MAE 263F HW3

Robotic Control of an Elastic Beam

Nathan Ge[1]

***Abstract*— In this homework, I will control the shape of a slender elastic beam with a planar robotic end effector. The beam is modeled as a 2D mass–spring chain. I will impose time-varying Dirichlet boundary conditions at the right end (position and orientation) to drive a middle node along a prescribed trajectory.**

I. PROBLEM STATEMENT

Task. Design a path planner that generates control inputs $x_c(t), y_c(t), \theta_c(t)$ such that the middle node of the beam tracks the prescribed trajectory $(x_*(t), y_*(t))$ while accounting for gravity.

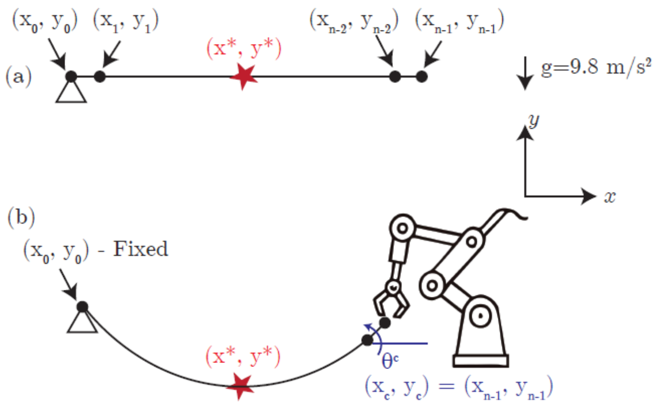

Fig. 1: (a) Elastic beam and (b) its discrete mass–spring representation.

External Forces: Gravity acts on each node:
$\mathbf{f}_g = [0, -mg]$ with $g = 9.81$ m/s^2

Beam Geometry and Material

Beam length	$L = 1$ m
Outer radius	$R = 0.013$ m
Inner radius	$r = 0.011$ m
Young's modulus	$E = 70$ GPa (aluminum)
Density	$\rho = 2700$ kg/m^3
Cross-sectional area	$A = \pi(R^2 - r^2)$
Area moment of inertia	$I = \frac{\pi}{4}(R^4 - r^4)$

Discretization

- Number of nodes: N, indexed $1, \dots, N$, uniformly spaced $\Delta L = \frac{L}{N-1}$
- Lumped mass at each node: $m = \frac{\rho A L}{N-1}$
- Use standard elastic energy for springs between adjacent nodes (bending and stretching), consistent with in-class discussion

Boundary Conditions

- Left end clamped at the origin:

$$x_1(t_{k+1}) = 0, \quad y_1(t_{k+1}) = 0$$

- Right end controlled by a planar robot with time-varying commands $x_c(t)$, $y_c(t)$, $\theta_c(t)$ and initial values $x_c(0) = L$, $y_c(0) = 0$, $\theta_c(0) = 0$
- Dirichlet constraints on the last two nodes:

$$x_N(t_{k+1}) = x_c(t_{k+1}) \tag{1}$$

$$y_N(t_{k+1}) = y_c(t_{k+1}) \tag{2}$$

$$x_{N-1}(t_{k+1}) = x_c(t_{k+1}) - \Delta L \cos(\theta_c(t_{k+1})) \tag{3}$$

$$y_{N-1}(t_{k+1}) = y_c(t_{k+1}) - \Delta L \sin(\theta_c(t_{k+1})) \tag{4}$$

Target Node and Trajectory

- Middle node: $* = \frac{N+1}{2}$ (e.g., $* = 10$ when $N = 19$; pick N odd for a unique middle node)
- Desired trajectory of the middle node (x_*, y_*) for $t \in [0, 1000]$ s:

$$x_*(t) = \frac{L}{2} \cos\left(\frac{\pi}{2} \frac{t}{1000}\right), \quad y_*(t) = -\frac{L}{2} \sin\left(\frac{\pi}{2} \frac{t}{1000}\right)$$

*This work was not supported by any organization

[1]Nathan Ge is a student in the Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, CA 90095 nzge@g.ucla.edu

II. Discussion

A. *Pseudocode*

```
Time-Stepping Main Loop
1) Set up initial guess of $\{x_c, y_c, \theta_c\}$ for the current time step.
2) Solve the equilibrium position of the beam at the given time step using the implicit Euler Newton–Raphson solver (2D Euler beam from class) with the current guess.
    • Modify the objective function to include a Dirichlet constraint on the $(N-1)$-th node.
    • Compute the $(N-1)$ node position based on $\{x_c, y_c, \theta_c\}$ before running the objective function to provide an initial guess.
3) Measure the midpoint error: the difference between the current midpoint and the desired midpoint from the prescribed trajectory.
4) Compute the Jacobian $J$ mapping $\{x_c, y_c, \theta_c\}$ to the midpoint node by perturbing each input and measuring the change in the midpoint.
5) Solve for the control update: $dU = -\text{pinv}(J)\,(\text{midpoint error})$ using the pseudo-inverse of $J$.
6) Update the guess of the control node: new guess $=$ old guess $+\, dU$.
7) Repeat steps 2–6 until the midpoint error is within a given tolerance or a maximum of 30 iterations is reached.
8) Repeat for all time steps.
```

Force/Energy Evaluation Use the unmodified 2D Euler beam implicit Euler Newton–Raphson solver from class to evaluate forces and energies at each equilibrium configuration.

B. *Plots*

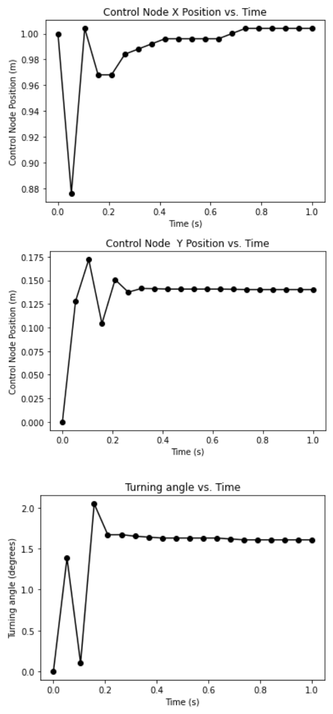

Fig. 2: Plots of the control inputs $x_c(t), y_c(t), \theta_c(t)$ over time.

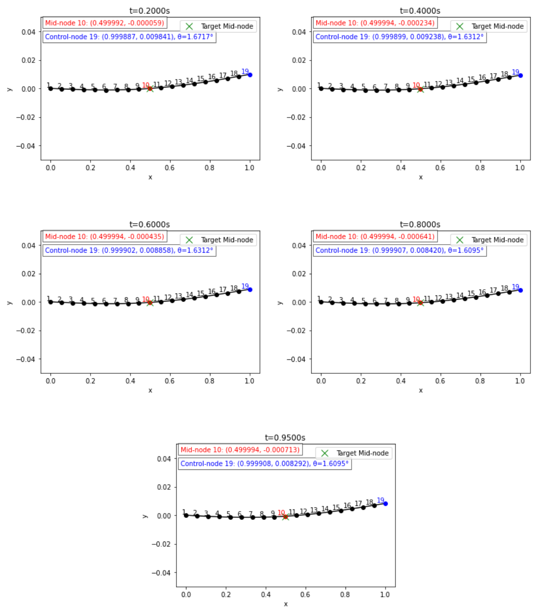

Fig. 3: Five snapshots of the beam shape (node positions) during the motion.

C. Feasibility Limits

The algorithm inherently accounts for feasibility limits through the equilibrium solver and iterative control updates. At each time step, the solver computes the physically achievable configuration for the current control input, ensuring that joint and workspace limits are respected. The numerical Jacobian-based control updates provide small, incremental corrections toward the desired mid-node position, preventing commands that exceed actuation limits. In addition, control updates are clipped to further ensure numerical feasibility. If a desired target is outside the feasible workspace, the iterative loop may fail to converge, effectively saturating the control input at the nearest achievable configuration. Thus, feasibility is handled implicitly through the combination of the solver and iterative control strategy.

The algorithm is quasi-static, meaning it computes robot configurations assuming each control input reaches equilibrium before advancing in time. It does not account for dynamic effects or temporal constraints, and it is not phase-based or trajectory-retimed. Desired motions are applied at fixed time steps without adjustment for joint velocity, acceleration, or workspace limits.

ACKNOWLEDGMENT

The author thanks Professor Khalid Jawed for guidance and instruction throughout the MAE 263F Soft Robotics course.