Charged Elastic Rod in 3D: Discrete Energy, Gradient, and BFGS Optimization

Numerical Methods Programming Assignment (Starter Note)

1 Overview

We model a closed filament in $\mathbb{R}^3$ (think: a charged DNA loop in solution) as N nodes

$$X = (\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}), \qquad \mathbf{x}_i \in \mathbb{R}^3,$$

with *periodic indexing* $\mathbf{x}_{i+N} = \mathbf{x}_i$. The goal is to minimize a total energy

$$E(X) = E_{\text{bend}}(X) + E_{\text{stretch}}(X) + E_{\text{coul}}(X),$$

using a quasi-Newton method (BFGS) in $\mathbb{R}^{3N}$.

The optimization is *nonconvex* and exhibits multiple local minima. Good numerical behavior requires: (i) correct analytic gradients, (ii) a line search, and (iii) robust updates.

2 Discrete energy

2.1 Bending (curvature penalty)

A standard discrete curvature surrogate is the second difference:

$$\mathbf{b}_i = \mathbf{x}_{i+1} - 2\mathbf{x}_i + \mathbf{x}_{i-1}.$$

We define

$$E_{\text{bend}} = k_b \sum_{i=0}^{N-1} \|\mathbf{b}_i\|^2.$$

Large k_b favors smoother curves.

2.2 Stretching (near-inextensibility)

Let segment vectors be $\mathbf{s}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$ with length $r_i = \|\mathbf{s}_i\|$. We penalize deviation from rest length ℓ_0:

$$E_{\text{stretch}} = k_s \sum_{i=0}^{N-1} (r_i - \ell_0)^2.$$

Typically $k_s \gg k_b$ to keep the filament close to inextensible.

2.3 Screened Coulomb repulsion

Charges on the filament repel. In ionic solution this is often modeled by Debye–Hückel screening:

$$U(r) = \frac{q^2 e^{-\kappa r}}{r},$$

with screening parameter $\kappa \geq 0$ (set $\kappa = 0$ for unscreened Coulomb). The total repulsion is

$$E_{\text{coul}} = \sum_{\substack{0 \leq i < j \leq N-1 \\ (i,j) \text{ not neighbors}}} U(\|\mathbf{x}_i - \mathbf{x}_j\|).$$

We exclude adjacent neighbors (including the wrap neighbor $(0, N-1)$) to avoid double-counting interactions already represented by stretching/bending.

3 Gradients (high level)

Your optimizer will require $\nabla E(X) \in \mathbb{R}^{3N}$. The starter repository provides a C++ routine that returns both E and ∇E.

Two typical patterns:

- For bending, each $\mathbf{x}_i$ appears in $\mathbf{b}_{i-1}, \mathbf{b}_i, \mathbf{b}_{i+1}$, yielding a local stencil.
- For stretching and Coulomb terms, derivatives follow from the chain rule:

$$\frac{\partial}{\partial \mathbf{x}} \|\mathbf{v}\| = \frac{\mathbf{v}}{\|\mathbf{v}\|},$$

  and for Coulomb:
$$\frac{d}{dr}\left(\frac{q^2 e^{-\kappa r}}{r}\right) = q^2 e^{-\kappa r}\left(-\frac{1}{r^2} - \frac{\kappa}{r}\right).$$

4 Self-avoidance via segment–segment WCA interaction

Point–point repulsion is insufficient to prevent self-intersection of a discretized filament. Instead, we model *self-avoidance* via short-range repulsion between *segments* of the polyline.

Let the i-th segment be

$$\mathbf{p}_i(u) = \mathbf{x}_i + u(\mathbf{x}_{i+1} - \mathbf{x}_i), \qquad u \in [0,1],$$

and similarly

$$\mathbf{p}_j(v) = \mathbf{x}_j + v(\mathbf{x}_{j+1} - \mathbf{x}_j), \qquad v \in [0,1].$$

Define the squared distance function

$$\phi(u,v) = \|\mathbf{p}_i(u) - \mathbf{p}_j(v)\|^2.$$

The segment–segment distance is

$$d_{ij} = \min_{u,v \in [0,1]} \|\mathbf{p}_i(u) - \mathbf{p}_j(v)\| = \sqrt{\phi(u^*, v^*)},$$

where (u^*, v^*) satisfies the first-order conditions

$$\partial_u \phi(u,v) = 0, \qquad \partial_v \phi(u,v) = 0,$$

together with the box constraints $u, v \in [0,1]$.

4.1 Weeks–Chandler–Andersen (WCA) potential

We use the repulsive part of the Lennard–Jones potential:

$$U(d) = \begin{cases} 4\varepsilon\Big[(\sigma/d)^{12} - (\sigma/d)^6\Big] + \varepsilon, & d < 2^{1/6}\sigma, \\ 0, & d \geq 2^{1/6}\sigma. \end{cases}$$

Define the self-avoidance energy

$$E_{\text{sa}} = \sum_{(i,j)\in\mathcal{P}} U(d_{ij}),$$

where $\mathcal{P}$ ranges over *non-adjacent* segment pairs (exclude $(i, i+1)$ with $(i \pm 1, i \pm 2)$ and the wrap neighbors).

4.2 Gradient requirement

You must compute ∇E_{sa} with respect to all node positions $\mathbf{x}_k$. This requires differentiating through the closest-point problem: the minimizers (u^*, v^*) depend implicitly on the segment endpoints $(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_j, \mathbf{x}_{j+1})$.

5 Confinement potential (to induce supercoiling-like states)

To obtain nontrivial equilibria (coiling/packing) with purely repulsive self-avoidance, we add a weak confining potential

$$E_{\text{conf}} = k_c \sum_{i=0}^{N-1} \|\mathbf{x}_i\|^2,$$

with $k_c > 0$ small. This encourages the filament to pack into a bounded region while self-avoidance prevents self-intersection.

6 BFGS task

You will implement BFGS in Python to minimize $f(x)$ where $x \in \mathbb{R}^{3N}$ is the flattened coordinate vector. At iteration k:

1. compute direction $p_k = -H_k \nabla f(x_k)$ where H_k approximates the inverse Hessian;
2. choose step length α_k using a line search (Armijo or Wolfe conditions);
3. update $x_{k+1} = x_k + \alpha_k p_k$;
4. update H_k using the BFGS inverse-Hessian update with curvature check $y_k^T s_k > 0$.

7 Suggested experiments

- Compare BFGS vs. gradient descent on the same initialization.
- Vary (k_b, k_s, κ) to see the qualitative change in equilibria.
- Observe sensitivity to the initial curve (multiple local minima).