COE 321K: Arbitrary Two-Dimensional Truss Solver

Ashton Cole
AVC687

February 13, 2023

1 Introduction

This assignment computes the reaction of an arbitrary two-dimensial truss system to applied forces and constrained displacements within a single Python script. It assumes the joints are pins, that bars are light, and that strains are small and proportional to stresses. It accepts input through files in the form provided in class, and outputs to the terminal and files. The product is applied to an example from class in Section 3.

2 Setup and Execution

The file `trusssolver2d.py` contains the entirety of the solver. The program requires the module `numpy` which must be installed separately.

Currently, hard-coded changes need to be made to the script for it to execute correctly. The paths for input files describing the system's nodes, elements, displacement constraints, and force constraints, as well as an output directory for all results files, need to be set at lines 460-464 in function `main()`.

After this, the script can simply be interpreted in a Python3 environment. It will solve the reactionary displacements and forces, as well as internal strains, stresses, and forces. Reactionary forces are output to `reactionary_forces`, reactionary displacements to `reactionary_displacements`, internal strains to `internal_strains`, internal stresses to `internal_stresses`, and internal forces to `internal_forces`, respectively.

3 Results

For this assignment, we reconsider the example provided in class, seen in Figure 1. The structure has a height and width L, and each element shares the same Young's modulus E and cross-sectional area A. An arbitrary force is applied downwards on node D. For the input files, coordinates are normalized to L,

Figure 1: Simple Truss Example Provided in Class

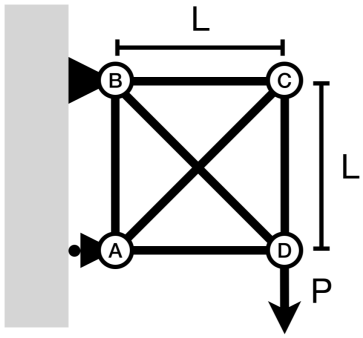


Table 1: Node and Element Identifier Translation Tables

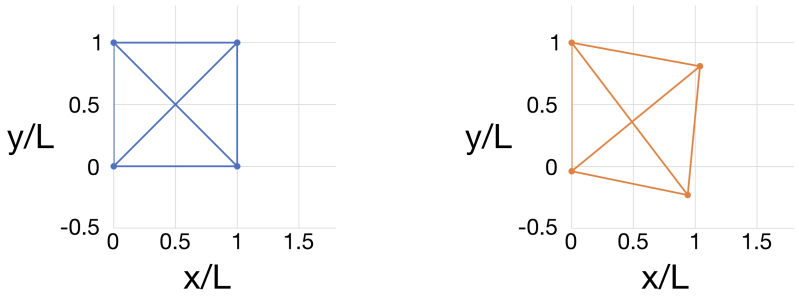
Node	Identifier
A	1
B	2
C	3
D	4

Element	Identifier
AB	1
AC	2
AD	3
BC	4
BD	5
CD	6

Table 2: Normalized Nodal Displacements

Node	Identifier	$\frac{uEA}{PL}$	$\frac{vEA}{PL}$
A	1	0	-0.3964
B	2	0	0
C	3	0.3964	-1.9142
D	4	-0.6036	-2.3107

Figure 2: Truss Before and After Displacements, with Displacements Scaled Down by a Factor of 0.1, in Terms of L



while values for E, A, and P are all set to 1. This means that the outputs can be thought of as nondimensional coefficients for expressions of these variables. For example, the output for the internal force of a given element could be applied to a congruent structure with any material properties by applying the coefficient to $\frac{PL}{EA}$. For the purposes of the program, the nodes and elements are given unique integer identifiers instead of letters. The translations are given in Table 1[1].

The resulting terminal output is attached in Figure 3. It includes all of the data which is saved into files. The nodal displacement values specific to this assignment are found in Table 2. Visualizations have also been created above to demonstrate how the force deforms the object. Dimensions are removed from the scales, and the displacement coefficients are reduced by tenfold to improve the visualization.[2]

4 Conclusions

The results are reasonable, and match what we found in class. As expected, the applied force deforms the truss downwards. Bars which are shortened experience compressive internal forces, marked by a negative sign, while stretched bars experience tension.

However, there is an immediately noticeable error. The reactionary force on node 2 in the vertical direction is not exactly 1, which makes the net force on the structure slightly off from 0. This, however, is likely not an issue with the script, but a numerical error associated with imprecisions in usng 32-bit floating decimals to store calculations. Further, the error is very small, many orders of

[1]It is worth noting that for the purposes of the script, the exact numbering is arbitrary. So long as the identifiers of the nodes match the identifiers referenced in the element configuration file, the program will run as expected.

[2]Technically, even this scale reduction is too exaggerated, since the linearizing assumptions of our solution process require deformations to be very small, but it is useful to see how the system responds.

Figure 3: Sample Terminal Output for the In-Class Example

```
Node ID: 1
Dimension: 1
Reactionary Force: 1.0
Dimension: 2
Reactionary Displacement: -0.3964466094067262
Node ID: 2
Dimension: 1
Reactionary Force: -1.0000000000000002
Dimension: 2
Reactionary Force: 1.0
Node ID: 3
Dimension: 1
Reactionary Displacement: 0.3964466094067263
Dimension: 2
Reactionary Displacement: -1.9142135623730956
Node ID: 4
Dimension: 1
Reactionary Displacement: -0.6035533905932738
Dimension: 2
Reactionary Displacement: -2.310660171779822
Element ID: 1
Strain: 0.3964466094067262
Stress: 0.3964466094067262
Force: 0.3964466094067262
Element ID: 2
Strain: -0.5606601717798214
Stress: -0.5606601717798214
Force: -0.5606601717798214
Element ID: 3
Strain: -0.6035533905932738
Stress: -0.6035533905932738
Force: -0.6035533905932738
Element ID: 4
Strain: 0.3964466094067263
Stress: 0.3964466094067263
Force: 0.3964466094067263
Element ID: 5
Strain: 0.8535533905932738
Stress: 0.8535533905932738
Force: 0.8535533905932738
Element ID: 6
Strain: 0.39644660940672627
Stress: 0.39644660940672627
Force: 0.39644660940672627
```

magnitude below the results, and does not detract from the overall character of the solution.

The script was also tested on truss problems from Homeworks 1 and 2. Likewise, it provided accurate results. Thus, it is fair to say that this appears to be an adequate solver for any truss which falls within our initial assumptions. Further testing on edge cases might be necessary to formally verify the solver's capabilities.

There are several ways that it could be improved upon. First, the program could be generalized to function for any number of spatial dimensions. It could deploy more rigorous error checking to recognize poorly formatted imputs and ill-defined systems. The algorithm could be more tightly scrutinized to remove ineffieciencies and reduce numerical erros from performing calculations with very large and very small numbers. Settings could be provided for the program to only solve certain parts of the solution, to save some time for the user. Finally, a graphic user interface and visualization features could be developed to make the program easier to use. For now, the current solution suffices.