Simulation of Cantilever Beam with End Load via Discrete Elastic Beam Theory

Philip Ng

***Abstract*— Beams are used in various applications of engineering, from infrastructure to vehicles. As rigid as they may seem, all beams are elastic; they deform under the loads they are given. The elastic properties of a beam are integral for understanding its various properties, such as its yield point and fracture point. Knowing how a beam will deform in various conditions helps inform engineers whether a structure is safe for use. In this article, we will explore numerical methods based on governing differential equations to generate a simulation of a cantilever elastic beam under an end load. A larger version of such a simulation can be utilized to predict the elastic behavior of beams of various geometries under any external condition.**

I. INTRODUCTION AND THEORY

Let us consider a N degree-of-freedom system acted on by conservative forces only. This is a valid assumption because most elastic structures, including beams, are primarily affected by such forces. In the case of the beam in viscous flow, we also have damping and weight forces acting on the system. Thus the general equation of motion for the i-th DOF would be

$$f = m_i\ddot{q}_i + \frac{\partial E_{potential}}{\partial q_i} + W + P = 0 \tag{1}$$

where $W + P$ are the weight and external force components, respectively. Let us also assume that the beam is a network of connected springs and spheres, like so:

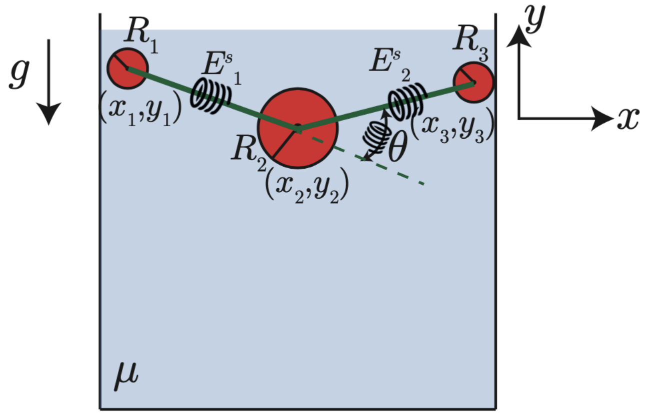

Fig. 1. Beam simplified as network of springs and spheres[1]

We will call each of the spheres *nodes* and the spring between each node *edges*. The angle θ is called the *turn angle* of the system. There are $a = N/2$ nodes and $b = N/2 - 1$ edges. Fig. 1 depicts a 3-node system with gravity oriented in the -y direction. The symbol E^s_i represents the elastic potential energy due to *stretching*, which occur at each edge. E^s_i represents the elastic potential energy due to *bending*, which occurs at each node except for the end nodes. The total elastic potential energy is summarized by

$$E_{potential} = \sum_{a=1}^{n} E^s_a + \sum_{e=1}^{n} E^b_b e \tag{3}$$

where a is the number of nodes and e is the number of edges.

To prepare for simulation, we must discretize (1). Below is the discretized equations for the i-th DOF using an implicit method. We will be using both in our simulation and comparing the results. (3) is an implicit equation,. Implicit equations are harder to program but converge at larger time-steps and thus require less time to compute.

$$\frac{m_i}{\Delta t^2}[\frac{q_i(t_{k+1} - q_i(t_k)}{\Delta t} - \dot{q}_i(t_k)] +$$

$$\frac{\partial E_{potential}}{\partial q_i} - q_i(t_k)\Delta t + W + P = 0 \tag{3}$$

Note that W is only in effect for degrees of freedom in the y-direction. We can now apply the discretized equations to vectors and matrices. Since the current problem is in 2-D, we can set up the following N-DOF vector, **q**:

$$\mathbf{q} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_a \\ y_a \end{bmatrix}$$

For the mass component, we will use an N X N diagonal lumped mass matrix **M**. The components m_{ii} represent the mass of node *i*. Note a represents the number of nodes.

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 & \dots & 0 \\ 0 & m_{11} & 0 & \dots & 0 \\ 0 & 0 & m_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & m_{aa} \end{bmatrix}$$

The weight matrix **W** only acts on y-oriented DOFs, so it is zero for all odd indices.

$$\mathbf{W} = \begin{bmatrix} 0 \\ W_1 \\ 0 \\ W_2 \\ \vdots \\ 0 \\ W_a \end{bmatrix}$$

With these matrices defined, the governing equation (1) becomes

$$\boldsymbol{f} = \boldsymbol{M}\ddot{\boldsymbol{q}} + \frac{\partial E_{potential}}{\partial \boldsymbol{q}} + \boldsymbol{W} + \boldsymbol{P} = 0 \tag{5}$$

To solve (5) explicitly, simple algebraic manipulation can be used. The implicit method will require Newton Rhapson Iteration, which further requires calculation of the Jacobian. The Jacobian is the gradient of the governing equation, which yields the discretized equation

$$\boldsymbol{J} = \frac{\boldsymbol{M}}{\Delta t^2} + \frac{\boldsymbol{C}}{\Delta t} + \frac{\partial^2 E_{potential}}{\partial^2 \boldsymbol{q}} \tag{6}$$

where

$$\frac{\partial^2 E_{potential}}{\partial^2 \boldsymbol{q}} = \sum_i^n \frac{\partial^2 E_i^s}{\partial^2 \boldsymbol{q}} + \sum_i^n \frac{\partial^2 E_j^b}{\partial^2 \boldsymbol{q}} \tag{7}$$

The gradients and Hessians of the potential energies can be calculated using the functions `gradEs`, `gradEb`, `hessEs`, `hessEb` in the Appendix. Using the vectorized equations (5) and (6), we can generate a simulation of a beam with N nodes with viscous and/or external forces acting open the system.

II. METHODOLOGY

A. Initial Parameters and Matrices

For proof of concept, this simulation involves a rectangular beam with aluminum spheres as nodes. We first set the simulation size to N = 3. Because this is a 2-D problem, there will be a total of 2N = 6 DOFs. Next, we define the necessary physical parameters: the density of the beam, and gravity. The beam's physical and geometrical parameters should also be defined: Young's Modulus, cross sectional area, and aerial moment of inertia. We also set the simulation parameters, such as time-step dt and the end-time of the simulation. With the initial parameters initialized, we can generate the radius vector, **M**, **C**, **W**, **P**, and **q**.

M is a matrix of size N X N with every two diagonal components equal, since both elements correspond to the same node. We generate this by doing a `for` loop from DOF i to N, and setting both `m(2*i-1,2*i-1)` and `m(2*i,2*i)` to the same mass using (8).

W is a weight vector of size N X 1, and can be populated using a `for` loop. Only the even-index components are populated by $W_a = m_a g.$, where a is the node index. Thus we can loop from DOF i to N, setting `W(2*i)` to weight in (9) and all other components to 0.

P is an external force vector of size N X 1, whose components are all 0 except at the node the force is applied, where its value is equal to P_y. We can find the node of action using the formula `nodeOfAction = round(c / l * (N-1))` where c is the distance along the beam at which P_y is applied, and l is the length of the beam.

Finally, the initial DOF vector **q0** is a N X 1 vector, where each even-index element q_i represents the y-position of a node, and each odd-index element the x-position of a node. In this problem, all y-components are set to 0 initially, while the x-compoennts are set to $l/(N-1)$, where l is the length of the beam. Thus the nodes are equally spaced along the beam. Thus a loop can be made from DOF i to n setting `q0(2*i-1) = l / (N-1)` and `q0(2*i) = 0`. We will also initialize a velocity vector **u0** for later analysis.

B. Boundary Conditions

For a cantilever beam, certain boundary conditions must be imposed. The beam is fixed at one end and free at the other, so we must fix the position of the first node. However, we must also fix the orientation of the first edge; if not, the external force will cause the entire beam to fall to vertical. To do so we must not only fix the position of the first, but also the *second* node. We can define fixed and free indices for the entire node network. Because each node has an associated x and y position in the DOF vector **q**, we must fix the first 4 indices and leave the remaining indices free:

```
fixed = 1:4;
free = 5:2*N;
```

When Newton Rhapson iteration is applied to the force and Jacobian matrices, only the free indices will be updated. This is done by defining an intermediate container `qFree = q(free)` and `JFree = J(free,free)`.

C. Implicit Method

To begin the implicit method, we set the timestep to $dt = 10^{-2}$ seconds, an ending time `maxTime`, and a variable measuring the number of time steps `steps = round(maxTime / dt)`. To measure the velocity of the middle node we create a variable `midNodeVelocity` with length `steps`. Finally, we set a force-dependent tolerance `eps = E*I/l2̂ * 1e-3` for the Newton-Rhapson method and an initial error `err = 10 * eps`. Now we can proceed with the time-marching scheme.

The pseudocode for the Newton-Rhapson iteration is as follows:

Algorithm 1 Implicit Iteration via Newton-Rhapson

```
1:  for every timestep from 0 to endtime do
2:      Update old DOFs and velocities
3:      Redefine error
4:      while error > tolerance do
5:          Define function f without elastic energy
6:          Define Jacobian J without elastic energy
7:          Update gradient and Jacobian of linear springs
8:          for all DOFs do
9:              get gradients using gradEs and hessEs
10:             Add results of gradEs to f
11:             Add results of hessEs to J
12:         end for
13:         Update gradient and Jacobian of bending springs
14:         for all DOFs do
15:             get gradients using gradEb and hessEb
16:             Add results of gradEb to f
17:             Add results of hessEb to J
18:         end for
19:         Update DOF vector and error
20:         Update q: $q = q - J^-1 f$
21:         Update error: err = absolute sum of f
22:         Update middle node velocity
23:     end while
24:     Store new DOF vector and velocities
25:     plot the beam positions in real time
26: end for
```

Because the results of `hessES` and `gradES` are 4 X 4 matrices, they can only take four DOF components as parameters. To ensure proper matrix addition, we use the following indices:

```
f(2*i-1:2*i+2) = f(2*i-1:2*i+2)
+ dF; J(2*i-1:2*i+2,2*i-1:2*i+2) =
J(2*i-1:2*i+2,2*i-1:2*i+2) + dJ;
```

For `hessES` and `gradES`, which are 6 X 6 matrices, we choose these six DOF components as parameters:

```
f(2*i-3:2*i+2) = f(2*i-3:2*i+2)
+ dF; J(2*i-3:2*i+2,2*i-3:2*i+2) =
J(2*i-3:2*i+2,2*i-3:2*i+2) + dJ;
```

Because the implicit method is computationally efficient, we can plot the beam position as it changes.

III. PROBLEM STATEMENT

The methodology discussed previously is applied to an aluminum cantilever beam with a point load applied at its free end. The physical parameters of the beam are as follows: the beam has a density of $\rho = 2700 kg/m^3$, elastic modulus $E = 200 \times 10^9 GPa$, length $l = 0.1m$, base $0.01m$, and height $0.002m$. The downward external force has magnitude $P_y = 10N$.

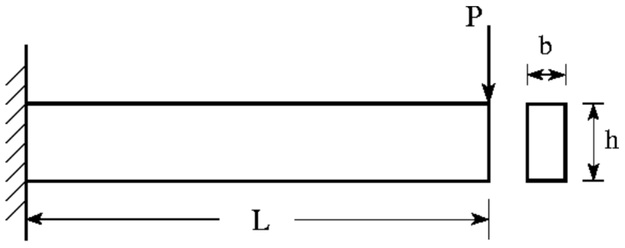

Fig. 2. Schematic of the specified problem.

Using discrete elastic beam theory, we plot the deflection of the beam under the specified load, then examine the relationship between the maximum delflection y_{max} and varying external forces P_y from 1 to 100N.

IV. RESULTS

A. External Force Case

The following figures depict the maximum displacement y_{max} of the beam at a force of $P_y = 10N$ over time.

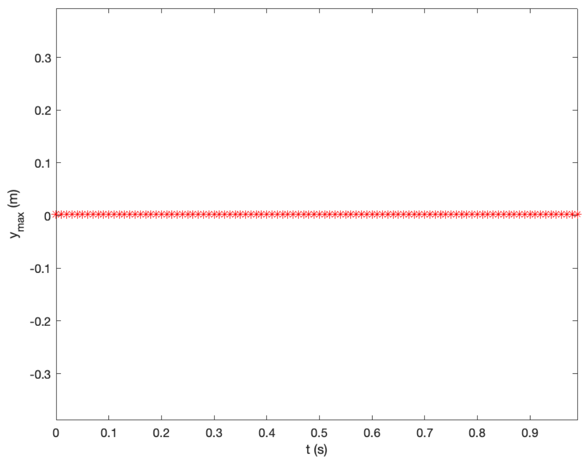

Fig. 3. Variation of y_{max} over time at $P_y = 10N$

The subsequent figure depicts the discrepancy between the Euler-Bernoulli solution and our numerical solution for y_{max} at values of P_y from 1N to 100N.

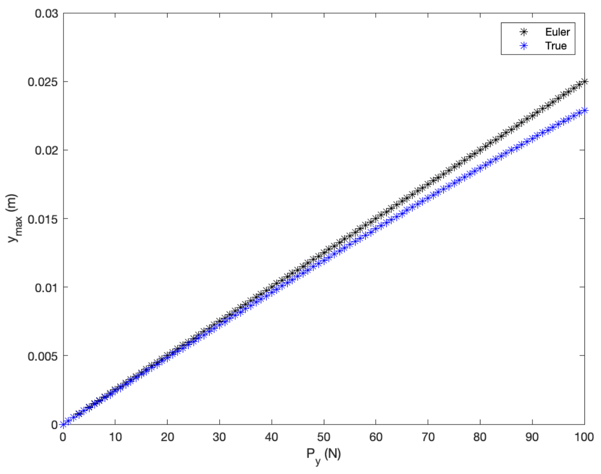

Fig. 4. Relationship between Euler-Bernoulli solution and numerical solution for y_{max} as a function of external force. The blue curve is the numerical solution. P_y

V. DISCUSSION

The maximum vertical displacement of the beam is given by Euler-Bernoulli Beam Theory:

$$y_{max} = \frac{PL^3}{3EI} \tag{13}$$

As shown in Fig. 3, with an external force of $P_y = 10N$, our simulation yielded a fairly steady maximum deflection of y_{max} of 2.46×10^-3m, compared to the Euler-Bernoulli theoretical value of 2.5×10^-3m. Although the maximum displacement calculated by Euler Bernoulli Beam Theory is fairly accurate at this specific value of P_y, further simulation showed deteriorating accuracy for larger external forces, as observed in Fig 4. Around $P_y = 50N$, the solutions begin to diverge, reaching a percent difference of nearly 10% by $P_y = 107N$. Our simulation is more robust for larger deformations because the assumptions Euler-Bernoulli Beam Theory only hold for small deformations. Euler-Bernoulli theory assumes that the beam is linearly elastic, when in reality materials exhibit nonlinear behaviors at higher deformations.

REFERENCES

[1] K. Jawed, Discretized Structures Notes.Structures Computer Interaction Lab. 2023.