MAE 263F: Homework 01

Pranav Sawant

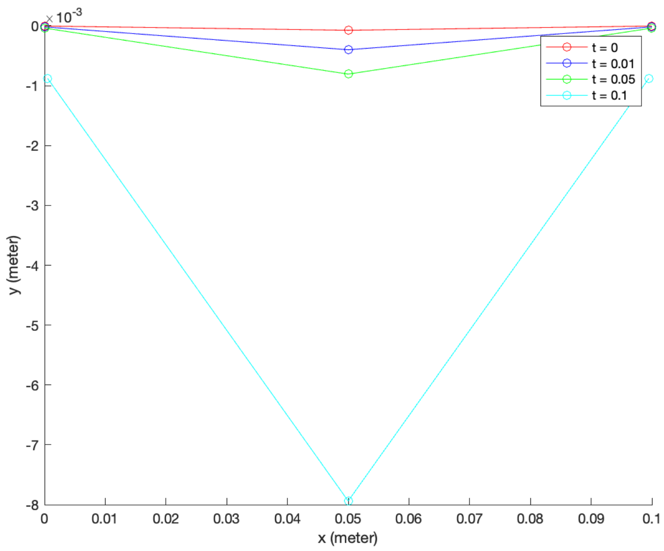
***Abstract*— This document contains the report of the homework 01 for the course MAE 263F taught at UCLA by Prof. M. Khalid Jawed. The homework has three assignments each of which has multiple sub questions. Detailed explanation and answers to those assignment is provided in the document.**

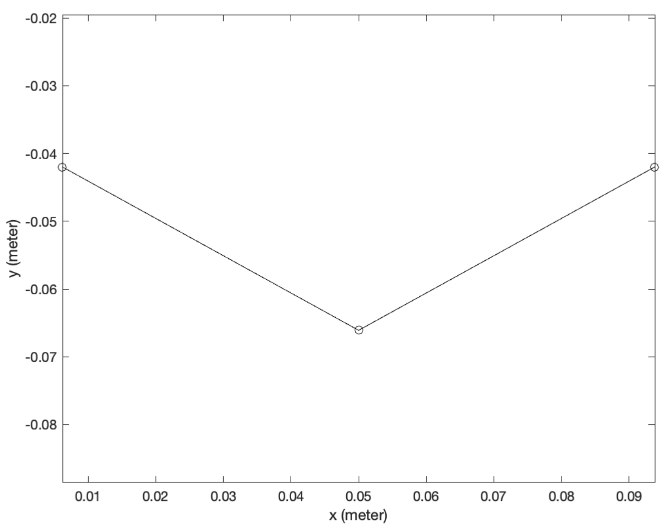
I. ASSIGNMENT 1

The file Problem1.m in the GitHub repository includes MATLAB code for a solver that simulates the sphere's position and velocity over time using both implicit and explicit methods, explicit method is commented. For the implicit method, a time step of $\Delta t = 10^{-2}$ s is applied, while for the explicit method, a time step of $\Delta t = 10^{-5}$ s is used.

A. Shape of the structure

The plots below illustrate the structure's shape at different time points, with the corresponding times indicated in the legend.





From these plots, it's clear that as time progresses, the structure gradually shifts downward while its shape changes, due to varying viscous forces at different nodes. Please note that the axis scales different between the two plots, adjusted to better visualize the structure's form.

B. Terminal Velocity

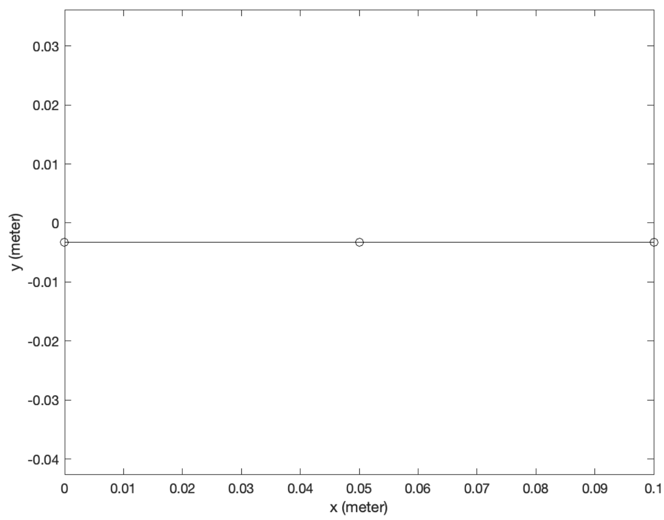
The terminal velocity of the system obtained through the simulation is:

$$v_t = -0.00597 \text{ m/s}$$

The negative sign indicated the system is going downwards.

C. Same Shape

In this section, we look at a case where all the spheres have the same radius. To make this happen, we can set the values of R1, R2, and R3 equal in the code. After running the simulation for a total of 10 seconds, it was noticed that the shape of the structure remains unchanged. This outcome align with our logic that, with equal radii, the viscous forces on all the spheres are equal. As a result, no torque is generated on the elastic rod between the nodes, which is why the structure's shape stays the same. The plot below shows the final shape of the structure.



D. Explicit VS Implicit Method

To answer this question, both the methods are implemented in the MATLAB file. After running the code it was found that the implicit method requires more computation per step and it allows for larger time steps. On the other hand, the explicit method, although it involves fewer calculations per step, needs much smaller time steps compared to the implicit method.

For instance, the explicit method works well with a time step of $\Delta t = 10^{-5}$ s, but if the time step is increased to $\Delta t = 10^{-4}$

s, the simulation quickly becomes unstable. In contrast, the implicit method performs well even with larger time steps, such as $\Delta t = 10^{-2}$s.

In summary, the explicit method is more efficient for short simulation times, but as the total simulation time increases, the number of steps required for the explicit method grows much faster than for the implicit method. Therefore, the implicit method is better suited for longer simulations.

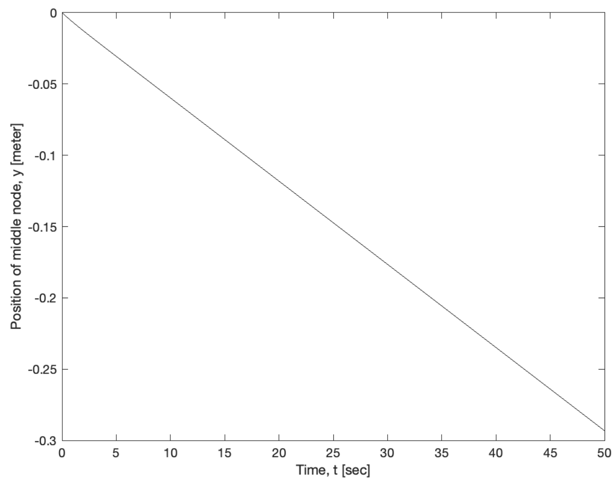
II. ASSIGNMENT 2

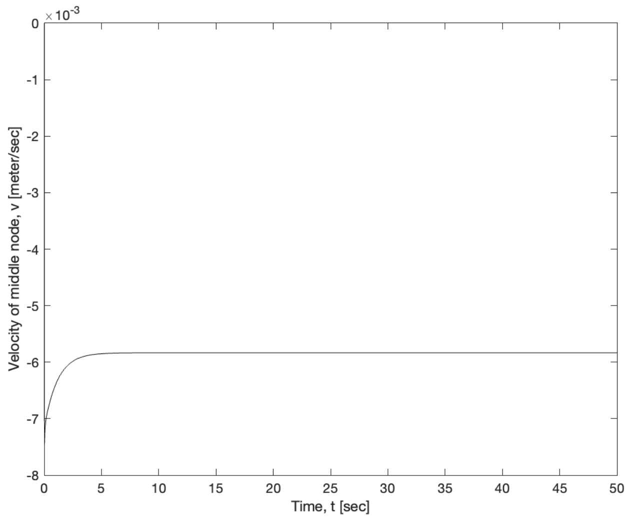
The second assignment is sort of an extension of the first assignment. Here we code for a general case of N spheres connected by N-1 elastic rods. The code for this simulation can be found in the file Problem2.m in the GitHub repository. In this case, the solver was run for a total of 50 seconds. Due to the long duration of the simulation, only the implicit method was used.

A. *Vertical Position and Velocity of the Middle Node*

Below are two graphs, the first one shows the vertical position over time, and the second one shows the vertical velocity of the middle node. From these graphs, we observe that the velocity of the middle node eventually stabilizes at a constant value. This value is the terminal velocity of the middle node, which is:

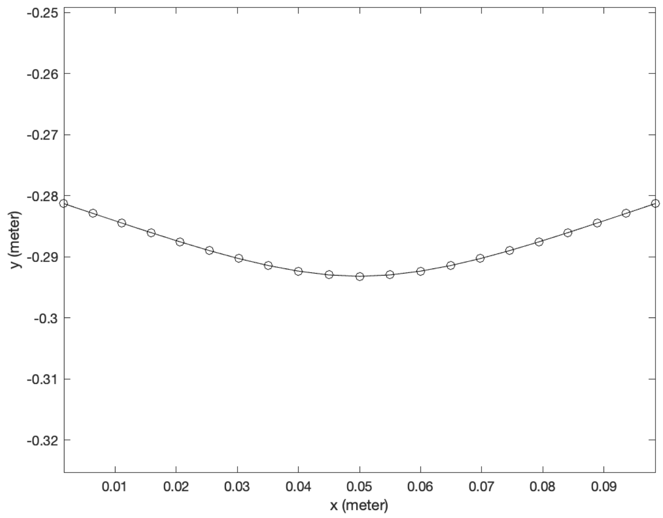
$$v_t = -0.00584 \text{ m/s}$$





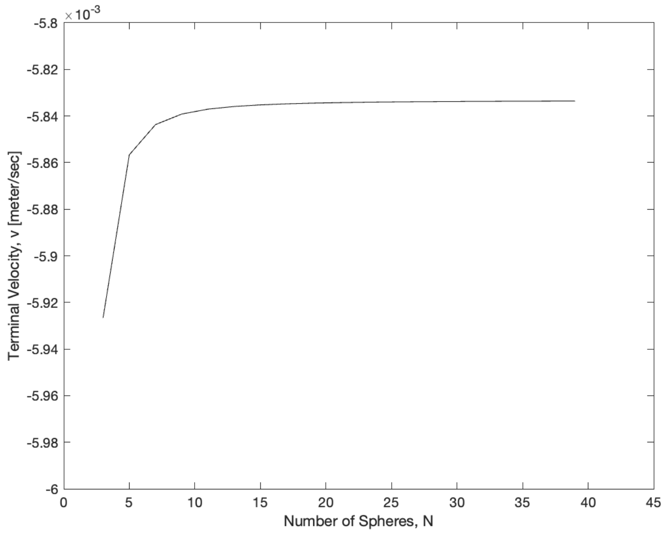
B. *Deformed Shape of the Bean*

After running the simulation for 50 seconds, below is the final deformed shape of the structure.



C. *Spatial and Temporal Discretization*

To explore the significance of spatial and temporal discretization, we analyze how quantifiable metrics like terminal velocity vary with discretization parameters such as N and Δt. Initially, we examine the plots of terminal velocity against the number of spheres and terminal velocity against the time step.



The terminal velocity vs. N graph shows that the simulation struggles when N is small. However, as N increases, the terminal velocity becomes steady and doesn't change much.

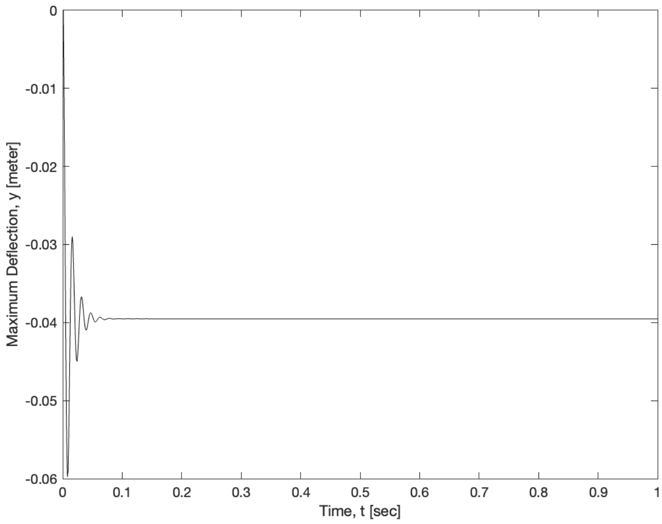
This highlights the importance of spatial discretization. Therefore, we need to choose a large enough value for N to ensure the simulation gives accurate results.

III. ASSIGNMENT 3

The code for this simulation can be found in the file Problem3.m in the GitHub repository.

A. Maximum Vertical Displacement

Below is a graph showing the maximum deflection, y_{max} of the beam over time. We can see that y_{max} is negative, and after some brief oscillations, it eventually settles at a steady value.



The final maximum deflection of the beam from the simulation comes out to be:

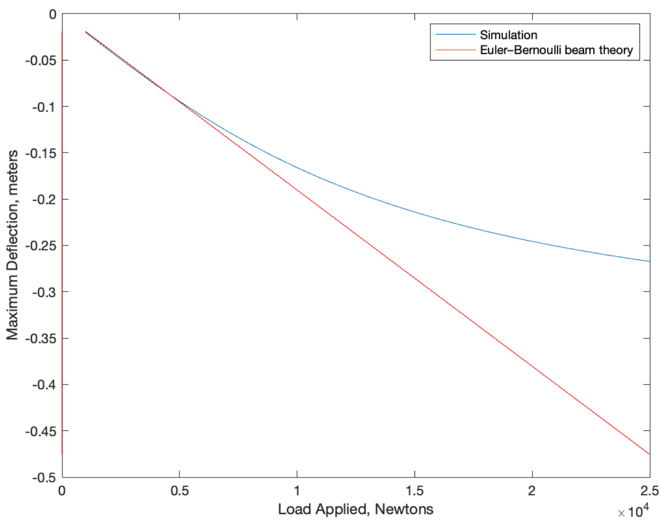
$$\delta max_{sim} = -0.03901 \text{ m}$$

And the theoretical deflection from the Euler-bernoulli beam theory is:

$$\delta max_{EB} = -0.03804 \text{ m}$$

B. Larger Loads

We now run the simulation for different load values P and create a plot showing the maximum deflection δmax against the load P, comparing the results from our simulation with those from the Euler-Bernoulli beam theory. The plot is shown below:



From the graph, we can see that the two solutions start to differ at around 5000 Newtons. The Euler-Bernoulli beam model works well for small deformations. However, as the deformations get bigger, the bending force increases, resisting those deformations. The Euler-Bernoulli model ignores this bending force, but in our simulation, we take the bending energy into account. As a result, our simulation provides more accurate results for larger loads.

References

[1] Professor M. Khalid Jawed, MechAE 263F Course Modules, Link: https://bruinlearn.ucla.edu/courses/193842/modules Portions of this code and the helper functions/files have been referenced from the class resources available on Bruinlearn (link provided above).