Homework 3 - Robotic Control of an Elastic Beam

Lovleen Kaur

I. Introduction: Robotic Control of an Elastic Beam

This homework covers the deflection of a beam with discrete representation through spherical masses with springs. The beam is being controlled by a robotic end that moves the last two nodes such that the mid point of the beam follows these location functions:

$$x^*(t) = \frac{L}{2}\cos\left(\frac{\pi}{2}\frac{t}{1000}\right)$$

$$y^*(t) = -\frac{L}{2}\sin\left(\frac{\pi}{2}\frac{t}{1000}\right)$$

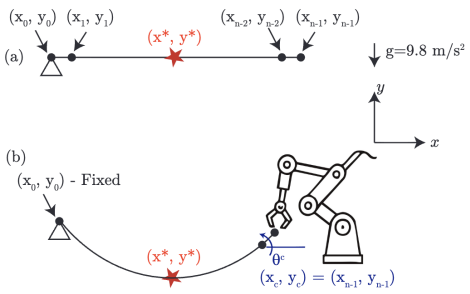

Fig. 1: Robotic Arm and Boundary Conditions

II. Methods:

A. Inputs:

The following are input parameters for the simulation:

Parameter	Value
L	1
R	0.013 m
r	0.011 m
d	0.75 m
g	9.8 $\frac{m}{s^2}$
E	70 GPa

I is the moment of inertia of the cross section,

$$I = \frac{\pi}{4}(R^4 - r^4)$$

The beam can be represented as a mass-spring system with a mass m located at each node, where

$$m = \pi(R^2 - r^2)l\rho/(N-1)$$

The boundary conditions on the robotic node and the node before the last node:

$$x_N(t_{k+1}) = x_c(t_{k+1}$$
$$x_{N-1}(t_{k+1}) = x_c(t_{k+1}) - \Delta L\cos\big(\theta_c(t_{k+1})\big)$$
$$y_N(t_{k+1}) = y_c(t_{k+1}$$
$$y_{N-1}(t_{k+1}) = y_c(t_{k+1}) - \Delta L\sin\big(\theta_c(t_{k+1})\big)$$

B. Force Evaluation::

Simulate the beam as a function of time (between $0 \leq t \leq 1000$ seconds) implicitly with $\Delta t = 0.05$ s for time step of 1s. With N = 19, for each 1s step in the 1000 s time range, the force enforced is the force of gravity that is applied to all the nodes and feeds into the implicit Newton Raphson solver as the external force input.

C. Time Stepping:

1 second is used as the time step on the global scale to each 1000 s but each 1 s is discretized with dt = 0.05 s to implicitly solve the system itself. The is done with each combination of varying $\{x_c, y_c, \theta_c\}$ inputs.

D. Enforcement of Dirichlet Constraints::

The the length between the last and the second to last node is computed at each time step from the previous state. This ΔL is used in enforcing the boundary condition for the x and y positions for the last two nodes using the expressions listed earlier with $\{x_c, y_c, \theta_c\}$ variables

E. Path-planning and Error Tracking

The main parameters that impact the last two nodes are $\{x_c, y_c, \theta_c\}$ and these are tested in a three possible iterations each so totaling a combination of 27 unique combination of $\{x_c, y_c, \theta_c\}$ that is tested every 1 second step. The range of x and y variables is limited to ± 0.01 and the range for

theta is limited to ± 1 rad for each 1 second step. This is to restrict the realistic movement of the robot over time and make small jumps in location so that robot does move an impossible amount over the 1s time span. This approach also helps reduce computation time as only a very small range of combinations need to be tested. In this case, only 3 were tested for each variable so a total 27 combinations were assessed for the lowest error.

The error can be reduced even further if a controller is implemented or by doing an iterative second layer search with in this range towards the number that has the lowest error. The later approach is still computationally expensive without parallelizing the code for each case. Another way to increase accuracy would be to increase the number of nodes along side better control algorithm. For the purposes of this homework, the 27 combinations approach with the limited movement range has yielded good results already, which can be improved upon if very high accuracy is needed. The absolute error in this solution is measured by computing the absolute distance between the simulation mid point and the analytical mid point. The absolute error results presented in this homework remained under 0.02 m at all times and the final state absolute error was 0.005 m.

F. Pseudo code:

```
for timestep 1:1000
    compute x and y midpoint analytical
    positions call the nested loop to
    test combinations
    #for xc,yc,and theta
        for xstep
            for ystep
                for thetastep
                    #Assign the boundary
                    conditions for N and
                    N-1 nodes

                    Call the implicit
                    solver Compute and
                    store error
                end
            end
        end
    compute the lowest error and the index
    store those xc, yc, and theta
    time = time + 1
    Plot beam for select time steps:
end
Plot the xc, yc, and theta results vs time
```

III. Results

This section overs the results of the simulation. The following are the beam shapes at some samples time steps

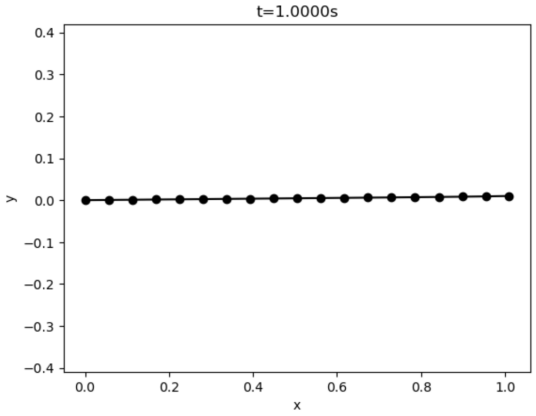

Fig. 2: Beam shape at t = 1

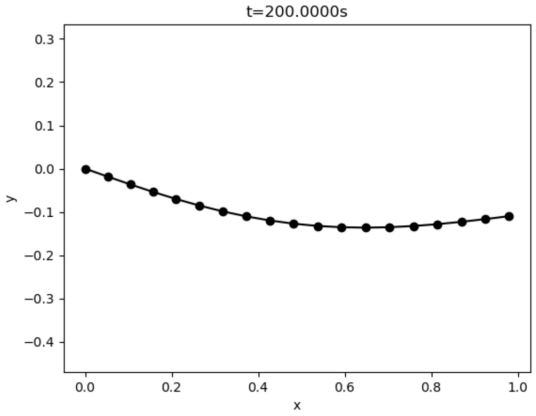

Fig. 3: Beam shape at t = 200

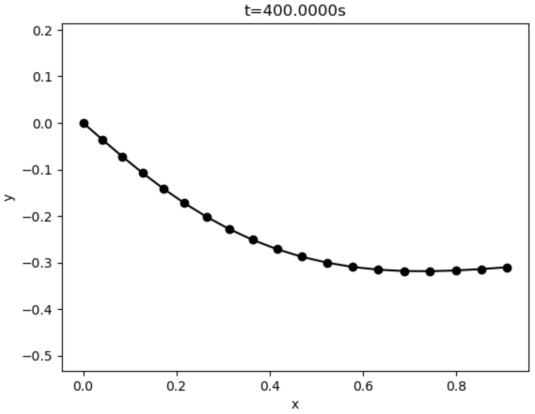

Fig. 4: Beam shape at t = 400

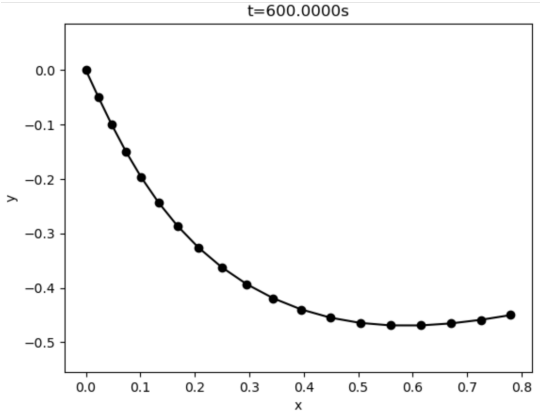

Fig. 5: Beam shape at t = 600

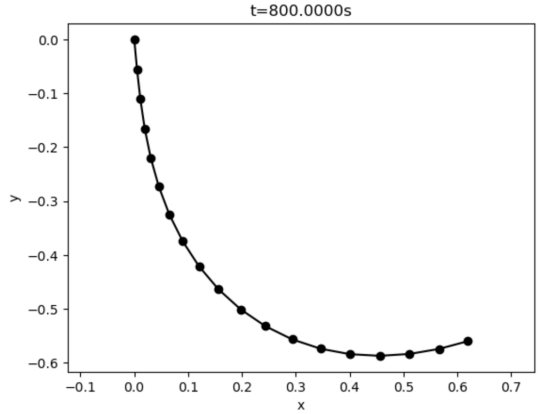

Fig. 6: Beam shape at t = 800

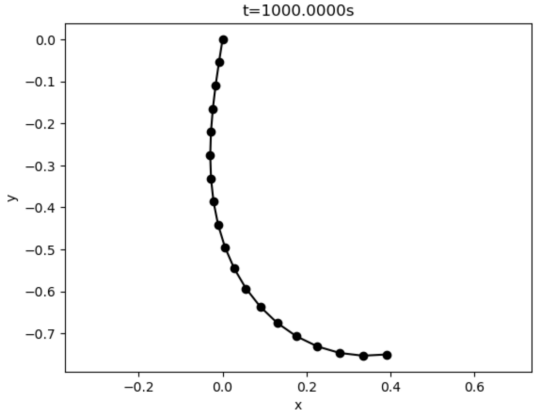

Fig. 7: Beam shape at t = 1000

The beam overtime researches this state with minimal error and without any sophisticated control approach. The following plots show the locaiton of the robotic control nodes and the absolute error from the path.

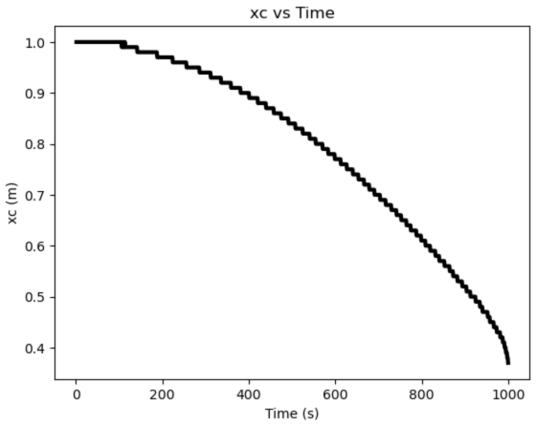

Fig. 8: Robotic boundary condition x_c over time

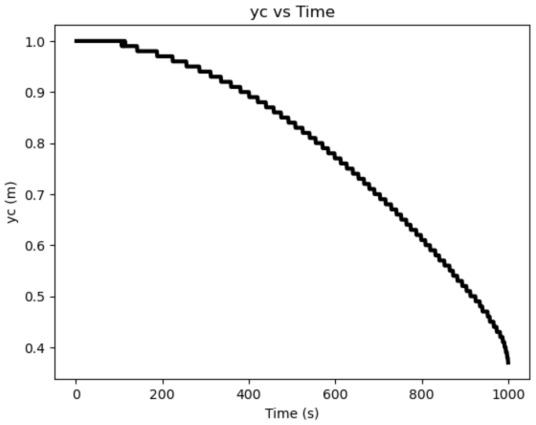

Fig. 9: Robotic boundary condition y_c over time

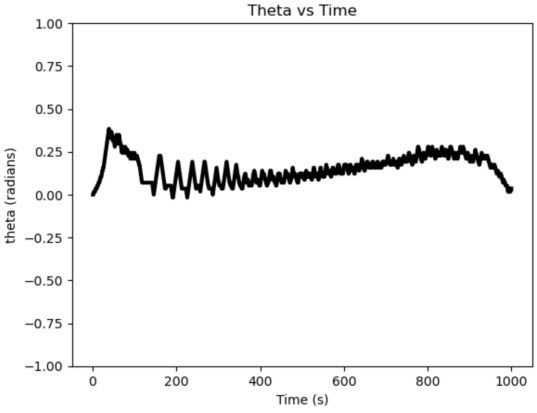

Fig. 10: Robotic boundary condition θ_c over time

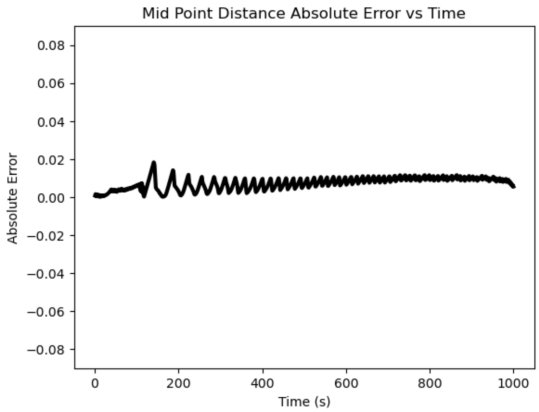

Fig. 11: Absolute Error Over Time

IV. CONCLUSIONS

The limited range allowed for the robot to move over each one second time interval allowed a slow and stable approach to the beam shape reached at 1000 s. This method works well despite having no sophisticated control algorithm. There are many solutions to this problems, a faster approach for the future would be to implement a control approach with feedback instead of running through multiple test cases and only then selecting the lowest error case.

Overall, the methods learned from previous homework and application in this homework yields a good result for this particular problem.