mathbf{s}). \tag{9}$$

We handle the remaining equality constraints (e.g., the Dirichlet boundary condition constraints) and the inequality constraints $\mathbf{C_I}$ which usually come from collisions by a classical active set framework [NW99]. To be specific, in each iteration of the active set method, we solve such a problem:

$$\begin{aligned} \mathbf{s}_{t+\Delta t} &= \arg\min_{\mathbf{s}} E(\mathcal{T}(\mathbf{s})), \\ \text{s.t.} \quad \mathbf{C}_{\mathcal{A}}(\mathcal{T}(\mathbf{s})) &= \mathbf{0}. \end{aligned} \tag{10}$$

The constraints $\mathbf{C}_{\mathcal{A}}$ are updated according to the current collision situation and the remaining equality constraints. Roughly speaking, the remaining equality constraints are always in the set $\mathbf{C}_{\mathcal{A}}$ and the active inequality constraints (e.g., active collisions) in $\mathbf{C_I}$ will be inserted into the set $\mathbf{C}_{\mathcal{A}}$ and processed as equality constraints. At the same time, non-active collisions in $\mathbf{C_I}$ will be removed from each iteration. In each iteration, the status can be back-traced to a non-penetration status. A pseudo-code can be found in App. B.

3.1. SQP framework

In general, we follow the method in [PBH15] to solve the nonlinear optimization problem in Eq. 10 under the sequential quadratic programming (SQP) framework (Alg. 1). We use the line search method [NW99] to improve the convergence rate of SQP, and the merit function to evaluate the step length of line search is shown as below:

$$\phi(\mathbf{s},\mu) = E(\mathcal{T}(\mathbf{s})) + \frac{1}{\mu}\|\mathbf{C}_{\mathcal{A}}\|_1. \tag{11}$$

To prevent the step from violating constraints a lot, μ is updated as the reciprocal of the largest value of the Lagrange multipliers with

regard to the current active constraints [NW99]. Specifically, we construct a quadratic model of Eq. 10 at the current $\mathbf{s}^k$ in each step as below:

$$\begin{aligned} \Delta\mathbf{s}^k &= \arg\min_{\Delta\mathbf{s}} \frac{1}{2}\Delta\mathbf{s}^\top \mathbf{A}^k \Delta\mathbf{s} + {\mathbf{b}^k}^\top \Delta\mathbf{s}, \\ \text{s.t.} \quad & \nabla_\mathbf{s}\mathbf{C}_\mathcal{A}\Delta\mathbf{s} + \mathbf{C}_\mathcal{A}(\mathcal{T}(\mathbf{s}^k)) = \mathbf{0}, \end{aligned} \tag{12}$$

where, with $\mathbf{J}(\mathbf{s}) = D\mathcal{T} = \partial\mathbf{x}/\partial\mathbf{s}$,

$$\mathbf{b}^k = \mathbf{J}(\mathbf{s}^k)^\top \left.\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{T}(\mathbf{s}^k)},$$

$$\mathbf{A}^k = \mathbf{J}(\mathbf{s}^k)^\top \left.\frac{\partial^2 E(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}=\mathcal{T}(\mathbf{s}^k)} \mathbf{J}(\mathbf{s}^k) = \mathbf{J}^{k\top}\mathbf{H}^k\mathbf{J}^k.$$

Here, we drop $\left.\frac{\partial \mathbf{J}(\mathbf{s})}{\partial \mathbf{s}}\right|_{\mathbf{s}=\mathbf{s}^k}$: $\left.\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{T}(\mathbf{s}^k)}$ like [PBH15] did. Therefore, we actually use an inexact Newton's method to solve each quadratic sub-problem.

Algorithm 1 SQP framework

Require: $\mathbf{x}^k$
  $\mathbf{s}^k = \mathcal{T}^{-1}(\mathbf{x}^k)$
  while $\left\|\mathbf{b} + \nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^\top\lambda\right\|_2 \neq 0$ **do** (loop of Newton iteration)
    Evaluate $\mathbf{A}^k$, $\mathbf{b}^k$, $\nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^k$
    $\Delta\mathbf{s}$, λ = active set($\mathbf{A}^k$, $\mathbf{b}^k$, $\nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^k$)
    α = line search ($\mathbf{s}^k$, $\Delta\mathbf{s}$, λ)
    $\mathbf{s}^{k+1} = \mathbf{s}^k + \alpha\Delta\mathbf{s}$
  end while

The QP problem in Eq. 12 leads to a KKT system:

$$\begin{bmatrix} \mathbf{A} & \nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^\top \\ \nabla_\mathbf{s}\mathbf{C}_\mathcal{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{s} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{C}_\mathcal{A}(\mathcal{T}(\mathbf{s}_k)) \end{bmatrix}. \tag{13}$$

By *Schur Complement* method, λ is solved first and then $\Delta\mathbf{s}$:

$$\begin{cases} \left(\nabla_\mathbf{s}\mathbf{C}_\mathcal{A}\mathbf{A}^{-1}\nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^\top\right)\lambda = \mathbf{C}_\mathcal{A} - \nabla_\mathbf{s}\mathbf{C}_\mathcal{A}\mathbf{A}^{-1}\mathbf{b}, \\ \mathbf{A}\Delta\mathbf{s} = \mathbf{b} - \nabla_\mathbf{s}\mathbf{C}_\mathcal{A}^\top\lambda. \end{cases} \tag{14}$$

As RedMax [WWB*19], we use preconditioned conjugate gradient (PCG) method to solve the equations for $\mathbf{A}$. The number of those equations is the number of rows in $\nabla_s\mathbf{C}_\mathcal{A}$ plus 2 in total and brings huge computational cost. Notice that the cost of solving a linear system by PCG is strongly related to the matrix-vector multiplication and its preconditioner, we fully exploit the structure of $\mathbf{A}$ to make the complexity of both kinds of operations in $\mathcal{O}(n)$ time and accelerate the convergence.

3.2. Matrix-Vector multiplication in $\mathcal{O}(n)$ time

PCG involves frequent matrix-vector multiply operations for the matrix $\mathbf{A} = \mathbf{J}^\top\mathbf{H}\mathbf{J}$. We show that multiplying $\mathbf{H}, \mathbf{J}, \mathbf{J}^\top, \mathbf{A}$ with a vector can all be implemented in $\mathcal{O}(n)$ time under our compact representation.

For simplicity, we use such notations:

- $\begin{bmatrix}\mathbf{I} & & \\ & \ddots & \\ & & \end{bmatrix}$: a block lower triangular matrix whose block elements are $\mathbf{I}$,
- $\begin{bmatrix} & & \\ & \ddots & \\ & & \mathbf{I}\end{bmatrix}$: a block upper triangular matrix whose block elements are $\mathbf{I}$,
- $\begin{bmatrix}\diagdown & & \\ & \ddots & \\ & & \end{bmatrix}$: a block diagonal matrix ignoring its specific value,
- $\begin{bmatrix}\diagdown\diagdown & & \\ \diagdown\diagdown & \ddots & \\ & \ddots & \end{bmatrix}$: a block tridiagonal matrix ignoring its specific value.

The pattern of H: For rod simulation, the system matrix before applying the transformation $\mathcal{T}$ has a simple band structure with such $(\mathbf{p}, \mathbf{q})$ ordering:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H_p} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_q} \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t^2}\mathbf{M_p} & \mathbf{0} \\ \mathbf{0} & \frac{1}{\Delta t^2}\mathbf{M_q} + \mathbf{K_q} \end{bmatrix} = \left[\begin{array}{c|c} \diagdown & \mathbf{0} \\ \hline \mathbf{0} & \diagdown\diagdown \end{array}\right], \tag{15}$$

where $\mathbf{H_p} = \frac{\partial^2 E}{\partial \mathbf{p}^2} = \frac{\partial^2 T_p}{\partial \mathbf{p}^2} = \mathbf{M_p}/\Delta t^2$ and $\mathbf{H_q} = \frac{\partial^2 E}{\partial \mathbf{q}^2} = \frac{\partial^2 T_q(\mathbf{q})}{\partial \mathbf{q}^2} + \frac{\partial^2 V(\mathbf{q})}{\partial \mathbf{q}^2} = \mathbf{M}_q/\Delta t^2 + \mathbf{K}_q$. Under this simple band structure, the $\mathcal{O}(n)$ time complexity of the matrix-vector multiplication for $\mathbf{H}$ is naturally guaranteed.

The Pattern of J: We split the Jacobian matrix $\mathbf{J}$ into two blocks:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J_{p,s}} \\ \mathbf{J_{q,s}} \end{bmatrix}. \tag{16}$$

$\mathbf{J_{q,s}}$ has a pattern as below due to $\mathbf{s} = [\mathbf{p_c}^\top, \omega^\top]^\top$:

$$\mathbf{J_{q,s}} = \begin{bmatrix} \frac{\partial \mathbf{q}}{\partial \mathbf{p}_c} & \frac{\partial \mathbf{q}}{\partial \omega} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \diagdown \end{bmatrix}. \tag{17}$$

The pattern of $\mathbf{J_{p,s}}$ is relatively complicated as below:

$$\begin{aligned} \mathbf{J_{p,s}} &= \begin{bmatrix} \mathbf{I} & & & \mathbf{0} \\ \mathbf{I} & \mathbf{J}^1_{\mathbf{p},\mathbf{s}} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{I} & \mathbf{J}^1_{\mathbf{p},\mathbf{s}} & \cdots & \mathbf{J}^n_{\mathbf{p},\mathbf{s}} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & & & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I} & & & \\ & \mathbf{J}^1_{\mathbf{p},\mathbf{s}} & & \\ & & \ddots & \\ & & & \mathbf{J}^n_{\mathbf{p},\mathbf{s}} \end{bmatrix}}_{\text{Diag}(\{\mathbf{J}^i_{\mathbf{p},\mathbf{s}}\})} \\ &= \begin{bmatrix} \begin{bmatrix}\mathbf{I} & & \\ & \ddots & \\ & & \end{bmatrix} \cdot \begin{bmatrix}\diagdown & & \\ & \ddots & \\ & & \end{bmatrix} \end{bmatrix}. \end{aligned} \tag{18}$$

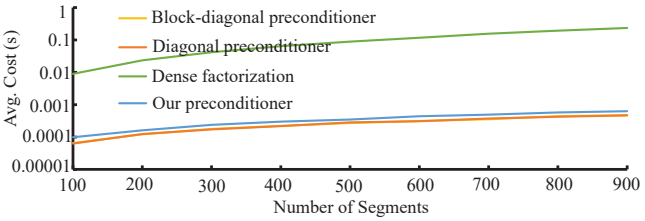
Obviously, we can multiply $\mathbf{J_{q,s}}$ with a vector in $\mathcal{O}(n)$ time. Furthermore, the time complexity of the matrix-vector multiplication for $\mathbf{J_{p,s}}$ is the same as $\begin{bmatrix}\mathbf{I} & & \\ & \ddots & \\ & & \end{bmatrix}$, which is identical to computing the prefix sum of the vector in $\mathcal{O}(n)$ time [HSO07]. Therefore, we can achieve the matrix-vector multiplication for $\mathbf{J}$ in $\mathcal{O}(n)$ time. $\mathbf{J}^\top$ shares the same property.

Finally, we can multiply $\mathbf{A}$ with a vector in $\mathcal{O}(n)$ time:

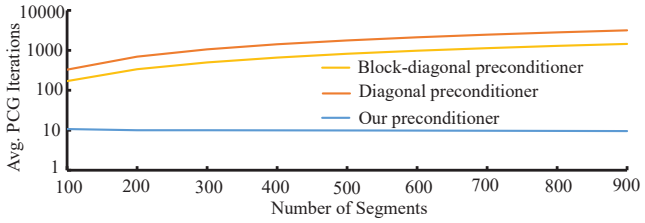
$$\mathbf{A} = \mathbf{J}^\top\mathbf{H}\mathbf{J} = \underbrace{\left[\begin{array}{c|c} \diagdown \cdot \begin{bmatrix}\mathbf{I} & & \\ & \ddots & \\ & & \end{bmatrix} & \begin{bmatrix}\mathbf{0} \\ \diagdown\end{bmatrix} \end{array}\right]}_{\mathbf{J}^\top} \underbrace{\left[\begin{array}{c|c} \diagdown & \mathbf{0} \\ \hline \mathbf{0} & \diagdown\diagdown \end{array}\right]}_{\mathbf{H}} \underbrace{\left[\begin{array}{c|c} \begin{bmatrix}\mathbf{I} & & \\ & \ddots & \\ & & \end{bmatrix} \cdot \diagdown & \\ \hline \begin{bmatrix}\mathbf{0} & & \end{bmatrix} & \diagdown\diagdown \end{array}\right]}_{\mathbf{J}}. \tag{19}$$

3.3. Hybrid preconditioner

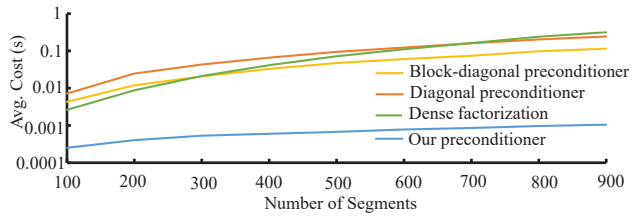
A suitable preconditioner is important for performance. Researchers usually use the diagonal, or block-diagonal part of $\mathbf{A}$ to construct the preconditioner considering its simplicity and low cost for construction. For instance, RedMax [WWB*19] adopts the block-diagonal strategy to construct their preconditioner. However,



(a) *Avg. precomputation cost w.r.t. different number of segments*



(b) *Avg. PCG iterations w.r.t. different number of segments*



(c) *Avg. computation cost w.r.t. different number of segments*

Figure 3: ***Comparison between our preconditioner and other preconditioners.*** *We simulate a helix under gravity with different preconditioners in 100 time steps and use 5 Newton iterations at most per time step. The construction cost of our preconditioner is slightly higher (a). However, solving the linear system using our preconditioner needs much less PCG iterations (b) and time (c) than using the diagonal or the block-diagonal preconditioners. As the rod is refined with more segments, the advantage of our method is more prominent.*

we notice that such a common way is not satisfactory, and the following preconditioner usually leads to much better convergence.

Specially, $\mathbf{A}$ in our chain structure can be expressed as below:

$$\begin{aligned}\mathbf{A} = \mathbf{J}^\top \mathbf{H} \mathbf{J} &= \begin{bmatrix}\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top & \mathbf{J}_{\mathbf{q},\mathbf{s}}^\top\end{bmatrix} \begin{bmatrix}\mathbf{H}_\mathbf{p} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_\mathbf{q}\end{bmatrix} \begin{bmatrix}\mathbf{J}_{\mathbf{p},\mathbf{s}} \\ \mathbf{J}_{\mathbf{q},\mathbf{s}}\end{bmatrix} \\ &= \mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}} + \mathbf{J}_{\mathbf{q},\mathbf{s}}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\mathbf{s}}.\end{aligned} \tag{20}$$

After further analysis, we note that $\mathbf{A}$ has a structure as below:

$$\begin{aligned}&\mathbf{J}_{\mathbf{q},\mathbf{s}}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\mathbf{s}} = \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \blacksquare\end{bmatrix}, \\ &\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}} = \mathrm{Diag}\{\mathbf{J}_{\mathbf{p},\mathbf{s}}^i\}^\top \cdot \underbrace{\blacksquare \cdot \mathbf{H}_\mathbf{p} \cdot \blacksquare}_{\tilde{\mathbf{H}}_\mathbf{p}} \cdot \mathrm{Diag}\{\mathbf{J}_{\mathbf{p},\mathbf{s}}^i\},\end{aligned} \tag{21}$$

where $\mathbf{J}_{\mathbf{q},\mathbf{s}} = \begin{bmatrix}\mathbf{0}, \mathbf{J}_{\mathbf{q},\omega}\end{bmatrix}$ and $\mathbf{J}_{\mathbf{p},\mathbf{s}} = \begin{bmatrix}\mathbf{I}, \mathbf{J}_{\mathbf{p},\omega}\end{bmatrix}$.

Clearly, $\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}$ in Eq. 21 has a block-tridiagonal structure. Inspired by this property, we select to extract the intact block-tridiagonal part of $\mathbf{J}_{\mathbf{q},\mathbf{s}}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\mathbf{s}}$ and the diagonal part of $\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}}$ to construct our hybrid preconditioner $\mathbf{P}$, for keeping the information after the transformation $\mathcal{T}$ as much as possible:

$$\mathbf{P} = \left(\text{blk-diag}\left(\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}}\right) + \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}\end{bmatrix}\right). \tag{22}$$

Such a hybrid preconditioner has some excellent properties: it is always a symmetric positive-definite matrix, constructing it and applying it are both in $\mathcal{O}(n)$ time.

The proof of positive definiteness: Without loss of generality, we suppose that the system matrix $\mathbf{H}$ is symmetric positive definitive. Otherwise, we can add $\alpha\mathbf{I}$ to ensure its positive definiteness. Obviously, its diagonal blocks $\mathbf{H}_\mathbf{p}$ and $\mathbf{H}_\mathbf{q}$ are also positive definite.

Our preconditioner has the following block strucutre based on Eq. 22:

$$\mathbf{P} = \begin{bmatrix}\sum_{i=0}^n \mathbf{h}_i & \mathbf{0} \\ \mathbf{0} & \overbrace{\text{blk-diag}(\mathbf{J}_{\mathbf{p},\omega}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\omega}) + \mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}}^{\mathbf{P}_{22}}\end{bmatrix}. \tag{23}$$

where $\mathbf{h}_i$ is a diagonal matrix whose diagonal elements are all greater than $\mathbf{0}$.

Obviously, the first diagonal block is positive definite, thus $\mathbf{P}$ is positive definite iff $\mathbf{P}_{22}$ is positive definite.

According to Eq. 20, we point out that $\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}}$ and $\mathbf{J}_{\mathbf{q},\mathbf{s}}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\mathbf{s}}$ are both at least positive semi-definite, since:

$$\forall \mathbf{v} \neq \mathbf{0}, \quad \mathbf{v}^\top \mathbf{J}_*^\top \mathbf{H}_* \underbrace{\mathbf{J}_* \mathbf{v}}_{\mathbf{y}} = \mathbf{y}^\top \mathbf{H}_* \mathbf{y} >= 0. \tag{24}$$

Therefore, blk-diag $\left(\mathbf{J}_{\mathbf{p},\omega}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\omega}\right)$ is still semi-definite. Next, we show that $\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}$ is positive definite.

The mapping $\mathcal{T}_{\mathbf{q},\omega}$ from ω to $\mathbf{q}$ is a locally injective mapping, so its Jacobian $\mathbf{J}_{\mathbf{q},\omega}$ has full column rank. Considering the positive definiteness of $\mathbf{H}_\mathbf{q}$, we can factorize it as $\mathbf{L}\mathbf{L}^\top$ and

$$\begin{aligned}\mathrm{rank}(\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}) &= \mathrm{rank}(\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{L}\mathbf{L}^\top \mathbf{J}_{\mathbf{q},\omega}) \\ &= \mathrm{rank}(\mathbf{L}^\top \mathbf{J}_{\mathbf{q},\omega}) \\ &= \mathrm{rank}(\mathbf{J}_{\mathbf{q},\omega}) \qquad = \dim(\omega).\end{aligned} \tag{25}$$

Thus, $\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}$ is still positive definite.

So far, our preconditioner is positive definite:

$$\begin{aligned}\mathbf{P} &= \left(\text{blk-diag}\left(\mathbf{J}_{\mathbf{p},\mathbf{s}}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\mathbf{s}}\right) + \begin{bmatrix}\mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}\end{bmatrix}\right) \\ &= \begin{bmatrix}\overbrace{\sum_{i=0}^n \mathbf{h}_i}^{\text{positive definite}} & \mathbf{0} \\ \mathbf{0} & \underbrace{\underbrace{\text{blk-diag}(\mathbf{J}_{\mathbf{p},\omega}^\top \mathbf{H}_\mathbf{p} \mathbf{J}_{\mathbf{p},\omega})}_{\text{positive semi-definite}} + \underbrace{\mathbf{J}_{\mathbf{q},\omega}^\top \mathbf{H}_\mathbf{q} \mathbf{J}_{\mathbf{q},\omega}}_{\text{positive definite}}}_{\text{positive definite}}\end{bmatrix}.\end{aligned} \tag{26}$$

Construction in $\mathcal{O}(n)$ time: Considering such a special [block diagonal matrix]-[block matrix]-[block diagonal matrix] multiplication:

$$\begin{bmatrix} \mathbf{A}_1 & & \\ & \mathbf{A}_2 & \\ & & \ddots \\ & & & \mathbf{A}_n \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1n} \\ \mathbf{B}_{21} & \mathbf{B}_{22} & \cdots & \mathbf{B}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \mathbf{B}_{n2} & \cdots & \mathbf{B}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{C}_1 & & \\ & \mathbf{C}_2 & \\ & & \ddots \\ & & & \mathbf{C}_n \end{bmatrix}. \tag{27}$$

The (i, j) block of the resulted matrix $\mathbf{ABC}$ is $\mathbf{A}_i\mathbf{B}_{ij}\mathbf{C}_j$. According to this, we can construct its block-tridiagonal or block-diagonal part in $\mathcal{O}(n)$ time clearly, assuming that three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are all prepared.

Back to our preconditioner, $\mathbf{J}_{\mathbf{q},\omega}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{J}_{\mathbf{q},\omega}$ naturally satisfies this structure, since $\mathbf{J}_{\mathbf{q},\omega}$ is a block diagonal matrix (Eq. 17). $\mathbf{J}_{\mathbf{p},\mathbf{s}}^{\top}\mathbf{H}_{\mathbf{p}}\mathbf{J}_{\mathbf{p},\mathbf{s}}$ is easily changed into such a structure in Eq. 27, using the associative law of multiplication shown in Eq. 21. Therefore, our concern converts to how we can access to $\tilde{\mathbf{H}}_{\mathbf{p}}$ in linear time. Fortunately, $\mathbf{H}_{\mathbf{p}}$ is a lumped block diagonal matrix:

$$\tilde{\mathbf{H}}_{\mathbf{p}} = \diagdown\!\!\!\diagdown \mathbf{H}_{\mathbf{p}} \diagdown\!\!\!\diagdown = \begin{bmatrix} \sum_{i=1}^{n}\mathbf{h}_i & \sum_{i=2}^{n}\mathbf{h}_i & \sum_{i=3}^{n}\mathbf{h}_i & \cdots & \mathbf{h}_n \\ \sum_{i=2}^{n}\mathbf{h}_i & \sum_{i=2}^{n}\mathbf{h}_i & \sum_{i=3}^{n}\mathbf{h}_i & \cdots & \mathbf{h}_n \\ \sum_{i=3}^{n}\mathbf{h}_i & \sum_{i=3}^{n}\mathbf{h}_i & \sum_{i=3}^{n}\mathbf{h}_i & \cdots & \mathbf{h}_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{h}_n & \mathbf{h}_n & \mathbf{h}_n & \cdots & \mathbf{h}_n \end{bmatrix}. \tag{28}$$

where $\mathbf{h}_i$ is the i-th diagonal block of $\mathbf{H}_{\mathbf{p}}$. Thus, $\tilde{\mathbf{H}}_{\mathbf{p}}$ has only n *distinct* non-zero blocks, which can be computed in $\mathcal{O}(n)$ time, just like the prefix sum. Conclusively, our preconditioner has only $\mathcal{O}(n)$ non-zero blocks, and we can construct it explicitly using this observation for each block.

Solving in $\mathcal{O}(n)$ time: Since the preconditioner has the standard block-tridiagonal structure, we use Thomas algorithm [Tho49] for solving in $\mathcal{O}(n)$ time.

Although the construction of our preconditioner could be relatively more complicated than the diagonal or block diagonal preconditioner, this specially tailored preconditioner improves the speed at least one order compared with the diagonal or block diagonal strategy as shown in Fig. 3. We also compare our PCG method with constructing $\mathbf{A}^{-1}$ by matrix factorization: the dense matrix structure causes more time consumption in both the matrix factorization (Fig. 3a) stage and the matrix-vector multiplication (Fig. 3c) stage.

Furthermore, our PCG solver has nearly linear complexity. Under the setting of Fig. 3, the average time of solving a linear system increases almost linearly as the number of segments increases (see Fig. 4). Therefore, our method is well scalable with respect to the number of segments. It should be noticed that the linear complexity does not apply to the whole time step because the collision constraints in Eq. 32 are nonlinear under our reduced representation and the number of Newton iterations to resolve the collisions may be different from frame to frame.

According to Eq. 20, $\mathbf{A}$ is the sum of two parts. As Young's modulus increases, $\mathbf{J}_{\mathbf{q},\mathbf{s}}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{J}_{\mathbf{q},\mathbf{s}}$ would be dominant increasingly. Therefore, in this situation, our preconditioner that uses the intact $\mathbf{J}_{\mathbf{q},\mathbf{s}}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{J}_{\mathbf{q},\mathbf{s}}$ could approximate $\mathbf{A}$ better, which leads to a better convergence rate as shown in Fig. 5a, while the simple block-diagonal preconditioner drops many large non-zero values in $\mathbf{J}_{\mathbf{q},\mathbf{s}}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{J}_{\mathbf{q},\mathbf{s}}$ and leads to a poor convergence rate as shown in Fig. 5b.

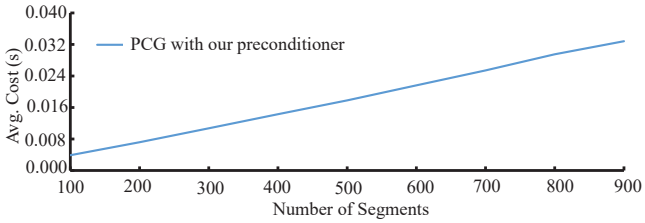
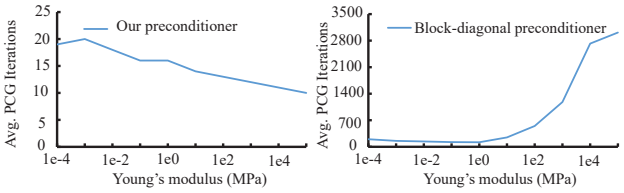


Figure 4: ***Average time cost of each time-step.*** *We simulate the helix under gravity in Fig. 6. Our PCG solves the linear systems using nearly fixed number of iterations. The time cost of each time-step is nearly linear with respect to the number of segments.*



(a) *Our preconditioner* **(b)** *Block-diagonal preconditioner*

Figure 5: ***PCG iterations w.r.t Young's modulus.*** *The superiority of our preconditioner is more obvious as Young's modulus increases.*

4. Comparisons and Results

We test the performance of our method in various experiments. All experiments in this section are performed on an AMD Core Ryzen 7-3800X 3.9GHz CPU, with 32GB of memory. In all our experiments, we terminate the Newton iteration (Alg. 1) if $\|\mathbf{b} + \nabla_{\mathbf{s}}\mathbf{C}_{\mathcal{A}}^{\top}\lambda\|_2 <$ 1e-5 or it reaches 5 iterations if there is no special declaration. In our experiments, it usually terminates at the former condition. For each linear system solving, we terminate the PCG iterations when the L_2 norm of the residual is smaller than 1e-7. Tab. 1 summarizes the statistics and performances of our experiments.

4.1. Comparisons

We compare our method with several existing methods, including the penalty method, the direct KKT method in Cartesian coordinates, an existing compact representation of the Cosserat rod [BAC*06, Ber09] and a potential compact representation based on RedMax [WWB*19]. To measure the violation of different constraints, we define $\frac{\sum_{i=0}^{n-2}\|\mathbf{p}_{i+1}-\mathbf{p}_i\|}{\sum_{i=1}^{n-1} l_i} - 1$ as the metric to measure the violation of inextensibility constraints. For measuring the violation of alignment constraints, we define $\sum_{i=0}^{n-2}\left\|\frac{\mathbf{p}_{i+1}-\mathbf{p}_i}{\|\mathbf{p}_{i+1}-\mathbf{p}_i\|} - d(\mathbf{q}_i)\right\|$ as the metric.

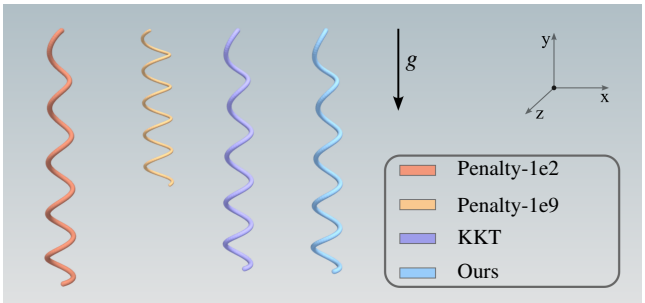
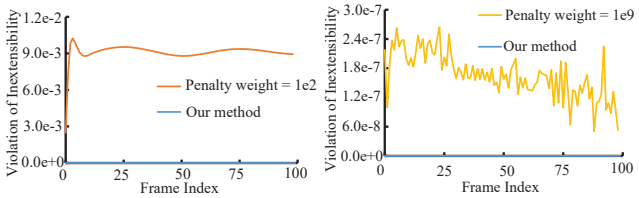


Figure 6: ***The helix under gravity.*** *We simulate the helix which is composed of 200 segments and use five Newton iterations at most per time step.*

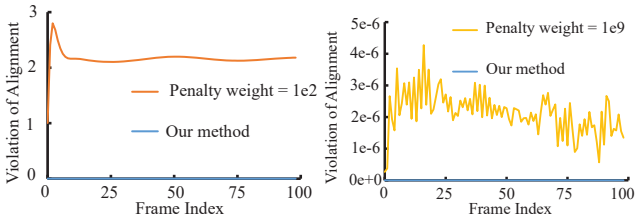
4.1.1. Comparison with the penalty method

For the experiments about the penalty method, we replace the energy term $E(\mathcal{T}(s))$ by $E(\mathbf{x})$ including the penalty terms about inextensibility and alignment constraints, and simulate the helix under gravity (Fig. 6). The weights of these two terms are the same. In each time step, this penalty method solves an unconstrained optimization problem about $E(\mathbf{x})$.

In the first experiment, both methods run 5 Newton iterations. As shown in Fig. 7a and Fig. 7b, it is hard for the penalty method to satisfy the inextensibility and alignment constraints in a few iterations. In other words, the accuracy of the penalty method could be poor. For instance, the inextensible rod could be seriously stretched if the weight is not large enough. However, as shown in next experiment, using large weight may bring artifacts in dynamics under a limited number of Newton iterations.



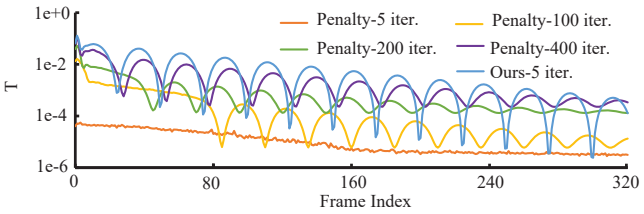
(a) *The violation of inextensibility constraints.*



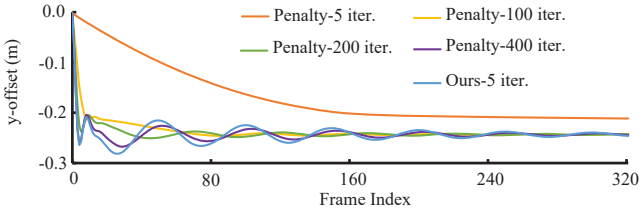
(b) *The violation of alignment constraints.*

Figure 7: ***Violation of constraints in the penalty method.*** *The penalty method can not exactly satisfy the inextensibility (a) and alignment constraints (b), and may bring large violations if the weight is not large enough. Note that the violations at frame* 0 *are all zero for all methods, and we plot these curves from frame* 1.

In the second experiment (Fig. 8), our method runs 5 Newton iterations while the penalty method with 1e9 as weight runs different Newton iterations. Here we calculate the kinetic energy by summing of $T_{trans} = \frac{1}{2}\sum_i m_i v_i^2$ and $T_{q,i}$ in Eq. 31. Fig. 8a shows that, as the number of Newton iterations of the penalty method increases, the period and phase of vibration get closer to our method. In Fig. 8b, the penalty method using weight 1e9 with 5 iterations is over-damped and does not vibrate at all. Indeed, as the weight increases, such artifacts will be more serious. Part of the reason is that, under larger weight, the optimization could be further away from converging in the limited Newton iterations.



(a) *The kinetic energy.*



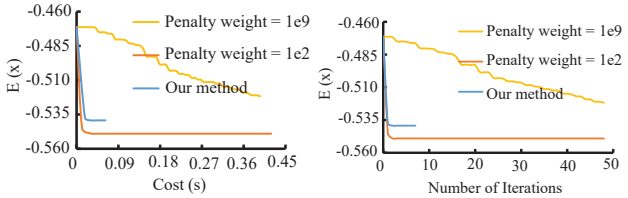
(b) *The y-offset of the free end node.*

Figure 8: ***Numerical damping of the penalty method (weight=1e9) using different number of iterations.*** *Larger weight introduces larger numerical damping and needs more Newton iterations to alleviate the artifacts. Note that kinetic energies at frame* 0 *are all zero, and we plot the kinetic energy from frame* 1 *in (a).*

In the third experiment, we compare the performance of our method with the penalty method on the first frame of the scenario in Fig. 6. To be fair, we run both methods to converge ($\|\mathbf{b}+\nabla_{\mathbf{s}}\mathbf{C}_{\mathcal{A}}^{\top}\boldsymbol{\lambda}\|_2 <$ 1e-5 for our method, and $\|\partial E(\mathbf{x})/\partial\mathbf{x}\|_2 <$ 1e-5 for the penalty method) without the restriction of iteration counts. As shown in Fig. 9, our method just needs a small number of iterations and converges quickly while the penalty method needs increasing time to converge as the penalty weight increases. Note that using a smaller penalty weight could reduce the time cost for convergence but introduce a larger violation of intrinsic constraints.

4.1.2. Comparison with direct KKT method

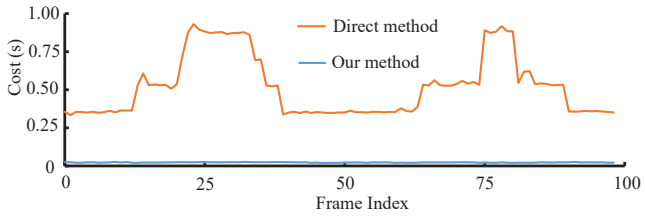
Directly solving the KKT system has no numerical damping artifacts but causes large computational cost, as shown in Fig. 6 and Fig. 10a. More importantly, solving the KKT system can not exactly satisfy the inextensibility and alignment constraints, especially when the Newton iterations are inadequate as shown in Fig. 10b and Fig. 10c. In contrast, our compact representation of Cosserat rod satisfies the constraints in Eqs. 1 to 3 naturally and we would not waste any computational cost to satisfy them. The



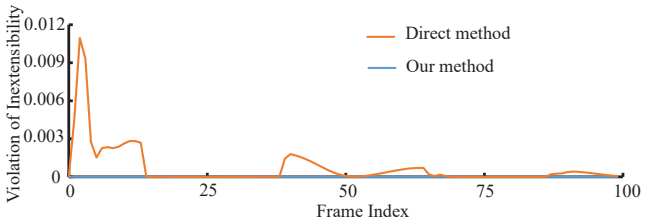
(a) *E in time.* **(b)** *E in number of iterations.*

Figure 9: ***Performance comparison between our method and the penalty method.*** *To converge, the penalty method with large weight needs more iterations and higher time cost than our method.*

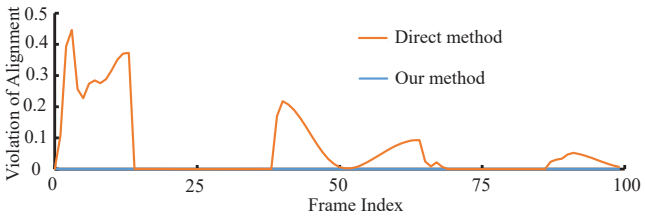
method in [DKWB18] also includes a KKT solver, but it alternates the KKT solver with the nonlinear Gauss-Seidel solver in each Newton step. The KKT solver is used to enforce the inextensibility constraints accompanied with bending and twisting forces, while the nonlinear Gauss-Seidel solver is used for loop-closing constraints and collision constraints. It also suffers from the problem of the violation issue when the Newton iterations are inadequate.



(a) *The comparison of the computational cost per time step.*



(b) *The comparison for the violation of inextensibility constraints.*



(c) *The comparison for the violation of alignment constraints.*

Figure 10: ***Comparison between our method and the direct KKT method in Cartesian coordinates.*** *We simulate helix under gravity as shown in Fig. 6. Our method not only spends less time than constructing KKT system and solving it in Cartesian coordinates (a) but also satisfies all the nonlinear intrinsic constraints (see (b) and (c)).*

4.1.3. Comparison with Super-Helices

The same as our chain representation, Super-Helices [BAC*06] can simulate rods exactly satisfying the constraints in Eqs. 1 to 3 by constructing a reduced-coordinate and [Ber09] improved its performance by introducing a linear-time solver. We note that different from our piecewise-linear-segment-based representation, Super-Helices uses a high-order-element-based representation to model the curved shape of a rod with fewer DOFs. This strategy is suitable when the simulated scene leads to smoothed deformation, but when complicated contact happens, the expressing ability of this curved element would decrease and more DOFs would be necessary to introduce.

Additionally, their method is not as unconditionally stable as our implicit time stepping method [Ber09], since the explicit computation of $\ddot{\kappa}_Q$ restricts that the time step has to be relatively small to suppress the stability issue with regard to the number of elements: a rod composed of one curved element can be simulated by $\Delta t = 35\,\text{ms}$ at most, and a rod composed of 40 curved elements can only be simulated at about $\Delta t = 4\,\text{ms}$. However, we simulate a similar scene on their benchmarks (see Fig. 11) and get stable simulation results using 200 segments and 250 ms as the time step because of our stable implicit time stepping.

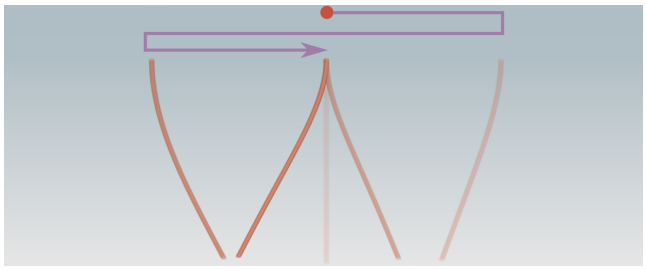


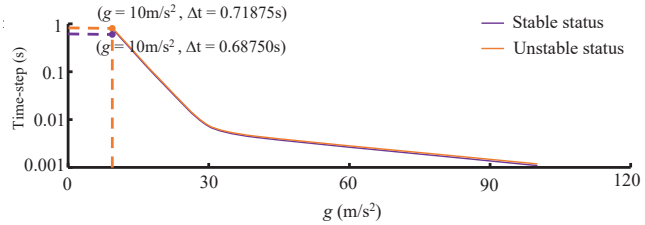
Figure 11: ***Dragging the root of a rod.*** *We simulate the rod using 200 segments, with* $1m$ *as reference length,* $1\,\text{kg}/\text{m}^3$ *as density,* $0.03\,\text{m}$ *as radius,* $1\,\text{MPa}$ *as Young's modulus,* $1\,\text{MPa}$ *as shear modulus, and* $\Delta t = 250\,\text{ms}$*. We move two vertices on the top of the rod in a sine wave and the displacement of the top two vertices can be analytically expressed as* $\phi \sin(2\pi f t)$*. We set* $f = 1$ *and* $\phi = 0.5\,\text{m}$*, which is the half of the reference length of the rod.*

To compare the performance with Super-Helices, we get similar results to them with better performance in a scene similar to Fig. 11 ($E = 0.2\,\text{MPa}, G = 0.1\,\text{MPa}, r = 0.03\,\text{m}, \rho = 1\,\text{kg}/\text{m}^3$, 20 segments): Super-Helices performed at 54 fps with about 8 ms as the time step in their paper, while our method performs at 344 fps with 30 ms as the time step. Our method gains 3.43 times acceleration, by setting the number of threads as one and only considering the clocks of the CPU (their CPU is 2.1 GHz while ours is 3.9 GHz), and ignoring the further benefit from our supported larger time step in this comparison.

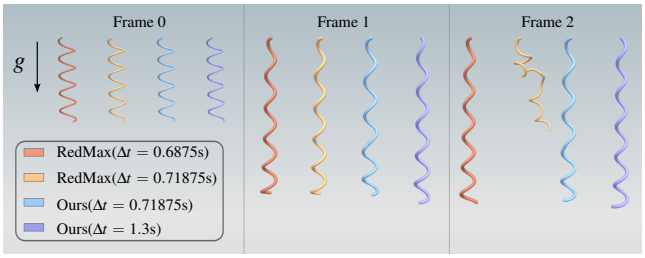
4.1.4. Comparison with RedMax

The recently developed RedMax [WWB*19] method can efficiently simulate articulated rigid bodies, which can also be extended to simulate an inextensible rod. To compare with this competitive idea, we extend RedMax to simulate the Kirchhoff rod in App. C. Roughly speaking, we introduce the spherical joints to

describe the change of the rotation field along the rod. Similar to Super-Helices, it makes the Jacobian and mass matrix more complicated. Besides, RedMax firstly applies coordinate transformation and then temporal discretization while our method is in the reversed order. Therefore, RedMax ignores more implicit forces than us, and thus brings the stability issue. To illustrate it, we simulate the same scenario as Fig. 6 by the extended RexMax and estimate its max time-step size without breaking the visual stability by the bisection method. As shown in Fig. 12, it becomes less stable as gravitational acceleration increases, and our method can simulate the rod stably under a larger time step.



(a) *Estimated max time-step size for the extended RedMax. The orange line and purple line are almost overlapped.*



(b) *The helix under gravity by our method and extended RedMax.*

Figure 12: ***The stability comparison with extended RedMax.*** *We use the bisection method to estimate the maximal time-step of the stable simulation using RedMax. For each g, the last trial of the bisection gives an interval of time-step size (a) whose begin (purple curve) and end (orange curve) associate with a stable and an unstable simulation respectively. RedMax becomes more and more prone to instability as gravitational acceleration increases. In (b), we exhibit the simulation results of the first sample points of the curves in (a). Our method can simulate under a much larger time-step.*

4.2. More results

In addition to the results shown in the comparisons, we further demonstrate the capacity of our method in some complex scenes.

Strand through springs. A rod ($E = 20\,\text{MPa}, G = 20\,\text{MPa}, r = 0.03\,\text{m}, \rho = 0.1\,\text{kg/m}^3$, 30 segments) with one end fixed and another end constrained in a sliding chute passes through two springs ($E = 2\,\text{MPa}, G = 2\,\text{MPa}, r = 0.03\,\text{m}, \rho = 0.1\,\text{kg/m}^3$, 150 segments for each spring) as shown in Fig. 14. The rod and the two springs are all simulated as inextensible Cosserat rods by our method with the time step 0.003 s. Hanging a heavy object in the middle, the two springs slide towards each other. Each frame involves about 16 collision contacts and is simulated at about 20 fps.

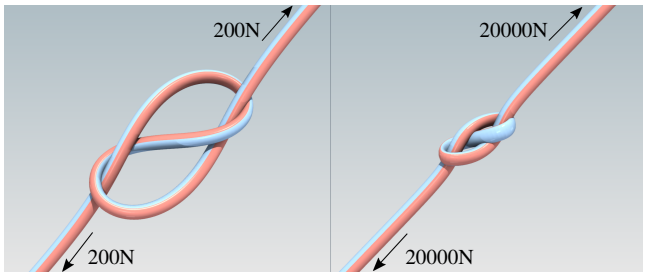


Figure 13: ***Knotting under forces with different magnitudes.*** *We can simulate tighter knot using larger stretching force.*

Knotting. As shown in Fig. 15, starting from a given knot structure with a stretching force (200 N), our method can simulate the knot to be tight as time goes on. When the knot is tight, it withstands large tension along the centerline. Due to our compact representation, the rod ($E = 20\,\text{MPa}, G = 20\,\text{MPa}, r = 0.03\,\text{m}, \rho = 0.1\,\text{kg/m}^3$, 100 segments) maintains the inextensibility perfectly. Our SQP framework with the active set method handles the self-collision well with 25 fps, when the time step is 0.003 s . After the minimal knot forms in Fig. 15, we continue to increase the stretching force to 20000 N, and the knot becomes much tighter as shown in Fig. 13.

Plectoneme formation. We can simulate the plectoneme formation of a Kirchhoff rod ($E = 1000\,\text{MPa}, G = 100000\,\text{MPa}, r = 0.003\,\text{m}, \rho = 200\,\text{kg/m}^3$, 50 segments), which is a typical phenomena where the twisting energy is transformed into the bending energy (Fig. 16). The two ends of the rod are stretched in the x-direction (1 N), and are only allowed to move freely in the x-direction. When twisting (10 rad/s) the two ends, the rod starts to writhe and form plectoneme.

Dropping a rod on three cylinders. As shown in Fig. 17, a rod ($E = 1\,\text{MPa}, G = 0.1\,\text{MPa}, r = 0.03\,\text{m}, \rho = 0.5\,\text{kg/m}^3$, 200 segments) is dropped on three cylinders. Because our method introduces less numerical damping, the rod bounces for a time before lying on the cylinders.

5. Conclusions, Limitations and Future Work

In this paper, we introduce a new representation of Cosserat rods which can perfectly satisfy the inextensibility, unit quaternions and alignment constraints at the same time. Based on this representation, we simulate the Cosserat rod under the SQP framework and make two further technical improvements for better performance: firstly we point out that the frequent matrix-vector multiplication can be implemented in $\mathcal{O}(n)$ time under our chain representation and then we construct a specific hybrid preconditioner in $\mathcal{O}(n)$ time. As a result, our method gains one or two orders of acceleration compared with a block-diagonal preconditioner and exactly satisfies the nonlinear constraints, under large time steps and in complex scenes.

One limitation of our method is that, as we stated in Fig. 5, the hybrid preconditioner works well when $\mathbf{J}_{\mathbf{q},\mathbf{s}}^{\top}\mathbf{H}_{\mathbf{q}}\mathbf{J}_{\mathbf{q},\mathbf{s}}$ dominates A and as its influence reduces, the superiority of our hybrid preconditioner decreases, where we need to increase PCG iterations to converge. Another limitation of our method is that we focus on the

Figure	#vert.	Avg. # of collision pairs	Avg. # of KKT systems	Avg. cost (ms) of grad. & Hes. eval.	Avg. cost (ms) of precomputation	Avg. cost (ms) of solving	Avg. total cost (ms)
Fig. 6	201	0	1	2.22	0.264	1.333	5.917
Fig. 14	333	16	7	4.421	0.947	40	52.69
Fig. 15	100	15	8	2.037	0.287	38	43
Fig. 16	51	8	7	2.392	0.375	13.089	20.160
Fig. 17	201	46	6	1.5613	1.56	12.172	36.888
Fig. 11	21	0	0	0.561	0.048	1.386	2.843

Table 1: ***Statistics and performances.*** *This table lists the average number of collision pairs, the average number of KKT systems to be solved and the average time cost spent in each time step on the experiments in Section 4.1.*

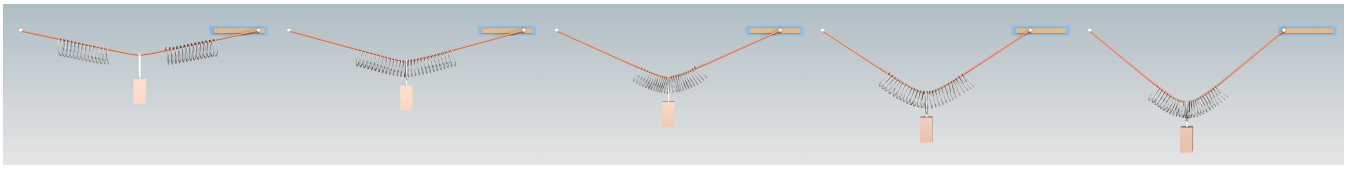


Figure 14: ***Strand through springs.*** *Two springs slide and collide with each other. Abundant collisions happen during this process and our method can handle it well.*

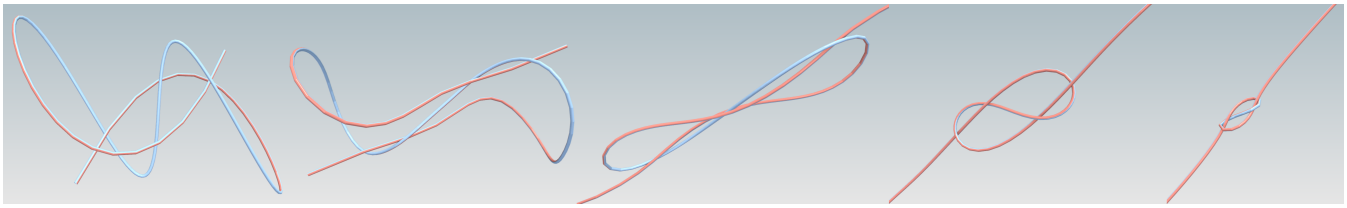


Figure 15: ***Knotting.*** *Our method reproduces the knotting process while preserving the inextensibility property no matter how much the force is. Here we use different colors to distinguish the different sides of the rod surface for better visualization.*

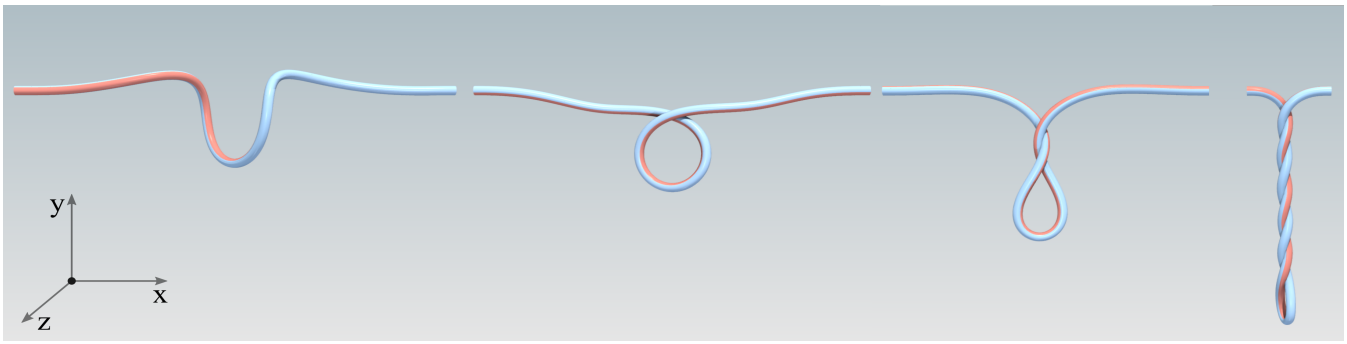


Figure 16: ***Plectoneme formation.*** *Plectoneme formation demonstrates the energy transformation from twisting energy to bending energy. During this process, two ends of the rod move only freely in the x-direction.*

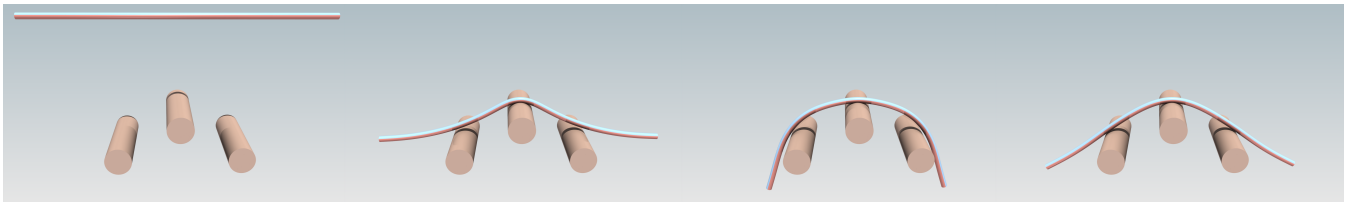


Figure 17: ***Dropping a rod on three cylinders.*** *This demo shows that our method has the ability to handle contacts with external obstacles.*

non-looped Cosserat rod structure and our method needs to be further extended into the Cosserat-net structure or multi-branch structure [ST09,DKWB18] and we leave it as future work to process this limitation in this paper. Looking into the future, we also would like to incorporate the adaptive discretization [WCU*20, SBRBO20] into our method for better capturing complex behaviors with sharp contacts. Besides, we use discrete collision detection only at the beginning of each SQP solving to collect proximity pairs. This strategy may miss collisions between time steps and collisions between internal Newton steps. This may bring passing-through errors or intersections. Now we use relatively small time steps to alleviate this issue. A better way could be incorporating the continuous strategy [OTSG09,SS15,LFS*20,WFS*21,YSC21]. Finally, we would like to investigate more efficient solvers or parallelize our method on GPU(s) for better performance.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and suggestions. Authors in Zhejiang University were partially supported by National Key R&D Program of China (No. 2020AAA0108901) and Zhejiang Provincial Science and Technology Program in China (No. 2021C01108). Authors in Zhejiang Sci-Tech University were partially supported by National Natural Science Foundation of China (No. 61702458, 61602416).

References

[BAC*06] BERTAILS F., AUDOLY B., CANI M.-P., QUERLEUX B., LEROY F., LÉVÊQUE J.-L.: Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3 (jul 2006), 1180–1187. doi:10.1145/1141911.1142012. 2, 6, 8

[BAV*10] BERGOU M., AUDOLY B., VOUGA E., WARDETZKY M., GRINSPUN E.: Discrete viscous threads. *ACM Trans. Graph. 29*, 4 (jul 2010), 116:1–116:10. doi:10.1145/1778765.1778853. 2

[Ber09] BERTAILS F.: Linear time super-helices. *Computer Graphics Forum 28*, 2 (2009), 417–426. doi:https://doi.org/10.1111/j.1467-8659.2009.01381.x. 2, 6, 8

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (jul 2002), 594–603. doi:10.1145/566654.566623. 12

[BWR*08] BERGOU M., WARDETZKY M., ROBINSON S., AUDOLY B., GRINSPUN E.: Discrete elastic rods. *ACM Trans. Graph. 27*, 3 (aug 2008), 1–12. doi:10.1145/1360612.1360662. 2

[Dil92] DILL E. H.: Kirchhoff's theory of rods. *Archive for History of Exact Sciences 44*, 1 (Mar 1992), 1–23. doi:10.1007/BF00379680. 3

[DKWB18] DEUL C., KUGELSTADT T., WEILER M., BENDER J.: Direct position-based solver for stiff rods. *Computer Graphics Forum 37*, 6 (2018), 313–324. doi:https://doi.org/10.1111/cgf.13326. 2, 8, 11

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3 (jul 2007), 49–es. doi:10.1145/1276377.1276438. 2

[Had06] HADAP S.: Oriented strands: Dynamics of stiff multi-body system. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2006), SCA '06, Eurographics Association, p. 91–100. 2

[HSO07] HARRIS M., SENGUPTA S., OWENS J. D.: Parallel prefix sum (scan) with cuda. *GPU gems 3*, 39 (2007), 851–876. 4

[KJM08] KALDOR J. M., JAMES D. L., MARSCHNER S.: Simulating knitted cloth at the yarn level. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, Association for Computing Machinery, pp. 65:1–65:9. doi:10.1145/1399504.1360664. 1

[KTS*14] KAUFMAN D. M., TAMSTORF R., SMITH B., AUBRY J.-M., GRINSPUN E.: Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Trans. Graph. 33*, 4 (jul 2014), 123:1–123:12. doi:10.1145/2601097.2601100. 1

[LFS*20] LI M., FERGUSON Z., SCHNEIDER T., LANGLOIS T., ZORIN D., PANOZZO D., JIANG C., KAUFMAN D. M.: Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph. 39*, 4 (jul 2020), 49:1–49:20. doi:10.1145/3386569.3392425. 11

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. *ACM Trans. Graph. 30*, 4 (jul 2011), 72:1–72:8. doi:10.1145/2010324.1964967. 2, 3

[NW99] NOCEDAL J., WRIGHT S. J.: *Numerical optimization*. Springer, 1999. 2, 3, 4

[OTSG09] OTADUY M. A., TAMSTORF R., STEINEMANN D., GROSS M.: Implicit contact handling for deformable objects. *Computer Graphics Forum 28*, 2 (2009), 559–568. doi:https://doi.org/10.1111/j.1467-8659.2009.01396.x. 11

[Pai02] PAI D. K.: Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum 21*, 3 (2002), 347–352. doi:https://doi.org/10.1111/1467-8659.00594. 1

[PBH15] PAN Z., BAO H., HUANG J.: Subspace dynamic simulation using rotation-strain coordinates. *ACM Trans. Graph. 34*, 6 (oct 2015), 242:1–242:12. doi:10.1145/2816795.2818090. 3, 4

[SBRBO20] SÁNCHEZ-BANDERAS R. M., RODRÍGUEZ A., BARREIRO H., OTADUY M. A.: Robust eulerian-on-lagrangian rods. *ACM Trans. Graph. 39*, 4 (jul 2020), 59:1–59:10. doi:10.1145/3386569.3392489. 11

[SH10] SPILLMANN J., HARDERS M.: Inextensible elastic rods with torsional friction based on lagrange multipliers. *Computer Animation and Virtual Worlds 21*, 6 (2010), 561–572. doi:https://doi.org/10.1002/cav.362. 2

[SLF08] SELLE A., LENTINE M., FEDKIW R.: A mass spring model for hair simulation. *ACM Trans. Graph. 27*, 3 (aug 2008), 1–11. doi:10.1145/1360612.1360663. 1

[SMSH18] SOLER C., MARTIN T., SORKINE-HORNUNG O.: Cosserat rods with projective dynamics. *Computer Graphics Forum 37*, 8 (2018), 137–147. doi:https://doi.org/10.1111/cgf.13519. 2

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (jul 2015), 70:1–70:9. doi:10.1145/2766947. 11

[ST07] SPILLMANN J., TESCHNER M.: CoRdE: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2007), SCA '07, Eurographics Association, p. 63–72. 1, 2, 3, 12

[ST09] SPILLMANN J., TESCHNER M.: Cosserat nets. *IEEE Transactions on Visualization and Computer Graphics 15*, 2 (2009), 325–338. doi:10.1109/TVCG.2008.102. 1, 11

[Tho49] THOMAS L.: Elliptic problems in linear differential equations over a network: Watson scientific computing laboratory. *Columbia Univ., NY* (1949). 6

[TWL*18] TANG M., WANG T., LIU Z., TONG R., MANOCHA D.: I-cloth: Incremental collision handling for GPU-Based interactive cloth simulation. *ACM Trans. Graph. 37*, 6 (dec 2018), 204:1–204:10. doi:10.1145/3272127.3275005. 2

[WCU*20] WEN J., CHEN J., UMETANI N., BAO H., HUANG J.: Cosserat rod with rh-adaptive discretization. *Computer Graphics Forum 39*, 7 (2020), 143–154. doi:https://doi.org/10.1111/cgf.14133. 1, 2, 3, 11

[WFS*21] WANG B., FERGUSON Z., SCHNEIDER T., JIANG X., ATTENE M., PANOZZO D.: A large-scale benchmark and an inclusion-based algorithm for continuous collision detection. *ACM Trans. Graph. 40*, 5 (sep 2021), 188:1–188:16. doi:10.1145/3460775. 11

[WWB*19] WANG Y., WEIDNER N. J., BAXTER M. A., HWANG Y., KAUFMAN D. M., SUEDA S.: Redmax: Efficient & flexible approach for articulated dynamics. *ACM Trans. Graph. 38*, 4 (jul 2019), 104:1–104:10. doi:10.1145/3306346.3322952. 2, 4, 6, 8

[YSC21] YU C., SCHUMACHER H., CRANE K.: Repulsive curves. *ACM Trans. Graph. 40*, 2 (may 2021), 10:1–10:21. doi:10.1145/3439429. 11

Appendix A: Energy Terms

These terms are derived in [ST07]. We briefly introduce them here.

The discrete bending and twisting potential energy is computed as

$$V_i(\mathbf{q}) = \frac{1}{2} l_i \sum_{k=1}^{3} \mathbf{K}_{kk} \left(\frac{2}{||\mathbf{q}_i||^2} (\mathbf{B}_k(\mathbf{q}_i + \mathbf{q}_{i+1}))^\top \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{l_i + l_{i+1}} - \bar{\varepsilon}_k \right)^2, \tag{29}$$

where the diagonal stiffness $\mathbf{K}$ has entries of $\mathbf{K}_{11} = \mathbf{K}_{22} = E\pi r^2/4$ and $\mathbf{K}_{33} = G\pi r^2/2$, where E and G are stiffness modulus which encodes the bending and torsional resistance respectively. $\bar{\varepsilon}_k$ is the natural bending and torsion of the rod based on its geometry at the rest shape. $\mathbf{B}_k \in \mathcal{R}^{4\times4}$ is the constant skew-symmetric matrix [ST07].

The kinetic energy is composed of translational one $T_p(\mathbf{p})$ and rotational one $T_q(\mathbf{q})$. $T_p(\mathbf{p})$ represents the incremental translational kinetic energy under the implicit Euler time discretization:

$$T_p(\mathbf{p}) = \frac{1}{2} \|\mathbf{p} - \mathbf{p}_t - \dot{\mathbf{p}}_t \Delta t\|^2_{\mathbf{M}/\Delta t^2}, \tag{30}$$

where $\mathbf{M}$ is the lumped mass matrix, Δt is the time step, $\mathbf{p}_t$ and $\dot{\mathbf{p}}_t$ are the position and velocity of vertices at the last time step respectively. For a rod with cross-section radius r and material density ρ, the discrete rotational kinetic energy per segment $\mathbf{e}_i$ is calculated as below:

$$T_{q,i}(\mathbf{q}) = \frac{1}{8} l_i \sum_{k=1}^{3} \mathbf{J}_{kk} \left((\mathbf{B}_k(\mathbf{q}_i + \mathbf{q}_{i+1}))^\top (\dot{\mathbf{q}}_i + \dot{\mathbf{q}}_{i+1}) \right)^2, \tag{31}$$

where the three entries of diagonal inertia tensor are $\mathbf{J}_{11} = \mathbf{J}_{22} = \rho\pi r^2/4$ and $\mathbf{J}_{33} = \rho\pi r^2/2$ respectively.

Appendix B: Active Set Method for Collisions

The inequality constraints $\mathbf{C_I}$ mainly stand for collision constraints to avoid intersections. Following [BFA02], the vertex-triangle constraints and the edge-edge constraints can be written as below:

$$\begin{cases} (\mathbf{p}_a - w_i\mathbf{p}_i - w_j\mathbf{p}_j - w_k\mathbf{p}_k) \cdot \mathbf{n} - \delta \geq 0, \\ \big((w_i\mathbf{p}_i + (1-w_i)\mathbf{p}_{i+1}) - (w_j\mathbf{p}_j + (1-w_j)\mathbf{p}_{j+1})\big) \cdot \mathbf{n} - \delta \geq 0. \end{cases} \tag{32}$$

Using a classical active set method (Alg. 2), the active constraints will be inserted into $\mathbf{C}_\mathcal{A}$ and processed as equality constraints. The non-active ones will be simply deleted.

Appendix C: Extended RedMax for Kirchhoff Rod Simulation

Kirchhoff rod describes a rod with no stretch and shear deformation, so the methodology for articulated rigid body simulation is potentially suitable to approximate it. To achieve this, we only need to use the spherical joints and use its DOFs to derive the potential energy for Kirchhoff model. We denote the DOFs of spherical joint by $\mathcal{Q}$ and express the change of material frame by $\exp(\mathcal{Q})$.

Algorithm 2 active set

Require: $\mathbf{s}^k$
  get the current active set $\mathcal{A}$ at $\mathbf{x}^k = \mathcal{T}\left(\mathbf{s}^k\right)$
  while true **do**
    Solve Eq (13) and get $\Delta\mathbf{s}$, λ
    if $\nabla_\mathbf{s}\mathbf{C_I}(\mathbf{s}+\Delta\mathbf{s}) - \mathbf{d}_I < 0$ **then** (In feasible Domain)
      if $\|\lambda\|_{-\infty} > 0$ **then** (all active constraints are tight (LCP))
        $\mathbf{s}^{k+1} = \mathbf{s}^k + \Delta\mathbf{s}$
        break
      else(some constraints are loose)
        delete loose constriant ($\lambda_i < 0$) from active set $\mathcal{A}$
      end if
    else(Not in feasible Domain, backtrack)
      $\beta = \min\{\frac{d_{\mathcal{A},i} - \nabla_\mathbf{s}\mathbf{C}^\top_{\mathcal{A},i}\Delta\mathbf{x}^k}{\nabla_\mathbf{s}\mathbf{C}^\top_{\mathcal{A},i}\Delta\mathbf{s}}\}$
      $\mathbf{s}^k = \mathbf{s}^k + \beta\Delta\mathbf{s}$
      add the tightest constraints to $\mathcal{A}$
    end if
  end while
  $\Delta\mathbf{s} = \mathbf{s}^{k+1} - \mathbf{s}^k$

Similar to Super-Helices, we discretize κ by n piecewise-constant basis functions. We set the material coordinates of joints (points) as $u_i \in [u_0, u_{n-1}]$ where $u_0 = 0$ and $u_{n-1} = L$, then

$$\kappa(u) = \kappa_j, \quad \frac{u_{j-1} + u_j}{2} \leq u \leq \frac{u_j + u_{j+1}}{2}, \tag{33}$$

where $j = 1, 2, \cdots, n-2$. The span of the basis is the Voronoi space of each joint.

Now we need to calculate κ using $\mathcal{Q}$. By the definition of the curvature, given the change of rotation, $\exp(\mathcal{Q})$, the curvature $\kappa = \frac{\log(\exp(\mathcal{Q}))}{l} = \frac{\mathcal{Q}}{l}$. Since this change only happens at the joint, $l = 0$ makes κ infinite. We average this change of rotation to the Voronoi space of the joint so that we can get the curvature of the joint κ_j by $\kappa_j = \frac{\mathcal{Q}}{l_j}$ where $l_j = \frac{u_{j+1} - u_{j-1}}{2}$. Now we can model the potential energy by

$$U = \frac{1}{2} \int_0^L \|\kappa(u) - \bar{\varepsilon}\|^2_\mathbf{K} \, du, \tag{34}$$

where $\bar{\varepsilon}$ and $\mathbf{K}$ are the same as Eq. 29. So the discrete energy is

$$U = \frac{1}{2} \sum_j l_j \left\| \frac{\mathcal{Q}_j}{l_j} - \bar{\varepsilon} \right\|^2_{(\mathbf{K})}, \tag{35}$$

and the derivatives of it are

$$\frac{\partial U}{\partial \mathcal{Q}_j} = (\mathbf{K}) \left(\frac{\mathcal{Q}_j}{l_j} - \bar{\varepsilon} \right), \quad \frac{\partial^2 U}{\partial \mathcal{Q}_j^2} = \frac{(\mathbf{K})}{l_j}. \tag{36}$$