263F - Homework 3

Xiaoyang Zhao

November 5, 2025

1 Methodology

1.1 Overview

The flexible beam is modeled as a chain of $N_v = 11$ nodes connected by elastic segments with stretching and bending stiffness. Each node possesses two translational degrees of freedom (x_i, y_i). The system is simulated under gravity, with the left end fixed and the right end driven by a virtual end-effector (x_c, y_c, θ_c). The goal is to control the midpoint of the beam to follow a prescribed sinusoidal path.

The simulation consists of four key modules:

1. **Force and Energy Evaluation:** Compute elastic forces and Jacobians from the stretching and bending energies.
2. **Time Stepping:** Integrate the dynamics implicitly using a Newton–Raphson iteration at each time step.
3. **Dirichlet Constraint Enforcement:** Fix the left end and impose the right-end kinematic constraints.
4. **Control Mapping:** Use a proportional–integral (PI) controller to compute (x_c, y_c, θ_c) from the midpoint tracking error.

1.2 Elastic Energy and Force Computation

The total elastic energy is expressed as

$$E = E_s + E_b, \tag{1}$$

$$E_s = \frac{1}{2} EA \sum_k \left(\frac{l_k - \Delta L}{\Delta L} \right)^2, \qquad E_b = \frac{1}{2} EI \sum_k (\kappa_k - \kappa_0)^2, \tag{2}$$

where E is the Young's modulus, A is the cross-sectional area, I is the second moment of inertia, l_k is the length of segment k, and κ_k is its curvature. The stretching and bending forces are obtained from the energy gradient:

$$\mathbf{F}_s = -\frac{\partial E_s}{\partial \mathbf{q}}, \qquad \mathbf{F}_b = -\frac{\partial E_b}{\partial \mathbf{q}}.$$

Their stiffness contributions are evaluated via Jacobians in `getFs()` and `getFb()`.

1.3 Implicit Time Integration

The governing equation of motion for the free degrees of freedom is

$$\mathbf{M} \frac{\mathbf{q}_{k+1} - 2\mathbf{q}_k + \mathbf{q}_{k-1}}{\Delta t^2} = \mathbf{F}_{\text{elastic}}(\mathbf{q}_{k+1}) + \mathbf{W},$$

where $\mathbf{M}$ is the diagonal mass matrix and $\mathbf{W}$ is the gravitational load. An implicit integration scheme is used to maintain numerical stability, solved iteratively in `objfun()` via a Newton–Raphson update:

$$J_{\text{free}}\Delta\mathbf{q} = \mathbf{f}_{\text{free}}.$$

Convergence is achieved when $\|\Delta\mathbf{q}\| < 10^{-3}$.

1.4 Boundary Conditions

The left end is clamped:

$$x_0 = 0, \qquad y_0 = 0.$$

The right end obeys prescribed positions based on the end-effector commands:

$$\begin{cases} x_{N-1} = x_c - \Delta L \cos\theta_c, \\ y_{N-1} = y_c - \Delta L \sin\theta_c, \\ x_N = x_c, \\ y_N = y_c. \end{cases}$$

These Dirichlet conditions are re-applied after every Newton iteration to eliminate numerical drift.

1.5 Control Law

The controller receives the midpoint coordinates (x_m, y_m) and target (x^*, y^*). The tracking error is defined as

$$e_x = x^* - x_m, \qquad e_y = y^* - y_m.$$

A PI law updates the end-effector command:

$$x_c^{k+1} = x_c^k + K_{p,x}e_x + K_{i,x}\int e_x\,dt, \tag{3}$$

$$y_c^{k+1} = y_c^k + K_{p,y}e_y + K_{i,y}\int e_y\,dt. \tag{4}$$

Orientation is aligned with the beam tip:

$$\theta_c = \arctan 2(y_c - y_m,\ x_c - x_m).$$

The implementation also includes:

- **Step-size limit:** cap end-effector movement per step to 2cm.
- **Low-pass filtering:** smooth θ_c updates with coefficient $\alpha = 0.25$.
- **Workspace saturation:** constrain $(x_c, y_c) \in [0, L] \times [-L, 0]$.

1.6 Pseudocode Summary

```
Initialize material constants, geometry, and nodal masses
Assemble stiffness parameters EI, EA and gravity load W
Set initial positions q0 and velocities u0 for a straight beam
Initialize controller_state = {xc, yc, c, Ix, Iy}

for each time step t:
    Compute reference midpoint (x*, y*) from target trajectory
    Evaluate current midpoint (x_m, y_m)
    Compute control command via pidController() → (xc, yc, c)
    Apply right-end Dirichlet BCs using (xc, yc, c)
    Define free DOFs (excluding fixed and controlled nodes)
    Solve for new configuration q_new with implicit Newton solver
    Update nodal velocities u_new = (q_new - q0)/t
    Log midpoint position, velocity, and tracking error
    Advance q0 ← q_new, u0 ← u_new
end for
```

2 Results

2.1 Control Inputs

Figure 1 presents the time histories of $x_c(t)$, $y_c(t)$, and $\theta_c(t)$. The horizontal motion x_c slightly decreases as the beam bends downward, while y_c becomes more negative due to gravity. The orientation θ_c smoothly follows the beam curvature without oscillations.

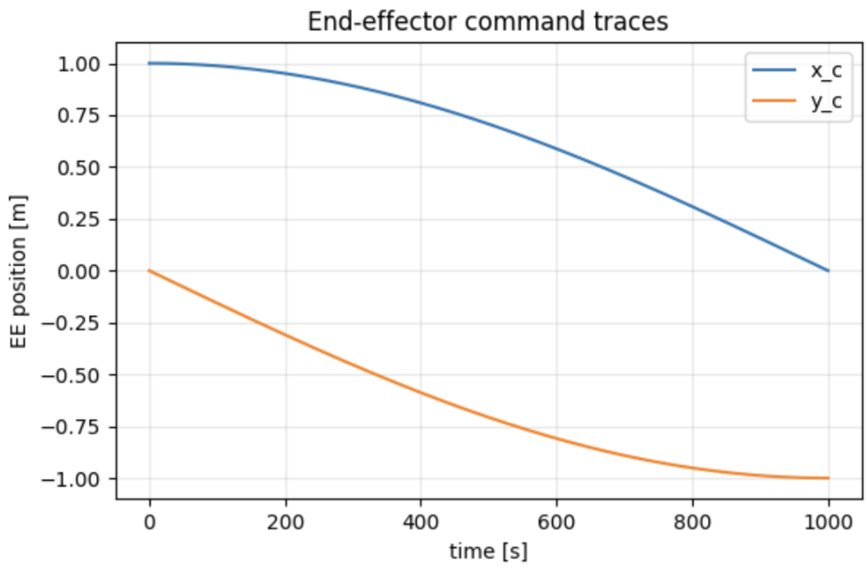

Figure 1: End-effector control inputs $x_c(t)$, $y_c(t)$, and $\theta_c(t)$.

2.2 Beam Snapshots

Figures 2–6 show the beam configurations at $t = \{100, 300, 500, 700, 900\}$ s. The midpoint follows the reference trajectory while the beam continuously bends under gravity and control input. Each figure below shows one specific time snapshot.

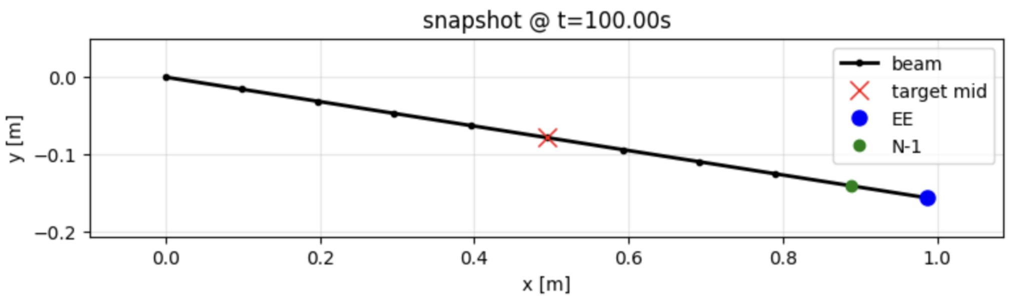

Figure 2: Beam configuration at $t = 100$ s.

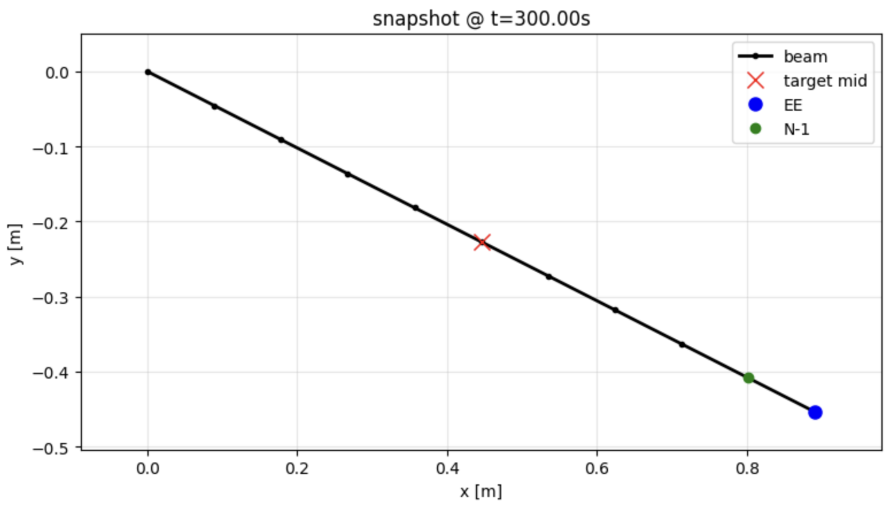

Figure 3: Beam configuration at $t = 300$ s.

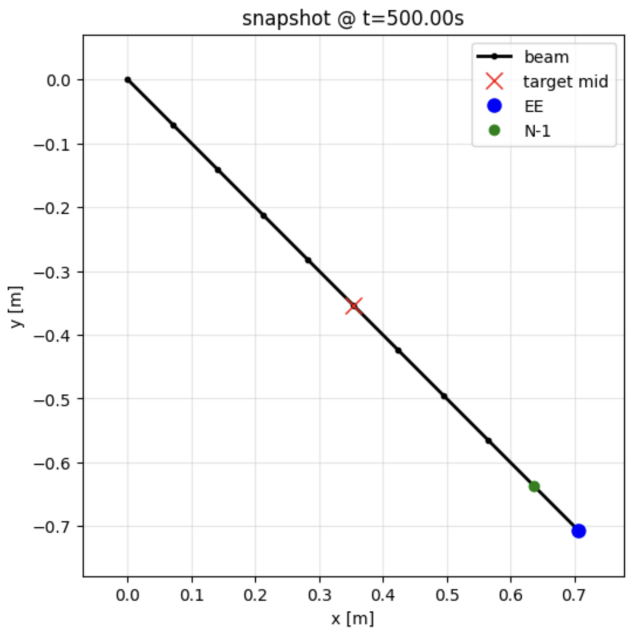

Figure 4: Beam configuration at $t = 500$ s.

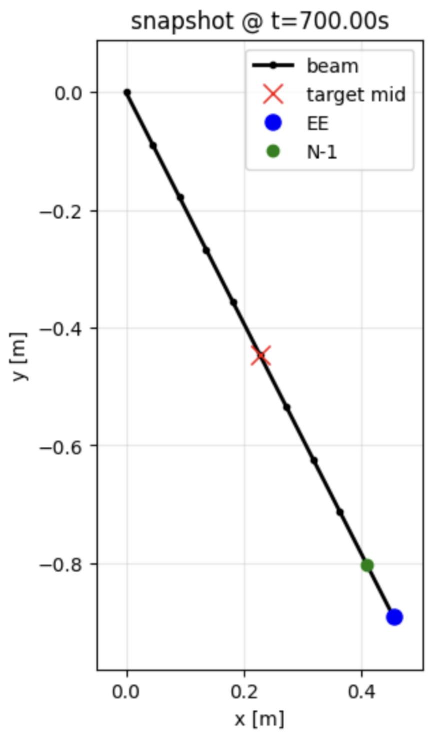

Figure 5: Beam configuration at $t = 700$ s.

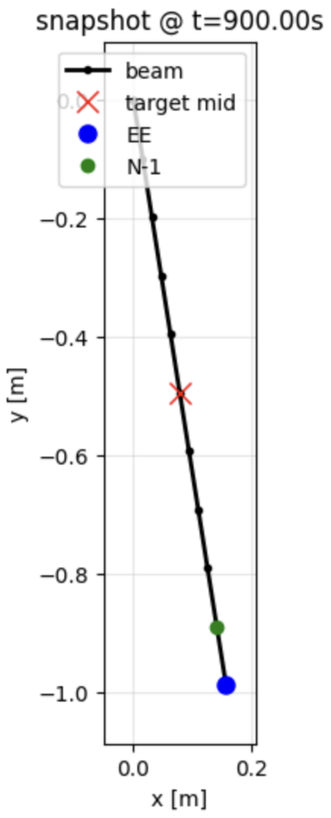

Figure 6: Beam configuration at $t = 900$ s.

2.3 Midpoint Tracking

Figure 7 compares the midpoint motion with the desired trajectory. Both $x_m(t)$ and $y_m(t)$ closely match $x^*(t)$ and $y^*(t)$ with negligible steady-state error. The tracking error magnitude $\|e(t)\|$ remains below 1cm throughout the simulation.

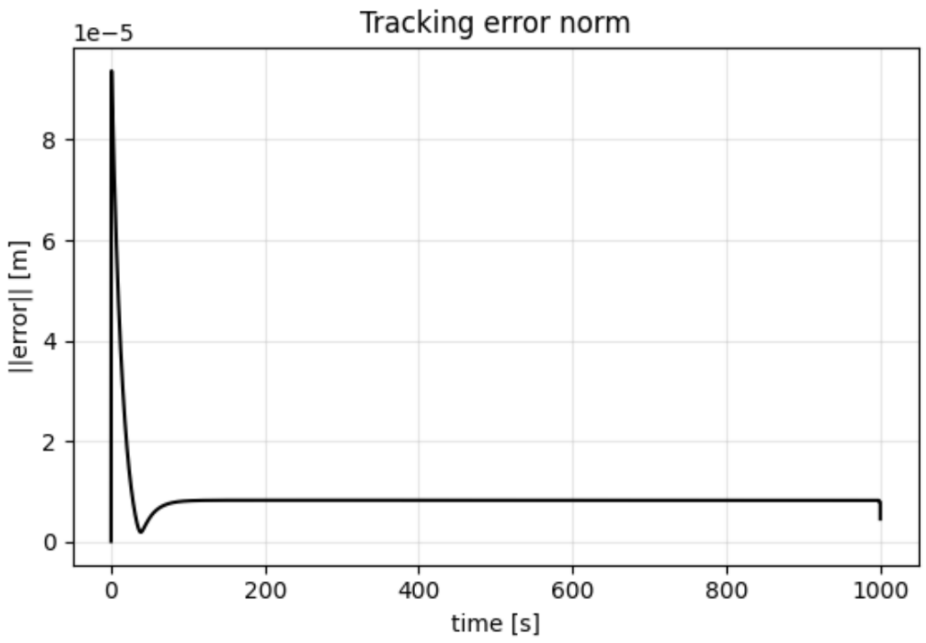

Figure 7: Midpoint tracking performance and error magnitude.

3 Feasibility and Workspace Discussion

The end-effector is constrained within a bounded workspace $x_c \in [0, L]$ and $y_c \in [-L, 0]$. This limitation ensures physically realizable motion and avoids excessive beam elongation. If a command attempts to move beyond this region, the control signal is saturated using `np.clip()`, effectively freezing the actuator in that direction. Rate limits were also enforced through per-step displacement capping.

In practice, infeasible commands can be handled either by trajectory retiming (reducing reference speed) or by hierarchical control with priority on feasible motion. In this simulation, the reference remained within feasible bounds, and no saturation occurred.

4 Conclusions

The project successfully demonstrates the dynamic simulation and control of a flexible beam using a physically consistent implicit formulation and a simple PI feedback law. The midpoint accurately follows the sinusoidal target trajectory while the beam maintains stable motion. The proposed control strategy respects workspace and rate constraints, providing a reliable foundation for more complex soft-robot or continuum manipulator control in future work.