Lab II

Benjamin Coveler

Northwestern University

Civil Engineering 216

27 April 2023

Contents

List of Figures

List of Tables

1 Problem Statement

Design a finite element method (FEM) solver capable of handling 1-dimensional beam problems. Then, use the solver on the system given in Figure 1. Find the support reaction forces at the two ends, the internal forces at each element, and the stresses at each element.

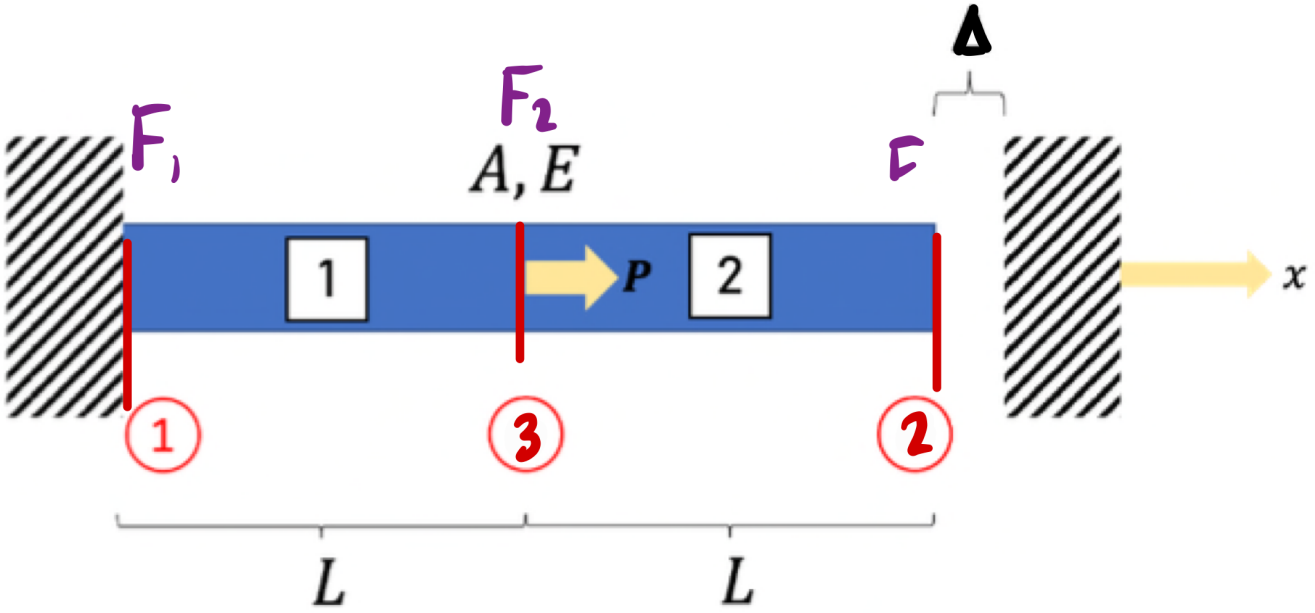

Figure 1: Given System (2 elements, 3 nodes)

1.1 Part A Question

Manually solve the system in Figure 1 for reactions, internal forces, and stresses at each element.

1.2 Part B Question

Write a computer program that solves any 1-dimensional system of bars with 2 walls (n elements, $n+1$ nodes).

1.3 Part C Question

After manually deriving the quantities and writing the program from the problem statement, verify the code using the system and the manual derivation results.

1.4 Unknowns

Table 1: Unknown Values to Solve For

Variables	**Symbol(s)**	**Units**
Reaction Forces	F_1 and F_2	Newtons
Internal Forces	F_{13} and F_{23}	Newtons
Elemental Stresses	σ_1 and σ_2	Newtons/millimetre^2

1.5 Givens

Table 2: Known Values from Problems

Variables	**Symbol**	**Values**	**Units**
Cross-Sectional Area of Bars	A	250	millimetres^2
Length of Bars	L	150	millimetres
Young's Modulus	Q	2*10^4	Newtons/millimetre^2
Applied Load	P	6*10^4	Newtons
Gap Distance	Δ	1.2	millimetres

1.6 Part A Process Outline/Task

In Part A, we aim to solve for the unknowns in Table 1. To do this, we will use the method of cuts for axial loading to find both types of forces. Then, we will use the equation $\sigma = E\epsilon$, known as Hooke's Law, to solve for the stresses.

1.7 Part B Process Outline/Task

In Part B, we use the method of finite elements to construct a MATLAB script that automates the process of Part A. This code must function for a general 1-dimensional system, so the code will necessarily include abstraction.

1.8 Part C Process Outline/Task

Finally, in Part C, we input the system parameters from Table 2 into the code from Part B. Then, we compare the results to the hand-calculated answers from Part A.

2 Theory Manual (Part A)

2.1 Equation Derivation

The free body diagrams for the system are shown in Figure 2 below, with internal cuts in red:

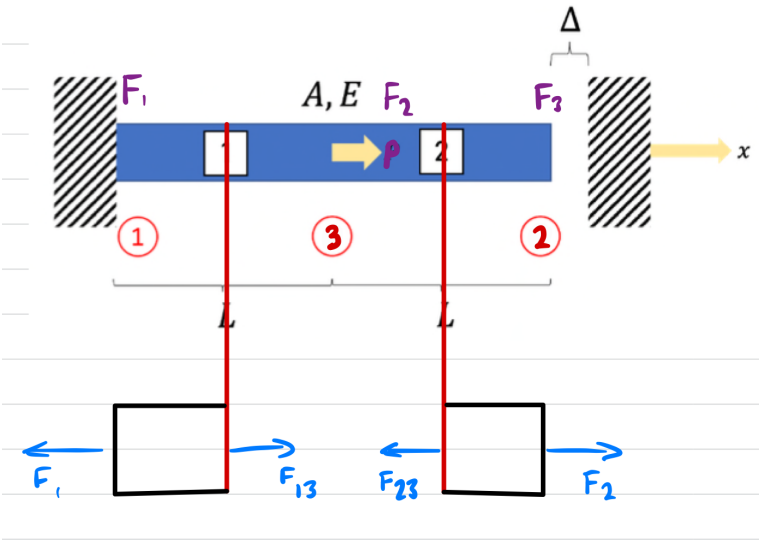

Figure 2: Free Body Diagrams

In Figure 2, we drew two cuts, one through each element. This will reveal the internal forces within each element, named F_{13} and F_{23} respectively. The node numbers are switched to match the notes nomenclature. After creating these internal forces (and reaction force equations), we may create our connectivity table. The connectivity table relates the nodes to the elements.

Table 3: Connectivity Table

Element	**Node** i	**Node** j
1	1	3
2	3	2

The next step is to create the k matrix for each element – that is, the stiffness. Note that both elements share the same values of E, A, and L, so the stiffness matrix is duplicated.

$$k_i = \frac{EA}{L} * \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Substituting in the values for the constants, the resulting stiffness matrix for *both* elements is:

$$k = \begin{bmatrix} 3.33 * 10^4 & -3.33 * 10^4 \\ -3.33 * 10^4 & 3.33 * 10^4 \end{bmatrix} \tag{1}$$

Next, we construct a global stiffness matrix. To complete this matrix, we "place" each elemental k value at each relevant node. In this case, the global stiffness matrix (note the capital K) is:

$$K = \begin{bmatrix} k_1 & 0 & -k_1 \\ 0 & k_2 & -k_2 \\ -k_1 & -k_2 & k_1 + k_2 \end{bmatrix}$$

Because each value of k is the same, we may substitute directly, and assume $k = k_1 = k_2$.

$$K = \begin{bmatrix} 3.33 * 10^4 & 0 & -3.33 * 10^4 \\ 0 & 3.33 * 10^4 & -3.33 * 10^4 \\ -3.33 * 10^4 & -3.33 * 10^4 & 6.67 * 10^4 \end{bmatrix} \tag{2}$$

From the notes, we know that the system is described by the equation $F = K * u$:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 3.33 * 10^4 & 0 & -3.33 * 10^4 \\ 0 & 3.33 * 10^4 & -3.33 * 10^4 \\ -3.33 * 10^4 & -3.33 * 10^4 & 6.67 * 10^4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

Now, we substitute in the values of P and Δ for F_3 and u_2 respectively. We also know that u_1 is 0, because the wall is a fixed support and there is no deflection of that element.

$$\begin{bmatrix} F_1 \\ F_2 \\ 6 * 10^4 \end{bmatrix} = \begin{bmatrix} 3.33 * 10^4 & 0 & -3.33 * 10^4 \\ 0 & 3.33 * 10^4 & -3.33 * 10^4 \\ -3.33 * 10^4 & -3.33 * 10^4 & 6.67 * 10^4 \end{bmatrix} \begin{bmatrix} 0 \\ 1.2 \\ u_3 \end{bmatrix} \tag{3}$$

Solving this system yields $F_1 = 50$ kN, $F_2 = -10$ kN, and $u_3 = 0.0015$ m. We know from our free body diagrams that the internal forces are equal to the reaction forces. **Therefore, the reaction forces are equal to 50 kN on Bar 1 and 10 kN on Bar 2. The internal forces are equal to 50 kN within Bar 1 and 10 kN on Bar 2.**

The final step of this derivation requires that we find the stresses in both bars. We know, via Hooke's law, that $\sigma = E\epsilon$.

$$\sigma_1 = E\epsilon_1 = 2 * 10^4 * \frac{u_2 - u_1}{L} = 200\frac{N}{mm^2} \tag{4}$$

$$\sigma_2 = E\epsilon_1 = 2 * 10^4 * \frac{u_3 - u_2}{L} = -40\frac{N}{mm^2} \tag{5}$$

Therefore, the stress in Bar 1 is 200 N/mm^2, and the stress in Bar 2 is -40 N/mm^2.

3 Programmer Manual

3.1 Variables and Constants

Table 4: Variables and Constants Used

Variable	Description
A	Var storing value of cross-sectional area
L	Var storing value of length of bars
E	Var storing value of Young's constant
P	Array storing location/value of applied load
delta	Var storing value of gap distance
elements	Var storing number of elements
nodes	Var storing number of nodes
iNode/jNode	Var storing i- and j-node numbers for elements
k	Array storing values of local stiffness matrix
K	Matrix storing global stiffness values
u	Array storing values of displacement for each element
stress	Array storing values of epsilon for each element

3.2 Functions and Operations

Table 5: Functions and Operations used in Script, in Order of Appearance

Function or Operator	Description
for	Loop statement that runs until a parameter is met
sprintf	Function that prints a specified statement
if	Conditional statement that runs code if cond met
elseif	Conditional statement that adds a second if condition
else	Conditional statement for all other cases not covered by `if`
end	Ends loops, conditionals, and other functions
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
\	Augmented matrix solver operator
.*	Element-wise multiplication operator for matrices

3.3 Code Outline

- Enter all constants and request force locations from user (to avoid having to manually enter elements of matrices)
- Construct connectivity table, following the nomenclature used in the FEM notes
- Create local stiffness matrix, then insert that local stiffness matrix into a global stiffness matrix. Locations determined by connected elements.
- Calculate displacements: P = K*U is equivalent to u = K \P
- Use Hooke's Law to calculate stresses from displacements matrix

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. We begin by examining the global stiffness matrix the code constructed:

$$K = \begin{bmatrix} 33333.3333333333 & 0 & -33333.3333333333 \\ 0 & 33333.3333333333 & -33333.3333333333 \\ -33333.3333333333 & -33333.3333333333 & 66666.6666666667 \end{bmatrix}$$

This indeed matches Equation 2, so we are on the right track. Next, we examine the actual results. We next pull up the force matrix, which is:

$$P = \begin{bmatrix} -50000 \\ -10000 \\ 60000 \end{bmatrix}$$

The 60000 matches the input for Node 3, and the two forces match our prediction of 50 and 10 kiloNewtons using Equation 3. Finally, let us examine the stress outputs.

$$\sigma = \begin{bmatrix} 200 \\ -40 \end{bmatrix}$$

This exactly matches our prediction from Equations 4 and 5. Combined with the successful results in the previous two tests, the finite element method calculator appears to behave as expected.

4.1 Axial Diagram

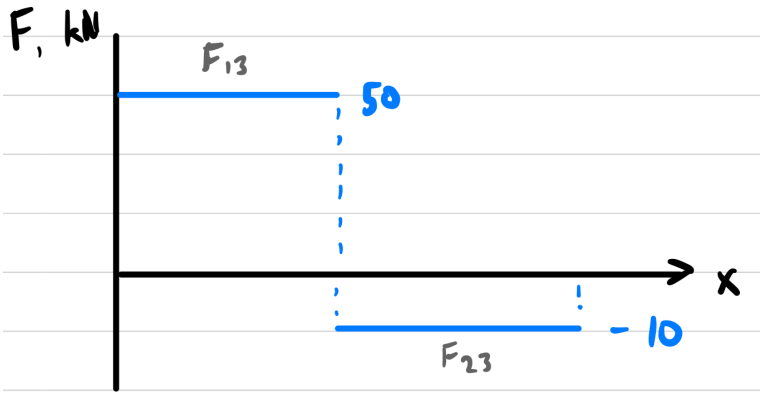

Figure 3: Axial Diagram: Force (kN) vs Distance x (mm)

5 Appendices and Attributions

5.1 MATLAB Script

```
% Benjamin Coveler
% 27 April 2023
% CEE 216 Lab 2

clear
close all

% Define shared properties of elements
A = 250; % mm^2
L = 150; % mm
E = 2.0e4; % N/mm^2

% Define load and displacement
P = 6.0e4; % N
Δ = 1.2; % mm

% Define number of elements and nodes
elements = 2;
nodes = 3;

% Request user for load at each node (for test case, only requests 1)
% Subtract 2 to get the initial position at the first element
% (arrays start at 1 in MATLAB)
% P matrix is for loads
for i = 1:nodes-2
      P(1, i+2) = input(sprintf('Load at node %d: ', i+2));
end

% Transpose load matrix to vertical orientation for later
P = P';

% Construct connectivity table
for i = 1:elements
      % If counter is 1, then manually assign according to notes
      % nomenclature system (3 is second item)
      if i == 1
            iNode(i) = 1;
            jNode(i) = 3;
      % If counter is at end of count, manually assign according to notes
      elseif i == elements
            iNode(i) = nodes;
            jNode(i) = 2;
      % Else, use pattern of 1 ahead for i and 2 ahead for j
      else
            iNode(i) = i+1;
            jNode(i) = i+2;
```

```
47        end
48  end
49
50  % Create local stiffness matrix. All values of k will have to be
51  % the same, since we are only taking in one value for E, A, and L
52  % k = (E*A/L) * [1, -1; -1,1];
53
54  matrix = [1, -1; -1,1];
55  for i = 1:elements
56        coeff = (E*A)/L;
57        k{i} = coeff.*matrix;
58  end
59
60  % Create global stiffness matrix by inserting values of k
61  % Initialise the global K matrix
62  K = zeros(nodes);
63  % Double loop through node count, to create node x node dimension matrix
64  for i = 1:nodes
65        for j = 1:nodes
66              if i == j
67                    % Funny business so we can copy nomenclature from notes
68                    % (less confusing for handwritten solution)
69                    if i > 2 && j > 2
70                          K(i,j) = k{1}(iNode(1))+k{2}(iNode(1));
71                    else
72                          K(i,j) = k{1}(iNode(1));
73                    end
74              % Elseif two conditions - OR (one, but not both, conditions)
75              elseif j == nodes || i == nodes
76                    K(i,j) = k{1}(jNode(1));
77               % If none of these conditions fit, leave entry as 0
78              end
79        end
80  end
81
82  % Initialise displacements vector
83  % Manually set displacement as gap (def of static indeterminate problem)
84  u = zeros(nodes, 1);
85  u(2) = Δ;
86
87  % Calculate displacements for all items using backslash operator
88  % to solve system without having to invert (AIA' method)
89  % Start at 3, because 1 and 2 are already known (u1 = 0, u2 = Δ)
90  % Run to end in case matrix is larger than 3 elements
91  u(3:end) = K(3:end, 3:end) \ (P(3:end) - Δ.*K(3:end, 2));
92
93  % Use equation F = K * u, with K as the global stiffness
94  P(1:2) = K(1:2,:)*u;
95
96  % Calculate stresses at every element using for-loop
97  % Hooke's Law: sigma = E * eps, eps = displacement/L
98  for i = 1:elements
```

```
99          % epsilon = uB - uA / L
100         stress(i) = (u(jNode(i)) - u(iNode(i))) * E/L;
101 end
```

5.2 Attributions

The author would like to thank Vina Sathtachotinun for her valuable help in creating the MATLAB code for this assignment.