MAE 263F Fall 2025 Homework_3

Luke Chang 905954847

I. Task Output

C. At least five snapshots of the beam shape (node positions) during the motion..

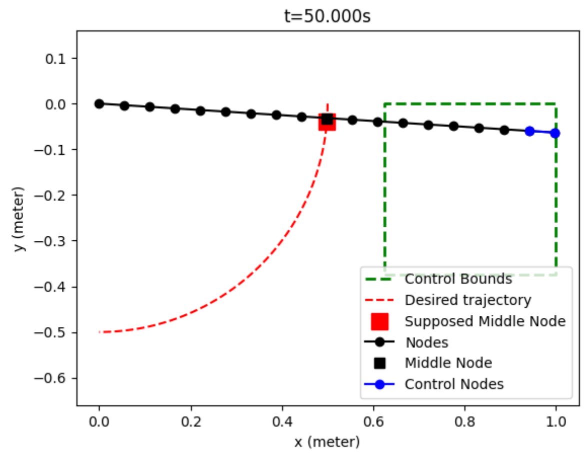

Fig. 1. Beam shape during motion at t = 50s

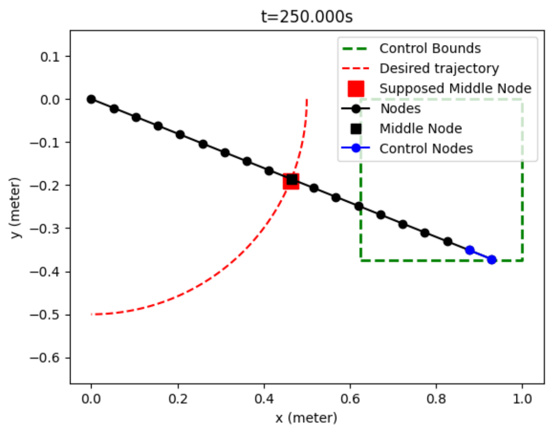

Fig. 2. Beam shape during motion at t = 250s

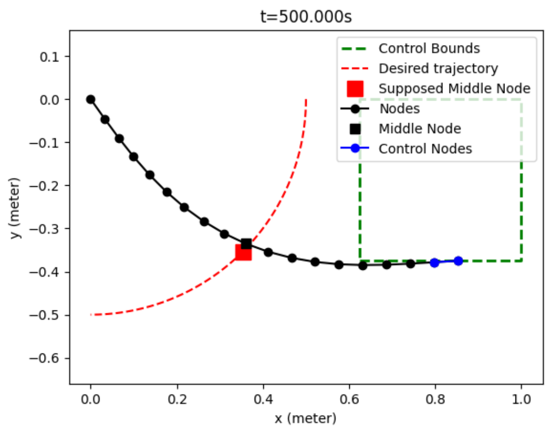

Fig. 3. Beam shape during motion at t = 500s

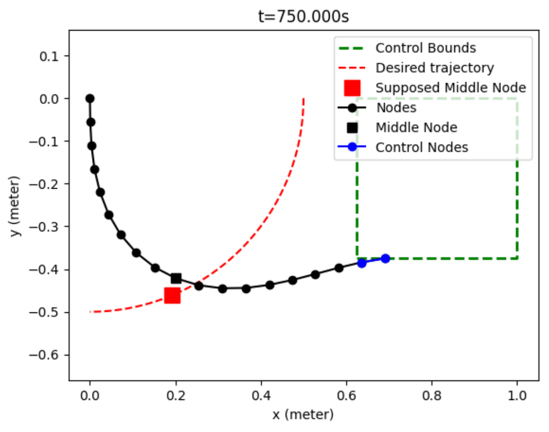

Fig. 4. Beam shape during motion at t = 750s

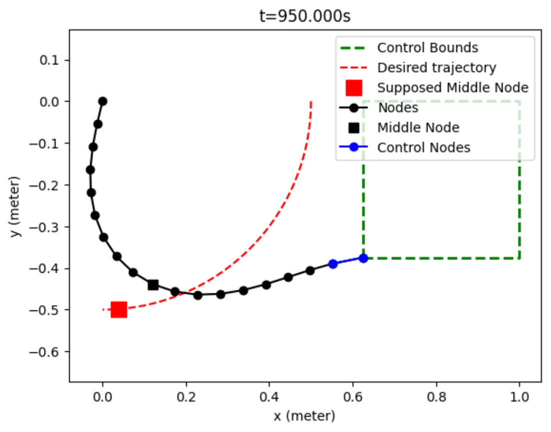

Fig. 5. Beam shape during motion at t = 950s

II. REPORT DELIVERABLES

A. A clear step-by-step description of your method, including pseudocode/algorithms for: (i) force/energy evaluation, (ii) time stepping, (iii) enforcement of Dirichlet constraints, and (iv) the path-planning or control law mapping the tracking error of node j to {xc, yc, θc}.

1. In this simulation a proportional controller is implemented using the spring beam model from previous weeks with the goal of extracting the target trajectory with the simplest method possible. All code additions were added in the main python script. The code structure can be broken down as such:
   a. Setup: Gravity is enabled in the y direction, while viscous damping and other external loads other than weight are omitted for energy evaluation. Boundary conditions wise, while only the left most node is pinned, the first node and last 2 nodes are fixed, where the last 2 nodes are later controlled and enforced Dirichlet constraints upon. Here bounding sizes for the control workspace are also setup including the rectangular bounding box, maximum loop iterations, movement rate, tolerance, and proportional control scaling factor.

```
x_size = .375
y_size = x_size

xc0, yc0 = RodLength, 0.0
xc_min, xc_max = xc0 - x_size, xc0
yc_min, yc_max = yc0 - y_size, yc0

max_rate = 0.05
max_loop = 5
max_tol = .01

x_mag = .25
y_mag = .25
```

   b.
   c. Time Stepping Loop: Here is the bulk of programming additions. Initially the target node trajectory and ideal, unbounded control trajectory are created.

```
x_star = 0.5 * RodLength * np.cos( (np.pi/2.0) * (t_arr/1000.0) )
y_star = -0.5 * RodLength * np.sin( (np.pi/2.0) * (t_arr/1000.0) )
x_c = RodLength * np.cos( (np.pi/2.0) * (t_arr/1000.0) )
y_c = -RodLength * np.sin( (np.pi/2.0) * (t_arr/1000.0) )
t_c = -t_arr*np.pi/2/1000
```

   Entering the time stepping loop, each iteration the objfun is first called to call on the default helper functions for energy evaluation with each time step. The new code is implemented right after such where the control input first follows the ideal unbounded controls with Dirichlet constraints implemented here until the control node reaches the control boundaries.

```
q0[nv*2-1] = y_c[timeStep]
q0[nv*2-2] = x_c[timeStep]
q0[nv*2-3] = q0[nv*2-1] - deltaL*np.sin(t_c[timeStep])
q0[nv*2-4] = q0[nv*2-2] - deltaL*np.cos(t_c[timeStep])
```

   Once hitting the boundary, a new nested For loop contains the error evaluation between the target middle node position vs reality and proportionally adjusts the control nodes based on this error multiplied by the factor set,

```
q0[nv*2-1] += y_error * y_mag
q0[nv*2-2] += x_error * x_mag
q0[nv*2-3] = q0[nv*2-1] - deltaL*np.sin(t_c[timeStep])
q0[nv*2-4] = q0[nv*2-2] - deltaL*np.cos(t_c[timeStep])
```

   continuously ensuring that the control node stays within the control boundaries and recalling the objfun. Finally within this loop, the beam is plotted at specific time intervals shown above.

B. Plots of the control inputs xc(t), yc(t), θc(t) over time.

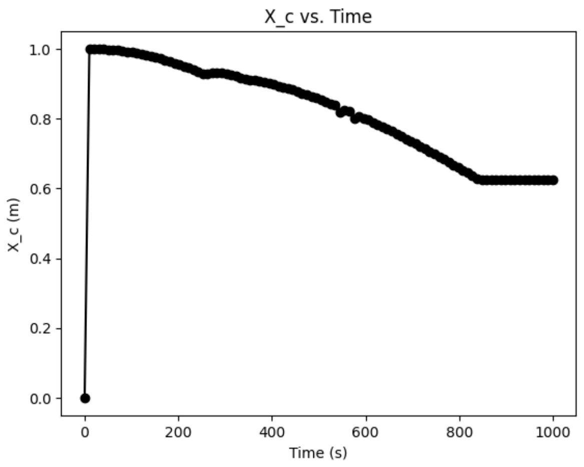

Fig. 6. X control over time

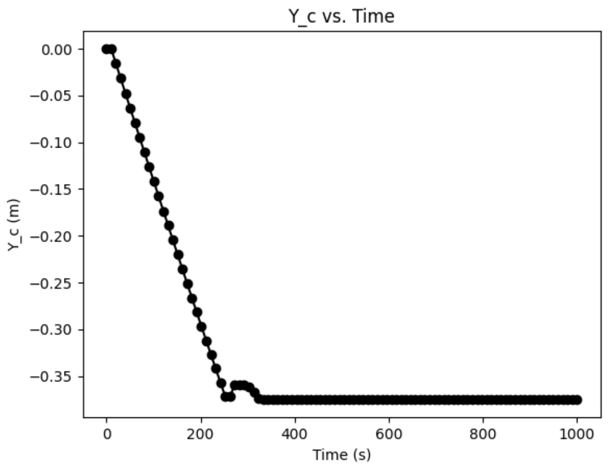

Fig. 7. Y control over time

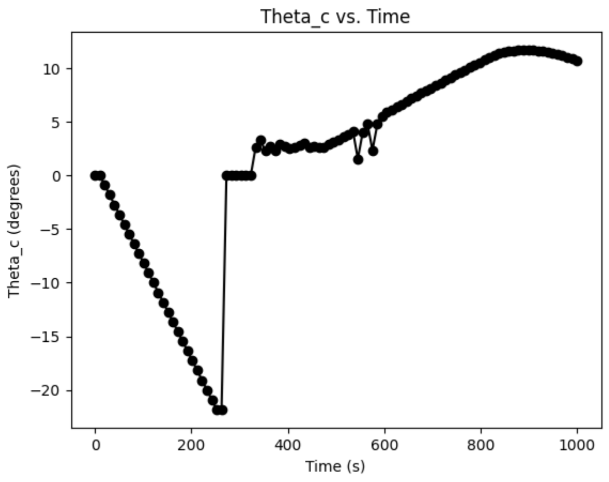

Fig. 8. Theta control over time

C. A brief discussion of feasibility limits due to the robot's workspace and joint limits; explain how you handle infeasible commands (e.g., saturation or trajectory re-timing)...

Additionally below shows the beam shape with time utilizing the ideal unbounded controls. Figures 9-13 compared to that of Figures 1-5 show the physical limits in reality due to robotic workspace of the controls. The unbounded controls show in reality the simplest workspace needed without large strains or stretching. The bounded controls show how realistically unfeasible it is to reach perfectly the target trajectory depending on the workspace or the heavier computation power needed with other control methods to optimize for such. Within this simulation saturation is incorporated making sure when the control nodes reach the control boundary that the node moves only within such constraints. Additional proportional controls wanting to go outside the boundary are essentially clipped.

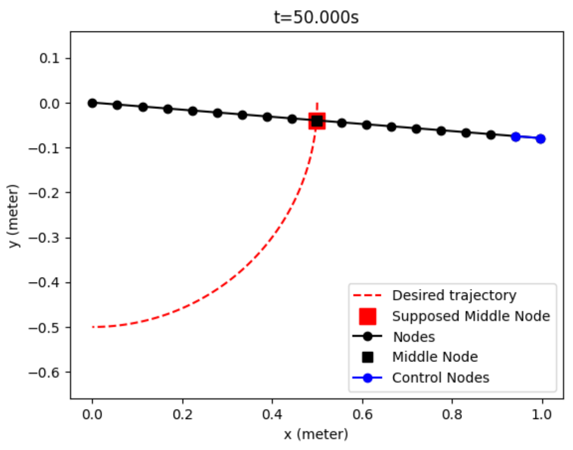

Fig. 9. Beam shape during motion WITHOUT workspace constraints at t = 50s

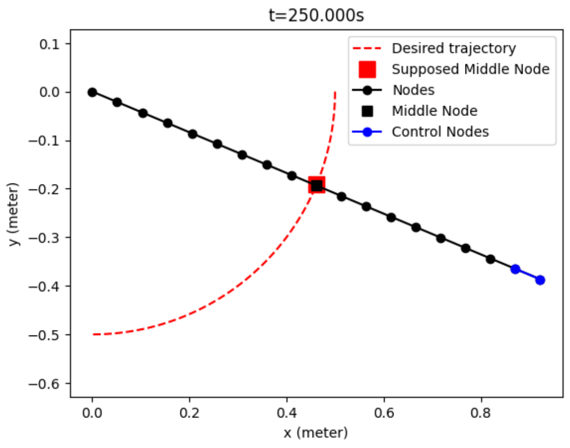

Fig. 10. Beam shape during motion WITHOUT workspace constraints at t = 250s

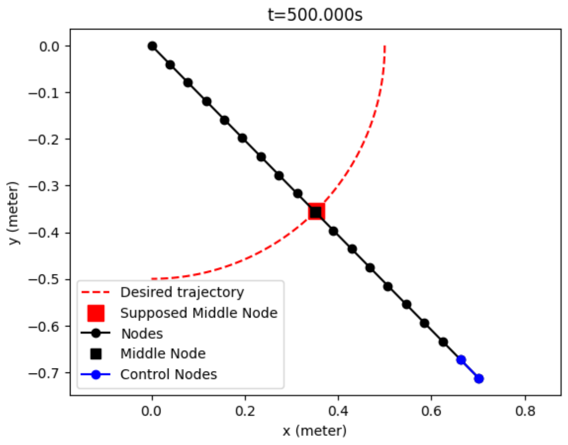

Fig. 11. Beam shape during motion WITHOUT workspace constraints at t = 500s

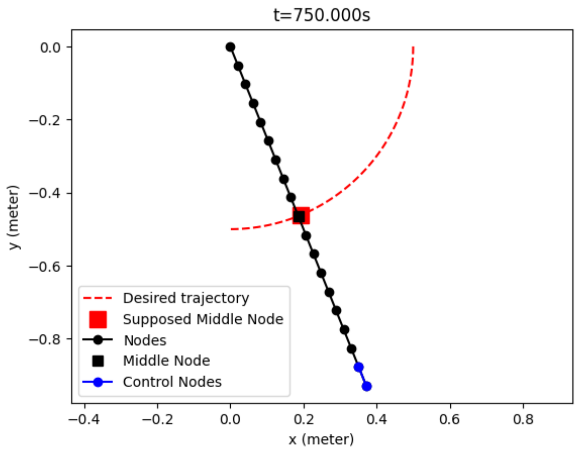

Fig. 12. Beam shape during motion WITHOUT workspace constraints at t = 750s

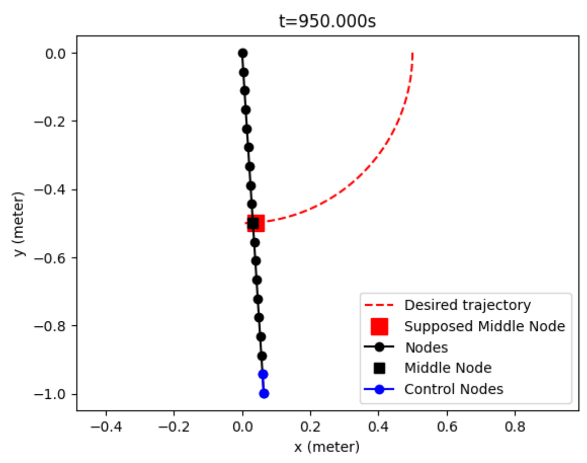

Fig. 13. Beam shape during motion WITHOUT workspace constraints at t = 950s

References

[1] K. J. Majeed, Colab notebook, MAE 263F: Mechanics of Flexible Structures and Soft Robots, University of California, Los Angeles, Fall 2025.