Methods for Truss Optimization (a work in progress

By Saul Chaplin

Spring 2022

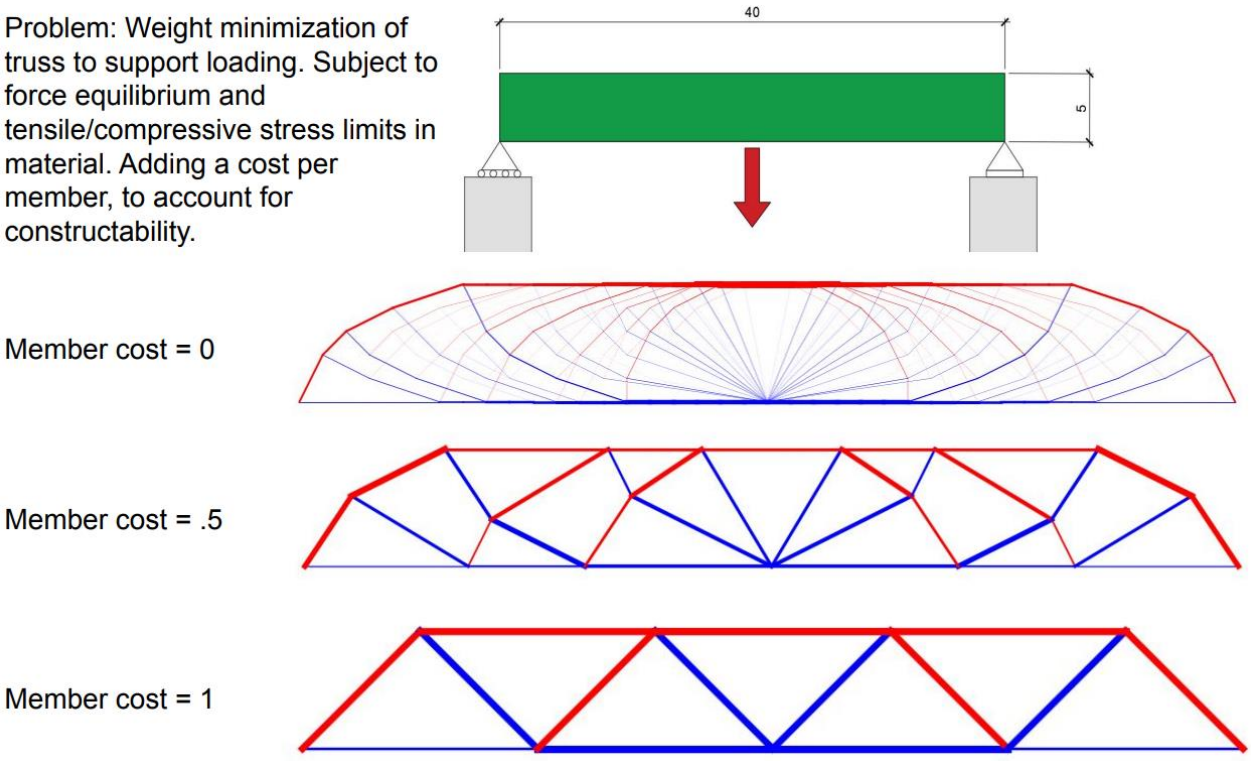


Figure: Example of truss optimization, as we move from a more optimized, less constructible bridge to a less optimized, more constructible bridge

CONTENTS

1. Optimization for weight 3
   1.1. Case 1: Default 5
   1.2. Case 2: Max section area 6
   1.3 Case 3: Double the height 7
   1.4 Case 4: Add a no-go zone for ships to pass 8
   1.5 Case 5: Optimizing for multiple load combinations 11
2. Constructibility Optimization with Member Costs 12
3. Minimization of Construction Costs 14
4. Stiffness via minimization of internal strain 17
5. Multi Objective Function with Weight, Stiffness, and Constructability 20
6. Multiple Materials with Different Costs, Weights, Properties (coming soon) 20
7. Where to go next 21

1. OPTIMIZATION FOR WEIGHT

The optimization process for weight will be shown with an example. Say we have possible truss geometry where B is the number of bars, N is the number of nodes, and the material is A. Material A has a max tensile stress, $\sigma_{max,comp} = 1$ and a max compressive $\sigma_{max,tensile} = 1$. The max section size on the market of a member of material A is taken as 3. The density of material A is given as ρ. (Units will be added later). The force applied at the midpoint at the bottom is $F = 1$. The envelope of space (that will be referred to as the domain) that can be built in is shown below in figure 1.1.

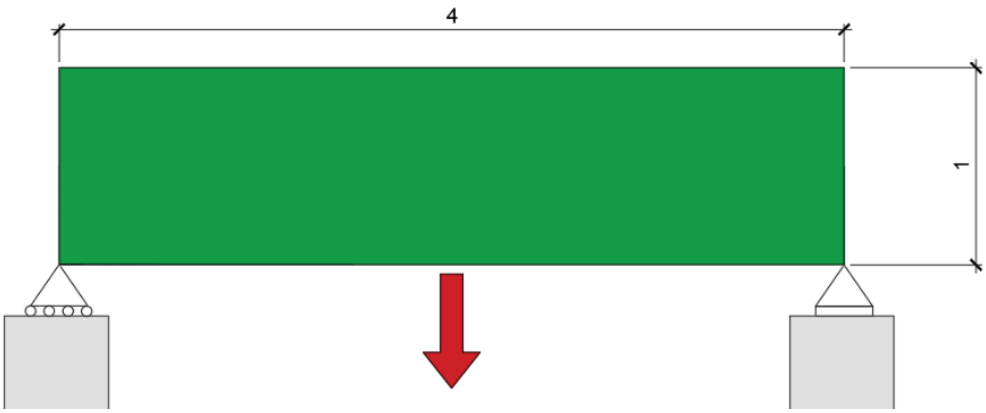


Figure 1.1: Truss envelope for first optimization task

For the first optimization task, we will attempt to minimize the weight of the truss, subject to the constraints that (1) a given loading pattern is supported, and (2) the stress in the members never exceeds the maximum material stress.

We can draw a grid of N nodes over the domain. We can find all the different iterations of bar connections, and call the number of bars B. The nodes and possible bars are shown in figure 1.2:

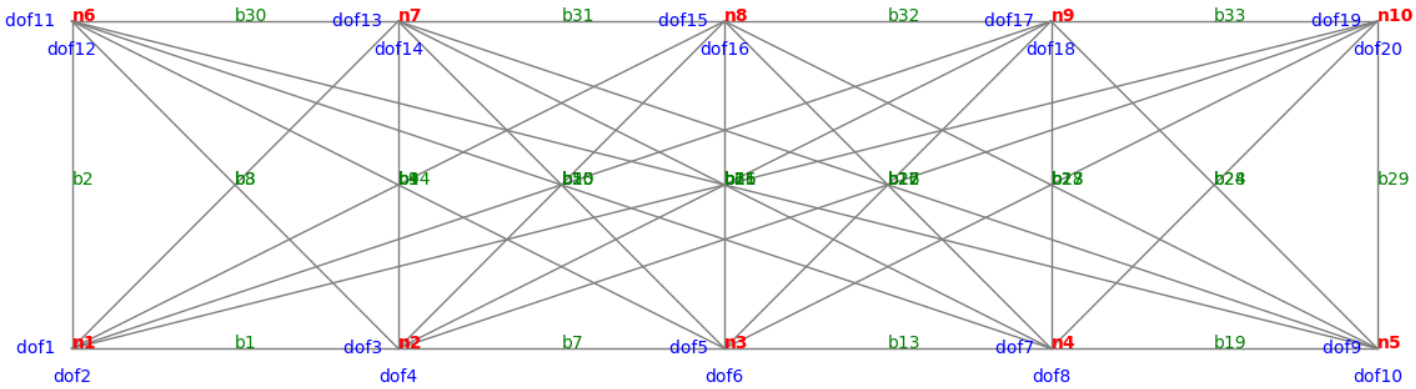


Figure 1.2: Possible nodes and members for truss optimization

Our goal is to minimize weight, so let's minimize mass. Let's define the objective function as

$$minimize(mass) = minimize(\rho * volume\ )$$

Thus, we can minimize for volume, and we will be also minimizing for mass. We can write the optimization problem as:

$$minimize(volume) = minimize\left(\mathbf{a}^{\mathrm{T}} \cdot \mathbf{l}\right)$$

, where **a** is a vector of the sizes of the sections of each member and $\boldsymbol{l}$ is a vector of the lengths of each element. In English, we are simply summing up, for every member, the length times the area, to obtain the total volume. The variables of optimization (aka the variables changed to minimize the objective function here are **a** and **fint).**

The problem is subject to two sets of conditions

(1)Force equilibrium constraint: we have that for all free dof, the sum of the internal forces must be equal to the external forces. In equation form, that looks like:

$$\mathbf{B}_{2\mathrm{NxB}} * \mathbf{fint}_{\mathrm{Bx1}} = \mathbf{Fext}_{2\mathrm{Nx1}} \odot \mathbf{dof}_{2\mathrm{Nx1}}$$

, where:

$\boldsymbol{B}$ is the connectivity matrix, with dimensions $2NxB$, where the element b_{ij} is the force in dof i from one unit of tension in bar j, multiplied by the corresponding $\mathbf{dof}(i)$. A good example is bar 8. For bar 8, the b_{ij} can be written as

$$b_{18} = \frac{\sqrt{2}}{2} * \mathbf{dof}(1) = \frac{\sqrt{2}}{2} * 1$$

$$b_{28} = \frac{\sqrt{2}}{2} * \mathbf{dof}(2) = \frac{\sqrt{2}}{2} * 0$$

In the case of the truss example above, writing the first few elements, we have:

$$\mathbf{B}_{2NxB} = \begin{matrix} \\ dof1 \\ dof2 \\ \vdots \end{matrix} \overbrace{\begin{matrix} b1 & b2 & \cdots \\ \end{matrix}}^{} \left\{ \begin{bmatrix} 1*1 & 0 & \\ 0 & 1*0 & \\ & & \ddots \end{bmatrix} \right.$$

$\mathbf{fint}_{\mathrm{Bx1}}$ is a vector that contains all the internal forces of the bars. (Note: although the f_{int} is not in the objective function it is still a variable of optimization).

$\mathbf{Fext}_{2\mathrm{Nx1}}$ is a vector of the external forces applied at each degree of freedom, $\odot$ denotes element-wise notation

$\mathbf{dof}_{2\mathrm{Nx1}}$ is a vector of 1s and 0s where $\mathbf{dof}(i) = 1$ if dof i is free and $\mathbf{dof}(i) = 0$ if dof i is fixed

(2) Section equilibrium constraint: We need the stress in the material in any member not to exceed the max compressive or tensile stress of the section. We can write this constraint as:

$$for\ all\ i\ in\ [1, B]\ , \qquad -\sigma_{max,comp} * \boldsymbol{a}(i) \leq \mathbf{fint}(i) \leq \sigma_{max,tension} * \boldsymbol{a}(i)$$

(In English, this is basically just that for all the members, the total force cannot exceed the force at which the section fails/yields in compression or tension).

The optimization problem above is written in Python[1], using a code modified from He 2019[2]. The solver CVXPY, an open-source Python package for convex optimization problems, is used. A well commented code called *case1to4.py* is available at https://github.com/saul-chaplin/TrussOptimization.git for the code for the following 4 cases. For those less familiar with Python, set-up instructions and packages to download are available in appendix B of He 2019. Plotting functions were written for the truss and the possible members, forces and supports were defined, max stresses were defined, the connectivity matrix **B** was found.

Note: to run this case, make sure case = 1, as shown below in the code *case1to4.py* and for the future cases, just change case = 2 or case =3 or 4, etc…

```
119  ##YOU CAN CHANGE THE OPTIONS HERE ##
120  case = 1 #Change between different cases
121  text = 1 #Switch to show/hide text on graphs. 1 to show, 0 to hide.
```

The optimization problem is run to obtain the following optimized truss shape:

1.1. Case 1: Default

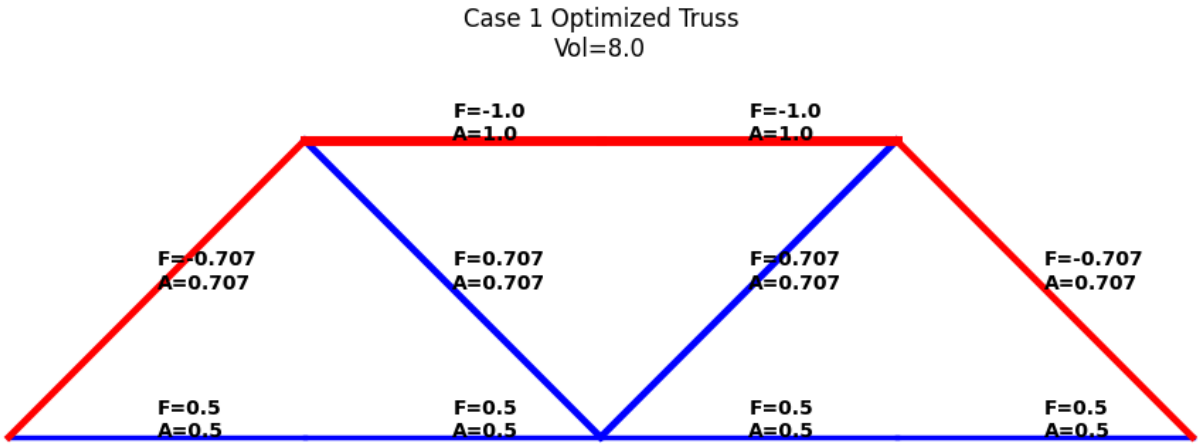


Figure 1.3: Optimized truss for weight

Some quick observations

- A quick numerical check reveals that the that each free node is in equilibrium, that the force of 1 occurring at node 3 (center bottom) is supported.
- Every section is either at its max tensile or compressive stress. This makes sense, because if they were not, then the optimization code would further reduce section size so that they were.
- Instead of continuous members across the top and the bottom, the members are "broken up" into 2 individual pieces. If you think about it, with our objective function, one member spanning the entire distance would provide exactly the same volume, and therefore there is no reason not to break up the member into two smaller pieces. It is notable, however, that (1) it would probably be easier to build the truss if the members were longer and there were less of them and (2) a joint in

[1] If you are new to Python (as I am!) I recommend this video https://www.youtube.com/watch?v=rfscVS0vtbw. Python is fairly similar to MATLAB, and you'll pick it up quickly! It is fun, useful, and free!
[2] It is recommended that you read this paper.

a pin compression member creates instabilities. Therefore, this "problem" will be remedied in section 2 by adding a cost per member.

It is fun to fiddle around with the optimization by playing around with some input parameters.

1.2. Case 2: Max section area

The parameters from case 1 are unchanged except for the fact that a max section size is added, of $a_{max} = .5$. Note how this is below the area of many of the members from the Case 1 scenario.

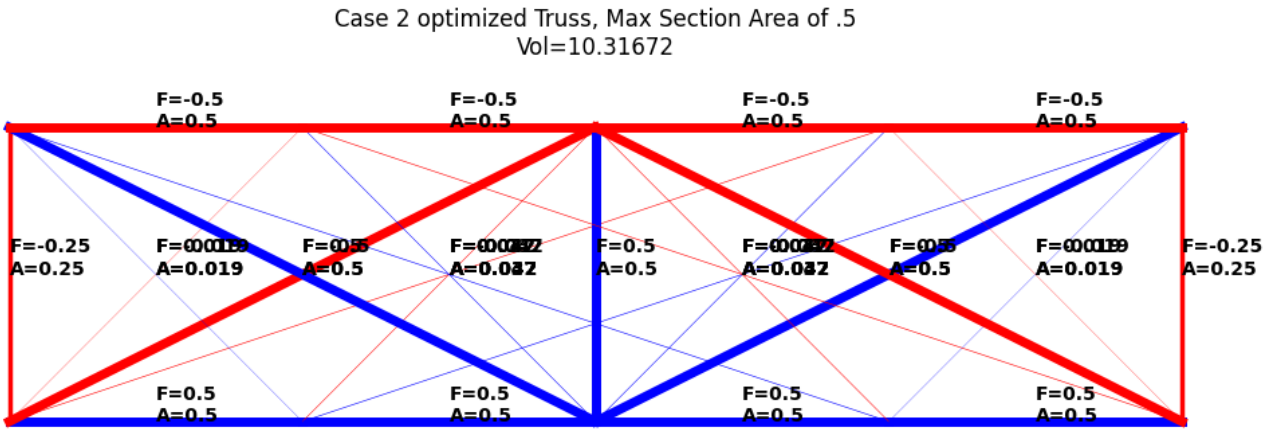


Figure 1.4: Output from a max section restrained truss

Comments:

- Redundant load paths have formed, because the determinant load path in case 1 is unable to take the load, as section sizes were limited to .5
- Objective function of the volume has climbed to 10.32 up from 8 (case 1). When we add another restriction, the objective must either increase or stay constant, and it has
- Tiny members have appeared. Those members are due to numerical errors in the optimization. Those members will be eliminated in part 2 when we add a member cost.

1.3 Case 3: Double the height

The parameters from case 1 are unchanged except that the height of the allowable building space was doubled from 1 to 2. The new bridge domain, and layout of possible members are shown in figure 1.5. The optimized truss is shown in figure 1.6.

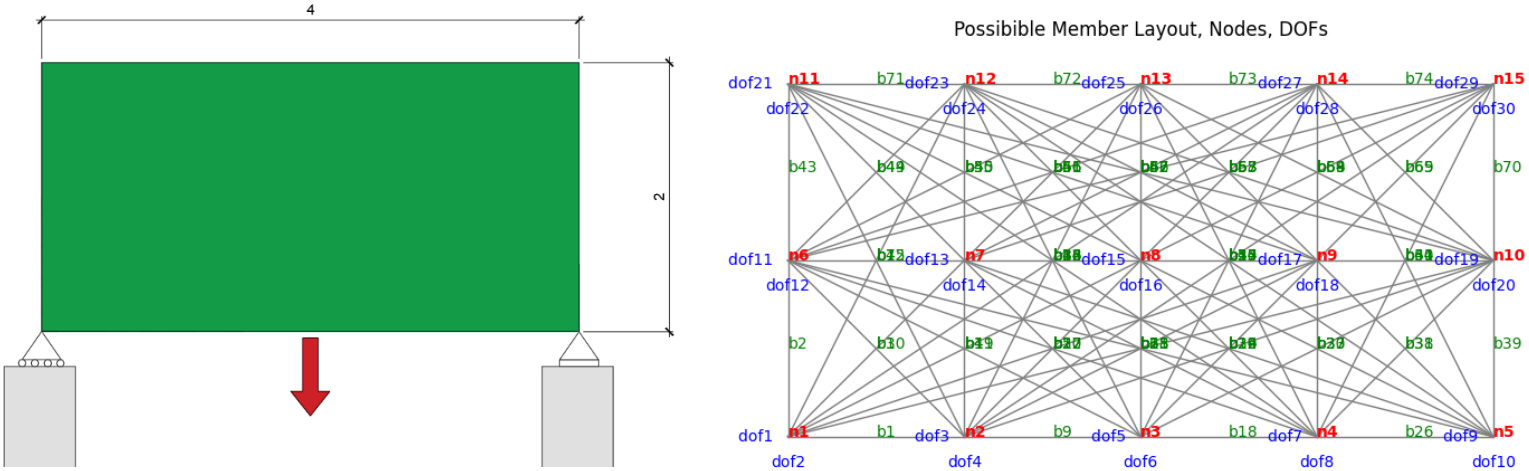


Figure 1.5: Bridge domain and possible members for a double-height scenario

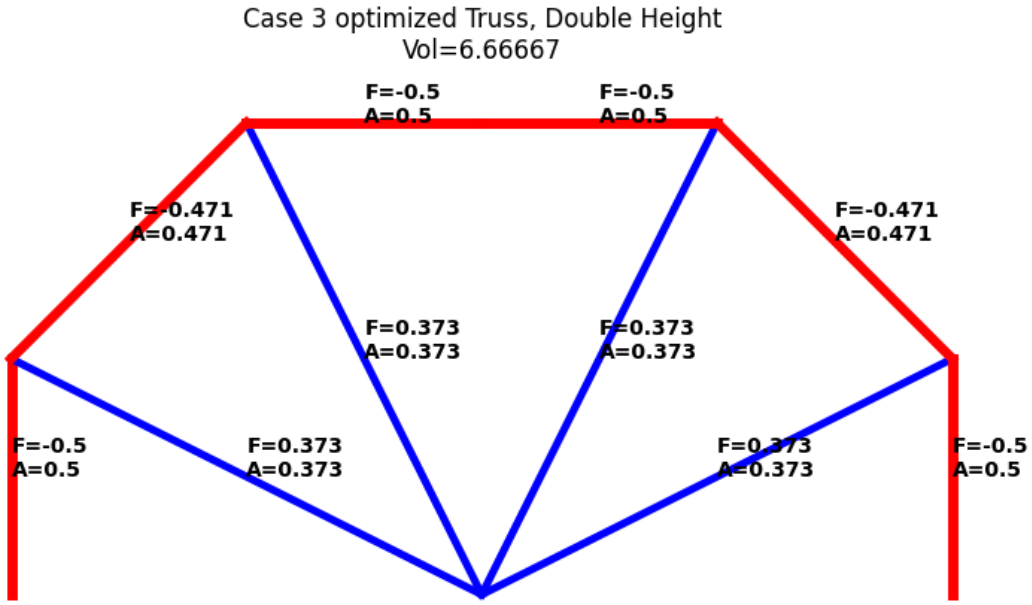


Figure 1.6: Optimized truss for double-height scenario

Comments:

- Volume has been reduced to 6.67, down from 8 in case 1. When we reduce the restrictions, (i.e. make the envelope larger), the objective function should either decrease or stay the same, and it has.
- The supports transfer no lateral force. Thus, the members to the left and right are perfectly vertical. Basically, we have a simplification of a half-circle arch. Also, the structure looks a bit like a half bike wheel, where the blue members are the spokes, in tension supporting the force which would be the weight of the bike carried through the hub.
- The structure, under pinned joints, has no lateral stability. If we were to design such a structure, the columns on the left and right would have to have to take significant moment at their bases.

1.4 Case 4: Add a no-go zone for ships to pass

To illustrate the versatility and limitations of the code, one can imagine a shipping channel like the one shown below in figure 1.7. A bridge is to be built above the channel, in a domain that provides enough room for the ship to pass underneath. The weight of the deck and the cars is simplified as point loads at deck level. A crane may be installed in the future at the midpoint, causing the need for a higher force resistance. Tollbooths will be built at either end, causing an increase in the force. The bridge will connect to approach tower on either side, and it is desired that the bridge itself be free-standing. The possible members, nodes, and dofs are shown in figure 1.8. The optimized truss is shown in figure 1.9.

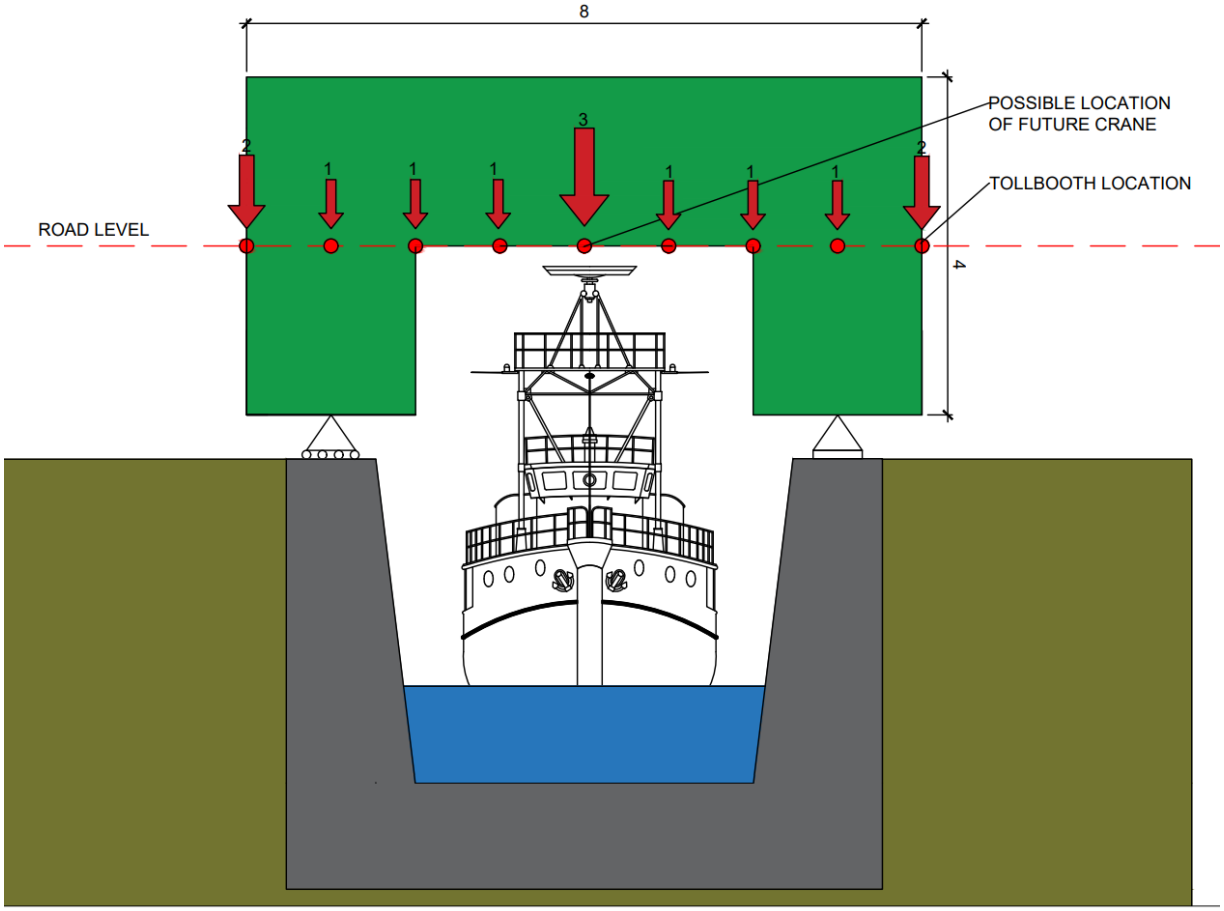


Figure 1.7: Domain, supports, and loads for case 5

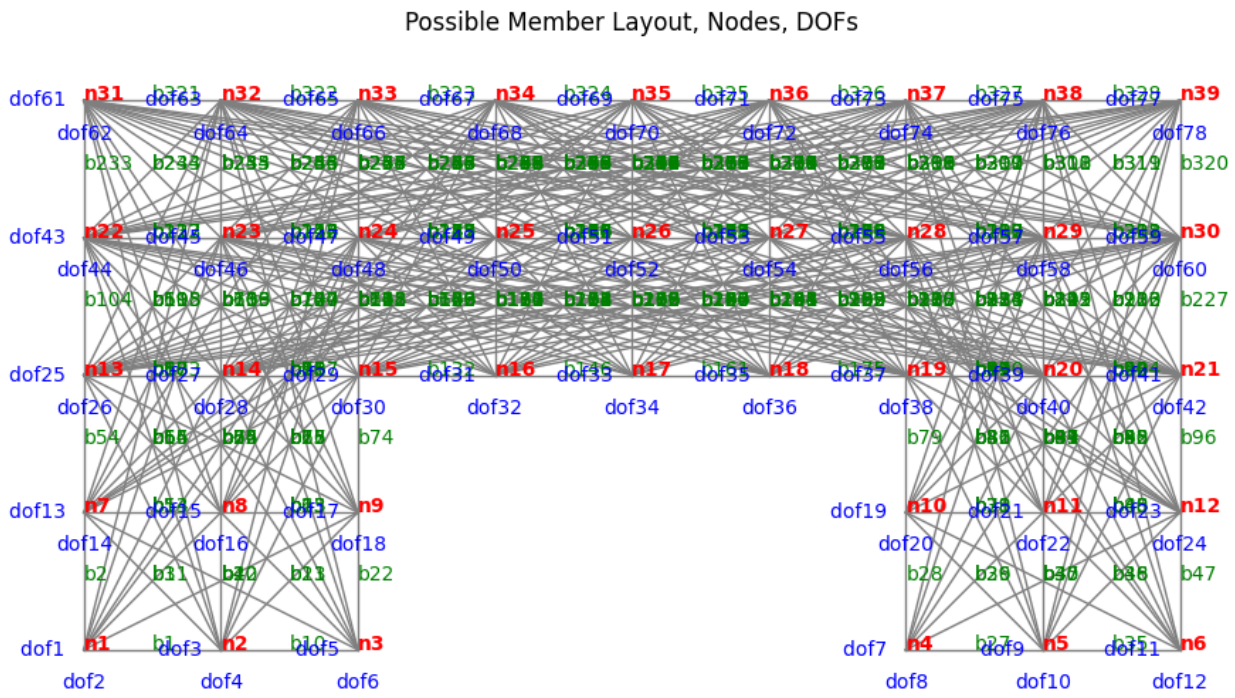


Figure 1.8: Layout of possible truss members, DOFs, and nodes

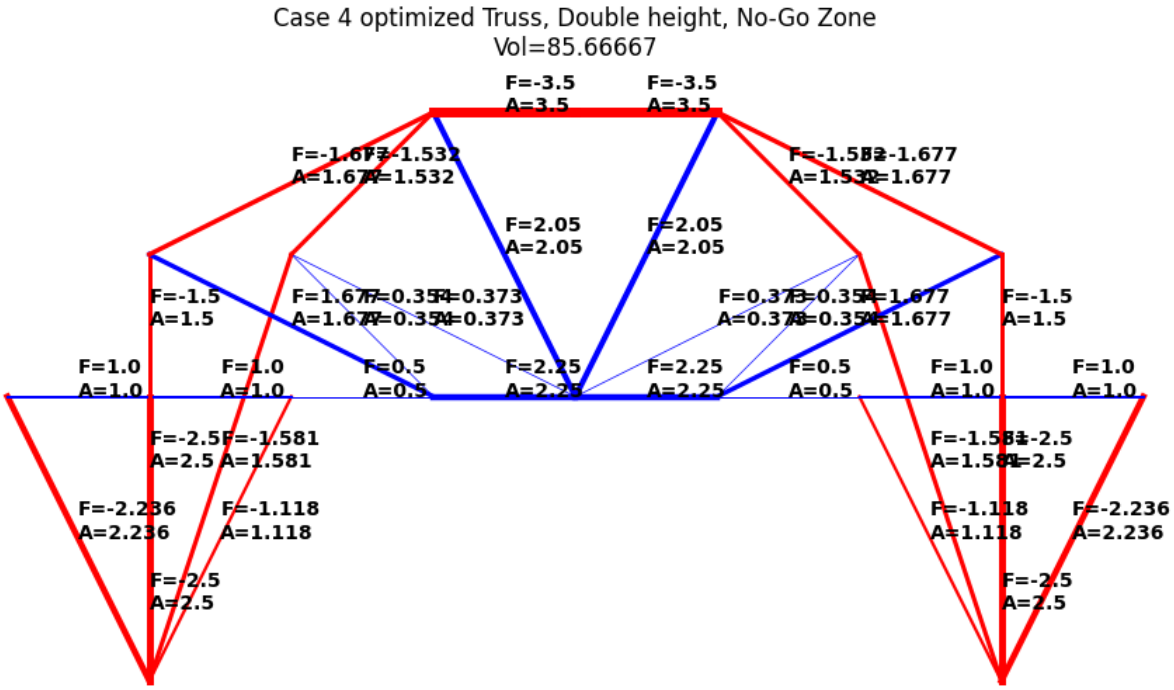


Figure 1.9: Optimized truss combination

Comments

- The shape is very weird looking. It is an optimized structure to support a deck that only puts loads at the specific given points in figure 1.7 and could not support loads in between the loading points. It is unclear where the given structure could support any lateral load.
- Several very small members are present that would make construction unnecessary difficult.
- Other things…

1.5 Case 5: Optimizing for multiple load combinations

(Coming soon!)

Example to be added. I derived the process mathematically and wrote it in AMPL (to be shown in section 3). Basically, just add another force-equilibrium constraint, and section stress constraint force that force combination. The process can be done in Python and is outlined in He 2019.

2. CONSTRUCTIBILITY OPTIMIZATION WITH MEMBER COSTS

To become more detailed.

Just change $\mathbf{l}_{\mathrm{Bx1}}$ the matrix of length of members, by adding a joint cost with value jc to each element. Our objective function is still:

$$minimize(volume) = minimize\left(\mathbf{a}^{\mathrm{T}} \cdot \mathbf{l}_{\mathrm{Bx1}}\right)$$

, but, in this case, $\mathbf{l}_{\mathrm{Bx1}} = \begin{bmatrix} l_{member\ 1} + jc \\ l_{member\ 2} + jc \\ \vdots \end{bmatrix}$

This method was originally suggested in Parkes 1978. The method is also outlined in He 2019. It means that two identical shorter members in line will be replaced by one longer element. The method, however, does not work to reduce the number of members to the smallest possible number.

Quick example (to be expanded) we can start with the case of a bridge, in a much larger problem than we had in part 1:

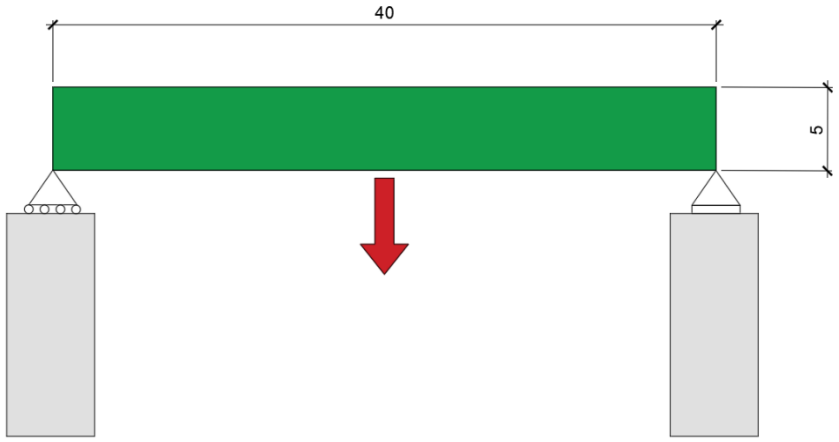


Figure 2.1: Truss envelope for first optimization task

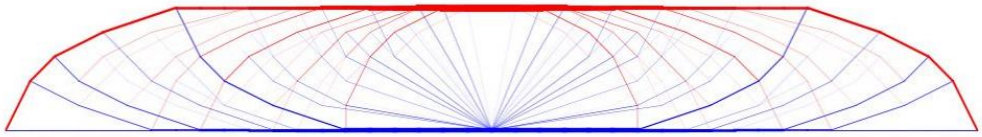
Figure 2.2: Possible nodes and members for truss optimization

The optimum truss was found using the code *BridgeWithMemberCost.py* available at https://github.com/saul-chaplin/TrussOptimization.git. In the code, to change member costs, just update the jc value in line 107.

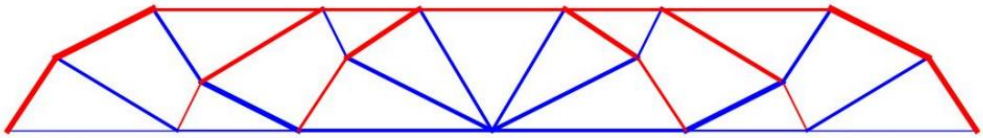
```
105  ##YOU CAN CHANGE THE OPTIONS HERE ##
106  case = 1 #Change between different cases
107  jc = 0 #Sets the joint cost value
108  plot_initial_layout = 0 #Plot initial layout
109  text = 0 #Switch to show/hide text on graphs. 1 to show, 0 to hide.
```

The following results were obtained:

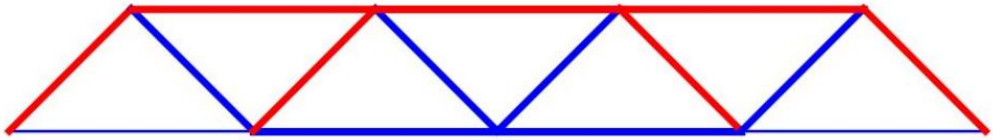
Member cost = 0



Member cost = .5



Member cost = 1



3. MINIMIZATION OF CONSTRUCTION COSTS

The previous inclusion of joint costs was insufficient to truly minimize for construction cost. For example, the solution, for a high joint costs, of problems like that in part 1.1, 1.2, or 2.1, does not converge to a triangle with a vertical in the center, as is clearly the intuitive optimal solution.

To attempt to minimize for construction costs, we can write an optimization statement:

Minimize the number of members

subject to equilibrium of forces

subject to max material stress

optional: subject to a max section size, or a member length

In mathematical terms, we can write:

$$Minimize(sum(\boldsymbol{ID}_{\mathrm{Bx1}}))$$

$$subject\ to\colon \mathbf{B}_{\mathrm{2NxB}} * (\mathbf{fint}_{\mathrm{Bx1}} \odot \boldsymbol{ID}_{\mathrm{Bx1}}) = \mathbf{Fext}_{\mathrm{2Nx1}} \odot \mathbf{dof}_{\mathrm{2Nx1}}$$

$$subject\ to\colon for\ all\ i\ in\ [1,B]\,, \qquad -\sigma_{max,comp} * \boldsymbol{a}(i) \leq \mathbf{fint}(i)\boldsymbol{ID}(i) \leq \sigma_{max,tension} * \boldsymbol{a}(i)$$

$$Optional\ subject\ to\colon \forall\ \mathrm{i} \in [1,B], \boldsymbol{a}(i) \leq \mathrm{a_{max}}\,, \boldsymbol{L}(i) \leq l_{max}$$

, where $\boldsymbol{ID}_{\mathrm{Bx1}}$ is a vector that of binary values, 1 or 0, where 1 represents the existence of a bar, and 0 represents that the bar does not exist. For instance, $\boldsymbol{ID} = [1,0,1,\dots]$ means that bar 1 exists, bar 2 doesn't and bar 3 exists, etc… The layout is shown below in figure 3.1. In this case, no max section size is set, and a max length of bars is set at $l_{max} = 3.5\ units$ to prevent a bar going all the way across the bottom. The possible members are shown in figure 3.1 Note the absence of a bar passing from bottom left to top right, as this would break the length constraint. The solution, which is completely intuitive, is shown in figure 3.2.

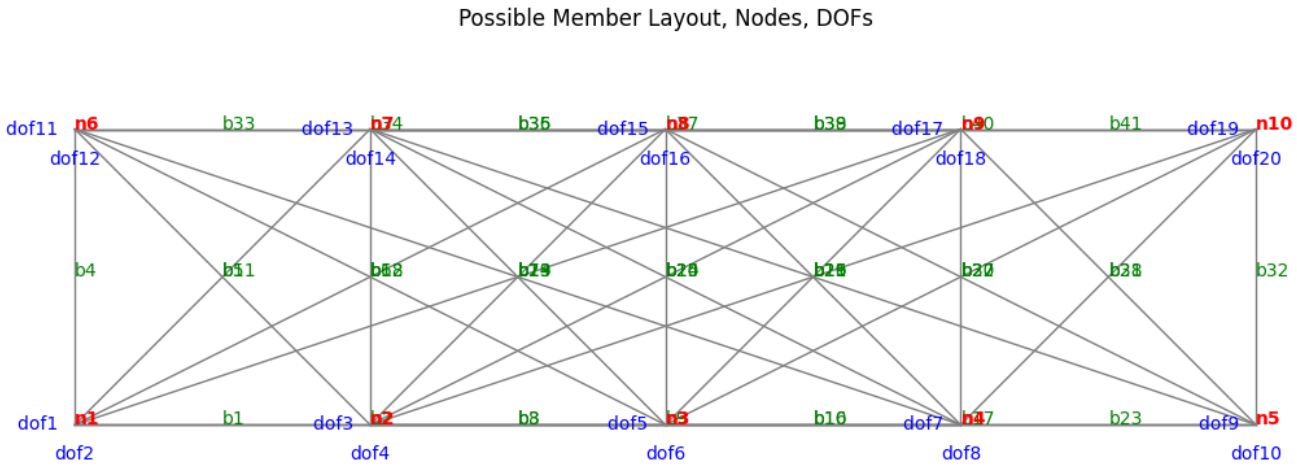


Figure 3.1: Layout

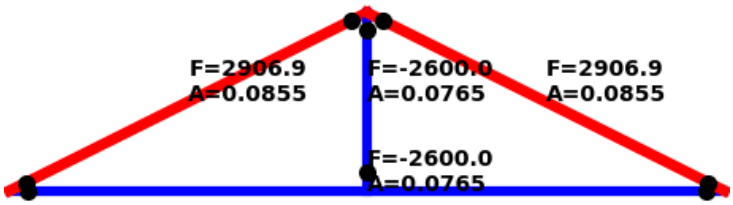


Figure 3(a): 4-bar truss without length restriction

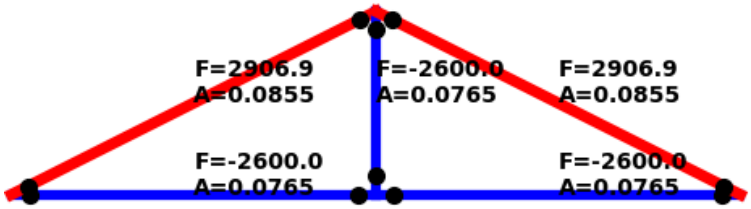
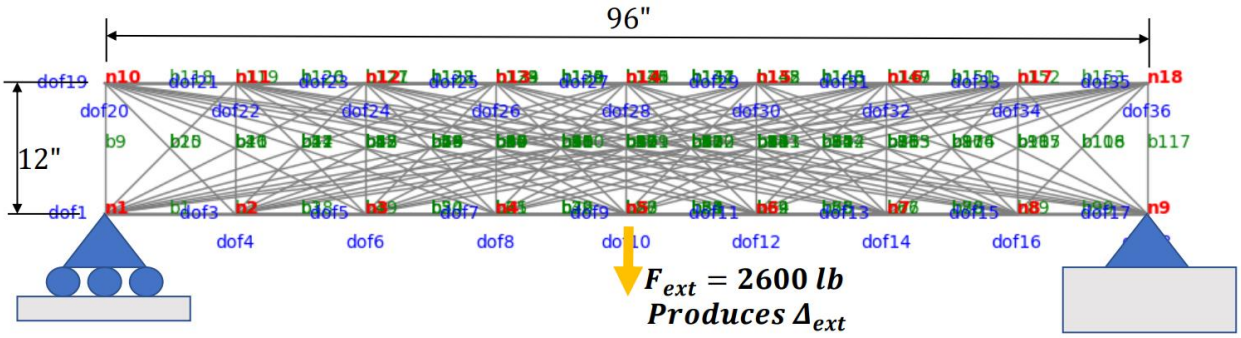
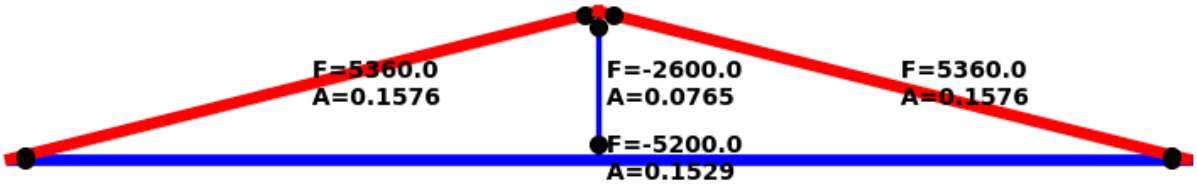


Figure 3(b): 5 bar truss with legth restriction preventing bar across the bottom of the bridge

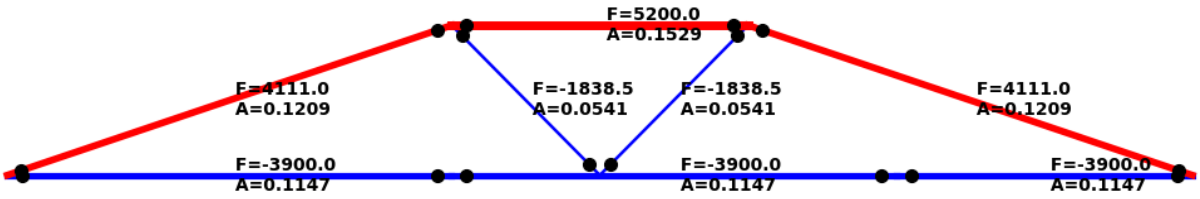
The above process can be scaled up to the following case, for a base structure with a layout grid that is 96”x1”, with the same loading and suppor conditions:



The optimized truss for no length constraints is shown below



The optimized truss for a ength constraints, $l_{max} = 3.5$ is shown below.



It is notable that the optimization quickly begins to take a long time (this last solution took a little over 5 minutes). This is because the problem is inherintelly a MILP (Mixed Integer Linear Programming) Problem, where 1’s and 0’s denote the presence or absense of bars. MILP problems are notoriously dififficult to solve. I am working on a way to simplify the problem to a linear programming problem, possibly by running successive linear integer optimization, and then penalizing an $\boldsymbol{ID}_{\mathrm{Bx1}}$ vector to become 0 or 1 if it is stuck in the middle, with an equaiton like:

$$\boldsymbol{ID} = \frac{\left(\frac{(1-\cos(\pi \boldsymbol{ID}))}{2} + r\boldsymbol{ID}\right)}{r+1}$$

, where r is the intensity of penalization. I will hopefully get this code working soon and add the results of a larger truss to this document.

4. STIFFNESS VIA MINIMIZATION OF INTERNAL STRAIN

In the previous examples, we attempted to minimize for (1) reduced weight and (2) constructability. But of course, deflection should also be of concern. Say we have a long bridge, and we calculate a truss form, that is sufficiently strong, and easy to build, but that when a truck passes over, the center of the bridge span deflects downward so much as to crack pavement (even though the members stay in the elastic region). Or what if we have designed a skyscraper, that is sufficiently strong in winds, but that moderate wind loads cause the top of the skyscraper to sway like a tree. This was the case for the Mjøstårnet, a 287 ft-tall wooden glu-laminated Highrise in Norway, for which the limiting factor of height was the displacements and resulting vibrations caused by winds.

Thus, it is clearly of interest to add the consideration of deflection to our optimization problem. The optimization for stiffness can be done in two ways. Firstly, we can optimize for stiffness by placing it in the objective function (statement 1). Secondly, we can add the stiffness as a constraint (statement 2).

Optimize for a low weight and high stiffness (statement 1)

Optimize for low weight, and the deflection cannot be more than 1" inches (statement 2)

To do these optimizations in the simplest way possible, we can use the concept of internal strain being equal to external work. If we have the truss below in figure 4.1, for instance, optimized in section 1 for minimum weight:

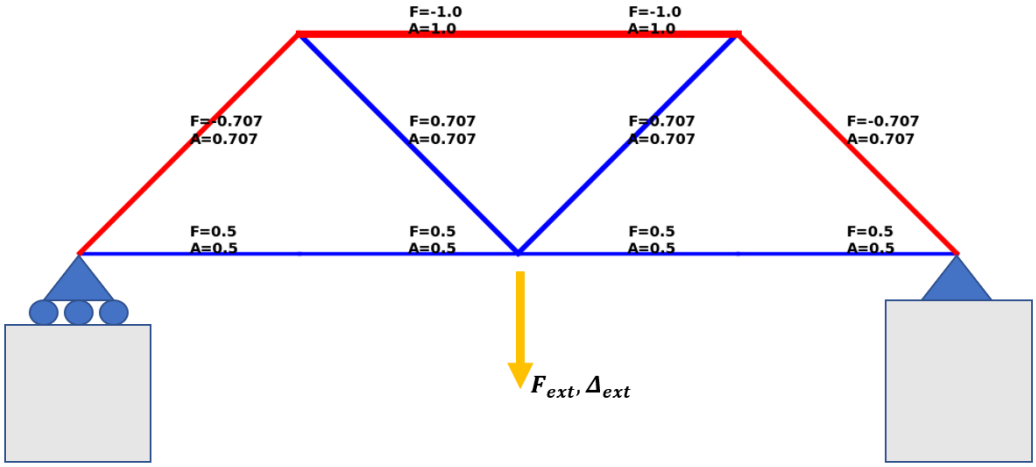


Figure 4.1: Truss layout

We can write that the external work is equal to the internal elastic strain energy.

$$W_{in} = Elastic\ Energy_{\ internal}$$

$$F_{ext} * \Delta_{ext} = \sum_{i=1}^{B} \frac{1}{2} k_i \Delta_i^2 \tag{4.1}$$

, where k_i is the stiffness of a member i, and Δ_i is the axial displacement in member i induced by the external loading.

We can express $k_{mem} = \frac{A_i E_i}{L_i}$ and $\Delta_i = \frac{f_i}{k_i} = \frac{L_i f_i}{A_i E}$. Substituting these values into equation (3.1), we have

$$F_{ext} * \Delta_{ext} = \sum_{i=1}^{B} \frac{1}{2}\left(\frac{A_i E}{L_i}\right)\left(\frac{L_i f_i}{A_i E}\right)^2 = \frac{1}{2}\sum_{i=1}^{B} \frac{L_i {f_i}^2}{A_i E} \tag{4.2}$$

, which can be solved for Δ_{ext} as

$$\Delta_{ext} = \frac{1}{2F_{ext}} \sum_{i=1}^{B} \frac{L_i {f_i}^2}{A_i E_i} \tag{4.3}$$

We can see that we have added no new variables to the problem, as the variables of optimization in section 1 were just L_i and A_i. (Assuming that E is a constant). We can now write the following optimization statement, that considers weight and stiffness:

$$minimize(volume, deflection) = minimize\left(\mathrm{C_w}(\mathbf{a}^{\mathrm{T}} \cdot \mathbf{l}) + C_\Delta \left(\frac{1}{2F_{ext}} \sum_{i=1}^{B} \frac{L_i {f_i}^2}{A_i E_i}\right)\right) \tag{4.4}$$

, where C_w and C_Δ are constants that specify the "cost" of weight and of deflection, respectively. The variables in the equation above will be subject to the same constraints shown in section 1., which are (1) Force equilibrium and (2) Section stresses must be within a certain acceptable range. Equation 4.4 is the mathematical equation for the (statement 1)

The difficulty of the objective function above is that the function is non-linear. This is because we now include the deflection term, which has several variables of optimization multiplied and divided by one another, in contrast to before in section (1) and (2), where we had just a linear combination of the variables of optimization. Unfortunately, the open-source Python CVXP solvers cannot solve this non-linear optimization problem. AMPL is a powerful optimization toolbox that can solve this non-linear problem (it is also used for everything from emergency room personnel scheduling in large hospitals to planning the extraction world's largest copper-mine in Chile based, to optimizing flight-paths and designs of electric drones, see: https://ampl.com/about/industry-use-cases/)

AMPL uses its own coding language, based on the syntax of C++. The handbook is available here: https://ampl.com/wp-content/uploads/BOOK.pdf. A free community version of the optimization toolbox is available here: https://ampl.com/ce/.

The optimization code previously written in Python was translated into AMPL. The structure of the optimization problem had to be changed to match the fact that AMPL uses tables as inputs, but all the concepts remain the same. The Python code that generates the AMPL model .mod file, as well as the AMPL .run file, is on my Github: https://github.com/saul-chaplin/TrussOptimization

The following input properties were used.

Matierial:	Steel
σ_{max}	$34\ ksi$
σ_{min}	$-34\ ksi$
Young's Modulus, E	$29000\ ksi$

An optimized version of the truss is shown below:

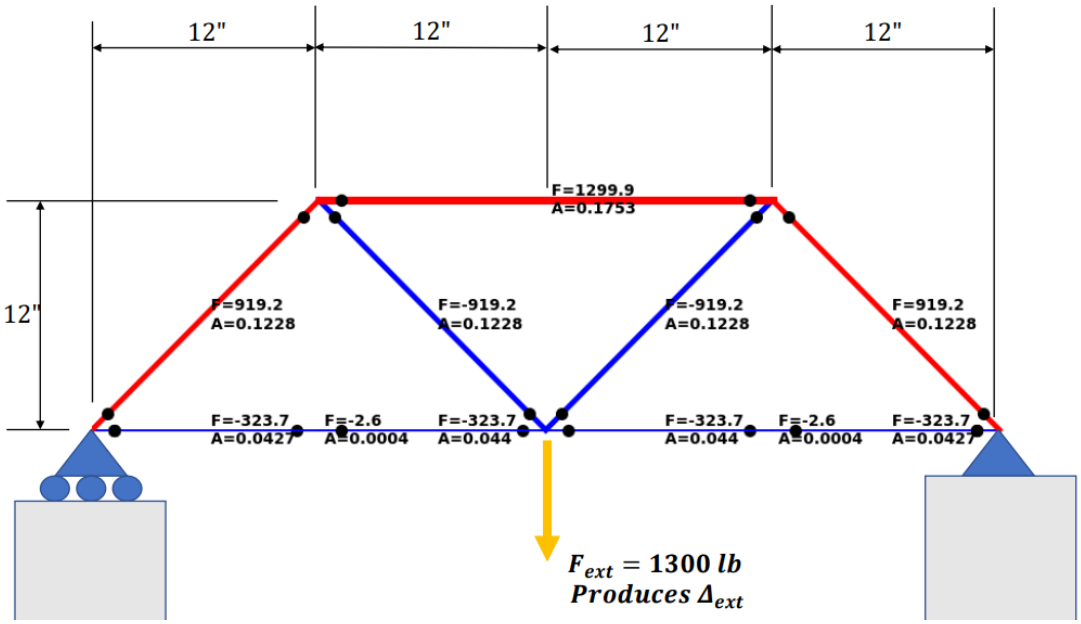


Figure 4.2: Optimized truss for a balance of weight and deflection

This optimized truss obtains a deflection of .012 inches, with a total weight of 4.83 lb. The solution was found with the solver KNITRO through AMPL with a run time of .0171 s.

5. MULTI OBJECTIVE FUNCTION WITH WEIGHT, STIFFNESS, AND CONSTRUCTABILITY

6. MULTIPLE MATERIALS WITH DIFFERENT COSTS, WEIGHTS, PROPERTIES (COMING SOON)

7. WHERE TO GO NEXT

- 3D optimization for everything
- Non-“discrete” domain optimization using finite elements domain, using homogenization, and then penalization to get something buildable.
- Figure out how tf SIMP optimization works. Here is a sick paper which references it’s use for trusses: https://www.e3s-conferences.org/articles/e3sconf/pdf/2019/66/e3sconf_eece18_04017.pdf
- Can I generalize for frame structures? Optimize moment frames? Allow members to take moment (so, for example, you can load in the middle of a beam).
- Soooo much more….

REFERENCES

Abrahamsen, R. (2017). Mjøstårnet - Construction of an 81 m tall timber building. Internationales Holzbau-Forum IHF. https://www.moelven.com/globalassets/moelven-limtre/mjostarnet/mjostarnet---construction-of-an-81-m-tall-timber-building.pdf

Agrawal A, Verschueren R, Diamond S, Boyd S (2018) A rewriting system for convex optimization problems. J Control Decision 5(1):42–60. https://www.cvxpy.org/

"A Python script for adaptive layout optimization of trusses", L. He, M. Gilbert, X. Song, Struct. Multidisc. Optim., 2019. Article Link: https://link.springer.com/article/10.1007/s00158-019-02226-6 Code download link: Python Script for Truss Layout Optimization (shef.ac.uk)

Parkes E (1978) Joints in optimum frameworks. Int J Solids Struct 11(9):1017–1022. https://www.sciencedirect.com/science/article/pii/002076837590044X

(by my prof at Universidad Catolica in Santiago, Chile, uses homogenization method)

- Gutiérrez, Sergio. OPTIMAL STRUT-AND-TIE MODELS USING FULL HOMOGENIZATION OPTIMIZATION METHOD. https://www.researchgate.net/publication/257194208_Optimal_Strut-and-Tie_Models_Using_Full_Homogenization_Optimization_Method
- Experimental evaluation of optimized strut-and-tie models for a dapped beam. Experimental evaluation of optimized strut-and-tie models for a dapped beam - Oviedo - 2016 - Structural Concrete - Wiley Online Library

Other sweet articles:

Fairclough, Helen et. Al. “Theoretically optimal forms for very long-span bridges under gravity loading.” https://royalsocietypublishing.org/doi/10.1098/rspa.2017.0726

https://www.e3s-conferences.org/articles/e3sconf/pdf/2019/66/e3sconf_eece18_04017.pdf

Zegard, Tomás, y Glaucio H. Paulino. “GRAND3 — Ground Structure Based Topology Optimization for Arbitrary 3D Domains Using MATLAB”. Structural and Multidisciplinary Optimization, vol. 52, no 6, diciembre de 2015, pp. 1161–84. DOI.org (Crossref), https://doi.org/10.1007/s00158-015-1284-2