Lab III

Benjamin Coveler

Northwestern University

Civil and Environmental Engineering 216-0

11 May 2023

Contents

List of Figures

List of Tables

1 Problem Statement

Write a program to find the necessary length and cross-sectional dimensions of the beam in Figure 1, given maximum allowable normal and shear stresses and various values for the applied loads.

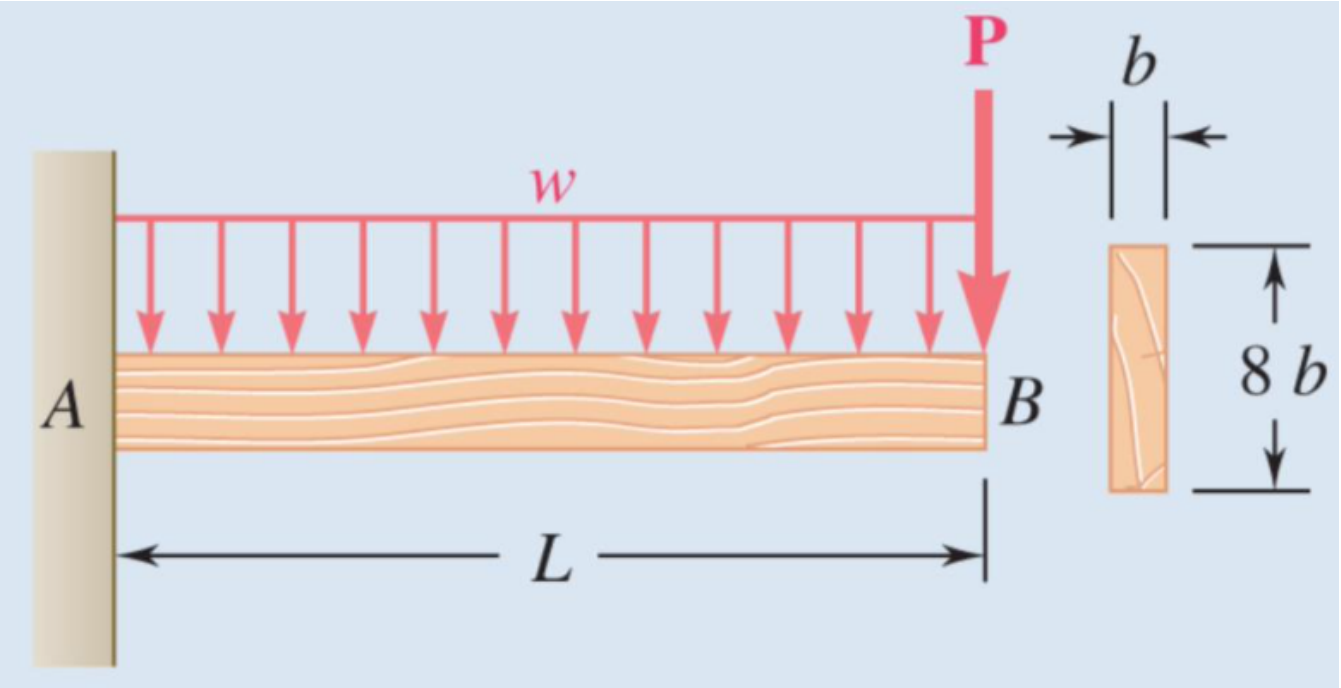

Figure 1: Given System (Cantilever Beam, Rectangular Cross-Section)

1.1 Dimension Results Required

1. Point load: 1000 lb, distributed load: 0
2. Point load: 0 lb, distributed load: 12.5 lb/in
3. Point load: 500 lb, distributed load: 12.5 lb/in

1.2 Unknowns

Table 1: Unknown Values to Solve For

Variables	Symbol(s)	Units
Length	L	Inches
b	b	Inches

1.3 Givens

Table 2: Known Values from Problem

Variables	Symbol	Values	Units
Maximum Normal Stress (Bending)	$\sigma_{allowable}$	1800	psi
Maximum Shear Stress	$\tau_{allowable}$	120	psi
Base Length (Cross-Section)	N/A	b	inches
Height Length (Cross-Section)	N/A	$8b$	inches

1.4 Process Outline

First, we will draw a free-body diagrams of the system, and identify the shear and moment functions ($V(L)$ and $M(L)$). Then, we will identify the necessary equations for normal and shear stress. We will substitute our shear and moment functions into the equations for the normal and shear stresses, plug in known values, and then solve the simultaneous equations for L and b.

2 Theory Manual

2.1 Shear and Moment Functions

The free body diagram for the system is shown in Figure 2 below:

Figure 2: Free Body Diagram

In Figure 2, we have two loads: a point load P and a distributed load w converted to a point load wL at the centre of the beam. We also add a shear V pointing downwards, according to the sign convention (on the right side of the element, positive shear is down).

The next step is to use the equilibrium equation $\sum F_y = 0$ to determine a function for the shear force.

$$\begin{aligned} \sum F_y &= 0 \\ \sum F_y &= w * L + P - V \\ 0 &= wL + P - V \\ V &= wL + P \end{aligned}$$

We now have an expression for the shear at any point. We may integrate it with respect to the variable length of the beam (in this case, L) to find the moment at any point.

$$V(L) = \mathbf{wL + P} \tag{1}$$

$$M(L) = \int V dL = \frac{\mathbf{wL^2}}{\mathbf{2}} + \mathbf{PL} \tag{2}$$

2.2 Stress Equations and Substitutions

In the case of bending with a rectangular cross-section, the following three equations are valid:

$$I = \frac{1}{12}bh^3 \tag{3}$$
$$\sigma = \frac{My}{I} \tag{4}$$
$$\tau = \frac{3}{2}\frac{V}{A} \tag{5}$$

where I is the second moment of inertia of the cross-section, M the moment, y the distance from the fixed axis, V the shear, and A the area of the cross-section. The shear stress τ is a simplified version of the general equation $\tau = \frac{VQ}{It}$, valid *only* for the rectangular cross-section case. The proof of this expression's validity is omitted, because it was completed during lecture time.

Note that, in order to make the program as general as possible, we avoid manually substituting past this point in Sympy. Instead, we input Equations 1-5 and let the `sym.solve()` operation do the rest.

For the hand calculations, we substitute Equations 1, 2, and 3 into Equations 4 and 5 to form System 6.

$$\begin{cases} \sigma = \frac{(\frac{wL^2}{2}+PL)*4b}{\frac{1}{12}b*(8b)^3} \\ \tau = \frac{3}{2}\frac{wL+P}{b*8b} \end{cases} . \tag{6}$$

Now, we substitute the known values of σ and τ...

$$1800 = \frac{(\frac{wL^2}{2} + PL) * 4b}{\frac{1}{12}b * (8b)^3} \tag{7}$$
$$120 = \frac{3}{2}\frac{wL + P}{b * 8b} \tag{8}$$

...and rearrange Equation 8 in terms of b:

$$80 = \frac{wL + P}{b * 8b}$$
$$640b^2 = wL + P$$

$$b = \sqrt{\frac{wL + P}{640}} \tag{9}$$

We plug in this value of b into Equation 7:

$$1800 = \frac{(\frac{wL^2}{2} + PL) * 4b}{\frac{1}{12}b * (8b)^3}$$

$$19200\left(\frac{wL+P}{640}\right)^{\frac{3}{2}} = \frac{wL^2}{2} + PL$$

From here, we plug in actual values to avoid painful math.

2.2.1 Part A

$$P = 1000 \text{ lb}, w = 0$$

$$19200\left(\frac{1000}{640}\right)^{\frac{3}{2}} = 1000L$$

$$19200\left(\frac{125}{64}\right) = 1000L$$

$$37500 = 1000L$$

$$\mathbf{L = 37.5} \text{ inches}$$

$$b = \sqrt{\frac{wL+P}{640}}$$

$$b = \sqrt{\frac{1000}{640}}$$

$$\mathbf{b = 1.25} \text{ inches}$$

2.2.2 Part B

$$P = 0, w = 12.5 \text{ lb/in}$$

$$19200\left(\frac{12.5L}{640}\right)^{\frac{3}{2}} = \frac{12.5L^2}{2}$$

$$\frac{12.5^{\frac{3}{2}}L^{\frac{3}{2}}}{640^{\frac{3}{2}}} = \frac{12.5L^2}{38400}$$

$$0.00273L^{\frac{3}{2}} = 0.000326L^2$$

$$\frac{0.00273L^{\frac{3}{2}}}{0.000326L^2} = 1$$

$$\frac{0.00273}{0.000326\sqrt{L}} = 1$$

$$\left(\frac{0.00273}{0.000326}\right)^2 = L$$

$$\mathbf{L = 70.31} \text{ inches}$$

$$b = \sqrt{\frac{12.5 * 70.31}{640}}$$

$$\mathbf{b = 1.17} \text{ inches}$$

2.2.3 Part C

$$P = 500 \text{ lb}, w = 12.5 \text{ lb/in}$$

$$19200\left(\frac{12.5L+500}{640}\right)^{\frac{3}{2}} = \frac{12.5L^2}{2} + 500L$$

$$38400\left(\frac{12.5^{\frac{3}{2}}L^{\frac{3}{2}} + 500^{\frac{3}{2}}}{640^{\frac{3}{2}}}\right) = 12.5L^2 + 1000L$$

$$38400\left(\frac{44.1942L^{\frac{3}{2}} + 11180.3}{16190.9}\right) = 12.5L^2 + 1000L$$

$$\frac{44.1942L^{\frac{3}{2}} + 11180.3}{16190.9} = 480000L^2 + 38400000L$$

$$44.1942L^{\frac{3}{2}} + 11180.3 = 7.771 * 10^9L^2 + 6.217 * 10^{11}L$$

$$\frac{44.1942L^{\frac{3}{2}} + 11180.3}{7.771 * 10^9L^2 + 6.217 * 10^{11}L} = 1$$

$$\frac{44.1942L^{\frac{3}{2}}}{7.771 * 10^9L^2 + 6.217 * 10^{11}L} + \frac{11180.3}{7.771 * 10^9L^2 + 6.217 * 10^{11}L} = 1$$

$$\frac{44.1942L^{\frac{3}{2}}}{7.771 * 10^9L^2} + \frac{44.1942L^{\frac{3}{2}}}{6.217 * 10^{11}L} + \frac{11180.3}{7.771 * 10^9L^2 + 6.217 * 10^{11}L} = 1$$

$$\frac{44.1942}{7.771 * 10^9\sqrt{L}} + \frac{44.1942\sqrt{L}}{6.217 * 10^{11}} + \frac{11180.3}{7.771 * 10^9L^2 + 6.217 * 10^{11}L} = 1$$

From here, the math is beyond messy, so the author used his TI-Nspire calculator to solve the equation to its conclusion.

$$\mathbf{L = 59.80} \text{ inches}$$

$$b = \sqrt{\frac{12.5 * 59.80 + 500}{640}}$$

$$\mathbf{b = 1.40} \text{ inches}$$

3 Programmer Manual

3.1 Variables and Constants

Table 3: Variables and Constants Used

Variable	Description
`L`	symbolic variable storing value of L
`b`	symbolic variable storing value of b
`sigma`	numeric variable storing the allowable normal bending stress
`tau`	numeric variable storing the allowable shear stress
`base`	dummy variable storing value of cross-sectional width
`height`	dummy variable storing value of cross-sectional height
`area`	symbolic variable storing `base * height`
`moment`	function's local symbolic variable storing calculated second moment of inertia
`V`	function's local symbolic variable storing expression for shear
`M`	function's local symbolic variable storing expression for moment
`sigmaEq`	Sympy equation relating normal bending to `sigma`
`tauEq`	Sympy equation relating shear to `tau`
`results`	function's local variable storing calculated `L` and `b` values
`partA`	array storing possible values of `L` and `b` for the Part A parameters
`partB`	array storing possible values of `L` and `b` for the Part B parameters
`partC`	array storing possible values of `L` and `b` for the Part C parameters

3.2 Functions and Operations

Table 4: Functions and Operations used in Script, in Order of Appearance

Function/Operator	Description
`MoI`	function that calculates moment of inertia for a rectangular section
`dimensionFinder`	function that calculates required lengths based on given bending loads
`sym.symbols`	creates symbolic variable from Sympy symbol library
`sym.eq`	creates symbolic equation with two sides
`sym.nonlinsolve`	nonlinear symbolic equation solver
`print`	displays input in the console
`list`	converts input to a list, elements can then be accessed with [n] notation
+	addition operator
-	subtraction operator
*	multiplication operator
/	division operator
**	exponential operator

3.3 Code Outline

- Enter all constants and symbolic variables to solve for
- Declare function for moment of inertia of rectangular section
- Declare function to find required dimensions of beam (inputs are loads, but requires V, M, b, L, sigma, tau, and MoI parameters to be predefined)
  - Use moment of inertia function as a helper function for this "main" function
- Run function for given load parameters and print results

4 Results and Analysis

The goal of this analysis section is to compare the results of the hand calculations in Section 2 to the results of the code.

4.1 Python Code Output

```
The results for Part A are:
L = 37.50 inches
b = 1.25 inches

The results for Part B are:
L = 70.31 inches
b = 1.17 inches

The results for Part C are:
L = 59.80 inches
b = 1.40 inches
```

Figure 3: Results from Python Script

4.2 Comparison to Theory Manual

4.2.1 Part A

The results from Part A's derivation in the theory manual are below:

$$\begin{cases} \mathbf{L = 37.5} \text{ inches} \\ \mathbf{b = 1.25} \text{ inches} \end{cases}$$

This matches exactly with the Part A Python code results in Figure 3, indicating that the model (with a single point load) is correct.

4.2.2 Part B

The results from Part B's derivation in the theory manual are below:

$$\begin{cases} \mathbf{L = 70.31} \text{ inches} \\ \mathbf{b = 1.17} \text{ inches} \end{cases}$$

These numbers also match the Python script's results, verifying the distributed-load-only model.

4.2.3 Part C

The results from Part C's derivation in the theory manual are below:

$$\begin{cases} \mathbf{L = 59.80} \text{ inches} \\ \mathbf{b = 1.40} \text{ inches} \end{cases}$$

This final set of numbers matches the Part C results in Figure 3, indicating that the model in the script is capable of handing a combined distributed/point load scenario.

5 Appendix

5.1 Python Code (.py)

```python
# Benjamin Coveler
# Lab 3 (CIV_ENV 216-0)
# 11 May 2023

# Find length L and width b given values of P and W

import sympy as sym

# Parameters given
L = sym.symbols(r'L')
b = sym.symbols(r'b')
sigma = 1800
tau = 120

# Cross-section parameters
base = b
height = 8 * b
area = base * height


# Moment of inertia calculator (rectangular section)
# Inputs: base [length], height [length]
# Output: moment of inertia [length^4]
def MoI(MoI_base, MoI_height):
    moment = 1/12 * MoI_base * MoI_height ** 3
    return moment


# Main function
# Inputs: point load [force], distributed load [force/length]
# Output: L [length], b [length]
def dimensionFinder(point_load, distributed_load):
    V = distributed_load * L + point_load
    M = (distributed_load * L ** 2) / 2 + point_load * L
    sigmaEq = sym.Eq(sigma, M * (4 * b) / MoI(base, height))
    tauEq = sym.Eq(tau, 3/2 * V / area)
    results = sym.nonlinsolve([sigmaEq, tauEq], [L, b])
    return results


# Call dimensionFinder for each set of inputs
print('The results for Part A are:')
partA = dimensionFinder(1000, 0)
print('L = %.2f inches\nb = %.2f inches' % (list(list(partA)[1])[0], ...
                                            list(list(partA)[1])[1]))

```

```
46 print('\nThe results for Part B are:')
47 partB = dimensionFinder(0, 12.5)
48 print('L = %.2f inches\nb = %.2f inches' % (list(list(partB)[0])[0], ...
                                               list(list(partB)[0])[1]))
49
50 print('\nThe results for Part C are:')
51 partC = dimensionFinder(500, 12.5)
52 print('L = %.2f inches\nb = %.2f inches' % (list(list(partC)[1])[0], ...
                                               list(list(partC)[1])[1]))
```