Homework 3

MAE 263F Marwan Fayed

• 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}.

(i) Force / Energy Evaluation

Stretching (per edge k between nodes k and k+1)

- Current edge length $L_k = \| x_{k+1} - x_k \|$.
- Rest length $l_k$(uniform = $\Delta L$).
- Edge stretching energy $E_k^s = \frac{1}{2} EA\, l_k (1 - L_k / l_k)^2$.
- From this, assemble the 4-vector gradient (forces on k and k+1) and 4×4 Hessian into global vectors/matrices.

Bending (per interior node k using nodes k−1, k, k+1)

- Compute a discrete curvature scalar $\kappa_k$from the two adjacent edge directions.
- Bending energy $E_k^b = \frac{1}{2} \frac{EI}{l_k} (\kappa_k - \kappa_k^0)^2$(here $\kappa_k^0 = 0$).
- Assemble the 6-vector gradient and 6×6 Hessian into global structures.

```
Fs = 0; Js = 0
for each edge k = 1..N-1:
  (g4, H4) = StretchGradHess(k)
  scatter Fs -= g4; Js -= H4
Fb = 0; Jb = 0
for each interior node k = 2..N-1:
  (g6, H6) = BendGradHess(k-1,k,k+1)
  scatter Fb -= g6; Jb -= H6
F_int = Fs + Fb
J_int = Js + Jb
```

(ii) Time Stepping (Implicit Backward Euler + Newton)

Goal: solve for positions at the next step $q_{n+1}$ and then update velocity.

Residual and Jacobian at iterate q

- Velocity $v = (q - q_n)/\Delta t$.
- Inertia: $F_{in} = m/\Delta t\,(v - u_n), J_{in} = M/\Delta t^2$.
- Internal: $(F_{int}(q), J_{int}(q))$ from (i).
- Damping: $F_v = C\,v, J_v = C/\Delta t$.
- Residual $r = F_{in} - F_{int} - F_v - W$.
- Jacobian $J = J_{in} - J_{int} - J_v$.

Newton loop (with constraints applied to free DOFs)

```
q = q_n
repeat:
  compute r, J
  take free dofs: r_free = r[free], J_free = J[free, free]
  Δq_free = solve(J_free, r_free)
  q[free] = q[free] - Δq_free
until ||r_free|| < tol or max_iters
q_{n+1} = q
u_{n+1} = (q_{n+1} - q_n)/Δt
```

(iii) Dirichlet Constraints (Clamped/Prescribed DOFs)

We fix:

- Left end node $(x_1, y_1)$ = constants.

- Right end two nodes to impose end-effector pose:
    - Tip node (x_N, y_N) = (x_c, y_c).
    - Previous node (x_{N−1}, y_{N−1}) = (x_c − ΔL·cos θ_c, y_c − ΔL·sin θ_c).

How it's enforced

1. Before each Newton solve, overwrite those DOFs in q with the prescribed values.
2. Build fixed = {x1,y1, xN,yN, xN-1,yN-1} and free = all \ fixed.
3. In Newton, only solve for the free subset (i.e., use r_free, J_free) and write back to q[free]. The fixed DOFs never change during the solve.

```
ImposeRightEnd(q, x_c, y_c, θ_c, ΔL):
  q[xN] = x_c; q[yN] = y_c
  q[xN-1] = x_c - ΔL*cos(θ_c)
  q[yN-1] = y_c - ΔL*sin(θ_c)
BuildBCSets():
  fixed = {x1,y1, xN,yN, xN-1,yN-1}
  free  = all_dofs \ fixed
```

(iv) Control Law (Tracking error of node j → {x_c, y_c, θ_c})

We drive the right end so that a chosen node $j$(here, the middle node) tracks a desired trajectory $(x^*(t), y^*(t))$using a simple PID on position error. We also rate-limit the commanded motion at the end effector.

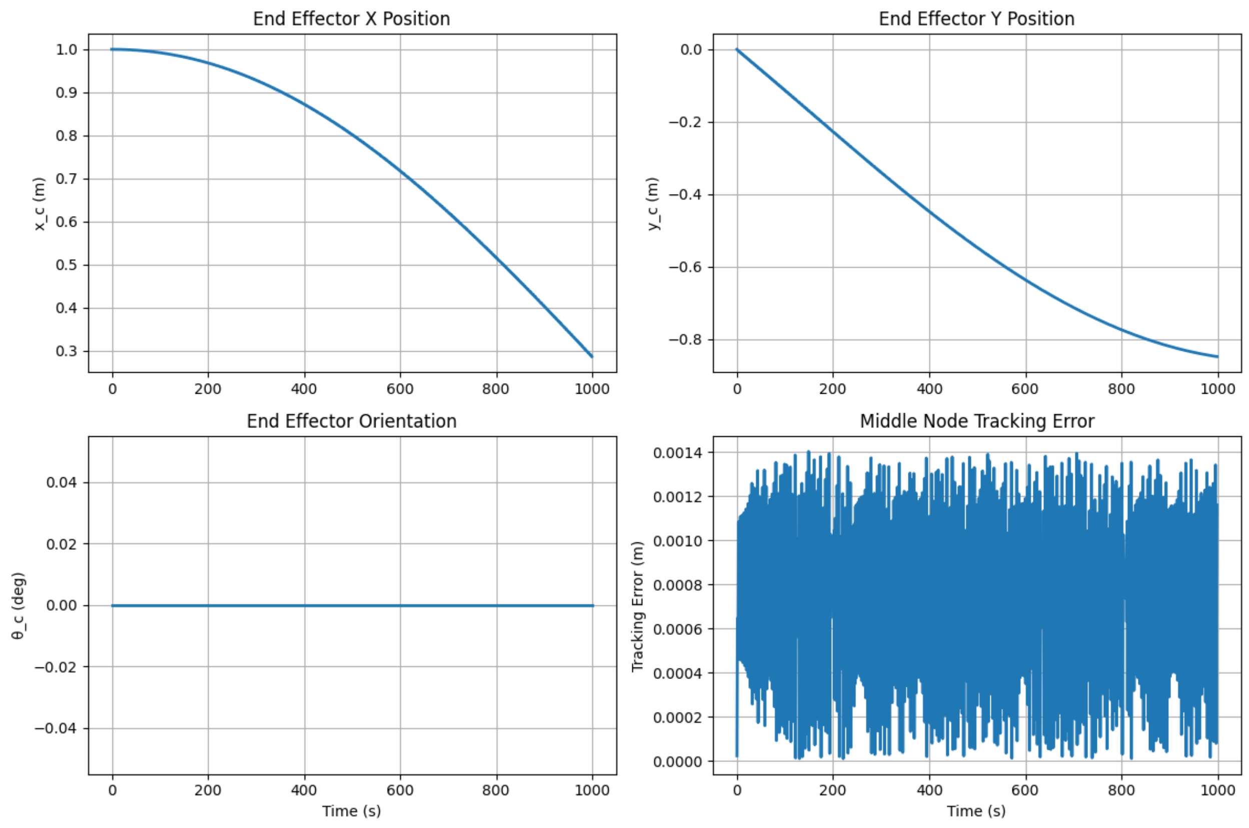
At each time step

1. Measure middle node: $(x_j, y_j)$.
2. Compute errors:
    - $e_x = x^* - x_j$, $e_y = y^* - y_j$.
    - Approx. derivatives: $\dot{e}_x \approx (x_j - x_j^{prev})/\Delta t$, similarly for $y$.
3. Update integrals: $I_x += e_x \Delta t$, $I_y += e_y \Delta t$.
4. PID increments:

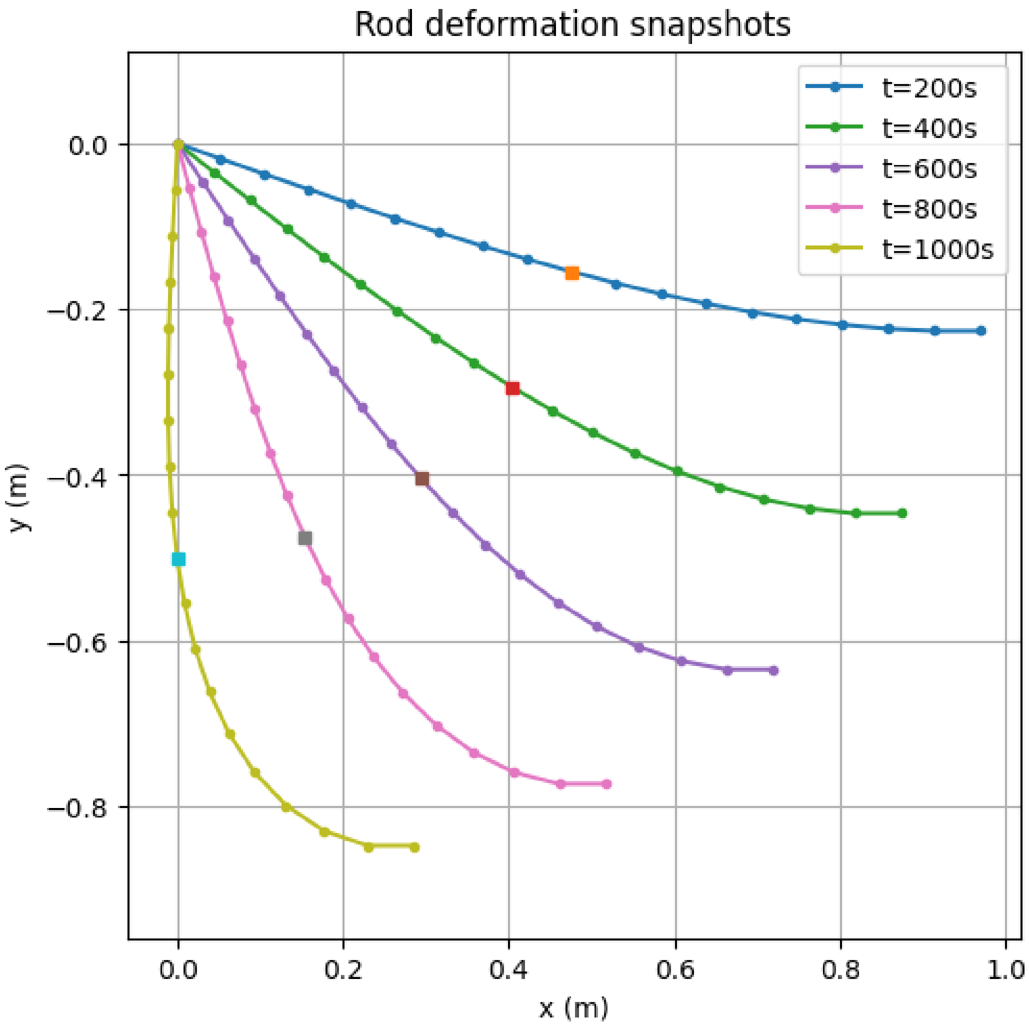
    - $\Delta x_c = K_p e_x + K_i I_x - K_d \dot{e}_x$
    - $\Delta y_c = K_p e_y + K_i I_y - K_d \dot{e}_y$
5. Proposed command: $(x_c, y_c) \leftarrow (x_c, y_c) + (\Delta x_c, \Delta y_c)$, set $\theta_c = 0$.
6. Rate limit (optional but used): limit the change from current tip to at most rmax_step.

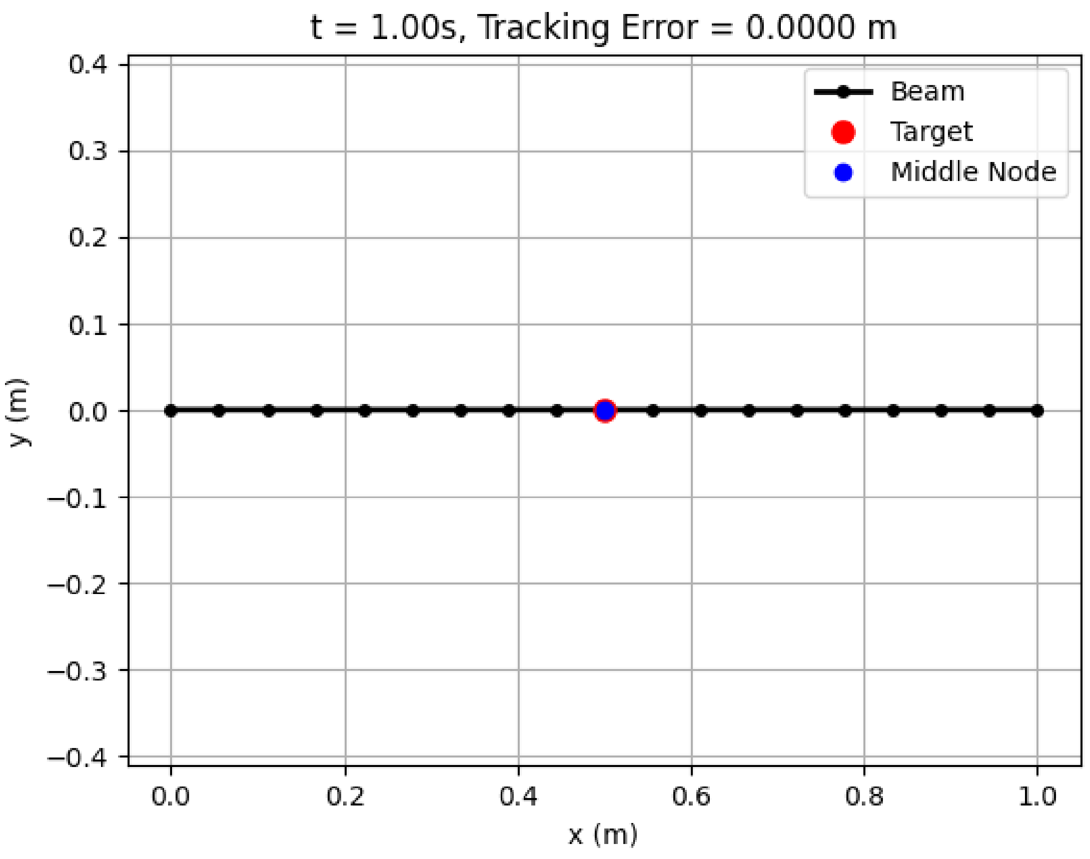
```
EndEffectorPID(x_star, y_star, xj, yj, xj_prev, yj_prev,
        x_c_prev, y_c_prev, I, Δt, Kp, Ki, Kd, rmax_step,
        x_tip_curr, y_tip_curr):
  ex = x_star - xj
  ey = y_star - yj
  ex_dot = (xj - xj_prev)/Δt
  ey_dot = (yj - yj_prev)/Δt
  I.x += ex * Δt
  I.y += ey * Δt
  dx_c = Kp*ex + Ki*I.x - Kd*ex_dot
  dy_c = Kp*ey + Ki*I.y - Kd*ey_dot
  x_c = x_c_prev + dx_c
  y_c = y_c_prev + dy_c
  θ_c = 0.0
  # rate limiter relative to current tip
  dx = x_c - x_tip_curr
  dy = y_c - y_tip_curr
  move = sqrt(dx*dx + dy*dy)
  if move > rmax_step:
    s = rmax_step / move
    x_c = x_tip_curr + s*dx
    y_c = y_tip_curr + s*dy

  return x_c, y_c, θ_c, l
```

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

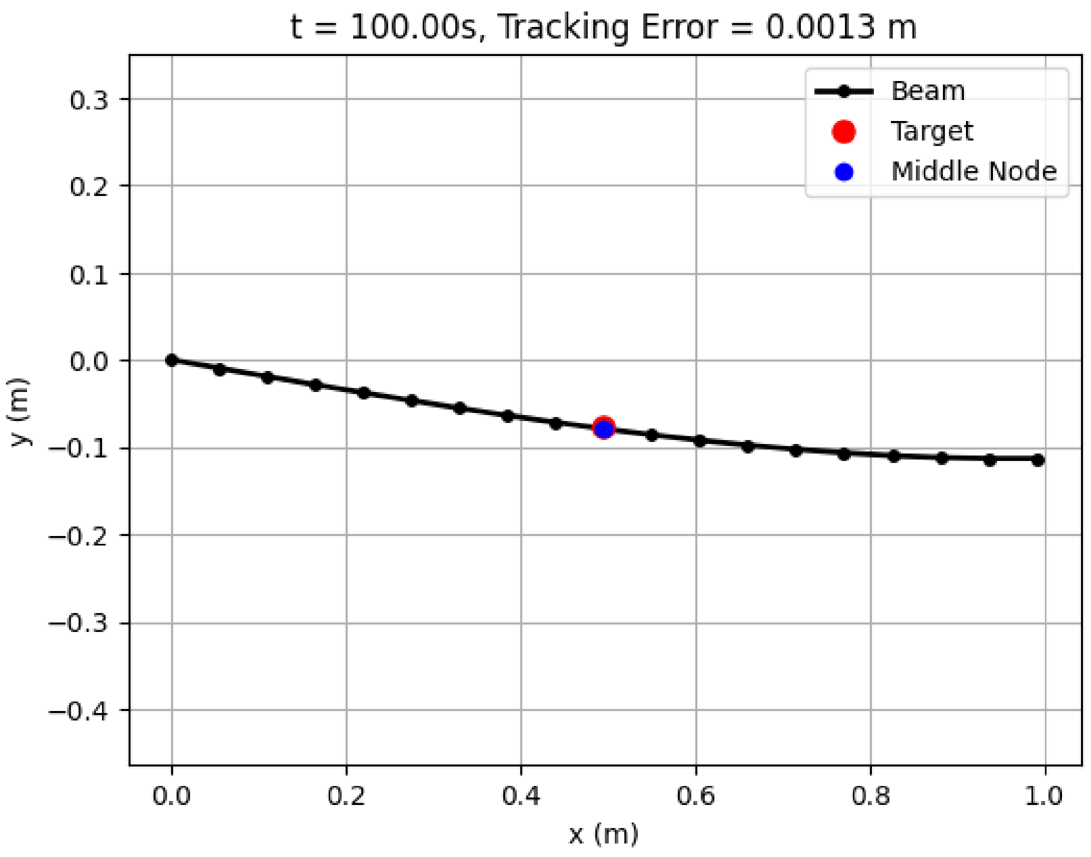


- At least five snapshots of the beam shape (node positions) during the motion

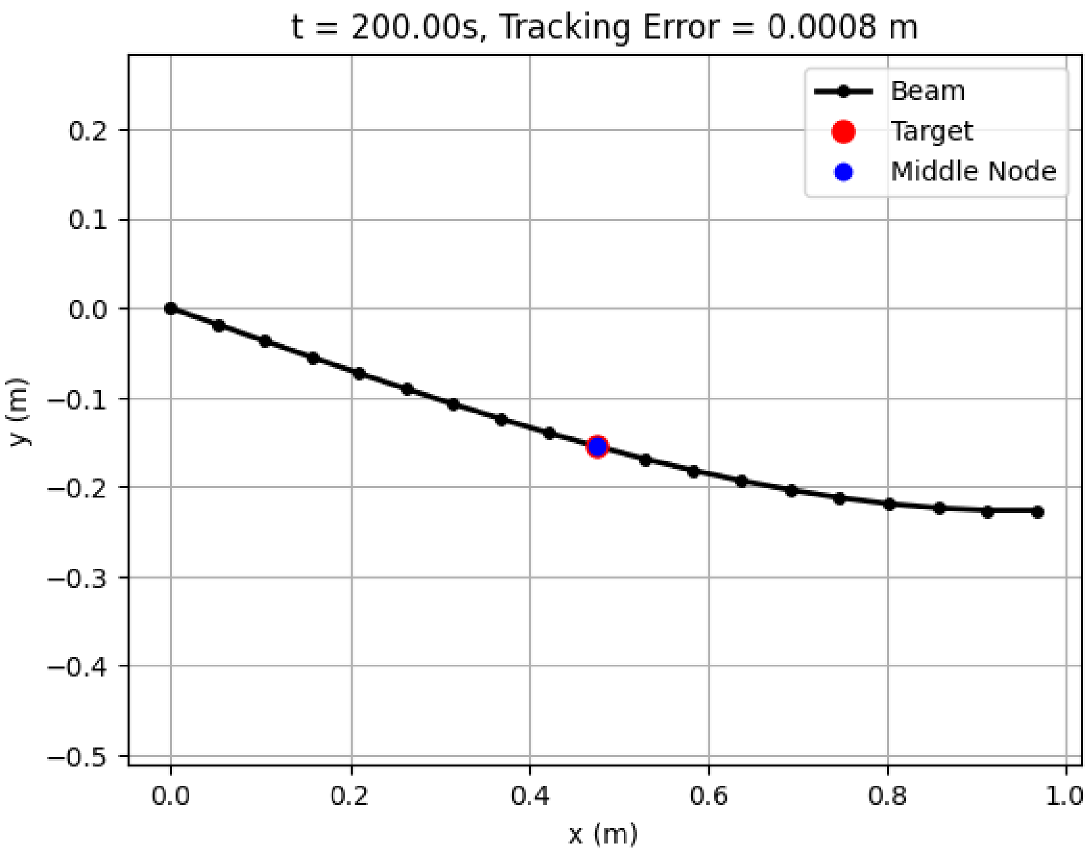




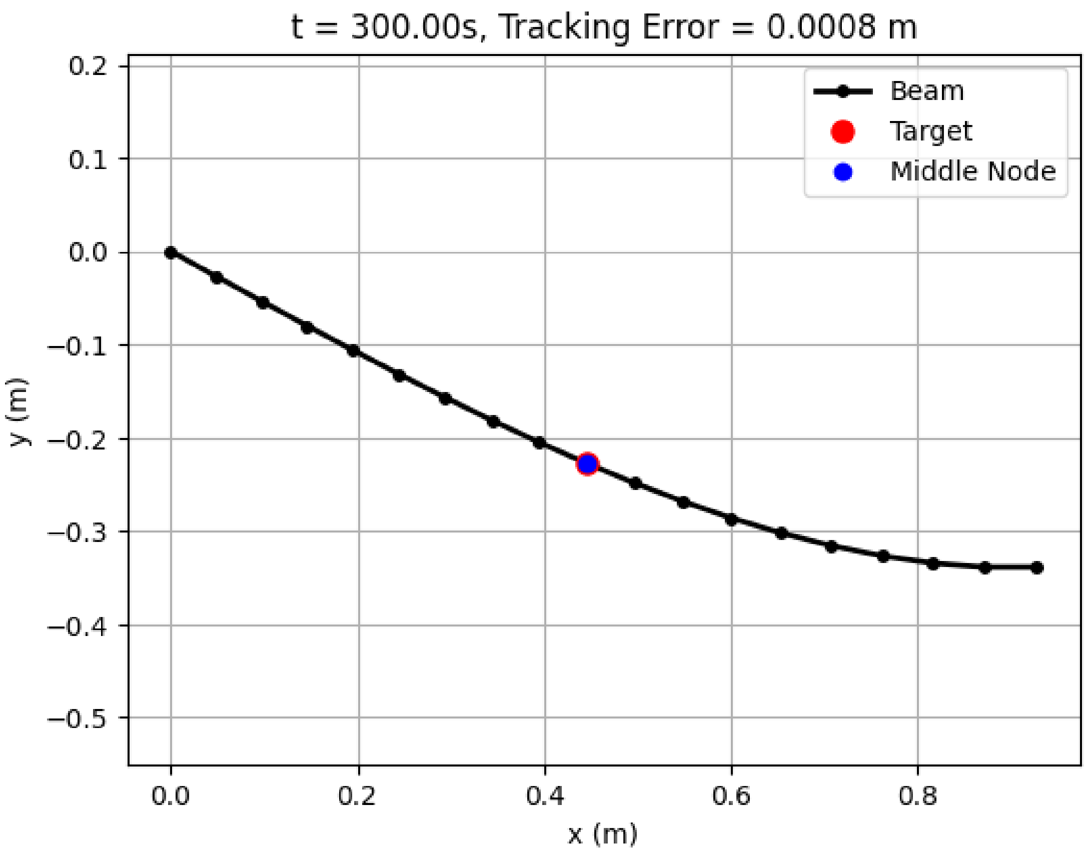
Step 1/1000, t = 1.00s, Err = 0.0000 m



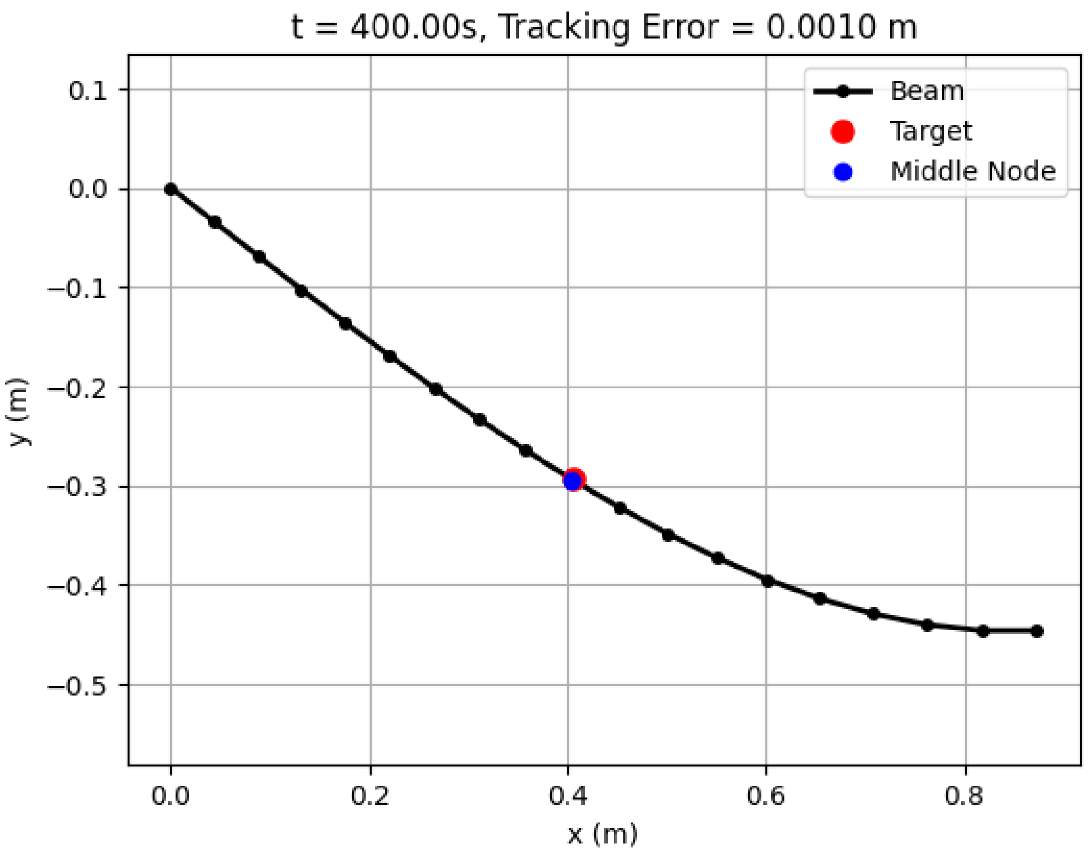
Step 100/1000, t = 100.00s, Err = 0.0013 m



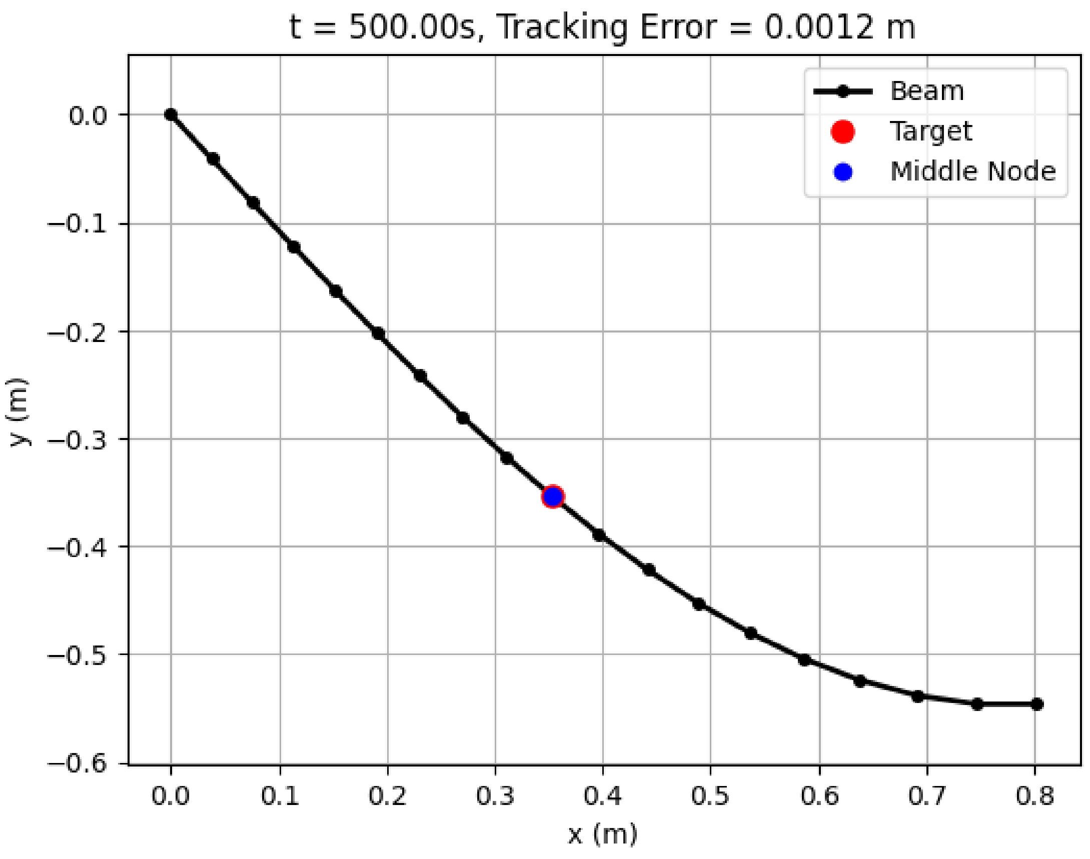
Step 200/1000, t = 200.00s, Err = 0.0008 m



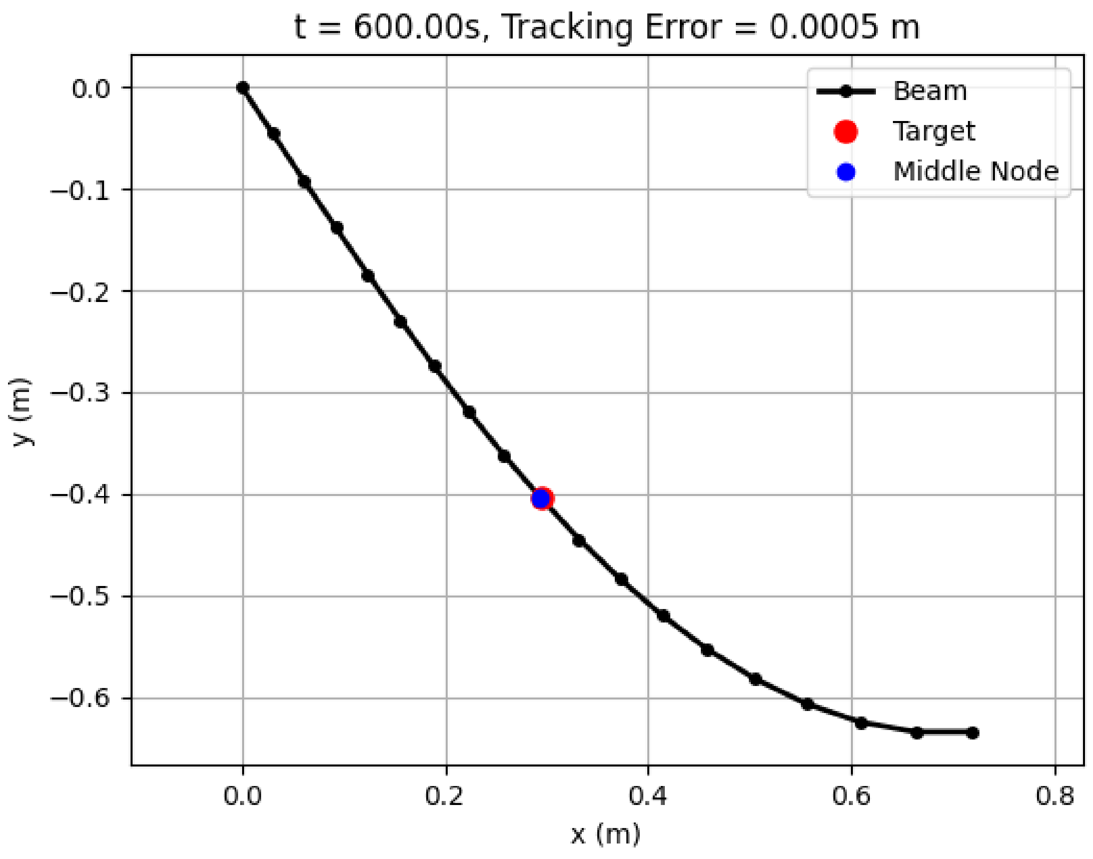
Step 300/1000, t = 300.00s, Err = 0.0008 m



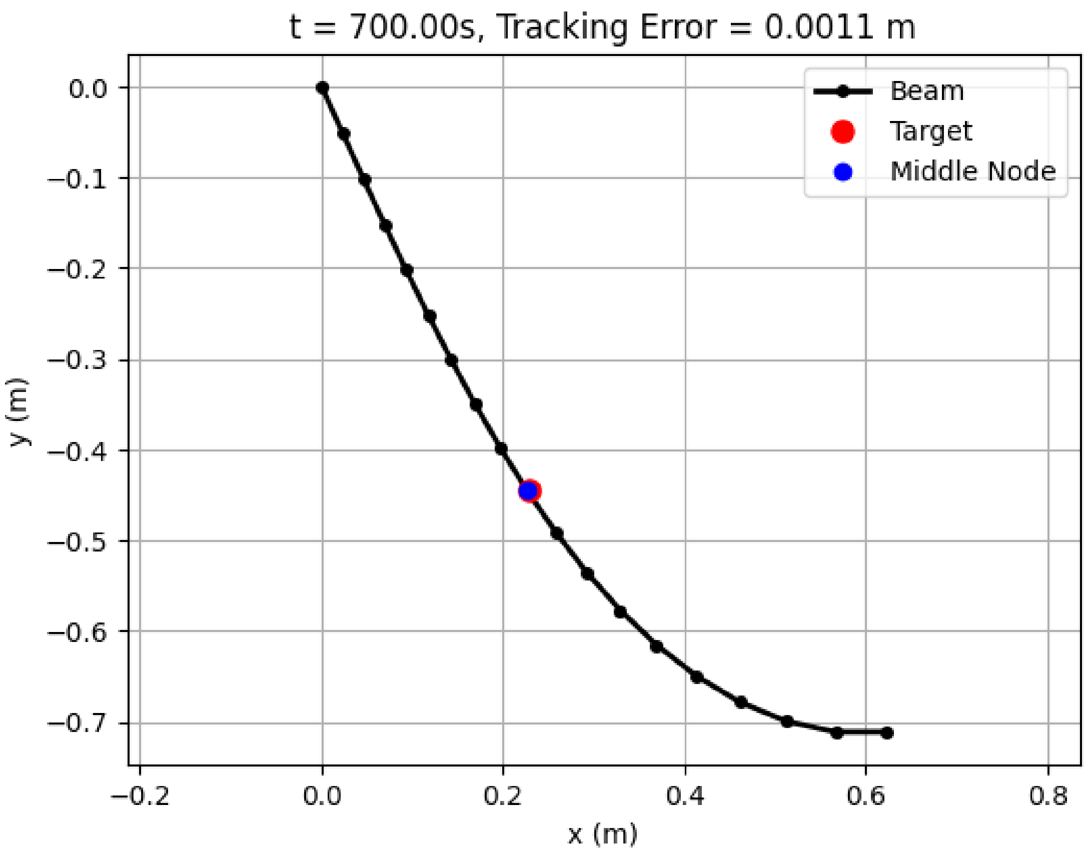
Step 400/1000, t = 400.00s, Err = 0.0010 m



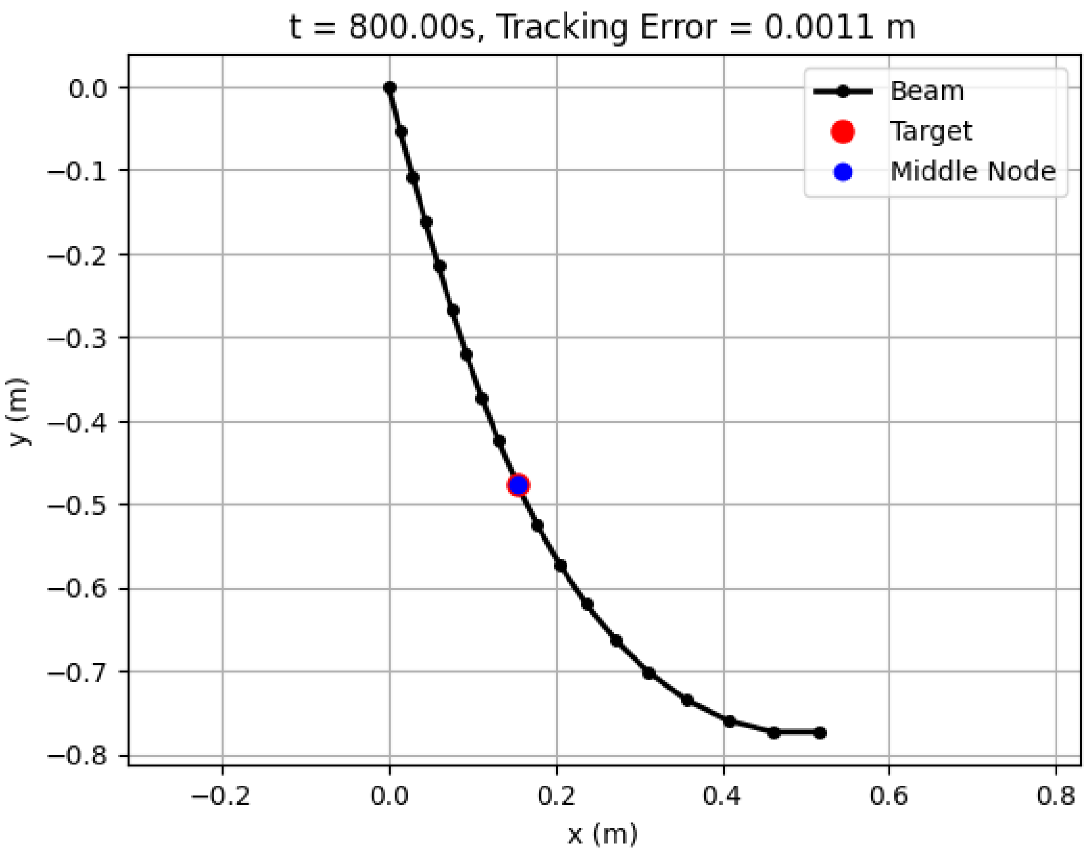
```
Step 500/1000, t = 500.00s, Err = 0.0012 m
```



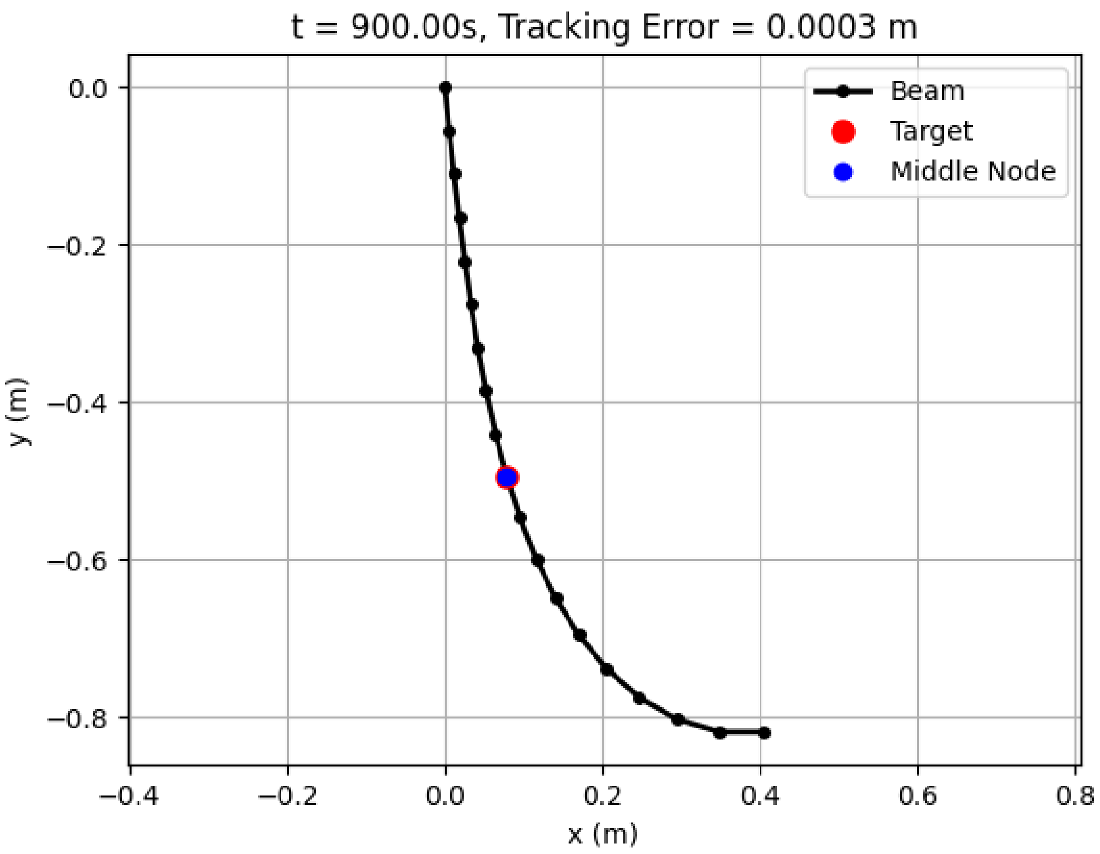
Step 600/1000, t = 600.00s, Err = 0.0005 m



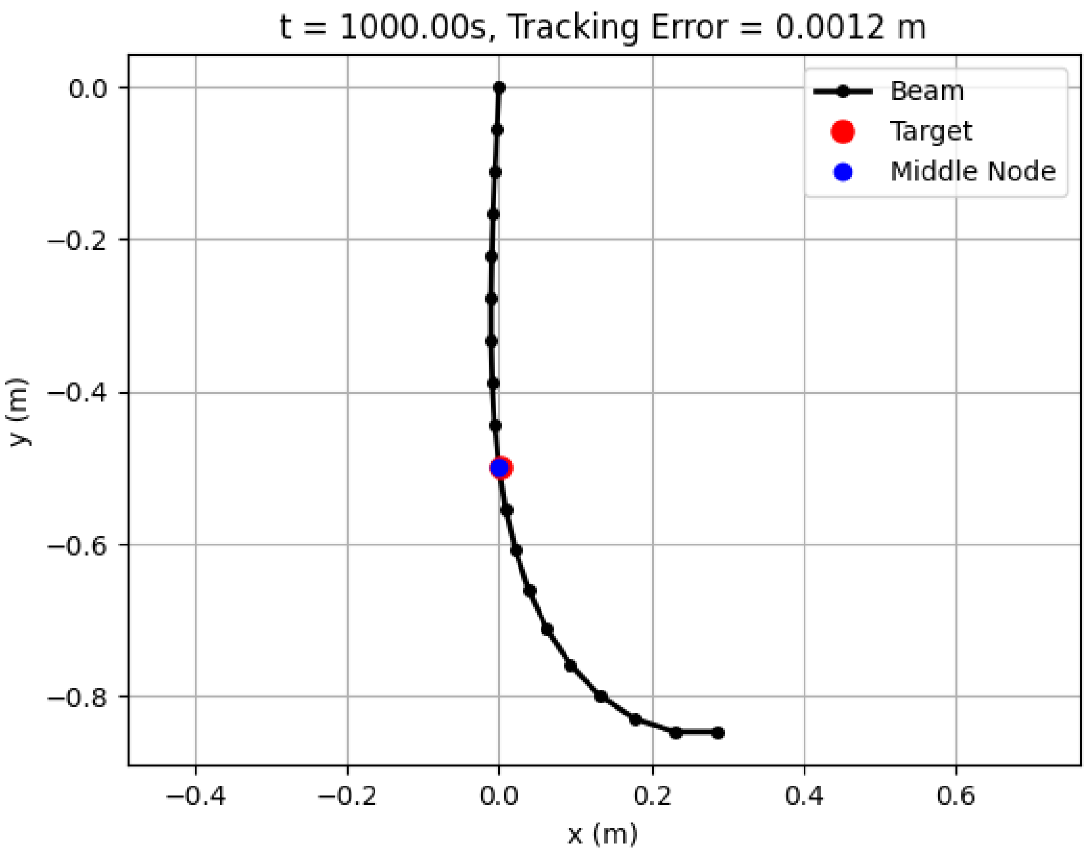
Step 700/1000, t = 700.00s, Err = 0.0011 m



Step 800/1000, t = 800.00s, Err = 0.0011 m



```
Step 900/1000, t = 900.00s, Err = 0.0003 m
```



```
Step 1000/1000, t = 1000.00s, Err = 0.0012 m
```

- 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).

The robot’s feasible motion is limited by its workspace (the rod length and curvature capacity), joint orientation limits, and actuator rate constraints. Commands that move the end effector beyond the rod’s reachable region, require excessive bending, or exceed maximum step size are considered infeasible. To handle these, the controller applies saturation and rate limiting—each command is clamped to a maximum allowable displacement per time step (max_step) to prevent unrealistic or unstable motions. If a command still exceeds workspace boundaries, it is projected back into the feasible region, and the controller’s integral term is frozen to avoid windup. In persistent cases, the reference trajectory can be slowed (re-timed) to maintain smooth and feasible motion within all physical limits.