Optimizing Parameters for Static Equilibrium of Discrete Elastic Rods with Active-Set Cholesky

Tetsuya Takahashi and Christopher Batty

***Abstract*—We propose a parameter optimization method for achieving static equilibrium of discrete elastic rods. Our method simultaneously optimizes material stiffness and rest shape parameters under box constraints to exactly enforce zero net forces while avoiding stability issues and violations of physical laws. For efficiency, we split our constrained optimization problem into primal and dual subproblems via the augmented Lagrangian method, while handling the dual maximization subproblem via simple vector updates. To efficiently solve the box-constrained primal minimization subproblem, we propose a new active-set Cholesky preconditioner for variants of conjugate gradient solvers with active sets. Our method surpasses prior work in generality, robustness, and speed.**

***Index Terms*—Parameter optimization, hair simulation, inverse problem, active-set method**

I. Introduction

When modeling deformable objects or fabricating elastic materials, sagging under external forces, such as gravity and applied loads, can ruin specific shapes carefully designed by artists and designers or captured from real world counterparts. This sagging problem arises because the external forces are often implicitly considered during modeling but simulators typically neglect it during initialization [1]–[5]. To prevent this issue, we aim to achieve static equilibrium of the objects preserving their original shapes at initialization. While stiffening materials can reduce the sagging, it alters the dynamic response of the elastic material. Alternatively, modifying rest shape parameters can achieve static equilibrium, though substantial rest shape changes can introduce stability problems or increase the likelihood of undesirable rest configurations [5].

In this paper, we focus on elastic objects with one-dimensional structures (e.g., hair, cables, strands, etc.). We employ discrete elastic rods (DER) [6, 7] for strand simulation due to their generality, flexibility, and efficiency, as adopted in various applications with forward simulations [8]–[11] and inverse problems [2, 12]–[14].

To address the sagging problem, we propose a new parameter optimization method that is guaranteed (at solver convergence) to achieve static equilibrium of DER-based strands. Our method simultaneously optimizes rest shape and material stiffness parameters while minimizing changes in the parameters and satisfying their corresponding box constraints; the latter are imposed to avoid violating physical laws or introducing stability issues in forward simulation. We formulate our problem as a constrained minimization and decompose it into primal and dual subproblems via the augmented Lagrangian method (ALM) [15], addressing the primal minimization subproblem with a Newton-type optimizer and the dual maximization subproblem via vector updates. To efficiently and accurately handle the primal subproblem, we take the box constraints into account and solve the symmetric positive definite (SPD) inner systems using a variant of conjugate gradient (CG) with active sets, modified proportioning with reduced gradient projections (MPRGP) [16]. Additionally, we propose a new preconditioner based on a Cholesky-based direct solver with active sets to accelerate the convergence of MPRGP. Figure 1 shows our method in action.

Tetsuya Takahashi is with Tencent America, USA; Christopher Batty is with University of Waterloo, Canada

II. Related Work

A. Elastic Rod Simulation

To capture bending and twisting of one-dimensional elastic strand structures, Pai [17] introduced the Cosserat theory into graphics, which was later extended to dynamical systems [18]. The Cosserat theory has since been adopted within position-based dynamics [19] and further extended with quaternions [4, 20], volume constraints [21], and projective dynamics [22]. Instead of evolving edge rotations, Shi et al. [23] proposed using a volume-like torsion energy to capture twisting effects. Curvature-based approaches were also presented to capture elastic rod dynamics with a smaller number of degrees of freedom (DOF) [24, 25]. From the various methods for simulating elastic strands, we selected DER [6, 7] because it can handle general bending and twisting cases with few DOFs.

B. Sag-Free Simulation

To achieve sag-free simulations (or equivalently, static equilibrium), one common approach is to enforce zero net forces in the dynamical system. Hadap [26] proposed using inverse dynamics to solve nonlinear force equations for reduced multibody systems, although such methods may not necessarily be applicable to general maximal coordinate systems (e.g., finite-element-method-based deformables and DER). To solve nonlinear force equations, the asymptotic numerical method (ANM) was employed, achieving faster convergence compared to Newton-type optimizers [27, 28], although it remains unclear how to incorporate box constraints, which help prevent stability problems and violations of physical laws. Curvature-based solvers have also been employed, since they need fewer DOFs to represent elastic rod dynamics. Derouet-Jourdan et al. [29] presented a method for achieving static equilibrium of curvature-based elastic strands [24] and extended their work with frictional contacts [30]. The curvature-based formulations

Fig. 1. Our method optimizes material stiffness and rest shape parameters to achieve static equilibrium for DER-based strands. It preserves the original hairstyle without sagging, demonstrates natural dynamic strand behaviors in response to prescribed motions of the root vertices, and eventually restores the strands to their initial configuration. Our parameter optimization required 713 seconds for 3.2k strands; forward simulation without collision handling took 25 seconds per frame.

have extensively been used for inverse problems to model, design, and fabricate elastic rods [31]–[34].

To achieve static equilibrium in more general contexts, Twigg and Kačić-Alesić proposed minimizing the force norm by optimizing rest shape parameters in mass-spring systems consisting of one-dimensional elements [1]. Takahashi and Batty extended their approach to DER [5] and proposed optimizing the rest length, rest curvature, and rest twist parameters while penalizing the force norm in the kinetic energy metric. However, this approach would fail to achieve static equilibrium because fully optimized rest shape parameters alone could still be insufficient to achieve zero net forces. Sag-free DER simulations have been attempted in other previous work as well [10, 35], albeit without offering formulation and algorithmic specifics. Rest shape optimization has also been applied to two-dimensional sheet-like structures [36] and three-dimensional deformable volumes [37, 38]. To enforce zero-net-force constraints, adjoint methods have been extensively used [2, 12, 39]. While this approach can be efficient and can ensure stable local minima by examining the Hessian of the DER objectives, handling the full Hessian due to differentiation of nonlinear force constraints can be complicated [9, 12]. Given that DER Hessian treatments are the subject of ongoing study [12, 23], we formulate our problem without explicitly incorporating the Hessian into the parameter optimization. While finite differences could bypass this complexity [40], they are known to be significantly slower than optimization with analytical gradients [15].

As an alternative, Hsu et al. [3, 4] proposed a global-local initialization method that first computes global forces to achieve static equilibrium and then adjusts local elements to generate such forces. However, this approach would require excessive stiffening of materials and is applicable only to specific simulation approaches that can locally define forces without coupling neighboring elements. As such, the global-local initialization is not applicable to DER [5].

C. Preconditioning with Active Sets

Preconditioners have been extensively used to reduce the number of necessary iterations for iterative solvers, such as CG. Preconditioners are also applicable and effective for variants of CG with an active-set method, e.g., MPRGP (which updates active sets in each iteration to find their optimal sets and solution) [16], by taking active sets into account during the preconditioning.

While Narain et al. [41] used modified incomplete Cholesky (MIC) preconditioning [42] for MPRGP, the derivations and rationale behind their approach are not explained, and it is not necessarily clear from their publicly available source code how to adapt it to support variants of Cholesky solvers (e.g., LDLT-type decomposition with 2×2 diagonal blocks [43]). Recently, Takahashi and Batty [44] proposed a smoothed aggregation algebraic multigrid (SAAMG) preconditioner for MPRGP, using an SPAI-0 smoother [45] with active sets (which can easily be modified to a diagonal, weighted Jacobi, or symmetric Gauss-Seidel (SGS) smoother/preconditioner). However, their focus is on systems with M-matrices, and thus the effectiveness of their preconditioning on non-M-matrices, such as those in our formulation, remains unverified.

III. Contributions

We propose optimizing both the material stiffness and rest shape parameters simultaneously while enforcing zero net forces as a hard constraint via ALM to ensure the static equilibrium of DER-based strands at solver convergence. To reduce the optimization cost, we propose two techniques. First, we arrange the optimization parameters in an interleaved way to yield a banded system by leveraging the one-dimensional strand structure. This arrangement eliminates the need for matrix reordering in Cholesky-based solvers. Second, given the overlapping force spaces with 4D curvatures in the bending formulation [6, 9], we consider only two rest curvature variables per vertex in the optimization, reducing the DOF count and improving efficiency. Moreover, instead of using penalty-based methods [5], we employ an active-set-based iterative solver, MPRGP [16], to enforce the box constraints precisely. We also propose a new active-set Cholesky preconditioner to accelerate the convergence of MPRGP. Because of these improvements, our method can more robustly and efficiently achieve static equilibrium of DER-based strands.

IV. Discrete Elastic Rods Preliminaries

For completeness, we briefly summarize key details of the DER formulation [6, 7] before explaining our parameter optimization method (Sec. V) and active-set Cholesky preconditioner (Sec. VI-A). In our framework, we exclude contacts (as our goal is to achieve static equilibrium which introduces no contacts if none exist at initialization) and thus process each strand independently, focusing on a single strand below.

Given a strand discretized into N vertices with positions $\mathbf{x} = [\mathbf{x}_0^T, \ldots, \mathbf{x}_{N-1}^T]^T \in \mathbb{R}^{3N}$ and $N-1$ edges with angles

$\boldsymbol{\theta} = [\theta_0, \ldots, \theta_{N-2}]^T \in \mathbb{R}^{N-1}$, the generalized positions can be defined by interleaving vertex and edge variables as $\mathbf{q} = [\mathbf{x}_0^T, \theta_0, \ldots, \theta_{N-2}, \mathbf{x}_{N-1}^T]^T \in \mathbb{R}^{4N-1}$. This arrangement leads to a banded Hessian due to the locally defined DER objectives and time-parallel transport [7]. For simplicity, we assume a strand with a perfectly circular cross-section and uniform radius r. The root end of the strand is minimally clamped, fixing $\mathbf{x}_0, \boldsymbol{\theta}_0$, and $\mathbf{x}_1$, which yields $4N-8$ active DOFs [5]. For more details, we refer to a book on DER [46].

We define a minimization problem for a forward simulation step with a DER objective $E(\mathbf{q})$ [5] as $\mathbf{q} = \arg\min_{\mathbf{q}} E(\mathbf{q})$, where $E(\mathbf{q}) = E_{\text{in}}(\mathbf{q}) + E_{\text{st}}(\mathbf{q}) + E_{\text{be}}(\mathbf{q}) + E_{\text{tw}}(\mathbf{q})$, and $E_{\text{in}}(\mathbf{q})$, $E_{\text{st}}(\mathbf{q})$, $E_{\text{be}}(\mathbf{q})$, and $E_{\text{tw}}(\mathbf{q})$ represent inertia, stretching, bending, and twisting objectives, respectively. We define these objectives in the following sections and provide their gradient in the supplementary material.

A. Inertia

We define the inertia objective as follows:

$$E_{\text{in}}(\mathbf{q}) = \frac{1}{2\Delta t^2} \left\| \mathbf{q} - \mathbf{q}^* \right\|_{\mathbf{M}}^2, \tag{1}$$

where $\mathbf{q}^* = \mathbf{q}^t + \Delta t \dot{\mathbf{q}}^t + \Delta t^2 \mathbf{M}^{-1} \mathbf{f}_{\text{ext}}$, $\mathbf{q}^t$ and $\dot{\mathbf{q}}^t$ denote the generalized positions and velocities at time t, respectively, Δt the time step size, $\mathbf{M}$ a diagonal generalized mass matrix [7, 46], and $\mathbf{f}_{\text{ext}}$ generalized external forces. While we set gravity as the only external force, one could specify spatially varying external loads and winds (even on edge angles) [5].

B. Stretching

We define the stretching objective [7] as $E_{\text{st}}(\mathbf{q}) = \sum_{i=1}^{N-2} E_{\text{st},i}(\mathbf{x}_i, \mathbf{x}_{i+1})$, where $E_{\text{st},i}(\mathbf{x}_i, \mathbf{x}_{i+1})$ is given by

$$E_{\text{st},i}(\mathbf{x}_i, \mathbf{x}_{i+1}) = \frac{1}{2} \left(\frac{s\boldsymbol{\alpha}_i \pi r^2}{\bar{\mathbf{l}}_i} \right) (\mathbf{l}_i - \bar{\mathbf{l}}_i)^2, \tag{2}$$

with s denoting a global constant scalar (defined in Sec. V-A) that scales stiffness parameters, $\boldsymbol{\alpha}_i$ as the stiffness parameter for stretching on edge i (we use different stiffness parameters for stretching, bending, and twisting to enable finer control [5, 47], unlike [7] which uses only Young's and shear modulus), $\mathbf{l}_i = \left\| \mathbf{x}_{i+1} - \mathbf{x}_i \right\|_2$ representing the length of edge i, and $\bar{\mathbf{l}}_i$ as its rest length.

C. Bending

We define the bending objective [6, 9] as $E_{\text{be}}(\mathbf{q}) = \sum_{i=1}^{N-2} E_{\text{be},i}(\mathbf{y}_i)$, where $E_{\text{be},i}(\mathbf{y}_i)$ is given by

$$E_{\text{be},i}(\mathbf{y}_i) = \frac{1}{2} \left(\frac{s\boldsymbol{\beta}_i \pi r^4}{4(\bar{\mathbf{l}}_{i-1} + \bar{\mathbf{l}}_i)} \right) \left\| \boldsymbol{\kappa}_i - \bar{\boldsymbol{\kappa}}_i \right\|_2^2, \tag{3}$$

and $\mathbf{y}_i = (\mathbf{x}_{i-1}^T, \theta_{i-1}, \mathbf{x}_i^T, \theta_i, \mathbf{x}_{i+1}^T)^T \in \mathbb{R}^{11}$, $\boldsymbol{\beta}_i$ denotes the bending stiffness at vertex i, and $\boldsymbol{\kappa}_i$ and $\bar{\boldsymbol{\kappa}}_i$ are the four-dimensional curvature and rest curvature, respectively [6, 9]. We use the formulation with four-dimensional curvatures and rest curvatures [6, 9] because a simplified bending formulation with two-dimensional curvatures and rest curvatures via averaged material frames [7] is known to fail to evaluate strand bending correctly due to the non-unit and non-orthogonal averaged material frames [12, 48]. We provide more details on these bending formulations in the supplementary material.

D. Twisting

We define the twisting objective [7, 46] as $E_{\text{tw}}(\mathbf{q}) = \sum_{i=1}^{N-2} E_{\text{tw},i}(\mathbf{y}_i)$, where $E_{\text{tw},i}(\mathbf{y}_i)$ is given by

$$E_{\text{tw},i}(\mathbf{y}_i) = \frac{1}{2} \left(\frac{s\boldsymbol{\gamma}_i \pi r^4}{(\bar{\mathbf{l}}_{i-1} + \bar{\mathbf{l}}_i)} \right) (\mathbf{m}_i - \bar{\mathbf{m}}_i)^2, \tag{4}$$

with $\boldsymbol{\gamma}_i$ representing the twisting stiffness parameter, and $\mathbf{m}_i$ and $\bar{\mathbf{m}}_i$ the twist (including the reference twist [7, 46, 49]) and rest twist, respectively.

V. Parameter Optimization

Our goal is, at the initialization stage, to find a set of optimal rest shape and material stiffness parameters which achieve static equilibrium of the strands. Letting $\mathbf{p}$ be the optimization variable (to be detailed in Sec. V-A), our parameter optimization task is formulated as a constrained minimization problem:

$$\mathbf{p} = \underset{\mathbf{p}_{\min} \le \mathbf{p} \le \mathbf{p}_{\max}, \mathbf{c}(\mathbf{p}) = 0}{\arg\min} R(\mathbf{p}), \; R(\mathbf{p}) = \frac{1}{2} \left\| \mathbf{p} - \mathbf{p}_0 \right\|_{\mathbf{W}}^2, \tag{5}$$

where $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$ denote the box constraints (lower and upper bounds for $\mathbf{p}$, respectively) to ensure compliance with physical laws and limit significant parameter changes [5], and $\mathbf{c}(\mathbf{p})$ (defined in Sec. V-C) is a nonlinear hard constraint to enforce zero net forces. We define $\mathbf{p}_0$ as the initial value for $\mathbf{p}$ before optimization, and $R(\mathbf{p})$ acts as a regularizer to minimize the deviation of $\mathbf{p}$ from $\mathbf{p}_0$ with a diagonal weight matrix $\mathbf{W}$. We aim to solve (5) using ALM and a Newton-type optimizer (see Algorithm 1 in Sec. V-F for the procedure).

A. Optimization Variable

We simultaneously optimize parameters of the rest shape $(\bar{\mathbf{l}}, \bar{\boldsymbol{\kappa}}, \bar{\mathbf{m}})$ and material stiffness $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ to achieve static equilibrium of an elastic strand. To improve robustness and efficiency, we use the following approaches to define the optimization variable $\mathbf{p}$.

Variable Scaling. While one typically specifies stiffness coefficients directly as $\mathbf{c}_{\text{st}}(= s\boldsymbol{\alpha})$, $\mathbf{c}_{\text{be}}(= s\boldsymbol{\beta})$, and $\mathbf{c}_{\text{tw}}(= s\boldsymbol{\gamma})$, we instead optimize $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$. This approach addresses the significant scale differences between the stiffness coefficients ($\mathbf{c}_{\text{st}}$, $\mathbf{c}_{\text{be}}$, $\mathbf{c}_{\text{tw}}$ on the order of 10^9 [7, 46]) and rest shape parameters ($\bar{\mathbf{l}}, \bar{\boldsymbol{\kappa}}, \bar{\mathbf{m}}$ up to 10), enabling stable convergence of numerical optimizers and mitigating numerical issues. To ensure similar value scales, we initialize the stiffness scaling constant s as $s = \frac{1}{3(N-2)} \sum_{i=1}^{N-2} (\mathbf{c}_{\text{st},i} + \mathbf{c}_{\text{be},i} + \mathbf{c}_{\text{tw},i})$, and subsequently set $\boldsymbol{\alpha} = \mathbf{c}_{\text{st}}/s$, $\boldsymbol{\beta} = \mathbf{c}_{\text{be}}/s$, and $\boldsymbol{\gamma} = \mathbf{c}_{\text{tw}}/s$. While equivalent formulations could be derived through proper scaling of (5) and (6), our change of variables ensures consistent scales in the Jacobians (see the supplementary material), thus avoiding catastrophic cancellation and enhancing numerical stability. In practice, we found that parameter optimization without this change of variables fails to achieve static equilibrium.

Variable Interleaving. Exploiting the one-dimensional structure and locally defined DER objectives of strands, we interleave the rest shape and stiffness parameters to ensure banded inner linear systems (which can be solved with fewer fill-ins via Cholesky-based direct solvers) within Newton-type optimizers, similar to the approach used in forward simulations

[7]. Note that we exclude $\bar{\mathrm{l}}_0$ on the fixed edge from the parameter optimization [5].

Reduced Rest Curvatures. Furthermore, we optimize only two rest curvature DOFs per vertex, $\bar{\boldsymbol{\kappa}}_{i,0}$ and $\bar{\boldsymbol{\kappa}}_{i,1}$, while excluding $\bar{\boldsymbol{\kappa}}_{i,2}$ and $\bar{\boldsymbol{\kappa}}_{i,3}$, in contrast to previous work [5]. This choice stems from the association of $\bar{\boldsymbol{\kappa}}_{i,0}$ and $\bar{\boldsymbol{\kappa}}_{i,2}$ with the second material frame, while $\bar{\boldsymbol{\kappa}}_{i,1}$ and $\bar{\boldsymbol{\kappa}}_{i,3}$ relate to the first [6]. Their overlapping force spaces lead to similar final values for $\bar{\boldsymbol{\kappa}}_{i,0}$ and $\bar{\boldsymbol{\kappa}}_{i,2}$ without expanding the force spaces necessary to achieve static equilibrium, while including $\bar{\boldsymbol{\kappa}}_{i,2}$ and $\bar{\boldsymbol{\kappa}}_{i,3}$ would increase the optimization cost due to the additional DOFs. In practice, we synchronize these variables by setting $\bar{\boldsymbol{\kappa}}_{i,2} = \bar{\boldsymbol{\kappa}}_{i,0}$ and $\bar{\boldsymbol{\kappa}}_{i,3} = \bar{\boldsymbol{\kappa}}_{i,1}$ whenever $\bar{\boldsymbol{\kappa}}_{i,0}$ and $\bar{\boldsymbol{\kappa}}_{i,1}$ are updated. This approach effectively reduces the dimensionality of the rest curvature from 4D to 2D; however, we still use 4D rest curvatures in (3) to ensure the same dimensionality for curvature $\boldsymbol{\kappa}$ and rest curvature $\bar{\boldsymbol{\kappa}}$.

Based on these choices, we define the optimization variable for the parameters as $\mathbf{p} = (\bar{\boldsymbol{\kappa}}_{1,0}, \bar{\boldsymbol{\kappa}}_{1,1}, \boldsymbol{\beta}_1, \bar{\mathbf{m}}_1, \boldsymbol{\gamma}_1, \bar{\mathrm{l}}_1, \boldsymbol{\alpha}_1, \ldots, \bar{\boldsymbol{\kappa}}_{N-2,0}, \bar{\boldsymbol{\kappa}}_{N-2,1}, \boldsymbol{\beta}_{N-2}, \bar{\mathbf{m}}_{N-2}, \boldsymbol{\gamma}_{N-2}, \bar{\mathrm{l}}_{N-2}, \boldsymbol{\alpha}_{N-2})^T \in \mathbb{R}^{7N-14}$. The subset corresponding to the rest shape parameters consists of $4N - 8$ DOFs, which matches the number of active DOFs for the generalized positions $\mathbf{q}$. Any set of rest shape parameter values can exactly be converted into any corresponding set of generalized positions (and vice versa), and thus either representation could (potentially) be optimized for. We choose to use the rest shape parameters in the optimization because this approach offers more precise control over the magnitude of forces via the box constraints, and makes the Jacobians of the generalized forces with respect to the rest shape parameters much simpler [5].

B. Augmented Lagrangian Method

To handle the nonlinear constraints $\mathbf{c}(\mathbf{p}) = 0$ in (5), we use ALM [15, 36] and define an augmented Lagrangian objective with a Lagrange multiplier $\boldsymbol{\lambda}$ and a penalty parameter ρ (> 0):

$$L(\mathbf{p}, \boldsymbol{\lambda}) = R(\mathbf{p}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{p}) + \frac{\rho}{2} \left\| \mathbf{c}(\mathbf{p}) \right\|_2^2 . \tag{6}$$

We then reformulate the parameter optimization (5) as

$$\mathbf{p}, \boldsymbol{\lambda} = \underset{\mathbf{p}_{\min} \le \mathbf{p} \le \mathbf{p}_{\max}}{\arg\min} \; \underset{\boldsymbol{\lambda}}{\arg\max} \, L(\mathbf{p}, \boldsymbol{\lambda}), \tag{7}$$

which we optimize by iteratively solving primal and dual subproblems [15]. The primal minimization subproblem, with fixed $\boldsymbol{\lambda}^k$ for iteration index k (initialized with $\boldsymbol{\lambda}^0 = 0$), is defined as

$$\mathbf{p}^{k+1} = \underset{\mathbf{p}_{\min} \le \mathbf{p} \le \mathbf{p}_{\max}}{\arg\min} \; L(\mathbf{p}, \boldsymbol{\lambda}^k), \tag{8}$$

while the dual maximization subproblem can be solved via a simple vector update:

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \rho \mathbf{c}(\mathbf{p}^{k+1}). \tag{9}$$

Thus, efficiently solving the primal subproblem (8) is crucial for fast parameter optimization.

C. Zero Net Force Constraints

Given the total force due to the DER objectives, represented as $\mathbf{f}(\mathbf{p}) = -\nabla E(\mathbf{q})$, it seems natural to define $\mathbf{c}(\mathbf{p}) = \mathbf{f}(\mathbf{p})$ to enforce zero net forces. However, in this case, the term $\frac{\rho}{2} \left\| \mathbf{c}(\mathbf{p}) \right\|_2^2$ in (6) becomes equivalent to a quadratic penalty on the total force, which is numerically ill-conditioned and fails to correctly account for the force contributions within the system [5]. To avoid these issues, we penalize the generalized forces in the kinetic energy metric [5] and define $\mathbf{c}(\mathbf{p})$ as

$$\mathbf{c}(\mathbf{p}) = \mathbf{M}^{-\frac{1}{2}} \mathbf{f}(\mathbf{p}). \tag{10}$$

Consequently, $L(\mathbf{p}, \boldsymbol{\lambda})$ in (6) can be rewritten as

$$L(\mathbf{p}, \boldsymbol{\lambda}) = R(\mathbf{p}) - \boldsymbol{\lambda}^T \mathbf{M}^{-\frac{1}{2}} \mathbf{f}(\mathbf{p}) + \frac{\rho}{2} \left\| \mathbf{f}(\mathbf{p}) \right\|_{\mathbf{M}^{-1}}^2 . \tag{11}$$

This objective (11) is numerically equivalent to the formulation proposed by Takahashi and Batty [5] when considering only the rest shape parameters (excluding stiffness parameters) and setting $\boldsymbol{\lambda} = 0$. Thus, their formulation, which treats $\mathbf{c}(\mathbf{p})$ as a soft constraint, can be regarded as a subset of ours. In the following, we assume $\Delta t = 1$ to simplify the formulations without affecting the optimization results [5].

D. Box Constraints

We impose box constraints on the optimization parameters to prevent stability problems (in forward simulation) which can be caused by too significant changes in the parameter values. Specifically, we define $\bar{\mathrm{l}}_{\min} \le \bar{\mathrm{l}} \le \bar{\mathrm{l}}_{\max}$ (except for $\bar{\mathrm{l}}_0$), $\bar{\boldsymbol{\kappa}}_{\min} \le \bar{\boldsymbol{\kappa}} \le \bar{\boldsymbol{\kappa}}_{\max}$, $\bar{\mathbf{m}}_{\min} \le \bar{\mathbf{m}} \le \bar{\mathbf{m}}_{\max}$, $\boldsymbol{\alpha}_{\min} \le \boldsymbol{\alpha} \le \boldsymbol{\alpha}_{\max}$, $\boldsymbol{\beta}_{\min} \le \boldsymbol{\beta} \le \boldsymbol{\beta}_{\max}$, and $\boldsymbol{\gamma}_{\min} \le \boldsymbol{\gamma} \le \boldsymbol{\gamma}_{\max}$. Unlike the penalty-based approach [5], which requires safety margins to mitigate the risk of violating physical laws (e.g., negative rest lengths and material parameters), we handle box constraints precisely using MPRGP [16]. Consequently, we set lower and upper bounds for $\bar{\mathrm{l}}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ as

$$\bar{\mathrm{l}}_{\min} = \epsilon, \quad \bar{\mathrm{l}}_{\max} = \infty, \quad \boldsymbol{\alpha}_{\min} = \epsilon, \quad \boldsymbol{\alpha}_{\max} = \infty, \tag{12}$$

$$\boldsymbol{\beta}_{\min} = \epsilon, \quad \boldsymbol{\beta}_{\max} = \infty, \quad \boldsymbol{\gamma}_{\min} = \epsilon, \quad \boldsymbol{\gamma}_{\max} = \infty, \tag{13}$$

where ϵ denotes a small positive value (we set $\epsilon = 10^{-10}$). While we set the upper bounds of stiffness parameters to ∞ to ensure that static equilibrium is achievable [3, 29], finite values can be used if one prefers to avoid excessively stiff materials that may compromise static equilibrium.

We set the lower/upper bounds for $\bar{\boldsymbol{\kappa}}$ and $\bar{\mathbf{m}}$ as

$$\bar{\boldsymbol{\kappa}}_{\min} = \bar{\boldsymbol{\kappa}}_0 - \mu \mathbf{e}_{\mathrm{be}}, \quad \bar{\boldsymbol{\kappa}}_{\max} = \bar{\boldsymbol{\kappa}}_0 + \mu \mathbf{e}_{\mathrm{be}}, \tag{14}$$

$$\bar{\mathbf{m}}_{\min} = \bar{\mathbf{m}}_0 - \eta \mathbf{e}_{\mathrm{tw}}, \quad \bar{\mathbf{m}}_{\max} = \bar{\mathbf{m}}_0 + \eta \mathbf{e}_{\mathrm{tw}}, \tag{15}$$

where $\bar{\boldsymbol{\kappa}}_0$ and $\bar{\mathbf{m}}_0$ denote the initial rest curvature and rest twist (before parameter optimization), respectively, μ and η the allowed change for rest curvature and rest twist, respectively, and $\mathbf{e}_{\mathrm{be}}$ and $\mathbf{e}_{\mathrm{tw}}$ vectors of all ones with the same dimensions as $\bar{\boldsymbol{\kappa}}$ and $\bar{\mathbf{m}}$, respectively. The stable range of $\bar{\boldsymbol{\kappa}}$ (i.e., which does not cause stability problems in forward simulations) and thus μ can vary depending on simulation settings (e.g., strand geometry and material stiffness) [5]; we use $\mu = 1$ by default based on our experiments. Additionally, since changes in rest twist are typically much smaller than those in the rest curvature, we set $\eta = \mu/4$ to reduce the number of tunable parameters.

E. Newton-based Box-Constrained Optimization

We solve the primal, box-constrained nonlinear minimization subproblem (8) using a Newton-type optimizer. While one could approximate the box constraints with a penalty objective to ensure fully unconstrained inner linear systems [5], penalty methods often require tedious parameter tuning to achieve acceptable results [15]. Even with carefully tuned parameters, these methods may lead to compromises that violate box constraints. Therefore, instead of relying on penalty methods, we incorporate box constraints directly when computing the Newton directions, which is equivalent to solving box-constrained quadratic programs (BCQPs) [50].

The Newton direction $\Delta\mathbf{p}$ at $k+1$ iteration can be computed by solving the following BCQP:

$$\Delta\mathbf{p}^{k+1} = \underset{\Delta\mathbf{p}^k_{\min}\le\Delta\mathbf{p}\le\Delta\mathbf{p}^k_{\max}}{\arg\min} F(\Delta\mathbf{p}), \tag{16}$$

$$F(\Delta\mathbf{p}) = \frac{1}{2}\Delta\mathbf{p}^T\nabla^2 L(\mathbf{p}^k)\Delta\mathbf{p} + \Delta\mathbf{p}^T\nabla L(\mathbf{p}^k,\boldsymbol{\lambda}^k), \tag{17}$$

where $\Delta\mathbf{p}^k_{\min}$ and $\Delta\mathbf{p}^k_{\max}$ denote lower and upper bounds for $\Delta\mathbf{p}$, respectively. We omit $\boldsymbol{\lambda}^k$ in the argument of $\nabla^2 L(\mathbf{p})$ as it is independent of $\boldsymbol{\lambda}^k$ (19). Assuming the ideal step length of one [15], we have $\mathbf{p}_{\min} \le \mathbf{p}^{k+1} = \mathbf{p}^k + \Delta\mathbf{p}^{k+1} \le \mathbf{p}_{\max}$, giving $\Delta\mathbf{p}^k_{\min} = \mathbf{p}_{\min} - \mathbf{p}^k$ and $\Delta\mathbf{p}^k_{\max} = \mathbf{p}_{\max} - \mathbf{p}^k$ [50].

Given the augmented Lagrangian objective (11), we can compute its gradient as

$$\nabla L(\mathbf{p},\boldsymbol{\lambda}^k) = \nabla R(\mathbf{p}) - \mathbf{J}^T\mathbf{M}^{-\frac{1}{2}}\boldsymbol{\lambda}^k + \rho\mathbf{J}^T\mathbf{M}^{-1}\mathbf{f}(\mathbf{p}), \tag{18}$$

where $\nabla R(\mathbf{p}) = \mathbf{W}(\mathbf{p}-\mathbf{p}_0)$, and we denote the Jacobian of $\mathbf{f}(\mathbf{p})$ with respect to $\mathbf{p}$ by $\mathbf{J}\left(=\frac{\partial\mathbf{f}(\mathbf{p})}{\partial\mathbf{p}}\right) \in \mathbb{R}^{(4N-8)\times(7N-14)}$ (details are given in the supplementary material). Note that we treat $\mathbf{M}$ (which is computed with the initial rest length $\bar{\mathbf{l}}$ [7, 46]) as constant, as $\mathbf{M}$ is introduced and intended to properly scale the generalized forces (see (10)) [5].

Since the generalized forces are linear with respect to the rest curvature, rest twist, and stiffness parameters, the second order derivatives with respect to these parameters are zero. While the second order derivatives with respect to $\bar{\mathbf{l}}$ are non-zero, these components would be projected to zero to ensure SPD inner systems and valid Newton descent directions [15]. In addition, the change in $\bar{\mathbf{l}}$ during the parameter optimization is typically small for less extensible strands, i.e., $\bar{\mathbf{l}}$ can be close to constant. Thus, using the exact Hessian with SPD projections does not have discernible benefits while increasing the implementation complexity and computational cost due to the involvement of third order tensors, as demonstrated and discussed in [5]. Therefore, we ignore the second order derivatives with respect to $\bar{\mathbf{l}}$ and approximate the Hessian in a Gauss-Newton style (excluding $\boldsymbol{\lambda}$ from arguments) by

$$\nabla^2 L(\mathbf{p}) \approx \mathbf{W} + \rho\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}. \tag{19}$$

As this approximated Hessian is guaranteed to be SPD, we can solve the convex BCQP (16) using MPRGP [16], accelerated with our active-set Cholesky preconditioner (see Sec. VI).

F. Algorithm and Implementation

Algorithm 1 outlines our parameter optimization method. To accelerate convergence and reduce the risk of converging to undesirable local minima, we use warm starting. We set $\rho = 10^6$, and use 10^3 and 10^0 for the stiffness and rest shape parts of diagonals in $\mathbf{W}$, respectively, so that we prioritize achieving (in order) zero net forces, small changes in the stiffness parameters, and then in rest shape parameters. We perform a single Newton iteration to solve our primal BCQP subproblem (8) as it serves as an inner problem within (7). To guarantee a decrease in the objective, we implement a backtracking line search [15]. The iterations are terminated when $\|\Delta\mathbf{p}\|_2$ falls below a threshold ϵ_p $(= 10^{-8})$ or iteration count exceeds $k_{\max}$ $(= 100)$. In addition, we can also terminate the solver iterations to prevent infinite loops if we cannot make any progress with the backtracking line search (although our method did not experience this case in our examples).

Algorithm 1 Parameter Optimization

1: Initialize generalized positions $\mathbf{q}$, radius r, stiffness coefficients $\mathbf{c}_{\mathrm{st}}$, $\mathbf{c}_{\mathrm{be}}$, $\mathbf{c}_{\mathrm{tw}}$, length $\mathbf{l}$, rest length $\bar{\mathbf{l}}$, generalized mass matrix $\mathbf{M}$, unit tangent vector $\mathbf{t}$, reference and material frames, curvature $\boldsymbol{\kappa}$, rest curvature $\bar{\boldsymbol{\kappa}}$, twist $\mathbf{m}$, rest twist $\bar{\mathbf{m}}$, stiffness scaling s, and material stiffness parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, penalty parameter ρ, weight matrix $\mathbf{W}$, Lagrange multiplier $\boldsymbol{\lambda}^k = 0$, iteration index $k = 0$
2: Set $\mathbf{p}_0 = \mathbf{p}$ and compute $\mathbf{p}_{\min}$ and $\mathbf{p}_{\max}$
3: **do**
4: Compute box constraints $\Delta\mathbf{p}^k_{\min}$ and $\Delta\mathbf{p}^k_{\max}$
5: Compute $\nabla L(\mathbf{p}^k,\boldsymbol{\lambda}^k)$, $\nabla^2 L(\mathbf{p}^k)$ with (18) and (19)
6: Solve $\nabla^2 L(\mathbf{p}^k)\Delta\mathbf{p}^{k+1} = -\nabla L(\mathbf{p}^k,\boldsymbol{\lambda}^k)$ with $\Delta\mathbf{p}^k_{\min}$ and $\Delta\mathbf{p}^k_{\max}$
7: Update to $\mathbf{p}^{k+1}$ using backtracking line search with (11) and synchronization for $\bar{\boldsymbol{\kappa}}$
8: Update Lagrange multipliers with (9)
9: $k = k + 1$
10: **while** $\epsilon_p < \left\|\Delta\mathbf{p}^{k+1}\right\|_2$ and $k < k_{\max}$

VI. Box-Constrained Quadratic Program Solver

Box constraints are a specific type of inequality constraint that can be handled efficiently through active-set methods, without relying on barriers or penalties [15]. Since our BCQPs arise as primal subproblems (8) within the constrained nonlinear minimization (7), we opt to solve them with a prescribed level of accuracy suitable for inexact Newton-like methods. This approach enables us to conserve computational resources by avoiding unnecessary precision [15]. Thus, instead of employing direct solvers with active sets (e.g., Lemke's method or Dantzig's simplex algorithm), we prefer iterative methods that permit early termination. Given the stiff yet SPD inner systems, we employ MPRGP [16] and propose an effective new preconditioner based on full Cholesky factorization and triangular solves with active sets, utilizing the banded system structures. We refer to this scheme as active-set Cholesky (ASC) preconditioning. Notably, since the Cholesky-based approach precisely solves a linear system,

Cholesky-preconditioned MPRGP (or CG) finds a solution in just one iteration when no box constraints are activated.

A. Active-Set Cholesky Preconditioning

Consider minimizing a quadratic objective $\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$ and corresponding linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. The associated linear preconditioning system can be expressed as $\mathbf{A}\mathbf{z} = \mathbf{r}$, where $\mathbf{z}$ and $\mathbf{r}$ are vectors corresponding to $\mathbf{x}$ and $\mathbf{b}$, respectively. Adopting an active set $\mathbf{a}$ that indicates whether $\mathbf{x}$ is limited or not (i.e., with an index i, $\mathbf{a}_i = 1$ or $\mathbf{a}_i = -1$ if $\mathbf{x}_i$ is limited by its lower or upper bound, respectively, and $\mathbf{a}_i = 0$ otherwise) [44], we can define a diagonal selection matrix $\mathbf{S}$ with $\mathbf{S}_{ii} = 1 - |\mathbf{a}_i|$, which can also be interpreted as a filtering matrix. Given $\mathbf{z}_u$ and $\mathbf{z}_c$ that denote unconstrained and constrained parts of $\mathbf{z}$, respectively, we need to solve the preconditioning system for $\mathbf{z}_u$ while enforcing $\mathbf{z}_c = 0$ [44]. The preconditioning system can be rewritten (incorporating the necessary constraints [51]) as $(\mathbf{S}\mathbf{A}\mathbf{S}^T + \mathbf{I} - \mathbf{S})\mathbf{z} = \mathbf{S}\mathbf{r}$, or given equivalently (by splitting it for $\mathbf{z}_u$ and $\mathbf{z}_c$) as $\mathbf{S}_u\mathbf{A}\mathbf{S}_u^T\mathbf{z}_u = \mathbf{S}_u\mathbf{r}$ and $\mathbf{z}_c = 0$, where $\mathbf{S}_u$ denotes a matrix consisting of $\mathbf{S}$'s rows associated with the unconstrained variables.

1) Problems with Naive Application of Cholesky Solver: While one approach to solving these preconditioning systems is to reassemble $(\mathbf{S}\mathbf{A}\mathbf{S}^T + \mathbf{I} - \mathbf{S})$ or $\mathbf{S}_u\mathbf{A}\mathbf{S}_u^T$ and factorize it using Cholesky decomposition, this procedure is costly because active sets can change in each MPRGP iteration [44], and thus it is ideal to perform Cholesky decomposition only once and reuse its resulting factor.

We consider Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ (where $\mathbf{L}$ is a lower triangular matrix). While its naive substitution to $(\mathbf{S}\mathbf{A}\mathbf{S}^T + \mathbf{I} - \mathbf{S})\mathbf{z} = \mathbf{S}\mathbf{r}$ gives $(\mathbf{S}\mathbf{L}\mathbf{L}^T\mathbf{S}^T + \mathbf{I} - \mathbf{S})\mathbf{z} = \mathbf{S}\mathbf{r}$, triangular solves are inapplicable to this form. Similarly, while $\mathbf{S}_u\mathbf{A}\mathbf{S}_u^T\mathbf{z}_u = \mathbf{S}_u\mathbf{r}$ can be transformed into $\mathbf{S}_u\mathbf{L}\mathbf{L}^T\mathbf{S}_u^T\mathbf{z}_u = \mathbf{S}_u\mathbf{r}$, we cannot perform triangular solves with $\mathbf{S}_u\mathbf{L}\mathbf{y} = \mathbf{S}_u\mathbf{r}$ and $\mathbf{L}^T\mathbf{S}_u^T\mathbf{z}_u = \mathbf{y}$ since $\mathbf{S}_u\mathbf{L}$ is generally not square. Another attempt is to solve $\mathbf{S}\mathbf{A}\mathbf{S}^T\mathbf{z} = \mathbf{S}\mathbf{r}$ (and thus $\mathbf{S}\mathbf{L}\mathbf{L}^T\mathbf{S}^T\mathbf{z} = \mathbf{S}\mathbf{r}$) while enforcing $\mathbf{z}_c = 0$ (e.g., by setting $\mathbf{z}_c$ to zero during the triangular solves). With this form, the forward substitution $\mathbf{S}\mathbf{L}\mathbf{y} = \mathbf{S}\mathbf{r}$ can be written in the elementwise notation as $\mathbf{y}_i = \frac{\mathbf{S}_{ii}\mathbf{r}_i - \mathbf{S}_{ii}\sum_{j<i}\mathbf{L}_{ij}\mathbf{y}_j}{\mathbf{S}_{ii}\mathbf{L}_{ii}}$, and back substitution $\mathbf{L}^T\mathbf{S}^T\mathbf{z} = \mathbf{y}$ as $\mathbf{z}_i = \frac{\mathbf{y}_i - \sum_{j>i}\mathbf{S}_{jj}\mathbf{L}_{ji}\mathbf{z}_j}{\mathbf{S}_{ii}\mathbf{L}_{ii}}$. However, this approach still has two problems. First, $\mathbf{S}_{ii} = 0$ for constrained variables, rendering these operations infeasible. Second, the forward and back substitution are asymmetric, violating the symmetry requirement for preconditioning in symmetric Krylov iterative solvers, such as CG and MPRGP [52].

2) Our Preconditioning: Considering the equivalence between forward/back substitution and forward/backward GS applied to lower/upper triangular matrices, we can interpret triangular solves as SGS. Thus, instead of directly filtering the system matrix $\mathbf{A}$ with $\mathbf{S}$, we perform the filtering in SGS to solve a system equivalent to the filtered preconditioning system. Given the elementwise form of forward GS, $\mathbf{y}_i = (\mathbf{r}_i - \sum_{j<i}\mathbf{L}_{ij}\mathbf{y}_j)/\mathbf{L}_{ii}$, we can filter this operation with $\mathbf{S}$ while ensuring symmetry by

$$\mathbf{y}_i = (\mathbf{S}_{ii}\mathbf{r}_i - \mathbf{S}_{ii}\sum_{j<i}\mathbf{S}_{jj}\mathbf{L}_{ij}\mathbf{y}_j)/\mathbf{L}_{ii}. \tag{20}$$

Similarly, given $\mathbf{z}_i = (\mathbf{y}_i - \sum_{j>i}\mathbf{L}_{ji}\mathbf{z}_j)/\mathbf{L}_{ii}$ for backward GS, we add filtering to obtain

$$\mathbf{z}_i = (\mathbf{S}_{ii}\mathbf{y}_i - \mathbf{S}_{ii}\sum_{j>i}\mathbf{S}_{jj}\mathbf{L}_{ji}\mathbf{z}_j)/\mathbf{L}_{ii}. \tag{21}$$

Note that it is unnecessary to filter the denominator $\mathbf{L}_{ii}$ in (20) and (21) because $\mathbf{y}$ and $\mathbf{z}$ are properly filtered in the numerator. Our filtered SGS preserves the symmetry of operations required for preconditioning of CG/MPRGP [52], and can be rewritten in the block form as $(\tilde{\mathbf{D}} + \mathbf{S}\tilde{\mathbf{L}}\mathbf{S}^T)\mathbf{y} = \mathbf{S}\mathbf{r}$ and $(\tilde{\mathbf{D}} + \mathbf{S}^T\tilde{\mathbf{L}}^T\mathbf{S})\mathbf{z} = \mathbf{S}\mathbf{y}$, respectively, where $\mathbf{L} = \tilde{\mathbf{D}} + \tilde{\mathbf{L}}$, and $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{L}}$ denote the diagonal and strictly lower parts of $\mathbf{L}$, respectively. Note that while (20) and (21) are equivalent to SGS (except for filtering), our $\mathbf{L}$ arises from Cholesky factorization, in contrast to the strictly lower triangular matrix $\hat{\mathbf{L}}$ from the traditional SGS (where $\mathbf{A} = \hat{\mathbf{D}} + \hat{\mathbf{L}} + \hat{\mathbf{L}}^T$, and $\hat{\mathbf{D}}$ is the diagonal part of $\mathbf{A}$). Additionally, while it is typical to initialize $\mathbf{y} = 0$ and $\mathbf{z} = 0$ in GS-type preconditioning [52], our scheme does not require such initialization because (20) and (21) directly overwrite $\mathbf{y}$ and $\mathbf{z}$.

In practice, we prefer sqrt-free Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, where $\mathbf{L}$ is unit lower triangular (for clarity, we redefine $\mathbf{L}$ here), and $\mathbf{D}$ is diagonal. This approach can detect negative pivots (to clamp them for robustness) and avoid sqrt operations for efficiency [47]. The triangular solve proceeds as $\mathbf{L}\mathbf{w} = \mathbf{r}$, $\mathbf{D}\mathbf{y} = \mathbf{w}$, and $\mathbf{L}^T\mathbf{z} = \mathbf{y}$. By merging the forward substitution and diagonal scaling (which preserves symmetry) and skipping unnecessary computations (e.g., multiplications with $\mathbf{S}_{ii}$ and $\mathbf{L}_{ii}(=1)$, and scanning rows in $\mathbf{L}$), we have forward and backward operations for $\mathbf{y}$ and $\mathbf{z}$, respectively, as

$$\mathbf{y}_i = \begin{cases} (\mathbf{r}_i - \sum_{j<i}\mathbf{S}_{jj}\mathbf{L}_{ij}\mathbf{y}_j)/\mathbf{D}_{ii} & \mathbf{S}_{ii} \neq 0, \\ 0 & \text{otherwise}, \end{cases} \tag{22}$$

$$\mathbf{z}_i = \begin{cases} \mathbf{y}_i - \sum_{j>i}\mathbf{S}_{jj}\mathbf{L}_{ji}\mathbf{z}_j & \mathbf{S}_{ii} \neq 0, \\ 0 & \text{otherwise}. \end{cases} \tag{23}$$

B. Active-Set Incomplete Cholesky Preconditioning

Our preconditioning strategy utilizing active sets can also be applied to IC preconditioning. Notably, when the system is banded and fully populated within the band, Cholesky and IC factorization are equivalent, making their preconditioning (even with active sets) identical.

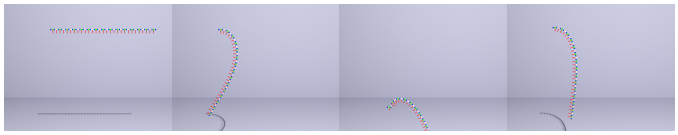
In our parameter optimization, while the system is banded, it is not entirely filled because, e.g., optimization variables for rest curvatures and stretching stiffness are not coupled (i.e., the Hessian (19) lacks off-diagonal elements relating them). Cholesky decomposition is guaranteed to succeed for SPD systems (except for the case where pivot elements become negative due to numerical error [53]), whereas IC factorization may fail unless the system matrix possesses certain properties, such as being an M-matrix [54]. This necessitates reordering the system (i.e., pivoting) potentially breaking the banded structures, adding positive diagonals [54], or reverting to GS [42], ultimately reducing preconditioning effectiveness. Given that approximately 97% of the band is filled, it is acceptable to introduce a small number of fill-ins, considering the IC issues above. We therefore prefer full Cholesky factorization.

(a) Naive initialization

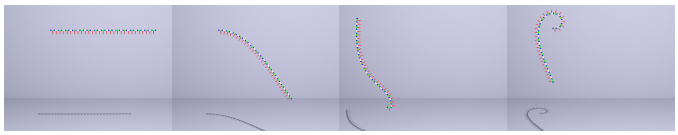
(b) 4D rest curvatures with unbanded/banded system

(c) 2D rest curvatures with banded system

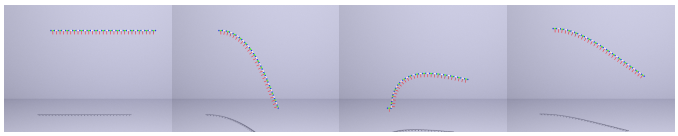
Fig. 2. Coil-like strand test. Rest shape optimization achieves static equilibrium unlike naive initialization (a). 4D rest curvatures with unbanded/banded systems generate identical results (b), while the reduced 2D rest curvatures also produce equivalent results given the redundant force spaces (c).



(a) Naive initialization



(b) Penalty method



(c) Active-set method

Fig. 3. We vertically translate the root position (initially the leftmost endpoint) of a horizontal strand up and down. The leftmost column shows the initialized states for each method and the remaining columns from left to right show the results of forward simulation. The penalty method fails to precisely enforce box constraints, causing significant rest curvature changes and allowing the strand's tail to end up on the left side after the motion stops (b). By contrast, our active-set method (c) strictly enforces the box constraints, keeping the tail on the right side and better preserving the original shape than the naive initialization (a).

VII. Results and Discussions

We implemented our method in C++20 with double-precision floating-point for scalar values and parallelized forward simulation and parameter optimization with OpenMP, processing each strand concurrently. We executed all the examples on a desktop machine with an Intel Core i7-9700 (8 cores) with 16GB RAM. For forward simulation, we use the exact Hessian with SPD projection for stretching objective and Gauss-Newton Hessian approximation for bending and twisting objectives [23]. We perform a single Newton iteration per simulation step [47], executing four simulation steps per frame, except for Figures 1 and 8, which used 12 simulation steps per frame. We use 60 frames per second. For MPRGP, we use a termination absolute or relative residual of 10^{-10}. Unless specified otherwise, we use $\mathbf{c}_{\text{st}} = 10^9, \mathbf{c}_{\text{be}} = 10^9, \mathbf{c}_{\text{tw}} = 10^9$. Material frames are rendered at the centers of the corresponding edges in red and yellow, with lengths matching the rest lengths of the edges.

A. Banded System with Reduced Rest Curvatures

To assess the performance improvement achieved through banded systems and reduced rest curvatures, we experiment with a coil-like strand discretized with 2k vertices, as shown in Figure 2 (for reference, we include naive initialization, which computes rest shape parameters with the generalized positions of the input strand). For a fair comparison with previous work [5], we optimize only the rest shape parameters and exclude box constraints to eliminate the influence of active sets. We compare the following:

1) Unbanded system with 4D rest curvatures [5];
2) Banded system with 4D rest curvatures;
3) Banded system with 2D rest curvatures.

Using the approximate minimum degree (AMD) reordering, the sequential arrangement of the rest shape parameters (rest length, rest curvatures, and rest twist) [5] aligns with the banded system, yielding identical results. The AMD reordering took 2.0 ms while the system solve took 5.0 ms. As the banded system (with 4D rest curvatures) eliminates the need for the reordering, we achieve around 29% performance gain.

Due to the redundant force spaces with rest curvatures, the virtually reduced 2D rest curvatures produce results comparable to those obtained with 4D rest curvatures. The computation times were 2.1 ms and 5.0 ms with 2D and 4D rest curvatures, respectively. As the DOF count is $4N - 8$ and $6N - 12$, with non-zero counts per row of 22.0 and 32.0 on average, the total number of non-zeros is approximately $88N (= 4N \times 22)$ and $192N (= 6N \times 32)$ for 2D and 4D rest curvatures, respectively. Thus, the performance ratio $0.42 = (2.1/5.0)$ is in good agreement with the ratio of the total number of non-zeros $0.46 \approx (88N/192N)$.

B. Penalty vs. Active-Set Methods

To demonstrate the efficacy of our active-set-based BCQP solver, we compare it to a penalty-based one, using a horizontal strand discretized with 30 vertices, as shown in Figure 3 (naive initialization included for reference). In this comparison, we optimize rest shape parameters only and use $\mu = 0.4$ to prevent significant rest curvature changes and thus to keep the tail end of the strand on the right. For the penalty method, we use a Cholesky solver [5]. As we perform only one Newton iteration to solve each BCQP, if we apply ALM (with Lagrange multipliers initialized to zero) to the BCQP, it becomes equivalent to the penalty method [15].

The penalty method failed to strictly enforce the rest curvature changes within their corresponding box constraints. Consequently, significant rest shape changes caused the tail end of the strand to flip to the left side after the root was perturbed. By contrast, our active-set method precisely

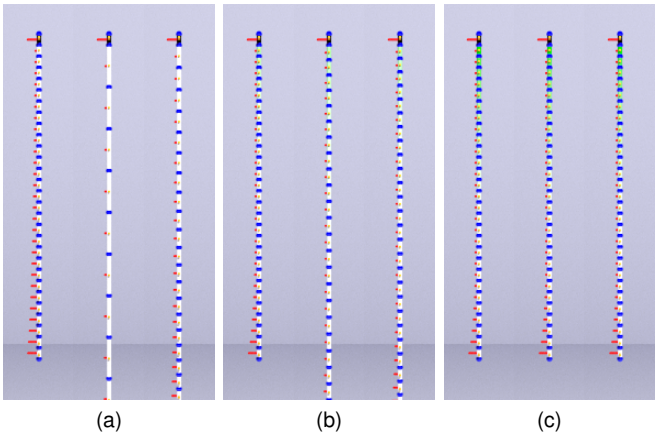


Fig. 4. Evaluation with a vertical strand. The first edge (black) is fixed so its stretching stiffness parameter is undefined. The white/green edges represent lower/higher stiffness parameters. (a) Rest shape only. (b) Rest shape and material stiffness with penalty. (c) Rest shape and material stiffness with ALM (ours). With the rest shape only optimization, material stiffness parameters are unchanged (edges stay white), failing to achieve static equilibrium with rest shape parameters within their box constraints (a). Optimizing the material stiffness parameters additionally stiffens the edges and thus reduces the necessary rest shape changes. Treating the zero net force constraint as a hard constraint enables more significant stiffness changes to achieve static equilibrium (c), compared to using a soft one (b).

constrains curvature changes via their box constraints, keeping the tail end of the strand on the right side. While we may need to compromise static equilibrium, our method better preserves the original shape compared to naive initialization.

The penalty method uses 4 Newton iterations with a total system-solving cost of 0.31 ms (0.078 ms per solve), whereas our method uses 2 Newton iterations with 8.0 MPRGP iterations on average per solve, resulting in a total cost of 0.33 ms (0.17 ms per solve). Although MPRGP needs multiple iterations, our method performs Cholesky factorization (which is costlier than triangular solves) only once per Newton iteration, leading to a moderate cost per MPRGP solve. Furthermore, the strict enforcement of box constraints enhances the convergence of Newton iterations. Consequently, the total cost of our method is comparable to that of the penalty-based approach.

C. Simultaneous Optimization of Rest Shape and Material Stiffness Parameters Under Hard Zero Net Force Constraints

To demonstrate the necessity of optimizing both rest shape and material stiffness parameters and enforcing $\mathbf{c}(\mathbf{p}) = 0$ as a hard constraint, we compare the following schemes:

1) Rest shape only: rest shape only optimization [5];
2) Rest shape and material stiffness with penalty: simultaneous optimization of rest shape and material stiffness parameters with enforcement of $\mathbf{c}(\mathbf{p}) = 0$ via quadratic penalty [15] (which is equivalent to ours with $\boldsymbol{\lambda} = 0$);
3) Rest shape and material stiffness with ALM (ours): rest shape and material stiffness parameter optimization with enforcement of $\mathbf{c}(\mathbf{p}) = 0$ as a hard constraint via ALM.

1) Vertical Strand: We test with a vertical strand discretized with 30 vertices (Figure 4) using $\bar{\mathrm{I}}_{\min} = 10^{-2}$ and $\mathbf{c}_{\mathrm{st}} = 10^3$.

Optimizing only rest shape parameters fails to achieve static equilibrium. Additionally optimizing stiffness parameters reduces the necessary rest shape changes, but enforcing $\mathbf{c}(\mathbf{p}) = 0$ as a soft constraint via a quadratic penalty also fails. This failure occurs because the penalty from material stiffness changes is large and comparable to the penalty from violation of $\mathbf{c}(\mathbf{p}) = 0$, leading to inadequate adjustment of the stiffness parameters. By contrast, our method guarantees perfect static equilibrium, as ALM enforces $\mathbf{c}(\mathbf{p}) = 0$ as a hard constraint. This allows for adequate adjustments to stiffness parameters, ensuring rest shape parameters remain within their box constraints while achieving zero net forces.

The rest shape only optimization takes 8 Newton iterations, 5.6 MPRGP iterations on average, and 8.1 ms for the entire optimization process, while the rest shape and material stiffness optimization with penalty takes 7 Newton iterations, 73.9 MPRGP iterations, and 19.5 ms for the optimization. The increased cost is primarily due to the larger system size ($7N-14$ for rest shape and material stiffness vs. $4N-8$ for rest shape only) with more non-zeros and increased MPRGP iterations (due to the increased complexity of the systems). Our method takes 25 Newton iterations, 66.0 MPRGP iterations, and 58.3 ms for the optimization. The increased cost compared to the approach with penalty is due to the updates of Lagrange multipliers, which modify the optimization problem. Consequently, our method requires around $3\times$ more Newton iterations and computational time. Although our method is approximately $7\times$ more costly than the rest shape only optimization [5], these methods are performed only once per scene as a preprocess (i.e., no additional cost during forward simulation). Thus, we believe that our approach, which guarantees static equilibrium with minimal stiffness changes while eliminating tedious manual stiffness tuning, is a practical and attractive alternative.

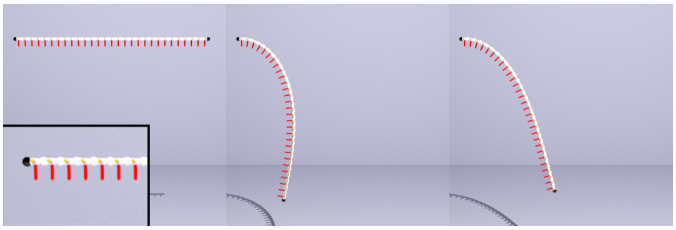
2) Horizontal Strand: We experiment with a horizontal strand discretized with 30 vertices (Figure 5) using $\mu = 0.2$. Similar to the vertical strand case, optimizing only the rest shape parameters fails to achieve static equilibrium (despite the relatively stiff strand with $\mathbf{c}_{\mathrm{be}} = 10^9$) due to tightly bounded rest curvature changes. While simultaneous optimization of the rest shape and material stiffness aims to achieve zero net forces, it still fails when enforcing $\mathbf{c}(\mathbf{p}) = 0$ as a soft constraint using the penalty method. By contrast, our method successfully attains static equilibrium. The computational costs follow a trend similar to that observed with the vertical strand. The optimization takes 7.5 ms for the rest shape only, 17.9 ms for the rest shape and material stiffness with penalty, and 47.7 ms for our method.

D. Box-Constrained Quadratic Program Solver Comparisons

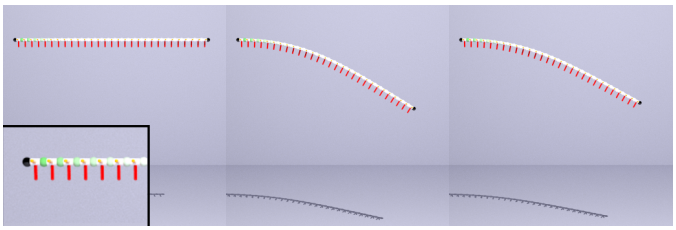
We compare our method with various schemes to solve the BCQP subproblem (16) using the scene shown in Figure 3.

1) MPRGP Preconditioners: We first examine our method against various preconditioning techniques for MPRGP. Specifically, we compare the following schemes:

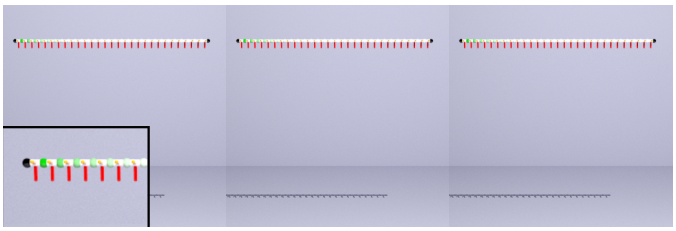
1) MPRGP: MPRGP without preconditioning [16];
2) D-MPRGP: MPRGP with diagonal preconditioning;
3) WJ-MPRGP: MPRGP with two weighted Jacobi iterations with a weighting factor $\omega = 0.5$;
4) SSOR-MPRGP: MPRGP with symmetric successive-over-relaxation (SSOR) preconditioning (serial forward and backward passes) with a weighting factor $\omega = 1.2$;



(a) Rest shape only



(b) Rest shape and material stiffness with penalty



(c) Rest shape and material stiffness with ALM

Fig. 5. Evaluation with a horizontal strand. Material stiffness parameters for bending are undefined for the first and last vertices (black). The white and green vertices represent lower and higher stiffness parameters, respectively. The insets provide enlarged views for clarity. The rest shape only optimization fails to achieve static equilibrium (a). While simultaneously optimizing rest shape and material stiffness parameters better enforces zero forces, treating the zero net force constraint as a soft constraint still fails (b), but as a hard constraint it succeeds in achieving the horizontal static equilibrium (c).

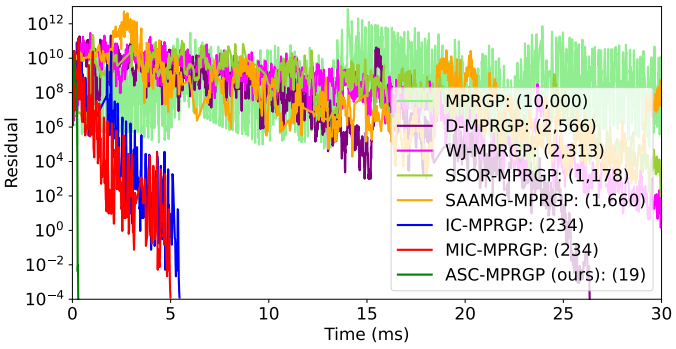


Fig. 6. Log-scale profiles of residuals over time with various preconditioning techniques in MPRGP. The numbers in the last parentheses represent MPRGP iteration counts. Our ASC-MPRGP converges $18\times$ faster than IC-MPRGP.

5) SAAMG-MPRGP: MPRGP with SAAMG preconditioning [44];
6) IC-MPRGP: MPRGP with IC preconditioning [41];
7) MIC-MPRGP: MPRGP with MIC preconditioning [41];
8) ASC-MPRGP (ours): MPRGP with our ASC preconditioning.

Figure 6 shows log-scale profiles of convergence over time. Due to the stiffness of our system, MPRGP without preconditioning failed to converge within 10,000 iterations and was terminated. While WJ-MPRGP and SSOR-MPRGP effectively reduce the number of MPRGP iterations required, their preconditioning costs are non-negligible, and D-MPRGP performs slightly faster. Although SAAMG-MPRGP can be effective for systems with an M-matrix [44], it underperforms on our non-M-matrix system. By contrast, Cholesky-based preconditioners are particularly effective because they exploit the banded structure of the system with minimal overhead for a single factorization per MPRGP solve. IC-MPRGP requires significantly fewer iterations than D-MPRGP, and its moderate preconditioning cost results in much faster overall performance. Given the nearly fully filled band, MIC-MPRGP performs almost identically to IC-MPRGP. Our ASC-MPRGP, which employs full Cholesky factorization, benefits from the complete Cholesky factors without the limitations of (M)IC-MPRGP, which uses incomplete Cholesky with a GS fallback to avoid breakdown [42]. Consequently, while IC-MPRGP converges in 234 iterations taking 5.4 ms, our ASC-MPRGP converges in just 19 iterations taking 0.3 ms, achieving around $18\times$ faster performance.

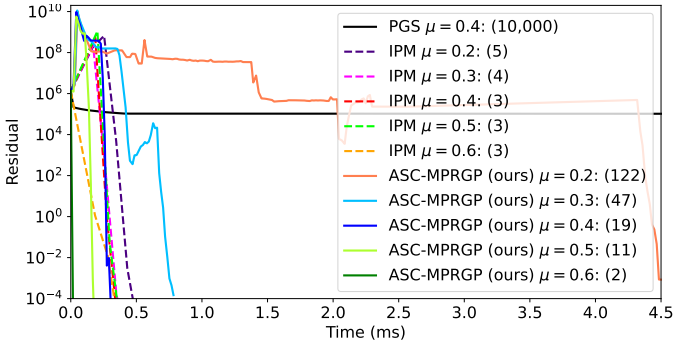


Fig. 7. Log-scale profiles of residuals over time with various BCQP solvers and permitted curvature change μ. The numbers in the last parentheses indicate solver iteration counts. Our ASC-MPRGP outperforms IPM for $\mu \geq 0.4$.

TABLE I
PERFORMANCE NUMBERS FOR IPM AND OUR ASC-MPRGP WITH VARIOUS μ. TIMING IS GIVEN IN MILLISECONDS, AND THE NUMBERS IN THE PARENTHESES REPRESENT THE ITERATION COUNTS.

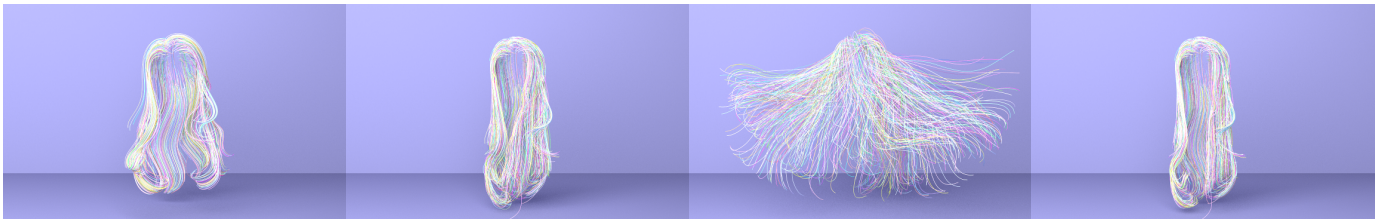
Scheme \ μ	0.2	0.3	0.4	0.5	0.6
IPM	0.517 (5)	0.434 (4)	0.361 (3)	0.388 (3)	0.375 (3)
ASC-MPRGP	4.518 (122)	0.783 (47)	0.305 (19)	0.184 (11)	0.039 (2)

2) BCQP Solvers: Next, we compare our method with other BCQP solvers. We consider the following schemes:

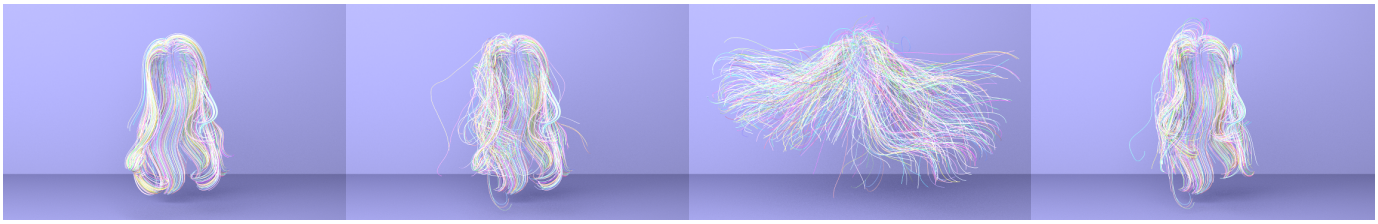
1) PGS: projected GS;
2) IPM: interior point method [55] with a Cholesky-based direct solver for fully unconstrained inner linear systems;
3) ASC-MPRGP (ours).

In this comparison, we exclude the penalty-based approach as it cannot accurately enforce the box constraints (comparisons between the penalty-based and active-set approaches are given in Sec. VII-B). Figure 7 shows log-scale profiles of convergence over time, and Table I summarizes the performance numbers (excluding PGS).

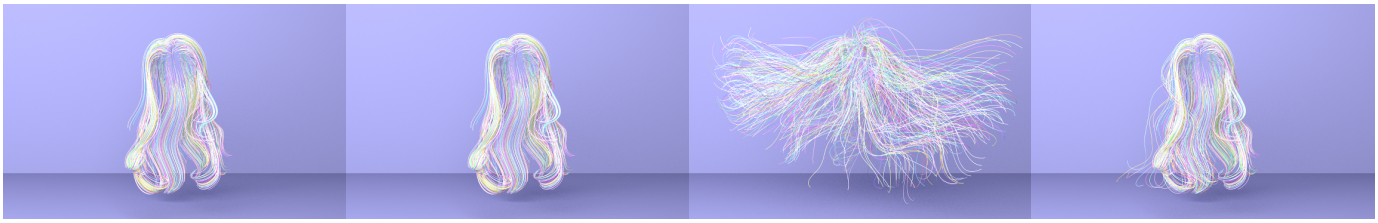
Although PGS should be able to strictly enforce the box constraints, it was slow to converge and terminated after 10,000 iterations. While IPM [55] can converge with a small



(a) Naive initialization



(b) Rest shape only optimization



(c) Rest shape and material stiffness optimization (ours)

Fig. 8. Hair simulation with complex strand geometry. Hair strands significantly sag with naive initialization. Rest shape only optimization suffers from some sagging and unnatural hair lifting. Our method successfully achieves static equilibrium, exhibits natural motions driven by the prescribed movements of the root vertices, and then restores the hairstyle to a form closely resembling the original.

number of iterations for all values of μ, each iteration is costly because IPM updates the system diagonals, necessitating a full Cholesky factorization for each updated system. By contrast, our ASC-MPRGP requires no system updates and enables the reuse of the Cholesky factor by combining it with active sets which may change in each MPRGP iteration. Thus, per-iteration cost is relatively small, enabling ASC-MPRGP to outperform IPM [55] with approximately $\mu \geq 0.4$ (our default is $\mu = 1$). While our method can be slower than IPM when μ is small (because MPRGP needs more iterations to update active sets and does not fully leverage our preconditioner), we believe that such cases are infrequent in practical applications.

E. Complex Strand Geometry

To evaluate the efficacy of our method with complex strand geometry, we experiment with hair data publicly released by Hu et al. [56]. Figure 8 compares our method with naive initialization and rest shape only optimization, using an asset with 915 strands (each is discretized with 100 vertices). In this comparison, we use an extra soft material with $\mathbf{c}_{\mathrm{st}} = \mathbf{c}_{\mathrm{be}} = \mathbf{c}_{\mathrm{tw}} = 10^8$ to clearly demonstrate differences, and $\mu = 0.5$.

With naive initialization, hair strands sag significantly due to gravity at the start of the simulation, eventually settling into sagged states after some root vertex movement. With rest shape only optimization, some strands still suffer from sagging since the allowed rest shape changes are limited by the box constraints. Conversely, other strands experience unnatural lifting because this approach attempts to achieve static equilibrium solely through adjustments to the rest shape parameters, yielding significant rest shape changes (even with the box constraints) and thus continuously unstable strand configurations that do not settle. Hence the rest shape only optimization fails both to achieve static equilibrium (with insufficient rest shape changes) and to ensure stable configurations (due to the excessive rest shape changes), indicating that the range parameter for box constraints μ is simultaneously too low for some strands and too high for others; therefore, merely adjusting μ cannot always fix these failures. By contrast, our method enables strands to achieve the perfect static equilibrium even with complex hair geometry through the smaller rest shape changes facilitated by simultaneous optimization of material stiffness. It achieves plausible motions and allows the strands to gradually settle and return towards the original hair styles, while naive initialization and rest shape only optimization do not.

Rest shape only optimization used 2.0 Newton iterations (with frequent failures in backtracking line search, leading to early termination) and 2.5 MPRGP iterations per solve on average (occasionally terminating at 100 iterations due to convergence issues), taking 0.6 s for the entire optimization process, whereas our method used 148.8 Newton iterations and 1.2 MPRGP iterations, taking 84.4 s. Although our method incurs higher costs than the rest shape only optimization (which fails to converge, thus suffering from sagging, unnatural lifting, and stability problems), we believe that the achieved static equilibrium justifies this one time initialization cost for forward simulations (which take 2.0 s per frame).

VIII. Conclusions and Future Work

We have proposed our parameter optimization framework that ensures static equilibrium of DER-based strands. Additionally, we presented an active-set Cholesky preconditioner to accelerate the convergence of MPRGP. We demonstrated the efficacy of our method in a wide range of examples. Below, we discuss tradeoffs inherent to our method and promising research directions for future work.

A. Undesirable Local Minima

While our method ensures that strands achieve static equilibrium, certain perturbations may cause them to fall into other local minima in forward simulations, which may not align with user expectations. Although our solver enforces rest shape changes within specified box constraints to mitigate the risk of encountering such local minima, determining optimal values for these constraints can be challenging. The desirability of the resulting behaviors is often subjective, and the optimal values may vary depending on strand geometry and material stiffness. To accommodate diverse scenarios, it could be advantageous to assign different values of μ for each strand.

In contrast to optimization methods focused solely on the rest shape parameters [5], our approach allows for modifications to the material stiffness, which can be perceived as undesirable. While $\mathbf{W}$ in (5) assists in balancing changes between the rest shape and material stiffness parameters, a potentially more effective strategy may involve conducting separate and iterative parameter optimizations for the rest shape and material stiffness, similar to block coordinate descent. Furthermore, to ensure smoothly varying material stiffness, it may be beneficial to employ a Laplacian-based regularizer or, alternatively, to optimize a limited set of representative stiffness parameters, which has the added advantage of reducing memory usage and optimization costs.

B. Active-Set Cholesky Preconditioner

As our ASC preconditioner is based on a Cholesky-based direct solver which is particularly effective for stiff SPD linear systems, it is worth evaluating our preconditioner with stiff BCQPs. In particular, our preconditioner should work with tree structures which form block-structured systems (instead of banded ones) without introducing any fill-in at off-diagonal blocks between the block matrices [53]. It also seems promising to extend our preconditioner to (incomplete) LDLT factorization [43, 57] and to explore symmetric indefinite solvers that can handle box constraints. With small μ, many box constraints can be activated, necessitating additional MPRGP iterations to manage active sets while failing to fully leverage our ASC preconditioning. In such cases, exploring active-set-free approaches, such as interior point methods [15, 55], may be promising. Although parallel execution over strands rendered extensive parallelization unnecessary in our framework, it would be valuable to optimize our preconditioner for parallel execution to fully utilize many-core architectures, particularly for strands discretized with a large number of vertices.

C. More General Inverse Problems

Supporting anisotropic and inhomogeneous strands, two-end clamping, and frictional contacts would be useful. In particular, incorporating contact forces in the parameter optimization would prevent extra stiffening and rest shape changes, giving more plausible results (e.g., for hairs on a head) [3, 4, 30]. To predict strand dynamics over time, developing differentiable physics approaches, such as those employing the adjoint method [2, 12], appears promising [58]. Extending our method to applications involving 2D/3D structured materials would also be of interest. While our method is guaranteed to reach static equilibrium at solver convergence, convergence conditions may depend on strand geometry and properties; thus, tuning solver parameters may be necessary in challenging scenarios. Additionally, solving the optimization problem without primal-dual decoupling could enhance the likelihood of convergence.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions, and Bo Yang, Qingyuan Zheng, Rundong Wu, and Yu Ju Chen for early discussions. We acknowledge the authors of [56] for publicly releasing the hair dataset. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (Grant RGPIN-2021-02524).

References

[1] C. D. Twigg and Z. Kačić-Alesić, "Optimization for sag-free simulations," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11, 2011, pp. 225–236.

[2] J. Pérez, B. Thomaszewski, S. Coros, B. Bickel, J. A. Canabal, R. Sumner, and M. A. Otaduy, "Design and fabrication of flexible rod meshes," *ACM Trans. Graph.*, vol. 34, no. 4, jul 2015.

[3] J. Hsu, N. Truong, C. Yuksel, and K. Wu, "A general two-stage initialization for sag-free deformable simulations," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022.

[4] J. Hsu, T. Wang, Z. Pan, X. Gao, C. Yuksel, and K. Wu, "Sag-free initialization for strand-based hybrid hair simulation," *ACM Trans. Graph.*, vol. 42, no. 4, jul 2023.

[5] T. Takahashi and C. Batty, "Rest shape optimization for sag-free discrete elastic rods," *Computer Graphics Forum*, vol. 44, no. 2, p. e70019, 2025.

[6] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM SIGGRAPH 2008 Papers*, ser. SIGGRAPH '08, 2008.

[7] M. Bergou, B. Audoly, E. Vouga, M. Wardetzky, and E. Grinspun, "Discrete viscous threads," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 116:1–116:10, 2010.

[8] E. Schweickart, D. L. James, and S. Marschner, "Animating elastic rods with sound," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017.

[9] Y. R. Fei, C. Batty, E. Grinspun, and C. Zheng, "A multi-scale model for coupling strands with shear-dependent liquid," *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019.

[10] S. Lesser, A. Stomakhin, G. Daviet, J. Wretborn, J. Edholm, N.-H. Lee, E. Schweickart, X. Zhai, S. Flynn, and A. Moffat, "Loki: A unified multiphysics simulation framework for production," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022.

[11] G. Daviet, "Interactive hair simulation on the gpu using admm," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23. New York, NY, USA: Association for Computing Machinery, 2023.

[12] J. Panetta, M. Konaković-Luković, F. Isvoranu, E. Bouleau, and M. Pauly, "X-shells: a new class of deployable beam structures," *ACM Trans. Graph.*, vol. 38, no. 4, jul 2019.

[13] Y. Ren, U. Kusupati, J. Panetta, F. Isvoranu, D. Pellis, T. Chen, and M. Pauly, "Umbrella meshes: elastic mechanisms for freeform shape deployment," *ACM Trans. Graph.*, vol. 41, no. 4, jul 2022.

[14] L.-J. Y. Dandy, M. Vidulis, Y. Ren, and M. Pauly, "Tencers: Tension-constrained elastic rods," p. 1–13, dec 2024.

[15] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[16] Z. Dostal and J. Schoberl, "Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination," *Computational Optimization and Applications*, vol. 30, no. 1, pp. 23–43, 2005.

[17] D. K. Pai, "STRANDS: Interactive Simulation of Thin Solids using Cosserat Models," *Computer Graphics Forum*, 2002.

[18] J. Spillmann and M. Teschner, "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*, ser. SCA '07. Eurographics Association, 2007, p. 63–72.

[19] N. Umetani, R. Schmidt, and J. Stam, "Position-based elastic rods," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '14, 2014, p. 21–30.

[20] T. Kugelstadt and E. Schömer, "Position and orientation based cosserat rods," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '16, 2016, p. 169–178.

[21] B. Angles, D. Rebain, M. Macklin, B. Wyvill, L. Barthe, J. Lewis, J. Von Der Pahlen, S. Izadi, J. Valentin, S. Bouaziz, and A. Tagliasacchi, "Viper: Volume invariant position-based elastic rods," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 2, no. 2, jul 2019.

[22] C. Soler, T. Martin, and O. Sorkine-Hornung, "Cosserat rods with projective dynamics," *Computer Graphics Forum*, vol. 37, no. 8, pp. 137–147, 2018.

[23] A. Shi, H. Wu, J. Parr, A. M. Darke, and T. Kim, "Lifted curls: A model for tightly coiled hair simulation," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 6, no. 3, aug 2023.

[24] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. Lévêque, "Super-helices for predicting the dynamics of natural hair," *ACM Trans. Graph.*, vol. 25, no. 3, p. 1180–1187, jul 2006.

[25] R. Casati and F. Bertails-Descoubes, "Super space clothoids," *ACM Trans. Graph.*, vol. 32, no. 4, jul 2013.

[26] S. Hadap, "Oriented strands: dynamics of stiff multi-body system," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '06, 2006, p. 91–100.

[27] X. Chen, C. Zheng, W. Xu, and K. Zhou, "An asymptotic numerical method for inverse elastic shape design," *ACM Trans. Graph.*, vol. 33, no. 4, jul 2014.

[28] K. Jia, "Sanm: a symbolic asymptotic numerical solver with applications in mesh deformation," *ACM Trans. Graph.*, vol. 40, no. 4, jul 2021.

[29] A. Derouet-Jourdan, F. Bertails-Descoubes, and J. Thollot, "Stable inverse dynamic curves," *ACM Trans. Graph.*, vol. 29, no. 6, dec 2010.

[30] A. Derouet-Jourdan, F. Bertails-Descoubes, G. Daviet, and J. Thollot, "Inverse dynamic hair modeling with frictional contact," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 159:1–159:10, Nov. 2013.

[31] F. Bertails-Descoubes, A. Derouet-Jourdan, V. Romero, and A. Lazarus, "Inverse design of an isotropic suspended Kirchhoff rod: theoretical and numerical results on the uniqueness of the natural shape," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2212, pp. 1–26, Apr. 2018.

[32] R. Charrondière, F. Bertails-Descoubes, S. Neukirch, and V. Romero, "Numerical modeling of inextensible elastic ribbons with curvature-based elements," *Computer Methods in Applied Mechanics and Engineering*, vol. 364, p. 112922, 2020.

[33] C. Hafner and B. Bickel, "The design space of plane elastic curves," *ACM Trans. Graph.*, vol. 40, no. 4, jul 2021.

[34] ——, "The design space of kirchhoff rods," *ACM Trans. Graph.*, vol. 42, no. 5, sep 2023.

[35] J.-M. Aubry and X. Xian, "Fast implicit simulation of flexible trees," in *Mathematical Progress in Expressive Image Synthesis II*, H. Ochiai and K. Anjyo, Eds. Springer Japan, 2015, pp. 47–61.

[36] M. Skouras, B. Thomaszewski, B. Bickel, and M. Gross, "Computational design of rubber balloons," *Comput. Graph. Forum*, vol. 31, no. 2pt4, p. 835–844, may 2012.

[37] B. Wang, L. Wu, K. Yin, U. Ascher, L. Liu, and H. Huang, "Deformation capture and modeling of soft objects," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 94:1–94:12, Jul. 2015.

[38] R. Mukherjee, L. Wu, and H. Wang, "Interactive two-way shape design of elastic bodies," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 1, no. 1, jul 2018.

[39] M. Ly, R. Casati, F. Bertails-Descoubes, M. Skouras, and L. Boissieux, "Inverse elastic shell design with contact and friction," *ACM Trans. Graph.*, vol. 37, no. 6, dec 2018.

[40] A. Choi, R. Jing, A. P. Sabelhaus, and M. K. Jawed, "Dismech: A discrete differential geometry-based physical simulator for soft robots and structures," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3483–3490, 2024.

[41] R. Narain, A. Golas, and M. C. Lin, "Free-flowing granular materials with two-way solid coupling," *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 173:1–173:10, 2010.

[42] R. Bridson and M. Müller, "Fluid simulation: Siggraph 2007 course," in *ACM SIGGRAPH 2007 Courses*, ser. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1–81.

[43] C. Greif, S. He, and P. Liu, "Sym-ildl: Incomplete ldlt factorization of symmetric indefinite and skew-symmetric matrices," *ACM Trans. Math. Softw.*, vol. 44, no. 1, Apr. 2017.

[44] T. Takahashi and C. Batty, "A multilevel active-set preconditioner for box-constrained pressure poisson solvers," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 6, no. 3, aug 2023.

[45] O. Bröker, M. J. Grote, C. Mayer, and A. Reusken, "Robust parallel smoothing for multigrid via sparse approximate inverses," *SIAM Journal on Scientific Computing*, vol. 23, no. 4, pp. 1396–1417, 2001.

[46] M. Jawed, A. Novelia, and O. O'Reilly, *A Primer on the Kinematics of Discrete Elastic Rods*, ser. SpringerBriefs in Applied Sciences and Technology. Springer International Publishing, 2018.

[47] L. Huang, F. Yang, C. Wei, Y. J. E. Chen, C. Yuan, and M. Gao, "Towards realtime: A hybrid physics-based method for hair animation on gpu," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 6, no. 3, 2023.

[48] G. Gornowicz and S. Borac, "Efficient and stable approach to elasticity and collisions for hair animation," in *Proceedings of the 2015 Symposium on Digital Production*, ser. DigiPro '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 41–49.

[49] J. M. Kaldor, D. L. James, and S. Marschner, "Efficient yarn-based cloth with adaptive contact linearization," *ACM Trans. Graph.*, vol. 29, no. 4, jul 2010.

[50] T. Takahashi and C. Batty, "FrictionalMonolith: A monolithic optimization-based approach for granular flow with contact-aware rigid-body coupling," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–16, 2021.

[51] R. Tamstorf, T. Jones, and S. F. McCormick, "Smoothed aggregation multigrid for cloth simulation," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 245:1–245:13, 2015.

[52] Y. Saad, "Iterative Methods for Sparse Linear Systems," *Notes*, vol. 3, no. 2nd Edition, pp. xviii+528, 2003.

[53] P. Herholz and M. Alexa, "Factor once: Reusing cholesky factorizations on sub-meshes," *ACM Trans. Graph.*, vol. 37, no. 6, dec 2018.

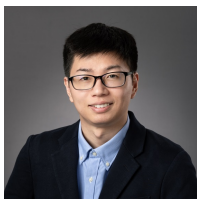
[54] J. Chen, F. Schäfer, J. Huang, and M. Desbrun, "Multiscale cholesky preconditioning for ill-conditioned problems," *ACM Trans. Graph.*, vol. 40, no. 4, jul 2021.

[55] T. Takahashi and C. Batty, "A primal-dual box-constrained qp pressure poisson solver with topology-aware geometry-inspired aggregation amg," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–12, 2024.

[56] L. Hu, C. Ma, L. Luo, and H. Li, "Single-view hair modeling using a hairstyle database," *ACM Trans. Graph.*, vol. 34, no. 4, jul 2015.

[57] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, "A survey of direct methods for sparse linear systems," *Acta Numerica*, vol. 25, p. 383–566, 2016.

[58] Y. Chen, Y. Zhang, Z. Brei, T. Zhang, Y. Chen, J. Wu, and R. Vasudevan, "Differentiable discrete elastic rods for real-time modeling of deformable linear objects," 2024.



Tetsuya Takahashi is a Senior Researcher at Tencent America. He was a Software Engineer at Adobe. He earned his M.S. and Ph.D. from the University of North Carolina at Chapel Hill in 2017 and 2020, respectively, and B.S. and M.S. from Keio University in 2012 and 2014, respectively. His research interests include physically-based simulation, numerical optimization, and geometry processing.

Christopher Batty is an Associate Professor in the David R. Cheriton School of Computer Science at the University of Waterloo in Ontario, Canada. He received his PhD from the University of British Columbia in 2010 and was a Banting Postdoctoral Fellow at Columbia University from 2011 to 2013. His research is primarily focused on the development of novel physical simulation techniques for applications in computer graphics and computational physics, with an emphasis on the diverse behaviors of fluids.