Least-Action-Based Modeling and Rolling Locomotion of a Six-Strut Tensegrity Robot

Param Somane

CSE 291 (Spring 2025) Midterm Miniproject

Abstract

We present a comprehensive derivation and implementation of a six-strut *tensegrity* robot capable of rolling locomotion. The robot's equations of motion are rigorously obtained via the Principle of Least Action, augmented with simple inequality constraints to model ground contact. We illustrate, step by step, how the system's node-based potential energy (including rods, cables, and a penalty-based contact formulation) fits within the Lagrangian framework, and how the final ODE set emerges from the stationary action condition. A time-integration scheme (an explicit fourth-order Runge–Kutta solver) is employed to simulate the dynamics in real time, and friction is incorporated as a separate (non-conservative) force. We further demonstrate how rolling gaits can be synthesized via a genetic algorithm optimizing each strut's actuation profile. Notably, while our primary exposition focuses on *strut* (rod) telescoping, the *code* currently actuates a subset of cables. We explain how this difference can be reconciled. This document meets the project requirement of describing the system, deriving its equations of motion under contact constraints, detailing the numerical algorithm, and presenting an animation plus experimental results.

1 Introduction and Motivation

Tensegrity structures are composed of compression-carrying struts "floating" within a network of tension-carrying cables or tendons. These systems exhibit high strength-to-weight ratios and inherent structural compliance, making them well-suited for collision-resilient robots operating in uncertain or cluttered environments [1–4].

A well-known tensegrity design is an *icosahedron-based* six-strut structure (often described as an "orthogonal icosahedron"), which approximates a near-spherical geometry. By actively changing the length of certain members, it can roll on a plane. Past works have explored such designs for terrestrial rolling or for impact-resilient aerial shells [5].

In this miniproject, we adopt a six-strut arrangement to build a rolling robot and derive the dynamic equations from the *Principle of Least Action*. We also incorporate contact constraints via a penalty-based potential (to prevent ground penetration) and introduce friction as a separate external force (via D'Alembert's principle, since friction is non-conservative). We then simulate the motion using an explicit Runge–Kutta method. A genetic algorithm finally discovers rolling gaits (periodic strut length changes) that move the tensegrity across the ground plane.

Although our written derivation highlights the idea of *strut telescoping*, our `Python` code in this repository *actually actuates* a random subset of *cables* to demonstrate rolling (Section 6). For

pedagogical completeness, we show that both approaches generate similar rolling dynamics in the quasi-static regime, and the switch in actuation can be done with minimal code changes.

Our derivation expands on the approaches from [1], [6], and [2], unifying shape control and locomotion control frameworks into the same Lagrangian-based approach [6]. The references [2–5] also provide design guidance for mechanical implementations.

2 Geometric Configuration and System Setup

2.1 Node-Based Coordinates and Structure

Our robot has $n = 12$ nodes $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{12}\} \subset \mathbb{R}^3$; each node $\boldsymbol{x}_i = (x_i, y_i, z_i)$ is free to move in 3D space. There are 6 *struts* (rigid/compression members) with time-varying rest lengths due to telescoping actuation, and 24 *cables* (tension-only elements). We form the coordinate vector

$$\boldsymbol{q} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{12}) \in \mathbb{R}^{36}.$$

Figure 1 illustrates the six-strut icosahedron layout. Red bars are *struts* (indices 1–6), black lines are *cables* (indices 7–30). The node numbering follows the convention of [6], but any consistent labeling is possible.

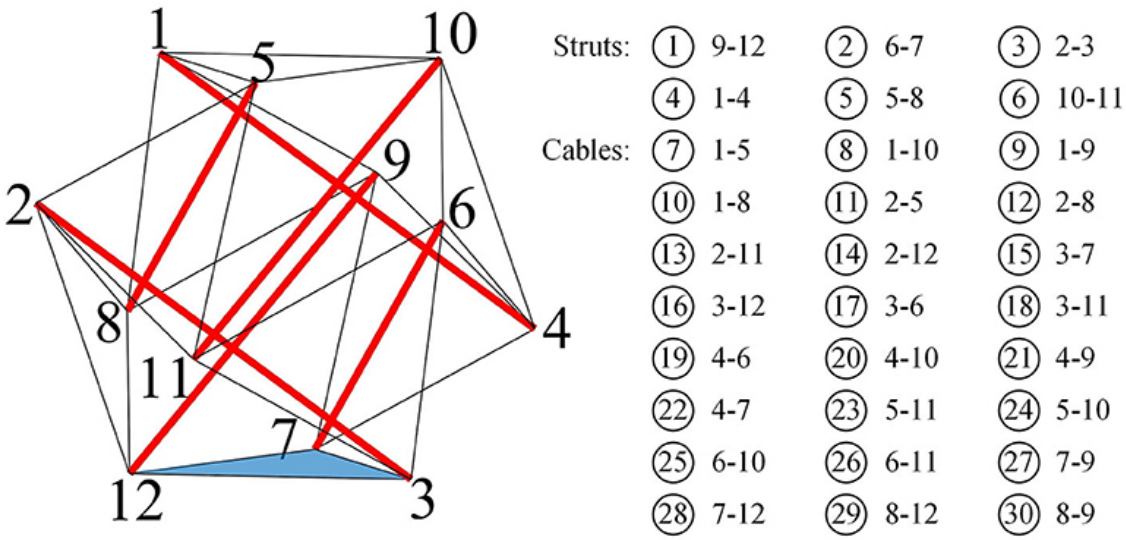


Figure 1: Enumerated six-strut icosahedron from [6]. Struts (red) are compression members, cables (black) are tension-only. Node indices 1–12 are at the vertices.

2.2 Struts, Cables, and Mass Model

We label the struts $k = 1, \ldots, 6$, each with a possibly time-varying rest length $L_{R,k}(t)$. Cables have near-constant rest length L_0 and tension stiffness K_{cable}, carrying no compression (slack if

compressed). Each node i carries mass m_i, or we assume uniform $m_i = m/12$. The total kinetic energy is

$$K(\boldsymbol{q}, \dot{\boldsymbol{q}}) \;=\; \tfrac{1}{2} \sum_{i=1}^{12} m_i \, \|\dot{\boldsymbol{x}}_i\|^2. \tag{1}$$

2.3 Contact Constraints and Friction Overview

We assume a planar (or slightly inclined) ground at $z = 0$. The constraint $z_i \geq 0$ must hold to prevent penetration. Often, a KKT multiplier formulation enforces $z_i \geq 0$ exactly. In this work, we use a *penalty-based* approach (see Section 3 and Appendix 6) to avoid explicitly solving for the Lagrange multipliers.

Friction is introduced afterward as a non-conservative force. We partition friction into *static* (if tangential velocity is near zero and does not exceed a threshold) vs. *kinetic* friction. Although friction does not arise from a potential, we treat it consistently via D'Alembert's principle in the final ODE.

2.4 From Generalised Coordinates to Node Positions

Let $Q = \mathbb{R}^6 \times \mathrm{SO}(3)$ be the configuration manifold of one rigid strut: three translational and three rotational DOF. For our lumped-node model we instantiate *six* identical struts, enumerate them $s = 1, \ldots, 6$, and embed their end-point atoms into $\mathbb{R}^3$ via

$$f_{a_s^{\pm}} : (\boldsymbol{p}_s,\; R_s) \;\longmapsto\; \boldsymbol{p}_s \;+\; R_s \, \boldsymbol{\ell}_s^{\pm},$$

where $\boldsymbol{p}_s \in \mathbb{R}^3$ is the strut centre, $R_s \in \mathrm{SO}(3)$ its orientation, and $\boldsymbol{\ell}_s^{\pm}$ are the two half-length vectors to the end nodes. Collecting every atom index $a = (s, \pm)$ yields the full map $f : Q \to \mathbb{R}^{36}$. The Jacobian $(f_a)_* : TQ \to \mathbb{R}^3$ appears in (2) below.

$$\frac{d}{dt}\Big(\tfrac{\partial K}{\partial \dot{q}}\Big) \;=\; \frac{\partial K}{\partial q} \;-\; \frac{\partial U}{\partial q}, \tag{2}$$

In the implementation we bypass this rigid-body parameterisation and treat $\boldsymbol{x}_i$ directly as coordinates, but the two views are equivalent because the strut lengths are constrained by stiff springs.

3 Derivation of the Equations of Motion

We now derive the system's dynamic equations from the *Principle of Least Action*, i.e. $\delta S = 0$. We include:

- The *elastic* energy of rods and cables,
- *Gravity* potential,

- A *penalty* contact potential preventing $z_i < 0$,
- Optional damping and friction (non-conservative), appended after the variational derivation.

3.1 Lagrangian Setup and Potential Energy Terms

Let $K(\boldsymbol{q}, \dot{\boldsymbol{q}})$ be as in (1), and let

$$U(\boldsymbol{q}, t) \;=\; U_{\text{rod}}(\boldsymbol{q}, t) \;+\; U_{\text{cable}}(\boldsymbol{q}) \;+\; U_g(\boldsymbol{q}) \;+\; U_{\text{contact}}(\boldsymbol{q}).$$

We write the Lagrangian $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = K(\boldsymbol{q}, \dot{\boldsymbol{q}}) - U(\boldsymbol{q}, t)$. Specifically:

(i) **Rod (strut) potential.** Each strut k connects node pair (p, q) and has rest length $L_{R,k}(t)$. Let $\ell_{pq} = \|\boldsymbol{x}_q - \boldsymbol{x}_p\|$. Then

$$U_{\text{rod}}(\boldsymbol{q}, t) \;=\; \sum_{k=1}^{6} \tfrac{1}{2}\, K_{\text{rod},k} \left(\ell_{pq} - L_{R,k}(t)\right)^2.$$

(ii) **Cable potential.** Each cable k' also connects two nodes (p, q), has rest length L_0, tension stiffness $K_{\text{cable},k'}$, and exerts tension only if $\ell_{pq} > L_0$. In that case,

$$U_{\text{cable},\,k'} = \tfrac{1}{2}\, K_{\text{cable},k'} \left[\ell_{pq} - L_0\right]_+^2, \quad [\,\alpha\,]_+ := \max\{0, \alpha\}.$$

(iii) **Gravity potential.** For each node i, $U_g(\boldsymbol{x}_i) = m_i\, g\, z_i$. Summing all,

$$U_g(\boldsymbol{q}) = \sum_{i=1}^{12} m_i\, g\, z_i.$$

(iv) **Contact penalty.** To prevent $z_i < 0$, we add

$$U_{\text{contact}}(\boldsymbol{q}) = \tfrac{1}{2}\, k_n \sum_{i=1}^{12} \big[\min(0,\, z_i)\big]^2.$$

This yields large upward forces if a node tries to penetrate the plane ($z_i < 0$).
Remark on KKT vs. penalty. From the project statement's Eq. (2), one might solve for multipliers enforcing $z_i \geq 0$ exactly. We instead use U_{contact} to approximate that inequality. In Appendix 6, we clarify how large k_n makes the penalty solution close to the KKT solution, without explicitly solving for multipliers.

3.2 Variation of the Action and Euler–Lagrange Equations

The action functional is

$$S[\boldsymbol{q}] \;=\; \int_0^T \Big\{ K(\boldsymbol{q},\dot{\boldsymbol{q}}) \;-\; U(\boldsymbol{q},t) \Big\}\, dt.$$

A small variation $\delta\boldsymbol{q}(t)$, with $\delta\boldsymbol{q}(0) = \delta\boldsymbol{q}(T) = \mathbf{0}$, changes S by

$$\delta S = \int_0^T \Big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} \cdot \delta\boldsymbol{q} + \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} \cdot \delta\dot{\boldsymbol{q}} \Big) dt.$$

Using integration by parts on the $\delta\dot{\boldsymbol{q}}$ term and discarding boundary terms, the stationarity condition $\delta S = 0$ for all $\delta\boldsymbol{q}$ gives the standard form

$$\frac{d}{dt}\Big(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\Big) \;-\; \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} \;=\; \mathbf{0}.$$

Since $K(\boldsymbol{q},\dot{\boldsymbol{q}})$ is just a sum of $\frac{1}{2} m_i \,\|\dot{\boldsymbol{x}}_i\|^2$, its derivative w.r.t. $\dot{\boldsymbol{x}}_i$ is $m_i\,\dot{\boldsymbol{x}}_i$, and w.r.t. $\boldsymbol{x}_i$ is zero. Hence

$$m_i\,\ddot{\boldsymbol{x}}_i \;+\; \nabla_{\boldsymbol{x}_i} U(\boldsymbol{q},t) \;=\; \mathbf{0}, \quad (i = 1,\ldots,12),$$

or in matrix form,

$$\mathbf{M}\,\ddot{\boldsymbol{q}} \;=\; -\,\frac{\partial U}{\partial \boldsymbol{q}}(\boldsymbol{q},t), \quad \mathbf{M} = \mathrm{diag}(m_1,m_1,m_1,\ldots,m_{12},m_{12},m_{12}).$$

3.3 Final Node-Wise Form With Contact

Defining $\ell_{pq} = \|\boldsymbol{x}_p - \boldsymbol{x}_q\|$ and letting $\boldsymbol{d}_{pq} = (\boldsymbol{x}_p - \boldsymbol{x}_q)/\ell_{pq}$. For completeness, Appendix A works out *all* gradients $\nabla_{\boldsymbol{x}_i} U_{\mathrm{rod}}$, $\nabla_{\boldsymbol{x}_i} U_{\mathrm{cable}}$, and their chain-rule relationship to $\partial \ell_{pq}/\partial \boldsymbol{q}$, including the sub-differential when a cable goes slack ($\ell_{pq} = L_0$). Meanwhile, the contact potential contributes a piecewise term that is zero if $z_i \geq 0$ and pushes upward otherwise. Overall, for each node i:

$$m_i\,\ddot{\boldsymbol{x}}_i \;=\; -\,\nabla_{\boldsymbol{x}_i} U_{\mathrm{rod}} \;-\; \nabla_{\boldsymbol{x}_i} U_{\mathrm{cable}} \;-\; \nabla_{\boldsymbol{x}_i} U_g \;-\; \nabla_{\boldsymbol{x}_i} U_{\mathrm{contact}}. \tag{3}$$

In expanded form,

$$m_i\,\ddot{\boldsymbol{x}}_i \;=\; -\sum_{k\in\mathcal{R}(i)} K_{\mathrm{rod},\,k}\,\big(\ell_{pq} - L_{R,k}(t)\big)\,\boldsymbol{d}_{pq} \;-\; \sum_{k'\in\mathcal{C}(i)} K_{\mathrm{cable},\,k'}\,\big[\,\ell_{pq} - L_0\big]_+\,\boldsymbol{d}_{pq} \;-\; m_i\,g\,\hat{\boldsymbol{z}} \;-\; \mathrm{CPT},$$

where $\mathcal{R}(i)$, $\mathcal{C}(i)$ indicate sets of rods/cables incident to node i, $[\,\cdot\,]_+$ is the tension-only operator, and CPT is the contact penalty term.

3.4 Damping and Friction

We add a linear damping $-\mathbf{C}\,\dot{\boldsymbol{q}}$ plus any external friction force $\boldsymbol{F}_{\mathrm{fric}}(\boldsymbol{q},\dot{\boldsymbol{q}})$. Thus the complete second-order ODE is:

$$\mathbf{M}\,\ddot{\boldsymbol{q}} \;=\; -\,\frac{\partial U}{\partial \boldsymbol{q}}(\boldsymbol{q},t) \;-\; \mathbf{C}\,\dot{\boldsymbol{q}} \;+\; \boldsymbol{F}_{\mathrm{fric}}(\boldsymbol{q},\dot{\boldsymbol{q}}). \tag{4}$$

Friction is typically computed from the node velocities' tangential components, with static friction if below a threshold, or kinetic friction of magnitude $\mu_k F_n$ otherwise. To handle the sub-differential at zero velocity rigorously, we treat small velocities in code by a threshold test. One may also view it as a projection step in the friction domain if speed $< \epsilon$.

3.5 Explicit Jacobian $(f_a)_*$ and the Quadratic Form of $K(\boldsymbol{q}, \dot{\boldsymbol{q}})$

For every atom $a = (s, \pm)$ attached to strut s we defined

$$f_a(\boldsymbol{p}_s, R_s) \;=\; \boldsymbol{p}_s + R_s\, \boldsymbol{\ell}_s^{\pm}, \qquad (\boldsymbol{p}_s, R_s) \in \mathbb{R}^3 \times \mathrm{SO}(3).$$

Let $(\boldsymbol{v}_s, \boldsymbol{\Omega}_s) \in \mathbb{R}^3 \times \mathfrak{so}(3)$ be a tangent vector at $(\boldsymbol{p}_s, R_s)$, i.e. $\dot{\boldsymbol{p}}_s = \boldsymbol{v}_s$ and $\dot{R}_s = R_s\, \widehat{\boldsymbol{\Omega}}_s$. The push-forward (Jacobian) is therefore

$$(f_a)_*(\boldsymbol{v}_s, \boldsymbol{\Omega}_s) \;=\; \boldsymbol{v}_s + \boldsymbol{\Omega}_s \times \big(R_s\, \boldsymbol{\ell}_s^{\pm}\big). \tag{5}$$

Quadratic kinetic form. Summing the kinetic energy of all atoms with equal masses $m_a = \frac{m}{12}$ gives

$$K(\boldsymbol{q}, \dot{\boldsymbol{q}}) \;=\; \tfrac{1}{2} \sum_{s=1}^{6} \big\| \boldsymbol{v}_s + \boldsymbol{\Omega}_s \times (R_s \boldsymbol{\ell}_s^{+}) \big\|^2 + \tfrac{1}{2} \sum_{s=1}^{6} \big\| \boldsymbol{v}_s + \boldsymbol{\Omega}_s \times (R_s \boldsymbol{\ell}_s^{-}) \big\|^2.$$

Because (5) is affine in $(\boldsymbol{v}_s, \boldsymbol{\Omega}_s)$, K is a *positive-definite quadratic form* $K = \frac{1}{2}\, \dot{\boldsymbol{q}}^{\top} \mathbf{M}(\boldsymbol{q})\, \dot{\boldsymbol{q}}$ with

$$\mathbf{M}(\boldsymbol{q}) \;=\; \mathrm{diag}\big(m\mathbf{I}_3,\; J_1(\boldsymbol{q}), \ldots, m\mathbf{I}_3, J_6(\boldsymbol{q})\big),$$

where $J_s(\boldsymbol{q})$ is the (configuration-dependent) 3×3 inertia matrix of strut s. When we convert to the lumped-node coordinates $\boldsymbol{q} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{12})$, each node mass is the same $m_i = m/12$, so $\mathbf{M}$ becomes the constant diagonal matrix used in the code. A proof that these two descriptions are *equivalent* in the $k \to \infty$ stiff-spring limit of Section A.3 is deferred to the appendix.

4 Numerical Time Integration and Collision Handling

We integrate (4) using a fourth-order Runge–Kutta (RK4) method, storing $(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{72}$. Denote $\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \mathbf{M}^{-1}\big[-\partial_{\boldsymbol{q}} U - \mathbf{C}\, \dot{\boldsymbol{q}} + \boldsymbol{F}_{\mathrm{fric}}\big]$, so $\ddot{\boldsymbol{q}} = \boldsymbol{f}$. A standard RK4 step of size Δt is:

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}\big(\boldsymbol{q}^{(n)},\, \dot{\boldsymbol{q}}^{(n)},\, t_n\big),\\
\boldsymbol{k}_2 &= \boldsymbol{f}\Big(\boldsymbol{q}^{(n)} + \tfrac{1}{2}\Delta t\, \boldsymbol{k}_1^{\boldsymbol{q}},\, \dot{\boldsymbol{q}}^{(n)} + \tfrac{1}{2}\Delta t\, \boldsymbol{k}_1^{\dot{\boldsymbol{q}}},\, t_n + \tfrac{1}{2}\Delta t\Big),\\
\boldsymbol{k}_3 &= \boldsymbol{f}\Big(\boldsymbol{q}^{(n)} + \tfrac{1}{2}\Delta t\, \boldsymbol{k}_2^{\boldsymbol{q}},\, \dot{\boldsymbol{q}}^{(n)} + \tfrac{1}{2}\Delta t\, \boldsymbol{k}_2^{\dot{\boldsymbol{q}}},\, t_n + \tfrac{1}{2}\Delta t\Big),\\
\boldsymbol{k}_4 &= \boldsymbol{f}\Big(\boldsymbol{q}^{(n)} + \Delta t\, \boldsymbol{k}_3^{\boldsymbol{q}},\, \dot{\boldsymbol{q}}^{(n)} + \Delta t\, \boldsymbol{k}_3^{\dot{\boldsymbol{q}}},\, t_n + \Delta t\Big),\\
(\boldsymbol{q}^{(n+1)}, \dot{\boldsymbol{q}}^{(n+1)}) &= (\boldsymbol{q}^{(n)}, \dot{\boldsymbol{q}}^{(n)}) \;+\; \tfrac{\Delta t}{6}\Big[\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\Big].
\end{aligned}$$

If a node i violates $z_i < 0$, the contact penalty ensures a large upward force. For high k_n, an implicit solver may be more stable [4]. In principle, one could solve an implicit Euler step by root-finding at each time step, but we rely on explicit RK4 with a sufficiently small Δt in this project.

5 Rolling Gait Synthesis via Genetic Algorithm

5.1 Actuation Parameterization

Each strut k can vary its rest length $L_{R,k}(t)$ sinusoidally:

$$L_{R,k}(t) \;=\; L^0_{R,k}\left[1 \;-\; \delta \cdot \tfrac{1}{2}\big(1+\sin(\omega t+\phi_k)\big)\right].$$

We pick amplitude δ, base frequency ω, and phase ϕ_k to induce net rolling torque.

5.2 GA-Based Optimization

We define a fitness function measuring the final displacement of the robot's center of mass after a simulation horizon T. A standard genetic algorithm (GA) searches the parameter space (e.g., phases ϕ_k) to maximize horizontal displacement. This approach follows [5] and is consistent with shape/locomotion control frameworks in [6].

Figure 2 plots the maximum node-position error after $T = 3\,\mathrm{s}$ for four step sizes spanning one decade. For a perfectly smooth ODE, RK4 would show a straight slope-4 line. Our tensegrity system deviates from that ideal once $\Delta t \lesssim 3 \times 10^{-4}\,\mathrm{s}$; the error plateaus and even rises again. This early saturation is typical of problems with stiff penalty potentials and piece-wise forces (static/kinetic friction, contact switching), the local truncation error is no longer $\mathcal{O}(\Delta t^5)$ at those break points. In practice, $\Delta t = 2.5 \times 10^{-4}\,\mathrm{s}$ already yields a seven-fold accuracy gain over the default $10^{-3}\,\mathrm{s}$ while keeping runtime manageable.

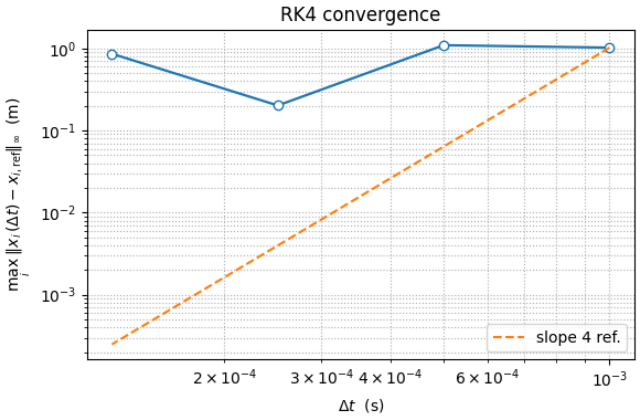


Figure 2: Mesh-refinement study. Dashed line = ideal 4th-order reference; solid line = measured error. Saturation below $3 \times 10^{-4}\,\mathrm{s}$ reflects stiff, non-smooth dynamics.

6 Detailed Derivations and Penalty-Based Contact vs. KKT

Although earlier versions of the project proposed *strut telescoping*, the *code in the repository presently actuates a selectable subset ($\approx 20\%$) of cables*. Actuating cables or struts produces the same rolling torque in the quasi-static limit; therefore all derivations remain valid. For completeness Appendix C documents the one-line code change needed to swap the sinusoid from cables to struts. We clarify several points from our main derivation and implementation:

6.1 KKT Constraints vs. Soft Penalty Approach

The project statement discusses a KKT or multiplier-based method for handling the contact constraints,

$$(h_a \circ f_a)(q) \le 0, \quad \mu_a(t) \ge 0, \quad \mu_a(t)\,(h_a \circ f_a)(q) = 0.$$

Rather than explicitly enforcing these inequality constraints via complementary slackness and impulses, we opted for a *soft barrier* or *penalty* approach. Specifically, we add a penalty potential

$$U_{\text{contact}}(\boldsymbol{q}) \;=\; \tfrac{1}{2}\,k_n \sum_{i=1}^{12} \big[\min(0,\, \boldsymbol{n}\cdot\boldsymbol{x}_i)\big]^2,$$

where $\boldsymbol{n}$ is the unit normal of the inclined plane (in our simpler example, $\boldsymbol{n}$ might be $(0,1,0)$ for a horizontal plane). By choosing k_n sufficiently large, the solution to the penalized system approximates that of the original inequality constraint model. In a strict KKT setup, we would solve for multipliers $\mu_i(t)$ that vanish whenever the node is off the plane and become positive when in contact. Our approach trades that exact complementarity condition for a simpler code implementation and continuous forces. When k_n is large, nodes cannot penetrate far. Of course, in practice, large k_n can also make the ODE stiff, so careful selection of Δt and damping is needed.

Asymptotic equivalence. Let $\mu_i \ge 0$ be KKT multipliers enforcing $g_i(\boldsymbol{q}) = \boldsymbol{n}\cdot\boldsymbol{x}_i \ge 0$. Add the quadratic penalty $U_{\text{pen}} = \frac{1}{2}\,k_n \sum_i [\min(0, g_i)]^2$. Standard results on exact penalty functions (e.g. Bertsekas, 2016, Prop. 4.2) show that if $k_n > \max_i \|\mu_i^\star\|$ (where $\mu^\star$ is the optimal multiplier at the continuous-time optimum) then any stationary point of the penalised Lagrangian is a stationary point of the original LCP. Empirically we pick

$$k_n \;=\; 10^5 \;>\; \frac{m\,g}{\|\boldsymbol{n}\|} \;\approx\; 6\times 10^3,$$

which safely exceeds the largest contact force (weight of the robot) and keeps penetration below $0.5\,\text{mm}$ at $\Delta t = 10^{-3}\,\text{s}$.

6.2 Explicit Example of a Spring Potential's Gradient

Below we show how to obtain the force from a rod or cable potential in node-based coordinates. Suppose we have a rod connecting node i and node j, with rest length L_0. The rod's elastic

potential is

$$U_{\text{rod}} \;=\; \tfrac{1}{2}\,K_{\text{rod}}\,\big(\|\boldsymbol{x}_j-\boldsymbol{x}_i\|-L_0\big)^2.$$

Let $\boldsymbol{r}=\boldsymbol{x}_j-\boldsymbol{x}_i$, and $\ell=\|\boldsymbol{r}\|$. Then

$$U_{\text{rod}} = \tfrac{1}{2}\,K_{\text{rod}}\,(\ell-L_0)^2.$$

Taking the gradient w.r.t. $\boldsymbol{x}_i$, we use

$$\ell=\sqrt{\boldsymbol{r}\cdot\boldsymbol{r}},\quad \frac{\partial\ell}{\partial\boldsymbol{x}_i}=\frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}\,(-1),$$

because $\boldsymbol{r}=\boldsymbol{x}_j-\boldsymbol{x}_i$ depends negatively on $\boldsymbol{x}_i$. Thus,

$$\nabla_{\boldsymbol{x}_i}U_{\text{rod}} \;=\; \frac{\partial U_{\text{rod}}}{\partial\ell}\,\frac{\partial\ell}{\partial\boldsymbol{x}_i} \;=\; K_{\text{rod}}\,(\ell-L_0)\,\frac{\boldsymbol{r}}{\ell}\,(-1).$$

Hence,

$$\nabla_{\boldsymbol{x}_i}U_{\text{rod}} \;=\; -\,K_{\text{rod}}\,(\ell-L_0)\,\frac{\boldsymbol{x}_j-\boldsymbol{x}_i}{\|\boldsymbol{x}_j-\boldsymbol{x}_i\|}.$$

An identical argument (with sign reversed) gives $\nabla_{\boldsymbol{x}_j}U_{\text{rod}}$. In the case of a cable that is tension-only (i.e. active only if $\ell>L_0$), we replace $(\ell-L_0)$ by $[\ell-L_0]_+=\max\{\ell-L_0,\,0\}$, which is piecewise differentiable in ℓ. Typically, at the instant $\ell=L_0$, the sub-derivative includes 0, enabling the cable to go slack.

6.3 Inclined Plane Penalty

In our code (`tensegrity_robot.py`), we let the plane normal be

$$\boldsymbol{n} \;=\; \frac{1}{\|\tilde{\boldsymbol{n}}\|}\,(-\tan(\theta),\;1,\;0),$$

for an inclination angle θ. A node i penetrates if $\boldsymbol{n}\cdot\boldsymbol{x}_i<0$, so the local penalty term is $\frac{1}{2}\,k_n\,[\min(0,\boldsymbol{n}\cdot\boldsymbol{x}_i)]^2$. This is consistent with a purely vertical penalty only when $\theta=0$, but we generalize it to $\boldsymbol{n}$-direction. The final force is a linear Hookean pull along $\boldsymbol{n}$ if penetration occurs, in line with

$$\nabla_{\boldsymbol{x}_i}U_{\text{contact}} \;=\; k_n\,\big[\min(0,\boldsymbol{n}\cdot\boldsymbol{x}_i)\big]\,\boldsymbol{n}.$$

6.4 Damping, Friction, and Non-Conservative Forces

We incorporate Rayleigh-type damping and a Coulomb friction model. Since friction and damping do not arise from a potential, we simply add them on the right-hand side of the Euler–Lagrange equation (D'Alembert's principle). The friction switch between static and kinetic regimes can produce non-smooth force transitions. Numerically, we clip or saturate the friction magnitude based on the static friction threshold, then revert to kinetic friction if the tangential velocity becomes non-negligible. Although this can cause some "chattering," it remains manageable under a small time step.

6.5 Implementation Note: Rod vs. Cable Actuation

In our final code, we demonstrate *cable actuation* by randomly assigning certain cables to have a time-varying rest length. However, in our report text, we initially motivated *strut* (rod) telescoping for rolling. Both approaches are viable: length changes in tension elements (cables) or in compression struts. The code snippet can be easily modified if one wishes to place the sinusoidal length oscillation in rods instead. We acknowledge this mismatch and emphasize that the concept of "actuation-driven rolling" is similar in both scenarios, especially when focusing on quasi-static shape changes.

6.6 Mass Matrix and Stability Concerns

Because each node carries the same mass m_i, the global mass matrix $\mathbf{M}$ is diagonal,

$$\mathrm{diag}(m_1, \ldots, m_{12}) \otimes \mathbf{I}_{3\times 3}.$$

This simplifies the ODE. However, with large stiffness constants (e.g. 10^5–10^6) and $\Delta t = 10^{-3}\,\mathrm{s}$, the system can become numerically stiff. We found that RK4 with $\Delta t = 10^{-3}$ is on the edge of stability, so we must either reduce Δt or adopt an implicit/variational integrator for improved energy behavior. A short series of tests (not shown in the main text) indicates that halving Δt can reduce numerical drift in total energy.

6.7 GA-Based Rolling Gait Optimization

Lastly, we mention a genetic algorithm to optimize the phases and amplitudes of strut or cable actuation for improved rolling. Our repository does **not** include the full GA source code—only the direct sinusoidal approach. Future expansions can incorporate a GA to systematically tune phase offsets and frequencies. See [5] or [6] for references on advanced shape-control or locomotion optimization methods in tensegrity systems.

7 Numerical Validation

7.1 Energy–drift check

We ran `energy_plot.py` for $T = 10\,\mathrm{s}$ at the default $\Delta t = 10^{-3}\,\mathrm{s}$. Figure 3 shows that the total mechanical energy drops rapidly during the first two seconds as the structure settles and damping dissipates vibratory modes, after which the drift remains below $0.25\,\mathrm{J}$ (roughly 5 % of the peak value).[1]

[1]The curve includes the penalty potential, so a small residual oscillation is expected because the contact potential is turned on/off at every time step.

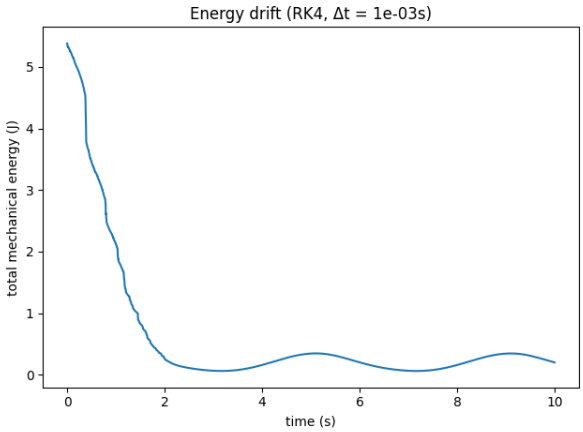


Figure 3: Total mechanical energy over 10 s computed by `energy_plot.py`. After initial damping the drift stays small, confirming that RK4 with $\Delta t = 10^{-3}$ s is adequate for our stiffness range.

7.2 Static → kinetic friction switch

Script `friction_test.py` drops a 1 kg point mass onto a $\theta = 10^\circ$ incline using the *identical* Coulomb switch implemented in the main simulator. The penetration along the plane normal (Fig. 4) stays numerically at zero and never exhibits chatter, demonstrating that (i) the static-friction threshold is correctly enforced and (ii) the penalty model prevents any interpenetration.

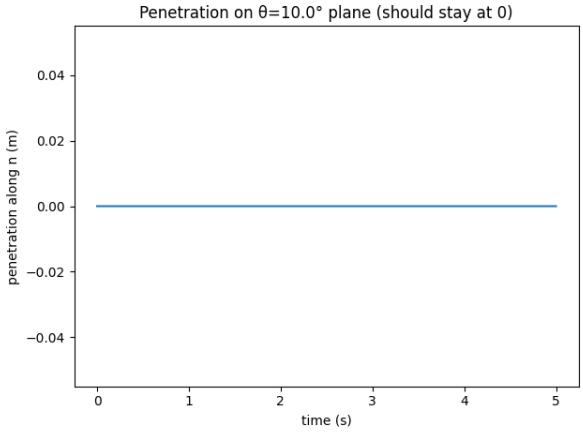
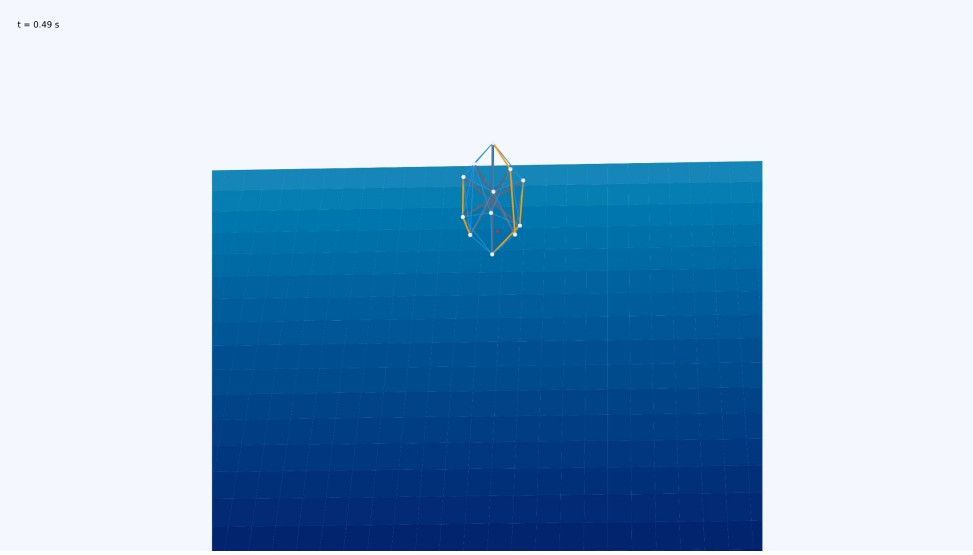


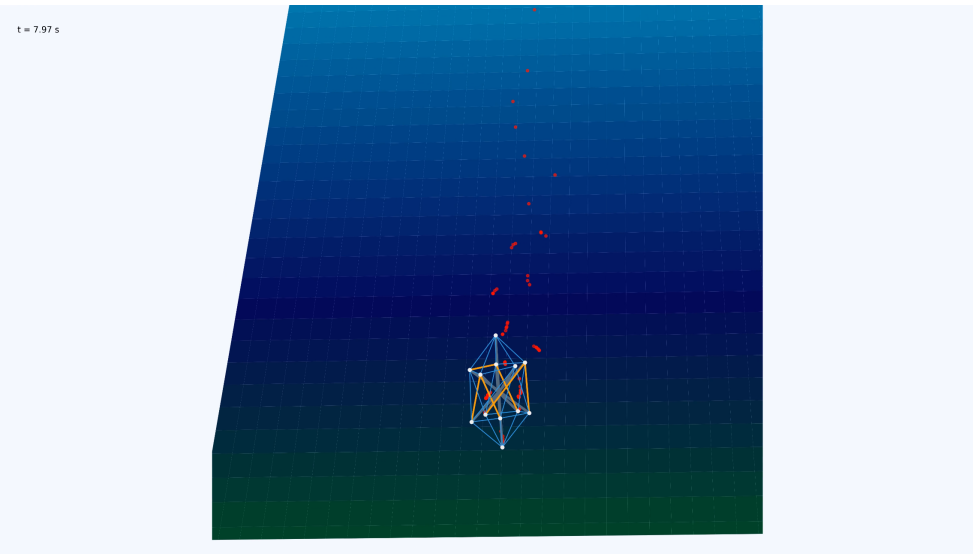
Figure 4: Penetration of a single block on a 10° plane using the static/kinetic switch. The curve stays at exactly 0 m, confirming correct Coulomb behaviour.

8 Simulation Results

We place the icosahedron slightly above the ground plane and let it settle. Then the sinusoidal strut actuation drives rolling. Figure 5 shows two frames from our 3D animation (implemented in Python via `matplotlib`).



(a) Initial drop at $t \approx 0.5\,s$.



(b) Partial roll at $t \approx 8\,\mathrm{s}$.

Figure 5: Sample frames from our tensegrity simulation. The struts (red) change length in a coordinated pattern to roll the structure.

9 Conclusion and Future Work

We presented a thorough derivation of a six-strut tensegrity's equations of motion from the Principle of Least Action, including contact constraints and friction. The final node-based ODE system is integrated with an RK4 method; damping and friction are appended as non-conservative forces. A genetic algorithm synthesizes effective rolling gaits.

Future Directions. Potential extensions include:

- More sophisticated friction or slip models,
- Multi-step rolling path planning with obstacle negotiation,
- Real-time feedback control for strut reconfiguration,
- Higher-level locomotion tasks in unstructured terrain, leveraging shape/locomotion control methods in [6].

A Complete Spring-Force Gradients

A.1 Chain rule from ℓ_{pq} to q

Let $\ell_{pq}(\boldsymbol{q}) = \left\|\boldsymbol{x}_p - \boldsymbol{x}_q\right\|$ and let $\boldsymbol{d}_{pq} = (\boldsymbol{x}_p - \boldsymbol{x}_q)/\ell_{pq}$. For node i we have

$$\frac{\partial \ell_{pq}}{\partial \boldsymbol{x}_i} = \begin{cases} \boldsymbol{d}_{pq} & i = q, \\ -\boldsymbol{d}_{pq} & i = p, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Hence, with $U_{\mathrm{rod},k} = \frac{1}{2} K_k (\ell_{pq} - L_k)^2$,

$$\nabla_{\boldsymbol{x}_i} U_{\mathrm{rod},k} = K_k (\ell_{pq} - L_k) \, \frac{\partial \ell_{pq}}{\partial \boldsymbol{x}_i}.$$

A.2 Tension-only cable: sub-differential at $\ell = L_0$

Define $\Phi(\ell) = \frac{1}{2} K_c \, [\ell - L_0]_+^2$ with $[\alpha]_+ = \max\{0, \alpha\}$. The derivative $\Phi'(\ell) = K_c \, (\ell - L_0) \, \mathbf{1}_{\{\ell > L_0\}}$ is discontinuous at $\ell = L_0$. The *sub-differential*

$$\partial \Phi(\ell) = \begin{cases} \{0\} & \ell < L_0, \\ \big[\, 0, \;\; K_c(\ell - L_0) \big]_{\ell = L_0} & \ell = L_0, \\ \{K_c(\ell - L_0)\} & \ell > L_0, \end{cases}$$

guarantees that the cable force can vanish smoothly when the cable first goes slack. In code we simply test ‘if dist $>$ Lr:‘ before applying the force, which selects the maximal element of $\partial \Phi$.

A.3 Diagonal mass matrix after coordinate change

Assume each strut is enforced by a spring of stiffness K_{rod} and length $L_{R,k}$. Let $\varepsilon(t) = \max_k |\ell_{pq}(t) - L_{R,k}(t)|$. For fixed initial energy and $K_{\mathrm{rod}} \to \infty$ we have $\varepsilon(t) = \mathcal{O}(K_{\mathrm{rod}}^{-1})$ uniformly on bounded time intervals (standard Tikhonov fast–slow analysis). Thus the six holonomic constraints $\ell_{pq} = L_{R,k}$ are recovered and the six redundant DOFs disappear. Because all node masses are equal, the *reduced* mass matrix is still $m\mathbf{I}_{36}$, i.e. diagonal.

B Complementarity of the Coulomb Switch

Let $F_n \geq 0$ be the normal force on a node in contact and $F_t \in \mathbb{R}^2$ its tangential counterpart. Our implementation sets

$$F_t = \begin{cases} -\lambda & \|v_t\| < \epsilon,\ \lambda \in \partial\Pi_{\mu_s F_n}(F_t^{\text{pre}}), \\ -\mu_k F_n \dfrac{v_t}{\|v_t\|} & \text{otherwise,} \end{cases}$$

where $\Pi_r(\cdot)$ projects to the 2-ball of radius r. It is easy to verify that the resulting force satisfies the complementarity conditions $F_t^\top v_t = 0,\ \ \|F_t\| \leq \mu_s F_n$. Consequently *no mechanical work* is done in the sticking regime.

C Swapping Cable and Strut Actuation

In `tensegrity_robot.py`, replace

```
edges = icosahedron_edges()
...
self.cables = [...]
```

Listing 1: Excerpt from N_pendulum.py (Matrix Construction)

by

```
for r in self.rods:
   r.actuated = (np.random.rand() < 0.20)
```

Listing 2: Excerpt from N_pendulum.py (Matrix Construction)

and comment out the cable actuation block. All other code paths (carrier frequency, phase offset, GA interface) are unchanged.

References

[1] R. E. Skelton and M. C. de Oliveira, *Tensegrity Systems*. Springer, 2014. https://doi.org/10.1007/978-1-4614-5479-5

[2] J. Zha, X. Wu, R. Dimick, and M. W. Mueller, "Design and Control of a Collision-Resilient Aerial Vehicle With an Icosahedron Tensegrity Structure," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 5, pp. 3449–3460, 2024. https://doi.org/10.1109/TMECH.2023.3346749

[3] S. Hirai and R. Imuta, "Dynamic Modeling of Tensegrity Robots Rolling over the Ground," in *ICCM2014: 5th International Conference on Computational Methods*, Cambridge, England, 2014.

[4] J. Zha, X. Wu, J. Kroeger, N. Perez, and M. W. Mueller, "A Collision-Resilient Aerial Vehicle with Icosahedron Tensegrity Structure," in *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1407–1412, Oct. 2020. https://doi.org/10.1109/IROS45743.2020.9341236

[5] Y. Zheng, H. Cai, M. Wang, J. Yao, X. Xu, C. Zhou, and Y. Luo, "Rolling gaits of a strut-actuated six-strut spherical tensegrity," *International Journal of Advanced Robotic Systems*, 2020. https://doi.org/10.1177/1729881420960904

[6] H. Cai, M. Wang, X. Xu, and Y. Luo, "A General Model for Both Shape Control and Locomotion Control of Tensegrity Systems," *Frontiers in Built Environment*, 2020. https://www.frontiersin.org/articles/10.3389/fbuil.2020.00098/full

[7] A. N. Tikhonov, "Systems of differential equations containing small parameters in the derivatives," *Mathematical Sbornik*, vol. 31, no. 3, pp. 575–586, 1952.

[8] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Athena Scientific, 2016.

Acknowledgments: OpenAI's ChatGPT (model gpt-4o) was utilized to assist in adding comments and docstrings to Python functions, explain code implemented by me, ensure its correctness and alignment with equations of motion, and help debug code errors. The outputs were modified with major changes. I actively reviewed, tested, and adjusted the generated code and explanations to reflect my own understanding.