Revisiting Discrete Elastic Rods

HANGQI, C., BHATIA, S., and ABDELWAHAB, N., Boston University, USA

A re-implementation of Discrete Elastic Rod from Bourgex, 2008. Starting from building the twist-free bishop frame, We analyzed the dynamics within a thin ideal rod and perform simulation to capture its behaviour when apply twist force to it.

Additional Key Words and Phrases: Discrete Elastic Rod, Computer Graphics, Simulation

ACM Reference Format:
Hangqi, C., Bhatia, S., and Abdelwahab, N.. 2024. Revisiting Discrete Elastic Rods. 1, 1 (May 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROBLEM STATEMENT- MICHELLS INSTABILITY HERE?

We aim to implement the Discrete Elastic Rods (DER) model from the 2008 SIGGRAPH paper of the same name. This model has revolutionized the film industry by enabling realistic, large-scale hair simulations. DER provides a framework for simulating complex, deformable rods, making it a fundamental tool in the computer graphics field. Its ability to accurately model the dynamics of inextensible rods under various physical forces makes DER an essential resource for researchers and practitioners looking to simulate and study the behavior of flexible materials in a realistic manner. The primary objectives of this project are to: (1) develop an interactive simulation interface that allows users to specify and manipulate cubic spline curves, observing the resulting torsion and curvature; and (2) implement advanced computational techniques such as the quasistatic optimization and symplectic integration to explore the dynamic and static behaviors of elastic rods under various conditions.

2 BACKGROUND/CONCEPTS

We define a Discrete Elastic Rod as a discretization of the Kirchhoff rod, which is a model of a long, thin cylinder following a space curve in $\mathbb{R}^3$ as its centerline (Fig. 1). The rod maintains a constant radius and exists in a plane orthogonal to the tangent of the curve at any given point.

To discretize the centerline of the Kirchhoff rod, we represent the curve as a polyline composed of vertices, which we refer to as primals, regularly sampled along the curve. These vertices are connected by edges, known as duals, that link each pair of adjacent vertices

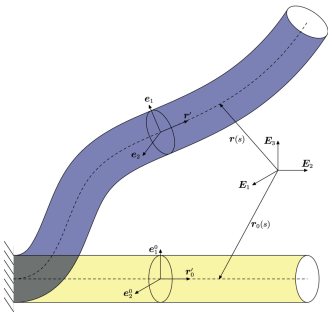

Fig. 1. Continuous Kirchhoff Rod. The cylinder "follows" the centerline, with a constant, small radius.

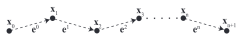

Fig. 2. Discrete Polyline. x_i are the primals, and e_j is the edge between x_j, x_{j+1}

Discrete elastic rods are distinguished by their inextensibility, meaning they do not stretch, and their resistance to bending and twisting. This is modeled by discretizing the rod's centerline into a series of vertices, referred to as primals, connected by edges known as duals. Each edge

The tangent of the curve at a vertex i is simply the vector defining the edge e_i, connecting vertices $i, i+1$. This sets up the foundation to construct the twist-free **Bishop frame**.The Bishop frame provides an orthonormal basis along the rod, crucial for visualizing how the rod deforms and responds when a twisting force is applied. This frame comprises a tangent (the edge itself), a normal, and a binormal (which is the cross product of the tangent and normal). Initiated at vertex 0 with a normal that is orthogonal to the tangent, the Bishop frames for subsequent vertices are calculated using Parallel Transport.

This is a method which uses a rotation matrix encoding the rotation from e_i to e_{i+1}, and applies the rotation to the previous Bishop frame to get the next one. We call the Bishop frame "twist-free" because the normal and binormal only rotate as much as the subsequent edge does. No additional, user-imposed twist is applied to the frame.

3 ROD ANALYSIS

Consider a long thin rod, whose length exceeds it's radius. Starting from the Kirchoff Rod model, there have been various models demonstrate such phenomenon. One common intuition is to conceptualize the rod as two distinct parts, separating the center line from the rod.

$$\text{rod} = \text{poly-center line} + \text{cross section}$$

Authors' address: Hangqi, C.; Bhatia, S.; Abdelwahab, N., hqcui19@bu.edu, Boston University, P.O. Box 1212, Boston, MA, USA, 02134.

ACM XXXX-XXXX/2024/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

This separation allow us to simplify the modeling process by treating the rod's geometry as basic polyline segments for the centerline. For now we simply set the rod to be isotropic, which means the cross section of the rod is a unit circle. We first perform the analysis to the continuous rod then manage to discretize the rod to be correctly presented by program.

3.1 Frenet Frame

As previously noted, our analysis focuses on the centerline of the rod, leading us to explore how to describe a curve in 3D space effectively. A straightforward method is using a continuous function, $f(t)$ or $f(x, y)$.Through arc-length parametrization, this function extends into a continuous line, $\Gamma(t)$. The **Frenet Frame**is a system of three orthogonal unit vectors that define the geometric properties of a curve in 3D space: the tangent vector (T) indicates the direction of the curve, the normal vector (N) points towards the curve's curvature, and the binormal vector (B) is perpendicular to both. These are all expressed through the Frenet-Serret formulas

$$\begin{aligned}\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s} &= \kappa\mathbf{N}\\ \frac{\mathrm{d}\mathbf{N}}{\mathrm{d}s} &= -\kappa\mathbf{T} + \tau\mathbf{B}\\ \frac{\mathrm{d}\mathbf{B}}{\mathrm{d}s} &= -\tau\mathbf{N}\end{aligned}$$

Where κ is the curvature of the point, τ is the torsion of the curve. For the Frenet Frame to be applicable, the curve must be non-degenerate (curvature $\kappa \neq 0$)

3.2 Bishop Frame

The Bishop Frame, therefore, modifies the traditional Frenet Frame to eliminate the torsion component, essentially allowing for the construction of a twist-free frame.

$$\begin{aligned}T(t) &= \frac{d\Gamma(t)}{dt}\\ N(t) &= \frac{dT(t)}{dt}\\ B(t) &= T(t) \times B(t)\end{aligned}$$

These vectors define an orthonormal basis where $T(t)$ is the tangent vector, $N(t)$ the normal vector, and $B(t)$ the binormal vector at each point on the curve. For each point on the curve, we also define a twist angle θ, enhancing the description of the curve's orientation in space. We can always start from a twist free line, the combine it with a cross section pattern and later applied twist deformation onto it.

3.3 Discrete Elastic Rod

The Discrete Elastic Rod (DER) model proposed by the paper of same title, Bregoue, 2008 improves the Kirchoff Rod to fit in the framework of Bishop Frame. The Rod is in-extensible, continous, Quasistatic updated described by discrete Bishop Frame. We now consider how to discretize our Bishop frame on the rod. A natural way to do so is resample the curve with arc-length parametraization and evenly distribute the points. The points we named as vertices. Connect the adjacency vertices we received the edges. For an open curve we would have $n + 1$ vertices and n edges. For a closed curve we have n vertices and n edges. The vertices are primal of the system while the edges serve as dual to the primal. For the discrete curve, the tangent can be derived as the unit vector of each edge. We still want the orthogonal frame on each edge. Thus we introduced the idea of parallel transportation, which ensure the connectivity and dependency of each frame to its neighbour.

3.4 Parallel Transportation

In the modeling of discrete elastic rods, we initialize the frame on the first edge using a reference vector $\{0, 0, 1\}$ and the cross product with the tangent vector to define the normal, ensuring orthogonality. This approach also establishes the binormal for the first frame.

As we progress along the curve, each segment or "beam" — defined as a pair of edges and their connecting vertex — determines the turning angles necessary for transferring the frame. Parallel Transport is employed to advance the orthonormal frame from one vertex to the next, adjusting it according to these angles without adding any additional twist. This method preserves the frame's orientation relative to the curve's local geometry across $n - 2$ beams in a curve defined by n vertices. We want the Parallel Transportation to satisfy such properties:

$$P_i\left(\mathbf{t}^{i-1}\right) = \mathbf{t}^i, \quad P_i\left(\mathbf{t}^{i-1} \times \mathbf{t}^i\right) = \mathbf{t}^{i-1} \times \mathbf{t}^i$$

Where P_i is the transportation matrix, t_i is the tangent on ith edge. The P_i matrix can be described as rotate the vector around the axis from the cross product of t_i and t_{i-1} by the angle of turning angle. Such matrix and be coded by Rodrigues' rotation formula or quarterion. In fact many popular languages and libraries have implemented this formula, thus you just need to calculate the axis and angle.

3.5 Pointwise property and Edgewise property

In our model, it is crucial to distinguish between properties associated with vertices and those with edges. The curvature on any point along the edges is zero due to the straight-line nature of the segments. At the vertices, the curvature κ_i at the ith vertex is given by:

$$\kappa_i = \tan\left(\frac{\theta_i}{2}\right),$$

where θ_i is the turning angle at that vertex. Various discretizations of curvature, such as $\sin(\theta)$ and $\sin\left(\frac{\theta}{2}\right)$, are noted in the literature, all converging to the continuous curvature as the sampling density increases.

Furthermore, the Darboux vector Ω, representing vectorized curvature, is defined in our model as:

$$\Omega = \kappa b = \frac{2\mathbf{e}^{i-1} \times \mathbf{e}^i}{\left|\overline{\mathbf{e}}^{i-1}\right|\left|\overline{\mathbf{e}}^i\right| + \mathbf{e}^{i-1} \cdot \mathbf{e}^i}.$$

Additionally, the effective weight of each vertex, influenced by a generalized Voronoi diagram, is calculated as:

$$l_i = \frac{|e^{i-1}| + |e^i|}{2}.$$

4 DYNAMIC SYSTEM

we now turn our attention to analyzing the dynamic systems of the curve, a phase where we encountered significant challenges.

4.1 Revisit to Energy and Force

Consider an ideal particle in a vacuum. With no external forces it would either remain stationary or continue in uniform linear motion depending on its initial state. This fundamental principle guides our initial analysis of the energy dynamics within the curve. In the context of the DER model, two specific forms of energy are critical to understanding and predicting the motion of the curve

4.2 Bending Energy and Force

Bending energy quantifies the deviation from a straight line for both continuous and discrete curves. For a continuous curve, it is defined as: The definition allows us to apply such an integration of curvature:

$$E_{\text{bend}}(\Gamma) = \frac{1}{2}\int \alpha\boldsymbol{\omega}^2 \,\mathrm{d}s = \frac{1}{2}\int \alpha\boldsymbol{\kappa}^2 \,\mathrm{d}s$$

where α is the bending modulus. ω here means the rod's curvature vector expressed in material coordinates and measure the bending of the material frame. Since we are considering isotropic rod for now, we only need the square of ω. For discrete curves, the bending energy is adapted as such:

$$E_{\text{bend}}(\Gamma) = \frac{1}{2}\sum_{i=1}^{n} \alpha\left(\frac{\boldsymbol{\kappa}\mathbf{b}_i}{\bar{l}_i/2}\right)^2 \frac{\bar{l}_i}{2} = \sum_{i=1}^{n} \frac{\alpha\,(\boldsymbol{\kappa}\mathbf{b}_i)^2}{\bar{l}_i}$$

where curvature along each edge is considered zero. The darboux vector, describing curvature changes at discrete points, is integrated into this calculation, adjusted by the Voronoi weight of each vertex, thus converting this edgewise attribute into a pointwise energy metric.

4.3 Twisting Energy and Force

Twisting energy differs from bending energy as it solely depends on the twisting angle, an extrinsic property associated with the Bishop frame. Calculated using the formula:

$$E_{\text{twist}}(\Gamma) = \frac{1}{2}\int \beta\theta^2 \, ds,$$

where β is the twisting modulus and θ is the twisting angle per unit length. This integral quantifies the total twisting energy along the curve, reflecting the rod's resistance to torsional deformations.

4.4 Quasistatic update

Discussing the dynamics of twist and bending waves, we note that twist waves, which are governed by the rotational inertia of the cross-section, travel faster than bending waves. Bending waves are dispersive and their velocity depends on the wavelength λ, calculated as v_h/λ, where h is the rod radius and v is the speed of sound in the material. For large wavelengths $\lambda \gg h$, bending waves travel significantly slower. Given that our focus is on large-wavelength bending modes, resolving the faster temporal evolution of twist waves in computations is inefficient and unnecessary.

4.5 Time integrator

Once we apply force F on it. The position, velocity, and acceleration can be described as:

$$a_t = \frac{F}{m}$$
$$v_t = \int_0^t a dt + v_0$$
$$x_t = \int_0^t v dt + x_0$$

Such integration is called time integrator. There are three popular time integrators:

- `Forward Euler`:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\frac{\mathbf{f}_t}{m}$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t\mathbf{v}_t$$

- `Sympletic Euler`:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\frac{\mathbf{f}_t}{m}$$
$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t\mathbf{v}_{t+1}$$

- `Backward Euler`:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta t\mathbf{v}_{t+1}$$
$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\mathbf{M}^{-1}\mathbf{f}\,(\mathbf{x}_{t+1})$$

Both Forward and Sympletic Euler take full use of known information also referred as explicit integration. While the Backwards Euler requires the information from the future named as implicit integration. To do such we need to set up a sparse solver to compute the velocity of the particle. Eliminate $\mathbf{x}_{t+1}$:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\mathbf{M}^{-1}\mathbf{f}\,(\mathbf{x}_t + \Delta t\mathbf{v}_{t+1})$$

Linearize (one step of Newton's method):

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \Delta t\mathbf{M}^{-1}\left[\mathbf{f}\,(\mathbf{x}_t) + \frac{\partial\mathbf{f}}{\partial\mathbf{x}}\,(\mathbf{x}_t)\,\Delta t\mathbf{v}_{t+1}\right]$$

Even though the implementation is quite difficult, the Backwards Euler can increase the size of time step significantly. For instance, to retrieve the similar output in Forward and Symplectic Euler in 1e-6, the Backward Euler only asks the time step at 1e-4. Such huge improvement save the time for CFD researchers and artists from film industries and make it possible to deliver a promising simulation. Following the DER paper, we choose to use symplectic Euler as our integrator.

5 CONSTRAINT TO THE SYSTEM

5.1 Inextensibility

In order to maintain the inextensibility, we use the fast projection method from Goldenthal, 2007

6 METHODOLOGY

6.1 Overview

Our first approach was to complete the basic goal- visualizing a cubic Bezier spline with a UI.

Algorithm 1 Fast Projection

$\mathbf{x}_0^{n+1} \leftarrow \mathbf{x}^n + \tilde{\mathbf{v}}^{n+1} \cdot h$
while strain exceeds threshold **do**
  $\delta\boldsymbol{\lambda} \leftarrow -\left[\nabla \mathbf{C}\left(\mathbf{x}_j^{n+1}\right) \nabla \mathbf{C}\left(\mathbf{x}_j^{n+1}\right)^T\right]^{-1} \mathbf{C}\left(\mathbf{x}_j^{n+1}\right)$
  $\delta\mathbf{x} \leftarrow \nabla \mathbf{C}\left(\mathbf{x}_j^{n+1}\right)^T \delta\boldsymbol{\lambda}$
  $\mathbf{x}_{j+1}^{n+1} \leftarrow \mathbf{x}_j^{n+1} + \delta\mathbf{x}$
  $j \leftarrow j + 1$
end while
$\mathbf{v}^{n+1} \leftarrow (\mathbf{x}^{n+1} - \mathbf{x}^n)/h$

6.2 Interactive Simulation Interface

We developed this using ImGui and the Polyscope visualization framework. The interface allows users to manipulate parameters and observe the effects immediately, hopefully giving them an intuition of the Discrete Elastic Rods dynamics.

6.2.1 Interface Implementation. The interface is implemented in C++, integrating ImGui for creating interactive sliders, buttons, and visualization windows. Users can input and modify parameters such as control points for Bezier curves, which are used to define the shape and properties of the elastic rods. These parameters include:

- Control points adjustments with real-time visualization.
- Selection between different continuity conditions (C0 and C1).

6.2.2 Simulation Control. The simulation's execution can be dynamically controlled through the interface. Users can start, pause, or stop the simulation, and reset or initialize curves based on the specified sample number and control points. This level of control is vital for conducting iterative experiments and observing the behavior under various conditions.

6.2.3 Visualization and Feedback. Polyscope is used to render the curves and provide a responsive feedback loop to the user. This includes visualizing the effects of adjustments in real-time, which is essential for educational and experimental purposes. The system supports displaying multiple curves and their evolution over time, highlighting the effects of different parameters on the rod dynamics.

6.2.4 Example Usage. For instance, a user can manipulate the bezier curve parameters to model the initial shape of an elastic rod, then apply different continuity conditions to see how they affect the rod's behavior. Adjustments to the physical properties like bending and twisting moduli can be made to study their impact on the stability and dynamics of the rod.

6.3 Reach Goal

We simultaneously worked on implementing the algorithm for the paper as follows:

6.4 Algorithm Description

The core of our simulation relies on the modified version of the DER algorithm, which includes the following key steps:

(1) **Initialization of Geometry:** The rods are discretized into vertices (primals) and edges (duals), forming the polyline that approximates the continuous rod. This discretization is critical as it impacts the precision and stability of the simulation.
(2) **Bishop Frame Construction:** For each segment, a twist-free Bishop frame is computed. The frame initialization starts at the first vertex, propagating through the polyline using parallel transport to maintain continuity and avoid the introduction of artificial torsion.
(3) **Energy Computation:** The system's potential energy is calculated based on bending and twisting energies. These calculations inform the dynamics of the rod under various forces and constraints.
(4) **Dynamic Simulation:** Utilizing symplectic Euler integration, the simulation advances in time, updating vertex positions and orientations according to computed forces derived from the potential energy gradients.

6.5 Implementation Details

The implementation was performed in C++ using a combination of libraries suited for high-performance computing and real-time interactive visualization:

- **Platform and Libraries:**
  - **Language:** C++
  - **Mathematical Computations:** Eigen, GLM
  - **Graphical Rendering and Interaction:** Polyscope
  - **Numerical Linear Algebra:** SuiteSparse (optional for performance optimization)

Eigen and GLM provide the tools for efficient matrix and vector operations, while Polyscope is our graphical interface to visualize the simulation results. SuiteSparse is optionally used to accelerate computations involving large sparse systems, which are common in dynamic simulations involving physical constraints.

6.6 Machines Used

All tests run were conducted on an ARM-based MacBook Pro (M1 Chip), and a Windows (Intel i7-13700H, 16 GB DDR4).

6.7 Code Specifics

The following pseudocode highlights the critical portion of the simulation loop, showcasing our contributions to the original DER algorithm:

```
function simulateDiscreteElasticRods(dt):
    for each rod in simulation:
        computeBishopFrames(rod)
        computeEnergies(rod)
        for each vertex in rod:
            if not fixed:
                updatePosition(vertex, dt)
        updateGraphics()
    end for
end function
```

This loop runs continuously, updating the state of each rod based on the forces acting on it.

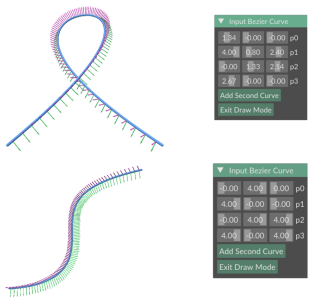

Fig. 3. Successful display of twist-free Bishop frame (pink: normal, green: binormal) on various user-defined Bezier curves

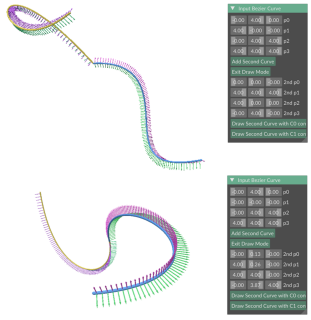

Fig. 4. Examples of successful Bezier Curves connected by C0 continuity (upper) and C1 continuity (lower)

7 RESULTS

Our implementation consists of two subprojects: Bishop frame visualization on a user-defined Bezier Curve, and an implementation of the Mitchell Buckling Instability on a closed, circular spline.

In the first sub-project we create an interface that allows the user to specify the four control points of a cubic Bezier curve, by dragging sliders for the x,y,z coordinates of the points (Fig.3).

The interface also allows the user to add a second curve, with C0 or C1 continuity (Fig.4). There were some issues with the C1 continuity implementation with the Bishop frame normal displaying a sharp change between the two halves of the curve, or sometimes the curve becomes discontinuous. This is thought to be due to a miscalculation in the rotation and translation matrices applied to the second curve to enforce .

The second sub-project was our implementation of Mithcell's Buckling instability. We start with a circular curve lying on the x-z plane, and apply an increasing twist angle to it. The expected result is that the circle will "buckle" out of the plane (Fig.5), (Fig.6). We adjust alpha and beta value parameters to observe how intrinsic properties of the material affect the buckling critical point. We observe that a higher β value results in more erratic buckling, which was not the expected result.

Some of the limitations we faced relate to the user interface. The current implementation lacks intuitive user interfaces for non-experts. There is currently no view of the control points when the user draws the spline curve, and instead we use sliders to draw each control point in. The intuition is still there, but it takes slightly longer to grasp an understanding of how everything is working, and it is difficult to create desired shapes without some trial and error.

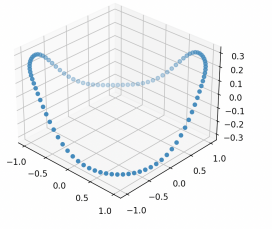

Fig. 5. This is the result from our python code using auto differentiation to calculate the forces. This was our most stable result, but still had some inconsistencies where it would become slightly unstable at points. This configuration had α,β =1, and θ_n =60

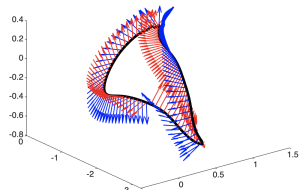

Fig. 6. MATLAB visualization, with parameters α, β = 1, and $\theta = \pi$. The rod is buckling out of plane, not adhering to the relationship presented in the Michell's instability equation.

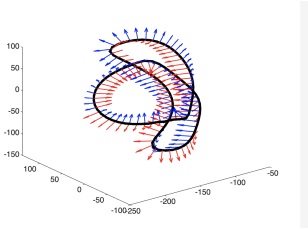

Fig. 7. Parameters α =1, β= 100, $\theta = \pi$. This configuration should buckle out of plane, if you plug in the values the θ_n exceeds the critical point. However it was not buckling the way we expected and gradually started to become more stable, which was also not expressing Michell's instability correctly.

8 REFLECTION

Overall, we were relatively content with the implementation we did, despite not achieving the reach goal we intended to. We implemented our basic goal well, but a point and click implementation is missing, where the user can click on the screen to generate control points. This visualization of control points (good way to say why we couldn't implement it- not available, too difficult etc.). Initially we were following a closely related assignment provided by Justin Solomon, an associate professor at MIT. This assignment asked us

to simulate the behavior of a closed loop from naturally straight, isotropic rods in MATLAB. This gave us a framework to base our code on, however created many problems in implementing the bending and twisting force calculations. Namely, our simulation didn't satisfy the Michell's instabily relationship presented in the paper: $\theta_c^n = 2\pi\sqrt{3}/(\beta/\alpha)$
This essentially states that the rod will buckle out of plane when the twisting angle θ^n reaches a critical value of θ_c, and θ_c correlates to α, the bending stiffness, and β, the twisting stiffness. which means how much the rod resists bending or twisting - higher stiffness means more twist to induce buckling. The rod is expected to remain stable until this critical value, so with the values we input, where $\alpha, \beta = 1, and \theta_n = \pi$ we can see that θ_n hasn't reached the 2π critical value, yet the model is unstable. It should be twisting uniformly along the rod but should not be buckling

At first we were just implementing the equations straight from the paper, which did not yield the results we wanted. We then went back and examined the paper closely, fixing up some of the syntactical errors, which improved the result but still didn't satisfy the Michell's instability relationship. The biggest issue we faced was a lack of theory knowledge, as we and we spent a large amount of time as a group reading through and discussing the paper. As we wanted to complete our visualization in Polyscope, we migrated the code to C++, strictly following the .

We narrowed down our issues to the bending and twisting force functions. As these functions presented so many issues we decided to try a different implementation, this time in Python using the JAX library. JAX specializes in automatic differentiation, a powerful technique that automatically computes the gradients (derivatives) of functions. Instead of the traditional method of manually deriving force equations for each segment of the rod, we defined a comprehensive energy function, computeEnergy. This function encapsulated all the physical laws affecting our system, such as elastic potentials and kinematic constraints, and described how the total energy of the system varied with changes in the rod's configuration. The ability to automatically calculate the gradient of this energy function with respect to the positions of the rod's vertices was pivotal. Using jax.grad, we were able to convert calculating forces into the much more straightforward task of energy differentiation. In physics, the forces acting on a system can be derived from the negative gradient of the potential energy. By leveraging JAX to compute these gradients, we effectively had JAX derive the necessary forces for us. This essentially cut out the force calculations that were causing issues previously.

Extensions

A possible extension to implement would be the visualization of knots to test out the simulation. This would present the user with an intriguing way of understanding how the discrete elastic rod works. We could also implement spline curves with c1 continuity.
The paper suggests enhancing boundary conditions beyond the current fixed and free configurations, where the ends of the rod are either held in place or allowed to move under forces. The proposed extensions include incorporating contact dynamics, such as frictional interactions and engagements with other bodies. This advancement would allow for more realistic simulations by accurately reflecting how rods interact with their environment, improving the fidelity and applicability of the model.
Further, there could be extensions to our current manifold projection method. Manifold projection is utilized to enforce constraints such as inextensibility and rigid body coupling by projecting the rod onto a configuration that satisfies these constraints. However, this technique can result in energy dissipation, as modifications to the positions or velocities of the rod may lead to a loss of kinetic energy. While the fast projection algorithm is effective in enforcing these constraints efficiently, it may not be suitable for applications where conserving energy is critical, as it can compromise the fidelity of energy conservation.
Another possible extension would be with the implementation of adaptive discretization. Adaptive discretization in the context of discrete elastic rods involves adjusting the complexity of the model's mesh based on the points of stress and strain observed during simulations. As the rod bends or twists, certain areas may experience more significant changes compared to others. The adaptive discretization algorithm identifies these high-stress areas and increases the mesh resolution there to capture the behavior more accurately. Meanwhile, regions with less activity can have fewer mesh points, conserving computational resources. This would serve as a way to make the algorithm even more efficient in terms of speed and resource usage.

9 CITATIONS AND BIBLIOGRAPHIES

[**?**] University, Miklós Bergou Columbia, et al. "Discrete Elastic Rods: ACM SIGGRAPH 2008 Papers." ACM Conferences, 1 Aug. 2008, dl.acm.org/doi/10.1145/1399504.1360662.

[**?**] Goldenthal, Rony, et al. (PDF) Efficient Simulation of Inextensible Cloth, 29 July 2007, www.researchgate.net/publication/210222835_Efficient_simula