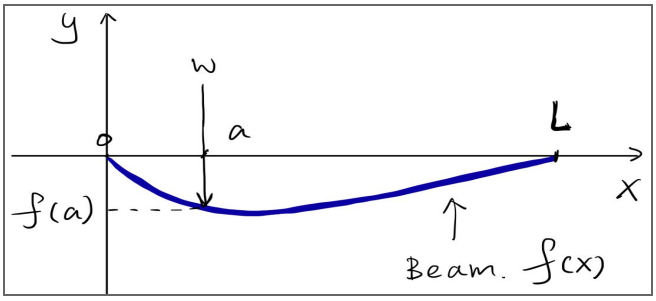
Minimizing Energy to Describe the Shape of an Elastic Beam

Shawn Wu 4338109

MOTIVATION & PROBLEM STATEMENT

Beam deflection is a classic engineering problem. The mathematics that precisely calculate how the structure deforms under stress gave rise to the architectural miracle during the First Industrial Revolution. In the same spirit, we attempt to recreate the math that describes how a simply supported beam changes under stress. The diagram below lays out the setup. We assume a point mass with weight W is applying a force on a uniform cylindrical beam of length L at position a. This beam is fixed at both ends and every point on the beam only moves vertically. The goal of this project is to find a function $f(x)$ that best models the shape of a beam both analytically and numerically. Notice that in this setup, $f(x)$ also denotes the deflection of the beam from its initial position for every points on the beam. For example, $f(a)$ means that the point $(a,\ 0)$ on the unbend beam has deflected by that much due to the loas mass.



The idea is that in this conservative structural system, eventually both the beam and the load mass are going to be at rest, reaching a stable equilibrium. And that is where the potential energy of the system is at its minimum. The potential energy, which is a combination of strain energy due to bending minus the work done by load mass over $f(a)$ is captured by the energy functional (Eq. 1). This means that the $f(x)$ we need is the $f(x)$ that could minimize the energy functional.

We first tried to give an analytical solution. We rewrote $f(a)$ as a δ-function. Then, we applied Euler-Lagrange equation (Eq. 2) to find the necessary condition on $f(x)$. This gives us an ordinary differential

$$\mathcal{E}[f] = \frac{EI}{2}\int_0^L (f''(x))^2 dx - Wf(a)$$
$$= \frac{EI}{2}\int_0^L (f''(x))^2 dx - W\int_0^L f(x)\delta(x-a)dx \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial f} - \frac{d}{dx}\left(\frac{\partial \mathcal{L}}{\partial f'}\right) + \frac{d}{dx}\left(\frac{\partial \mathcal{L}}{\partial f''}\right) = 0 \quad (2)$$

where, $\mathcal{L}(x, f, f', f'') = \frac{EI}{2}(f''(x))^2 + Wf(x)\delta(x-a)$

$$f^{(4)}(x) = -\frac{W}{EI}\delta(x-a) \quad (3)$$

$$f(0) = f(L) = 0$$
$$f''(0) = f''(L) = 0 \quad (4)$$

$$\begin{cases} f(x) = -\frac{Wbx}{6EIL}(L^2 - x^2 - b^2), & \text{if } x < a \\ f(x) = -\frac{Wav}{6EIL}(L^2 - v^2 - a^2), & \text{if } x \geq a \end{cases} \quad (5)$$

for $b = L - a$, and $v = L - x$.

equation (Eq. 3) regarding $f(x)$. By letting the directional derivative of the energy functional equal to zero, we were able to deduce the boundary conditions (Eq. 4) for function $f(x)$. Their physical interpretation is that at the ends of the beam, both the deflection and the bending moment is zero. Therefore, we eventually obtained an unique solution (Eq. 5) for any load mass placed at any location on the beam. Notice that the antiderivative of a δ-function is a θ-function, which explains the piecewise property of $f(x)$. Next, we attempt to numerically approximate this function.

METHODS USED

The general steps we took to numerically approximate $f(x)$ are as followed. By above setup, we first need to locate three points: two fixed endpoints (0,0), $(L,\ 0)$ where L = length of the beam, and a third point denoting the position of the load mass $(a,\ d)$, where d is the deflection of the beam measured from x-axis by a load mass initially placed at position $(a,\ 0)$. Here, d is our variable input to approximate $f(x)$, while a is just a constant that we choose. We then performed the natural cubic spline interpolation through these three points to approximate the shape of the beam. The algorithm for natural cubic spline outputs the coefficients for the piecewise cubic polynomial. Using these coefficients, we then implemented a python function, using the 'Symbol' function from SymPy, that would input our coefficients and spit out the interpolating polynomial with the "x" variable inside it that we could later substitute for a value. We then numerically calculated the potential energy with respect to the interpolating polynomial. The idea is that by varying d, we could locate a interpolating polynomial that would make the potential energy minimum within certain accuracy, and that is the approximated solution for $f(x)$. Now, let's explain in details of those steps.

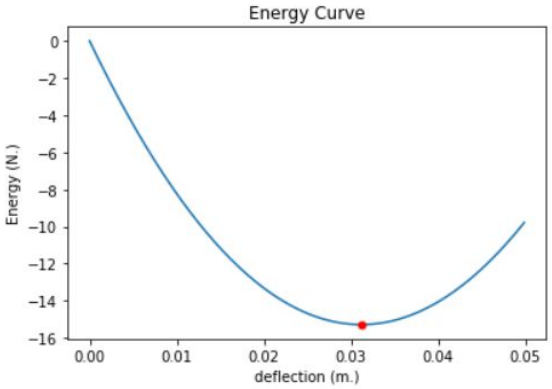
The reason we chose natural cubic spline is because it fits our assumptions. In our scenario, both the second derivative and the value of $f(x)$ is zero at the endpoints, which satisfies the boundary condition of the natural cubic spline. And at the third point, the value, the first and second derivative of $f(x)$ are the same whether to the left or the right of the interpolating polynomial, which fits the gluing conditions. This also means that the final interpolating polynomial is the piecewise polynomial through natural cubic spline. Thus, natural cubic spline had greatly reduced the dimension of our problem.

$$f'(x) = lim_{h \Rightarrow 0} \frac{f(x+h) - f(x-h)}{h}$$

$$f''(x) = lim_{h \Rightarrow 0} \frac{f'(x+h) - f'(x-h)}{2h}$$

In calculating the potential energy by the energy functional, we had decided to defined the derivative and integration function ourselves. To find the second derivative, we used the definition of the derivative, but replaced the $f(x)$ term with $f(x-h)$ so as to increase the accuracy of the result. To yield a second derivative, we used $f'(x)$ terms in the numerator, shown in the image provided. Of course, our derivative function does not calculate the value in exact. The function have an absolute error of $\frac{1}{100{,}000}$.

For the integral, we decided to apply gaussian quadrature for the maximum accuracy. Since our spline would be returning two cubic polynomials, we used 2-point gaussian quadrature because it perfectly integrates polynomials up to cubic degree. With these methods, we started to implement an energy python function that outputs the potential energy of the beam given the various assumptions and calculations we made previously. Before moving on, we had tested all of these functions on simpler problems where we knew the answers to ensure that our algorithms were computing correctly.



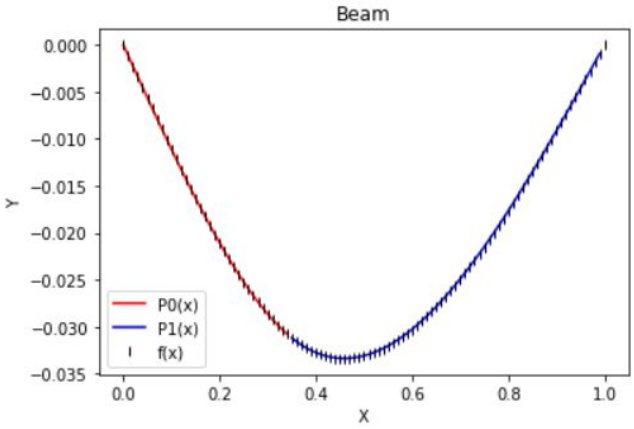
Once we had made sure that this energy function was working, we can now found the approximated $f(x)$ by finding the d that would make the potential energy minimum. We also graphed the potential energy verses d, and indeed there exists a minimum point based on the concave shape of the energy curve. Therefore, we designed an algorithm to find that minimum point. The finding minimum algorithm works in two steps.

Step 1: The finding minimum algorithm first initializes a list of coordinates (deflection, energy), where each item in the list has a larger deflection input than in the item before it. We let the deflection values start at 0. The algorithm then scans the potential energy values in that list, and stops when it detects an energy value larger than the one in the previous coordinate. When this occurs, we stored that deflection value which corresponds to this larger energy value in a variable "D1".

Step 2: The algorithm repeats Step 1, but starts looping at a slightly smaller deflection value than D1 and the step size in the deflection values is 100 times smaller than that in Step 1. By reducing the step size, we were able to make a better approximation. After completing Step 1 with modified initial deflection and deflection step size, the algorithm returns the coordinates that have the energy input right before it starts increasing. These are the coordinates which approximate the minimum of potential energy.

From these coordinates, we have found the d that corresponds the interpolating polynomial which describes the beam after bending. Now, let's see the results.

RESULTS



On the graph to the right, the red and blue line are the interpolating polynomials through the points (a, d), and two endpoints. Here, d is the deflection value we had found using above algorithm. And the vertical dash lines are the true deflection values for every points on the beam using our analytical results (Eq. 5). We can see that they matched up very nicely. We also computed the deflection value at

position a and compared that with $f(a)$ using Eq. 5. We had screenshot the result below. The absolute error is within $\frac{1}{10000}$, which is very likely due to the derivative function we defined

```
The analytically derived value of deflection for this beam problem is: 0.03122997111480312

The deflection from our calculations:  0.031197999999999965
```

earlier.

In summary, we think that we have successfully answered our motivating question that given the physical properties of a beam and a given load mass/position, we can find a function $f(x)$ which best describes the shape of the beam after it comes to a stop, analytically and numerically. Our numerical results matched up with theory very nicely. It is interesting that the analytical solution for the shape of the beam is actually just a cubic piecewise polynomial, which is exactly what natural cubic spline would output. This is probably one of the reasons why our numerical results are very closed to the analytical answers.