ASEN 2001 Lab 2: Design, Testing, and Construction of a Space Truss

Abdullah Al Ameri
University of Colorado Boulder, Sophomore, Aerospace Engineering

Eric Hunnel
University of Colorado Boulder, Sophomore, Aerospace Engineering

Mia Abouhamad
University of Colorado Boulder, Sophomore, Aerospace Engineering

Kunal Sinha
University of Colorado Boulder, Sophomore, Aerospace Engineering

Johann Kailey-Steiner
University of Colorado Boulder, Sophomore, Aerospace Engineering

Fig. 1 Group picture with the Built Truss

Two space trusses were designed using MATLAB in conjunction with the fundamental principles of static structures and in utilizing the method of joints. These trusses had to be statically determinant and able to stand under their own weight on three or four supports locations on specific support platforms (figure 1,2). In both cases, six support reactions were provided by the support platforms. MATLAB code was written to model and analyze the forces on the truss, provide an output file with the results, run error and failure analysis, and plot the results. From these two designs, one was chosen after careful analysis of its properties with the assist of MATLAB. This truss was built using the provided materials and tested under the conditions simulated in the code. The truss was then loaded with extra weight loads until failure and then the maximum load that could be carried by the truss was compared to the predicted maximum load from the MATLAB code. The final had a length of 36.5 inches and could carry a load of 1.86 Newtons.

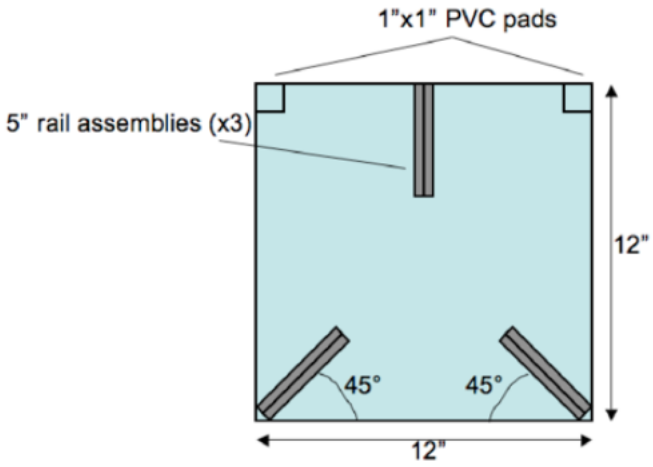


Fig. 2 Pads Used to provide support reactions

I. Nomenclature

POF	=	Probability of failure
FOS	=	Factor of safety
g	=	gravitational acceleration
σ_m	=	Uncertainty in magnets strength
μ_m	=	Mean magnets strength
P_{dsr}	=	Assumed failure probability
F_{dsr}	=	Desired force in the bars
$icdf$	=	Inverse cumulative distribution function
ΣF_x	=	Sum of the forces in the x direction.
ΣF_y	=	Sum of the forces in the y direction.
ΣF_z	=	Sum of the forces in the z direction.

II. Introduction

A truss is one of the most practical and most common structures used by engineers. A truss is essentially a triangulated system of straight interconnected structural elements. The most common use of trusses is in supporting frames for buildings, bridges, airplanes fuselages, and any body that needs to support loads. The main reasons for using trusses are their long spans, lightweights reduced deflection [1] compared to plain members. They give engineers an opportunity to support considerable loads.

The reaction forces on a space truss can be solved using the method of joints. This method takes advantage of the static equilibrium equations at each joint, and hence the name method of joints. The static equilibrium equations states that all forces in the x, y, and z, directions should be equal to zero:

$$\Sigma F_x = 0, \Sigma F_y = 0, \Sigma F_z = 0 \tag{1}$$

The main purpose of this lab was to study the statics behind a truss and build two different statically-determinate trusses with the guidelines given to us. The truss was required to stand on either three or four supports on provided support pads and span between two tables. This had to be achieved using the materials provided for the lab including: steel balls for joints, fiberglass tubes, sleeves for members, and neodymium magnets to be inserted in the tubes and used to connect members to joints. The tubes were given in eleven different lengths spanning from 6 to 24 inches and sleeves could be used to combine two tubes creating a custom member length [6]. After both designs were found to be statically determinant, one design was chosen based on careful analyses of the factor of safety, the support reactions, and

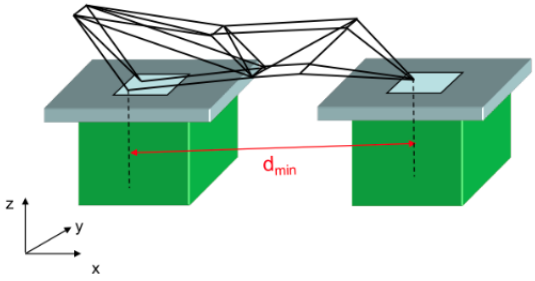


Fig. 3 How Truss Will Be Set Up With Pads

maximum forces within the bars. A thorough failure analysis was done. The main goal was to make sure that the joints will not experience more than its strength, which is 4.8 ± 0.4 N. The chosen truss was then built with the materials provided. The truss was then tested for its maximum loading capacity. The experimentally found value of external load was then compared with the predicted load that the truss could handle. The difference between them was justified by a thorough inspection of the assumptions made while designing the truss and analyzing it.

III. Truss Design

The idea behind design 1 (figure 4) was to have two right triangles comprised of smaller internal components meet each other in the middle to form one side. For the most part, only the 6in, 8in, and 10in bars were used to make the truss, but one 12in was used in the middle of both truss halves giving them their max height of 12in. The base was made out of 8in by 8in squares while the idea for the sides was to use 6, 8, 10 triangles. In order to make the truss statically determinant, bars were put into the diagonals of the squares in the base and the top of the truss. The bars in the diagonals were all made using sleeves to get the required lengths.

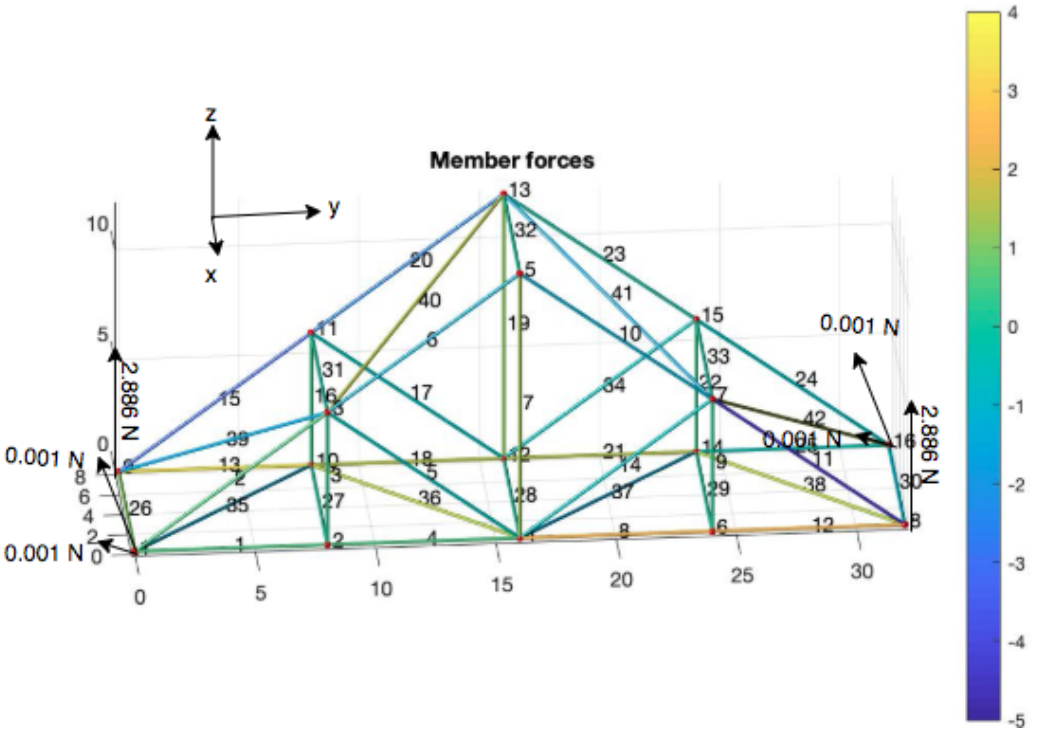


Fig. 4 Internal forces and FBD of Design 1

The goal of design 2 (figure 5) was to create a truss using only 6, 8, and 10-inch bars. In order to make the truss statically determinate, right triangles were made using these bars. Five equilateral triangles were made out of six-inch bars, which were connected by eight-inch bars horizontally, and ten-inch bars along the hypotenuses. It was important to use the eight-inch bars horizontally along the base to extend the length of the truss. Design 2 was very repetitive making it easy to assemble. All of these qualities made design 2 simple, reliable, and easy to modify.

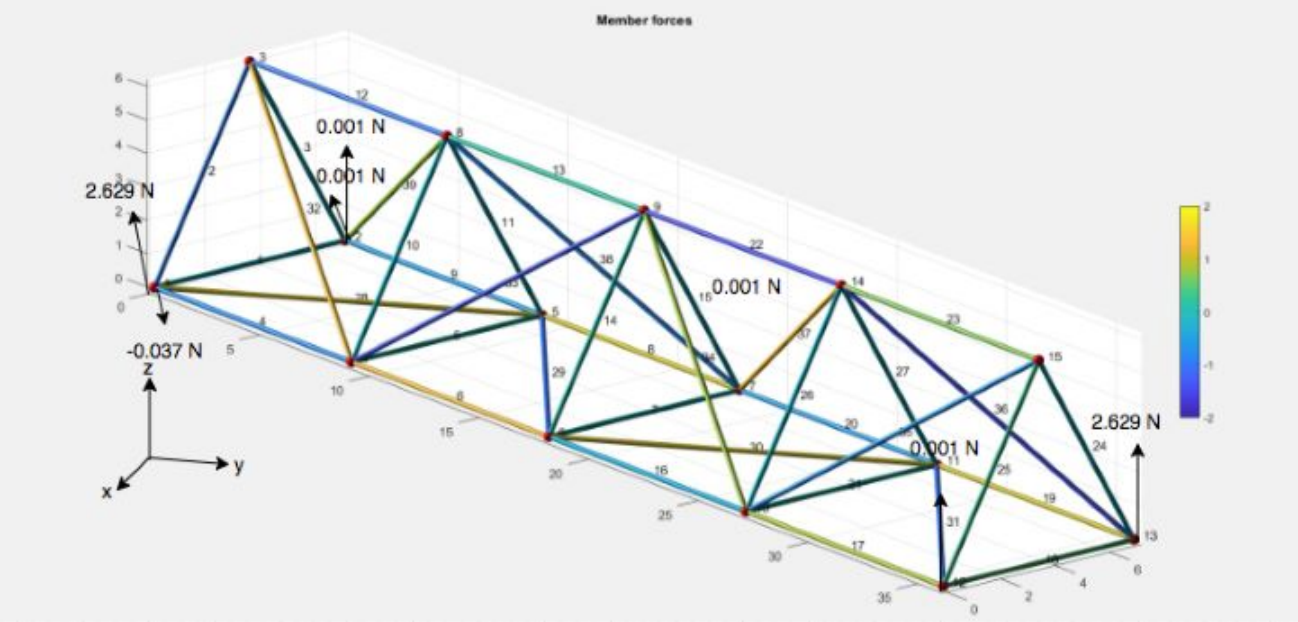


Fig. 5 Internal forces and FBD of Design 2

IV. Design Analyses

A. Concept Models

An important first step in the design process is to develop concept models. These give an idea of the geometry that can be used to create larger designs. In this lab, two simple example truss input files were given to serve as concept models. The code was formulated using these models. Given the results from the code, the best geometry for a larger truss could be determined. The analysis of these example trusses as well as testing simple shapes such as triangles and squares with the code revealed that triangles were the most effective at distributing load and preventing critically loaded joints. The primary design component of trusses would be triangles. (Free body diagrams of these example cases can be found in the appendix in the CDR).

B. Detailed Design and Computational Analysis

When analyzing the trusses, it was assumed that the only external load acting on the truss was from the weight of the truss itself. In addition to this, it was assumed that the bars have a uniform linear density. For the sake of simplification, half of the weight of each bar including internal magnets was estimated to act at each joint that a bar is attached to. The magnets that were used to connect the bars to joints were also assumed to be uniform in mass and density, and their weight was applied at the end of each member. For each design, support reactions and member forces were calculated using code formulated with MATLAB. A GUI provided by the Teaching Assistants to create and visualize statically determinant trusses. This GUI allowed for individual joint, member, support, and reaction force adjustment and output a file used as the input for structural analysis code. This input file contained the number of joints, members, reaction forces, however it did not contain external load line, which should be added manually to avoid code problems. The coordinates of each joint, bar connectivity, reaction force vectors, and external load vectors were also included in the input file. The code used the information provided in the input file to find reaction forces, tension and compression

in each bar, and a graph displaying the truss and its internal forces. The code also runs a Monte Carlo simulation to analyze the likelihood of failure given a reasonable factor of safety. The safety factor is given by

$$FOS = F_{nominal,max}/F_{dsr} \tag{2}$$

where F_{dsr} is estimated by the $icdf$ function with inputs P_{dsr}, μ_F, and σ_F for a normal distribution, given n:

$$F_{dsr} = icdf('normal', P_{dsr}, \mu_F, \sigma_F) \tag{3}$$

C. Sensitivity Analysis

When analyzing design 1, the maximum tension was found to be 3.69N in bar 13 between joints 9 and 10 and the maximum compression was 4.6253N on bar 11 between joints 7 and 8 (Fig. 1). With a safety factor of 1.086 and an assumed probability of failure of 17%, the Monte Carlo simulations calculated a probability of failure of 18.03%. To achieve a probability of failure of 100%, 36 small magnets and 36 steel balls could be attached to the truss (Table 1). In design two, the maximum internal tension and compression forces were 4.146N and 5.034N respectively. The maximum tension occurred in bar 32 between joints 3 and 4 and the maximum compression occurred in bar 22 between joints 9 and 14 (Fig. 2). With a safety factor of 1.01058 and an assumed probability of failure of 45% the Monte Carlo simulations calculated a probability of failure of 69%. To achieve a probability of failure of 100%, 24 small magnets and 24 steel balls could be attached to the truss totaling in an added external load of 7.4957N (Table 1).

At first, after determining the overall safety of the truss and accounting for the probability for failure and safety factor, Design 1 was chosen as the superior design because it had a higher safety factor and lower failure probability (table 1). Unfortunately, in the process of checking over the design before CDR, an error in member length calculations was discovered. An attempt to back-track the error and fix it was made but it was not successful. The diagonal members spanning the base of the truss (bars 35 through 38) required lengths of approximately 11.3 inches. As the shortest provided bar length was six inches, the shortest member length if sleeves were used was 12 inches. Clearly, this design was no longer viable and Design 2 would need to be used as the final truss. Although Design 2 is not as stable as Design 1 based on the analysis estimations, it was the backup plan for the group, and it had to be used. The main differences between the two designs are stated in table 1. Design 1 is stronger, but it is heavier and shorter than Design 2.

Design	Assumed POF	Calculated POF	# Magnets	# Spheres	FOS	Truss Mass (kg)	Truss Length (in)
1	17%	18.03%	36	36	1.0863	0.6311	32
2	50%	60%	24	24	1.00	0.5424	36

Table 1 Designs comparison and Maximum Added Magnets and Spheres Before a Failure Probability of 1

In short summary, the outcomes of the final assembly and test turned out to match the predictions at some occasions and it did not at other occasions. A detailed discussion is provided at section VI.

V. Model Validation

The chosen design was constructed in lab and was stable under its own weight. To test the strength of the design, extra magnets were added to certain joints at the bottom of the truss until it failed. In total, the truss was able to withstand 19 small magnets and 19 steel balls. The predicted amount was 24 magnets, and 24 steel balls (Table 1). This discrepancy is to be expected and can be explained. This discrepancy is likely due to the imperfect nature of the building method and supplies. It is also likely that some of the assumptions might have resulted in different expectations than reality, and this is where factor of safety plays a big role. A deeper and thorough reflection about this process is to be found at section VI.

VI. Discussion

The built design exhibited a fragile behavior and was likely to fail if the external loads were not placed carefully. Accidentally, the truss was broken due to the placement of bars or random physical interactions with the surroundings in the lab environment (people, tables, etc). This was predicted by a Monte Carlo simulation (Table 1). In addition, it was predicted that the design is likely to fail at its critical points, bar 22, which was connected to joint 9 and joint 14 (See Design Analysis). When the truss was tested to its fullest capacity, the maximum forces at the bars at moment of failure were at the same location (Fig. 5, Appendix) and hence the theoretical prediction was matched.

During the experiment, however, loading the truss to its fullest capacity was not an attainable goal due to assumptions made during analysis that were not necessarily valid during the build. Firstly, it is very likely that imperfections in the cut of the bars, the surfaces of magnets and joints, and not including the mass and the force the glue (that was used to connect magnets to bars) affected the actual truss, but in the analysis, those were assumed to be negligible. Second, another problem was due to the inexact location that the joints and bars should be placed at compared to the MATLAB code that assumed exact locations of the bars and joints. Thirdly, during the analysis, the weight load was distributed at the joints rather than the center of volume of the whole truss or each bar. As a result, the analysis assumed more force on the joints that there actually is. This was noticed in the practice of building the truss. The Truss managed to hold its own weight and did not reflect the high chance of failure Monte Carlo simulations predicted (See table 1). Lastly, the analysis did not account for the likelihood of repulsive magnetic forces when the truss is built. Those repulsive forces can cause some of the truss members to fail during the building process. This being said, one of the major elements disregarded during the analysis is the tendency of the magnets to interact with each other throughout the truss.

Overall, it is more likely that the probability of failure will increase, and thus the actual safety factor is more likely to decrease. Hence, it is highly recommended that the implied safety factor decrease as analytical assumptions are made. The chosen safety factor should decrease as well. A better performance would require a reasonable set of assumptions, yet as the assumptions increase it might be necessary to compensate for them by a smaller choice of safety factor. The desired safety factor was intended to be a value higher 1.6 according to the recommendation of California Engineering and Technology Alliance for trusses safety factor [4]. However, the actual applied safety factor was below this number (table 1). The actual applied safety factor was largely governed by being within the bounds that the Monte Carlo simulations predict. In a typical space truss, the safety of factor should have been selected and taken into account before proceeding to design.

VII. Conclusions

To sum up, it is very important to notice that the final analyses for both designs reveal something extremely crucial to the design process: the choice of good geometry and redundancy in designs. It is likely that Design 1 had a better geometric structure compared to design 2 even though both had a very close length and mass. Given the material learned in ASEN 2001, informed decisions were able to be made based on an understanding of the forces involved in a 3d truss made from bars and joints. The factor of safety helped determine how sturdy the truss was and how much weight the truss was designed for. However, the engineering skills of the group are clearly not fully-fledged at the moment: more is to be learned before applying these skills in a real-world setting.

Part of the engineering process that may have helped in preventing the error made in design one is the PDR. According to Defense Acquisitions group, the PDR process can provide an "updated risk assessment for the Engineering, Manufacturing and Development (EMD) Phase" [5]. Some of the assumptions that have been made may have affected the analysis process (See Discussion). These assumptions can be very difficult to re-create in a real-world setting so it is important to choose assumptions carefully and replicate the analytical conditions as best as possible. Making these improvements to the engineering process can improve results dramatically.

VIII. References

[1] "Trusses," *SteelConstruction.info* [online], https://www.steelconstruction.info/Trusses [retrieved 03 Nov. 2018]

[2] Hibbeler, R. C., "Chapter 5 Structural Analysis," *Statics and Mechanics of Materials*, fifth ed., Pearson Education,

Inc., 2016, pp. 239

[3] "Factors of Safety - FOS - Are Important for Engineering Design," *The Engineering Toolbox* [online], https://www.enginee
ringtoolbox.com/factors-safety-fos-d_1624.html[retrieved 3 Nov. 2018]

[4] Dawson, G., "Truss Factor of Safety and Characterization - ppt video online download," *SlidePlayer, California Engineering and Technology Alliance* [online], https://slideplayer.com/slide/8718011/ [retrieved 3 Nov. 2018].

[5] "Preliminary Design Review (PDR)," *AcqNotes* [online], http://acqnotes.com/acqnote/acquisitions/preliminary-design-review.[retrieved 3 Nov. 2018]

[6] "ASEN 2001 Lab 2: Design, Construction, and Testing of a Truss Structure – Fall 2018," *AcqNotes* [online], https://canvas.colorado.edu/courses/21255/files/1851273?module_item_id=964201.[retrieved 10 Oct. 2018]

IX. Appendices

External loads:	Joint-id	Force-x	Force-y	Force-z
	1	0.00	0.00	-0.29
	2	0.00	0.00	-0.27
	3	0.00	0.00	-0.47
	4	0.00	0.00	-0.52
	5	0.00	0.00	-0.30
	6	0.00	0.00	-0.27
	7	0.00	0.00	-0.47
	8	0.00	0.00	-0.29
	9	0.00	0.00	-0.30
	10	0.00	0.00	-0.39
	11	0.00	0.00	-0.34
	12	0.00	0.00	-0.40
	13	0.00	0.00	-0.44
	14	0.00	0.00	-0.39
	15	0.00	0.00	-0.34
	16	0.00	0.00	-0.30

Fig. 6 External Loads on Design 1

```
25
26    External loads: Joint-id  Force-x      Force-y      Force-z
27                     1         0.00         0.00        -0.28
28                     2         0.00         0.00        -0.28
29                     3         0.00         0.00        -0.38
30                     4         0.00         0.00        -0.79
31                     5         0.00         0.00        -0.79
32                     6         0.00         0.00        -0.69
33                     7         0.00         0.00        -0.69
34                     8         0.00         0.00        -0.40
35                     9         0.00         0.00        -0.40
36                    10         0.00         0.00        -0.79
37                    11         0.00         0.00        -0.79
38                    12         0.00         0.00        -0.28
39                    13         0.00         0.00        -0.28
40                    14         0.00         0.00        -0.40
41                    15         0.00         0.00        -0.28
42
```

Fig. 7 External Loads on Design 2

```
25
26    External loads: Joint-id  Force-x      Force-y      Force-z
27                     1         0.00         0.00        -0.28
28                     2         0.00         0.00        -0.28
29                     3         0.00         0.00        -0.38
30                     4         0.00         0.00        -0.79
31                     5         0.00         0.00        -0.79
32                     6         0.00         0.00        -0.69
33                     7         0.00         0.00        -0.69
34                     8         0.00         0.00        -0.40
35                     9         0.00         0.00        -0.40
36                    10         0.00         0.00        -0.79
37                    11         0.00         0.00        -0.79
38                    12         0.00         0.00        -0.28
39                    13         0.00         0.00        -0.28
40                    14         0.00         0.00        -0.40
41                    15         0.00         0.00        -0.28
42
```

Fig. 8 External Loads for Failure Design 2

CDR

The following pages contains the CDR.

CDR: LAB 2

ASEN 2001, STATICS AND STRUCTURES

GROUP 1:

1- ABDULLAH AL AMERI

2-KUNAL SINHA

3-MIA ABOUHAMAD

4-ERIC HUNNEL

5-JOHANN KAILEY-STENIER

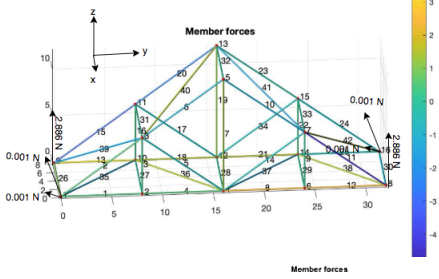
MECHANICAL MODELING

ESSENTIAL FREE BODY DIAGRAMS, EXTERNAL LOAD

Figure 1: FBD for Design 1 and 2 respectively.

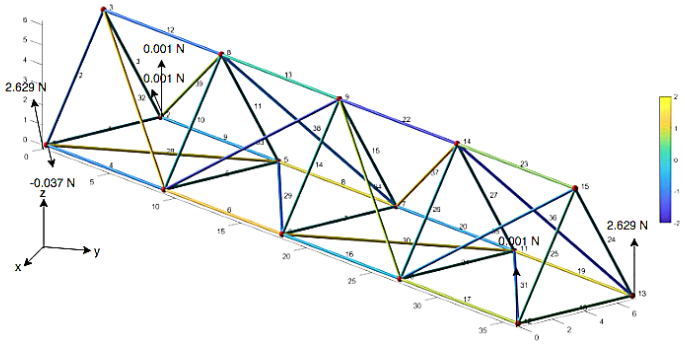
Design 1:

- LENGTH: 32 INCH
- MASS: 0.6311 KG
- # BARS : 42
- # JOINTS : 16
- CRITICAL POINTS: BAR 11 (JOINT 7 AND 8)



Design 2:

- LENGTH: 36 INCH
- MASS: 0.5424 KG
- # BARS : 39
- # JOINTS : 15
- CRITICAL POINTS: BAR 22 (JOINT 9 AND 14)



EXTERNAL LOAD ASSUMPTIONS

We are assuming that the only external load acting on our truss is the weights of the items used to make it. In addition, we assume that the bars have uniform linear density. The weight of the bars, magnets, joints, and sleeves act equally on both joints that it is connected to. The weights is to be distributed equally on each joint (for instance the weight of the bars will be carried by each joint the bars connected to, each joint takes half of the load).

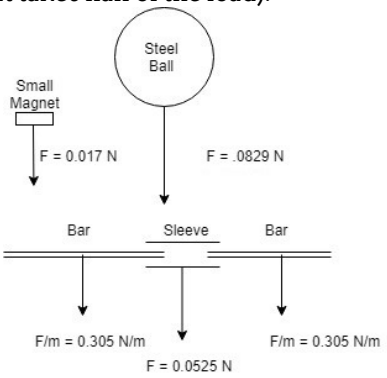


Figure 2: FBD For the truss components and their weight contribution.

CODE VERIFICATION

CLASS EXAMPLES

The following examples were used to test the validity of the MATLAB code, provided a picture from the the output file and the resultant plot, adjusted by adding the reaction forces. The external loads are not displayed but they can be seen at the output file

EXMAPLE 1

```
HOBBY 3-D Truss analysis
------------------

Date: 08-Oct-2018 11:43:03

Input file: test3d_2.inp

Joints:          Joint-id  x-coordinate y-coordinate
                       1          0.00         4.00         0.00
                       2         -3.00         4.00         0.00
                       3         -3.00         0.00         3.00
                       4          0.00         0.00         3.00
                       5          0.00         0.00         0.00
                       6         -3.00         0.00         0.00


External loads: Joint-id  Force-x     Force-y
                       1      200.00       300.00      -500.00
                       2        0.00       400.00         0.00

Bars:            Bar-id    Joint-i      Joint-j     Force    (T,C)
                       1          1            2       300.000     (C)
                       2          1            5       366.667     (C)
                       3          1            3       971.825     (T)
                       4          1            4         0.000     (C)
                       5          2            3         0.000     (C)
                       6          2            5       500.000     (T)
                       7          2            6         0.000     (C)
                       8          3            4       500.000     (C)
                       9          3            6       300.000     (C)
                      10          4            5         0.000     (C)
                      11          4            6       424.264     (T)
                      12          5            6       300.000     (C)

Reactions:       Joint-id  Uvec-x       Uvec-y      Uvec-z       Force
                       3        0.00         1.00       0.000    -666.667
                       3        0.00         0.00       1.000     200.000
                       4        1.00         0.00       0.000    -200.000
                       4        0.00         1.00       0.000       0.000
                       4        0.00         0.00       1.000     300.000
                       5        0.00         1.00       0.000     -33.333
```

Figure 3: Output results for test case 1

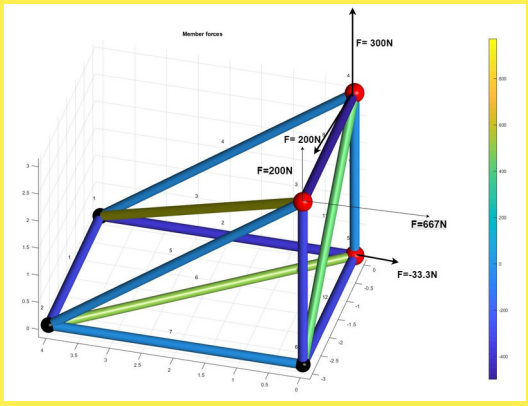


Figure 4: 3D Plot and FBD for test case 1

EXMAPLE 2

```
3-D Truss analysis
------------------

Date: 21-Oct-2018 20:28:55

Input file: test3d_1.inp

Joints:          Joint-id  x-coordinate y-coordinate z-coordinate
                       1         6.00         0.00         0.00
                       2         3.00         4.50         0.00
                       3         0.00         0.00         0.00
                       4         3.00         2.00         6.00


External loads: Joint-id  Force-x     Force-y     Force-z
                       4        0.00         0.00        -8.00

Bars:            Bar-id    Joint-i      Joint-j     Force    (T,C)
                       1          1            2       0.890     (T)
                       2          2            3       0.890     (T)
                       3          3            1       0.617     (T)
                       4          1            4       2.593     (C)
                       5          2            4       3.852     (C)
                       6          3            4       2.593     (C)

Reactions:       Joint-id  Uvec-x       Uvec-y      Uvec-z       Force
                       1        1.00         0.00        0.00      -0.000
                       1        0.00         1.00        0.00      -0.000
                       1        0.00         0.00        1.00       2.222
                       2        0.00         0.00        1.00       3.556
                       3        0.00         1.00        0.00       0.000
                       3        0.00         0.00        1.00       2.222
```

Figure 5: Output results for test case 2

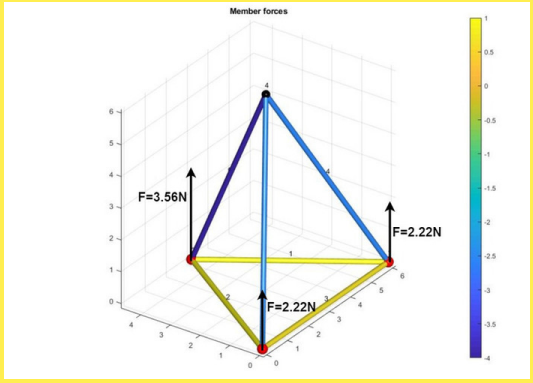


Figure 6: 3D Plot and FBD for test case 2

SAFETY FACTOR RATIONAL AND DESIGNS ROBUSTNESS AND FAILURE TOLERANCE

It is important to differentiate between the chosen safety factor and the actual applied safety factor that is calculated and predicted from the code. The chosen safety factor should be taken into consideration before proceeding into design. It is clear to us that the first design is less likely to fail based on the obtained data from the analysis and failure simulations. The lowest possible safety factor is a safety for Aerospace applications is 1.3, and thus something around that range was chosen. For Design 2 Monte Carlo simulations predicted a very high chance of fail, and thus to stay within the safe range a higher safety factor must be considered. Although the range for Aerospace applications goes up to 2, for this application we are allowed to go beyond that, and thus 2.5 was chosen. The implied Safety factors are not the same as the chosen, for the implied is what is actually there based on the analysis.

Design 1:

Assumed probability of fail: 17%
Monte Carlo Probability of fail: 18.03%
Chosen safety factor: 1.4
Applied safety factor: 1.0863

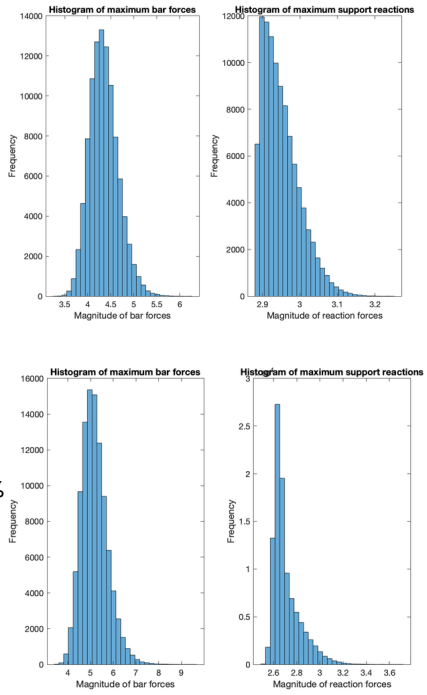
Maximum loading capacity (theoretical): 36 magnets and and 36 spheres

Design 2:

Assumed probability of fail: 50%
Monte Carlo Probability of fail: 60%
Chosen safety factor: 2.5
Applied safety factor: 1.00

Maximum loading capacity (theoretical) : 24 magnets and 24 spheres

Figure 7: Monte Carlo simulations result for Design 1 and 2 respectively.



SENSITIVITY ANALYSIS
AND CHOSEN DESIGN

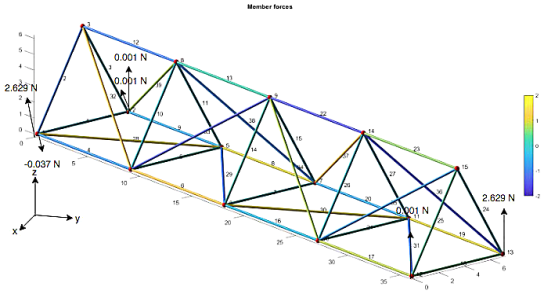
DESIGN 2

After looking through the truss designs, their free-body diagrams, their factors of safety, and their Monte Carlo simulations, we ended up choosing design 2.

This was because the design was easier to build with the added error of the size of the bar with the magnets. Moreover, when this design inevitably failed due to too much stress on a point, it was easier to put back together since all the triangles are the same.

This design was also longer, which made up for its higher probability of failure. We are confident that the truss will still stand under its own weight.

Along with these reasons, when checking over the final design for design 1, we found an error in some of the member length calculations.. The diagonals along the base of the truss should be about 11.3 in but this length is not possible. Design 2 is perfectly viable.



BUILDING AND FABRICATION PLAN

YES!

Design 2 can be made with the truss bars and sleeves given to us.

	Design 1	Design 2
Bar Lengths / Item	Number used	Number used
12"	2	0
10"	12	12
8"	18	12
6"	18	15
Sleeves	8	0

Table 1 : Initial Design Material Use Estimations

All the lengths of our bars were checked by looking at the data from our MATLAB code and found that in Design 2 every bar complies with the lengths of the bars which will be given to us. In addition, we don't exceed the amount of each bar given to us.

MATLAB codes

The MATLAB Analysis utilizes some a group of generic functions that will be the same, and the only thing that will change is the main script that will call each function. Each script is different, and there is a total of two scripts, one for each design. Detailed REAMME.md files are provided to guide any user. It is important to notice that the output files from the GUI do not have any external loads by default. It is is the user responsibility to add manually an external line the creates external loads in the beginning as pre-existing loads. If there is no external loads, initially, a 0 external load line should be inputted, for instance something like *1 0 0 0* could be placed as external loads, which means simply that there is 0 loads in x, y, z direction at joint 1.

Furthermore, one DesignInNOut.m file could be used to run the analysis, the only thing that needs to be changed will be line 5 and 6 from the code which corresponds to the name of the input file and the output file. However, there is two different folders for each design provided, and again, each folder has its own README.md to guide the user.

The group of the generic/uniform codes are the following:

1 - ExtractTruss.m: This function extracts information from a given .txt file. The text file is commented by # symbols and the code will go and get everything in between.

2- addweight.m: This function will adjust given vectors of external loads and their locations and return two vectors, one about the location of the joints, and one about their actual magnitude in x y z.

3 - FAA.m: Force analysis function. It runs the static equilibrium equations and outputs analysis results.

4 - writeoutput.m: function made by Kurt Maute, provided to us by the TA's. The function will write the outputs of the force analysis in a .txt file.

5 - MonteCarls.m: Stochastic analysis of 3-D statically determinate truss by Monte Carlo Simulation. Only positions and strength of joints treated as random variables.

6- plottruss.m: plot 3D-truss, bars colored according to force supported nodes are plotted in red non-supported nodes are plotted in black.

The Main scripts, TrussInNOutDesign1.m,TrussInNOutDesign2.m : The main script that controls everything. All you have to do is to run it and it will call all other functions. The first lines has the name of the input/output files, thus it can be used with other input files that has the same format to do the exact same thing. All you have to do to use it with another file is to change the name of the input file and maybe the output if you do not want the output to be over-written.

Generic, uniform functions/Used by all scripts

ExtractTruss.m

```
function [num,magnets,rods_and_connections,Reactions_forces,External_Loads] = ExtractTruss(Input)
% Oct 1th, 2018. ASEN 2001 Statics and Structures, Lab 2
% Done By:
%           - Abdulla Al Ameri
%           - Eric Hunnel
%           - Kunal Sinha
%           - Johann Kailey-Steiner
%           - Mia Abouhamad
%-------------------------------------------------------------------------
% This codes extracts information
% from a given txt file. The text file is commented by # symbols and the
% code will go and get everything in between, designed for Lab 2.
% ------------------------------------------------------------------------
% INPUTS:
```

```
%           - .txt file name
%           - Important: the name needs to include the extension
%
%-------------------------------------------------------------------------
% OUTPUTS: In order of output (total of 8)
%
%           - num: Number of joints, bar, reactions, and external loads  (in array respectively)
%
%           - magnets: Magnet joints and their coordinates (Norm x y z)
%
%           - rods_and_connections : Rigid rods and their associated joints
%
%           - location at which external couple moments are applied ( x y
%           z )
%
%           - Reactions_forces (Joint location   fx  fy   fz)
%
%           - External loads (Joint location   fx  fy   fz)
%
%-------------------------------------------------------------------------
% If you are not fimilar with CLI ( Command Line interface ) function in
% matlab, you would've to do something like
% [a,b,c,d,e,f,g,h] = Extraction ('Lab2_Input.txt');
% where a,b,c ... are your outputs in order

%% Housekeeping
clc

%% Extract the whole file

%----------------------------(GET THE DATA)-------------------------------

handle = fopen(Input); %give the input file ID.

i = 'initial run'; %initialize the looping variable
j = 0; %loop counter

input_file = {};
%Create a file with cells input to extract each line from the data file to a cell.

while i ~= -1 %fgetl returns -1 when the lines have all been read

  i = fgetl(handle); %Opens the file line by line
    j = j+1; %Keeping count to make sure data goes in the right cell
    input_file {j} = i; %Put the data extracted in cell j of input_file

end

%% See where the #comments are in the cell
% 1 = cell is comments, 0 = cell is data

comments_indi = zeros(1,length(input_file)-1);
%Initialize an array of the length of the cell vector minus 1

for j = 1 : (length(input_file)-1) %loop for the length of the index

    if contains(input_file(j),'#') == 1 %check if the cell is comment
    comments_indi(j) = 1; %return 1 at the indicie of the comment cell

    end
end

%% Extract the info between the comments

% The following sections will extract data blocks and write them in seperate
% arrays.

data_indi = find(comments_indi==0); %Finds the locations of the zeros (the data)
```

```
%It's like a treasure map

%From there, if data is consecutive they will be
%extracted together in one array.

% There's 7 total blocks of data.


%% Data Set 1: num bars reactions ext f

set = {}; %Initialize set
i = 1; %Initialize counter variable i
num = [ 0 0 0 0 ]; %Initialize num_f_m

if data_indi(1+i) - data_indi(i) == 1 %If two lines of data are consecutive

    while data_indi(i+1) - data_indi(i) == 1 % Then while lines are consective

        set{1} = input_file{data_indi(i)}; %Store the data in set
        num_new = str2num(set{i}); %Then convert the contents of set to numbers

    end

    num = [ num ; num_ew ]; %Store the contents in num_f_m


else

    set{i} = input_file{data_indi(i)}; %Store the line in set
    num = str2num(set{i}); %Then convert the contents of set to numbers
    %And store them in num_f_m
    i = i+1; %Tick up the counter variable

end

num_joints = num(1);
num_bars = num(2);
num_reactions = num(3);
num_ext_loads = num(4);

%% Data Set 2: Magnets cords and their cordinates

set = {}; %Clear set
magnets = [ 0 0 0 0 ]; %Initialize ext_f_cord

while data_indi(1+i) - data_indi(i) == 1 %While data is consecutive

        k = 1; %Initialize k to 1
        set{1} = input_file{data_indi(i)}; %Extract the data to set
        manets_new = str2num(set{k}); %Put the data from set into a new variable
        k = k+1; %Tick up k
        i = i+1; %Tick up line counter
        magnets = [ magnets ; manets_new ]; %Concatenate ext_f_cord and the data

end

    if data_indi(i+1) - data_indi(i) ~= 1 %If data is not consecutive, such as the last line

    set{1} = input_file{data_indi(i)}; %Extract the data to set
    manets_new = str2num(set{1});
    magnets = [ magnets ; manets_new ]; %Concatenate data with the array
    %k = k+1;

    end

magnets(1,:) = []; %Clear the first line of zeros

%% Data Set 3: Rods and their asocciated joints
```

```
151
152 i = i+1; %Tick up the line counter
153 set = {}; %Clear set
154 rods_and_connections = [ 0 0 0 ]; %Initialize
155
156 while data_indi(1+i) - data_indi(i) == 1 %While data is consecutive
157
158         k = 1; %Initialize k
159         set{1} = input_file{data_indi(i)}; %Extract the data to set
160         rods_and_connections_new = str2num(set{k}); %Put the data from set into a new variable
161         k = k+1; %Tick up k
162         i = i+1; %Tick up line counter
163         rods_and_connections = [ rods_and_connections ; rods_and_connections_new ]; %Concatenate
        data with the array
164
165 end
166
167     if data_indi(i+1) - data_indi(i) ~= 1 %If data is not consecutive, such as the last line
168
169     set{1} = input_file{data_indi(i)}; %Extract the data to set
170     rods_and_connections_new = str2num(set{1});
171
172
173     rods_and_connections = [ rods_and_connections ; rods_and_connections_new ]; %Concatenate data
         with the array
174     %k = k+1;
175
176     end
177
178 rods_and_connections(1,:) = []; %Clear the first line
179
180 %% Data Set 4: Reactions forces
181
182 i = i+1; %Tick up line counter
183 set = {}; %Clear set
184 Reactions_forces = [ 0 0 0 0 ]; %Initialize array
185
186 while data_indi(1+i) - data_indi(i) == 1 %While data is consecutive
187
188         k = 1; %Initialize k
189         set{1} = input_file{data_indi(i)}; %Extract the data to set
190         Reaction_forces_New = str2num(set{k});%Put the data from set into a new variable
191         k = k+1; %Tick up k
192         i = i+1; %Tick up line counter
193         Reactions_forces = [ Reactions_forces ; Reaction_forces_New ]; %Concatenate data with the
        array
194
195 end
196
197     if data_indi(i+1) - data_indi(i) ~= 1 %If data is not consecutive, such as the last line
198
199     set{1} = input_file{data_indi(i)}; %Extract the data to set
200     Reaction_forces_New = str2num(set{1});
201     Reactions_forces = [ Reactions_forces ; Reaction_forces_New ]; %Concatenate data with the
        array
202     %k = k+1;
203
204     end
205
206 Reactions_forces(1,:) = []; %Clear the first line
207
208
209 %% Data Set 5: Externa Loads
210
211 i = i+1; %Tick up line counter
212 set = {}; %Clear set
213 External_Loads = [ 0 0 0 0 ]; %Initialize array
214
```

```
if i == length(data_indi)

    External_Loads = str2num(input_file{data_indi(i)});

else

while data_indi(1+i) - data_indi(i) == 1 %While data is consecutive

        k = 1; %Initialize k
        set{1} = input_file{data_indi(i)}; %Extract the data to set
        External_Loads_New = str2num(set{k}); %Put the data from set into a new variable
        k = k+1; %Tick up k
        i = i+1; %Tick up line counter



        External_Loads = [ External_Loads ; External_Loads_New ];

       % the last itteration

            if i == length(data_indi) %If data is not consecutive, such as the last line
    set{1} = input_file{data_indi(i)};
    External_Loads_New = str2num(set{1});
    External_Loads = [ External_Loads ; External_Loads_New ]; %Concatenate data with the array
    %k = k+1;
    break
            end
end



    External_Loads(1,:) = []; %Clear the first line

end

end
```

Listing 1 Function that reads the input

addweight.m

```
function [barweight_m, reacjoints_w]=addweight(connectivity,joints,loadjoints,loadvecs,lin_dens,
    sleve,slevemass,jointmass,magnetmass)
% Oct 1th, 2018. ASEN 2001 Statics and Structures, Lab 2
% Done By:
%           - Abdulla Al Ameri
%           - Eric Hunnel
%           - Kunal Sinha
%           - Johann Kailey-Steiner
%           - Mia Abouhamad
%-------------------------------------------------------------------------
% The following function will adjust given vectors of external loads and their
% locations and return two vectors, one about the location of the joints,
% and one about their actual magnitude in x y z
%
% -------------------------------------------------------------------------
% INPUTS:
%           - connectivity: what joints has bars connected to them
%           - joints: the location of joints
%           - loadjoints: where loads already applied.
%           - laoadvecs: external loads (weight is to be added
%           - lin_dens = linear density in  kg / m
%           - sleve: Array of 0's and 1's, will be as big as how many bars
%           we have and if at the location of the bar 1 is returned that
%           mean there's a sleve there
%           - sleveweight: in N.
%           - jointmass: the mass of the balls that acted as joints
```

```
%           - magnetmass: mass of the magnets that were glued to the side
%           of each bar.
%
%-------------------------------------------------------------------------
% OUTPUTS: In order of output (total of 8)
%
%           - barweight_m: vector returns all external loads with weight
%           added
%
%           - reacjoints_w: the joints the load is applied to.
%
%
%-------------------------------------------------------------------------
% If you are not fimilar with CLI ( Command Line interface ) function in
% matlab, you would've to do something like
% [a,b,c,d,e,f,g,h] = Extraction ('Lab2_Input.txt');
% where a,b,c ... are your outputs in order

%find length of each bar

[r c]=size(connectivity);
[r1 c1]=size(joints);
barlength = zeros(r,1);

% define weight of magnet balls that are acting as joints and weight of
% magnets that are glued at the end of each bar

magnetweight = (magnetmass)*9.81;
jointweight = (jointmass)*9.81;
sleveweight = (slevemass)*9.81;


%loop over conectivity to find length.

for i = 1:r
    %see each joints each bar is connected to.
    joint_2 = connectivity(i,2);
    joint_1 = connectivity(i,1);
    %get length and convert to meters
    barlength(i) = sqrt(((joints(joint_1,1)-joints(joint_2,1))^2)+((joints(joint_1,2)-joints(
    joint_2,2))^2)+((joints(joint_1,3)-joints(joint_2,3))^2))*(0.0254);
end


% barweight given bar dens and length
barweight = barlength.*(lin_dens*9.81);
barweight_m = zeros(r1,3);

% add pre-exisiting loads if there's any

for j = 1:length(loadjoints)

    barweight_m(loadjoints(j),:) = loadvecs(j,:);

end

%add half of the bar weight into each side and with that half there's also
%magnet that's glued at each side.

for k = 1:r

barweight_m(connectivity(k,1),3) = barweight_m(connectivity(k,1),3) - barweight(k)/2 -
    magnetweight ;
barweight_m(connectivity(k,2),3) = barweight_m(connectivity(k,2),3) - barweight(k)/2 -
    magnetweight ;


end
```

```
91
92 % after hardcoding sleves with 1 and 0, if there is a sleve add the weight
93 % to the bar
94
95 % the sleve vector is hard coded, check TrssInNOutDesign.m
96
97 for i = 1:length(sleve)
98     if sleve(i) == 1
99        barweight_m(connectivity(i,1),3) = barweight_m(connectivity(i,1),3) - (sleveweight/2);
100       barweight_m(connectivity(i,2),3) = barweight_m(connectivity(i,2),3) - (sleveweight/2);
101    end
102 end
103
104 % add weight of magnet balls that are acting as joints
105
106 barweight_m(:,3) = barweight_m(:,3) - jointweight ;
107
108 %restablish load joints, where each row will represent the number of joint
109 %or joint id
110 reacjoints_w = 1:r1;
111 reacjoints_w = reacjoints_w';
112
113
114
115
116
117 end
```

Listing 2 The function that accounts for the weight

FAA.m

```
1  function [barforces,reacforces]=FAA(joints,connectivity,reacjoints,reacvecs,loadjoints,loadvecs)
2  % function [barforces,reacforces]=forceanalysis(joints,connectivity,reacjoints,reacvecs,
       loadjoints,loadvecs)
3  %
4  % compute forces in bars and reaction forces
5  %
6  % input:  joints       - coordinates of joints
7  %         connectivity - connectivity
8  %         reacjoints   - joint id where reaction acts on
9  %         reacvecs     - unit vector associated with reaction force
10 %         loadjoints   - joint id where external load acts on
11 %         loadvecs     - load vector
12 %
13 % output: barforces    - force magnitude in bars
14 %         reacforces   - reaction forces
15 %
16 % Author: Kurt Maute, Sept 21 2011
17
18 % extract number of joints, bars, reactions, and loads
19 numjoints = size(joints,1);
20 numbars   = size(connectivity,1);
21 numreact  = size(reacjoints,1);
22 numloads  = size(loadjoints,1);
23
24 % number of equations
25 numeqns = 3 * numjoints; %changed to 3 for 3D
26
27 % allocate arrays for linear system
28 Amat = zeros(numeqns);
29 bvec = zeros(numeqns,1);
30
31 % build Amat - loop over all joints
32
33 for i=1:numjoints
34
```

```
   % equation id numbers
   idx = (3*i)-2;
   idy = (3*i)-1;
   idz = (3*i); %added for 3D

   % get all bars connected to joint
   [ibar,ijt]=find(connectivity==i);

   % loop over all bars connected to joint
   for ib=1:length(ibar)

       % get bar id
       barid=ibar(ib);

       % get coordinates for joints "i" and "j" of bar "barid"
       %Actually, for 3D, get coordinates for joints "i", "j", and "k" of
       %bar "barid"
       joint_i = joints(i,:);
       if ijt(ib) == 1
           jid = connectivity(barid,2);
       else
           jid = connectivity(barid,1);
       end
       joint_j = joints(jid,:);

       % compute unit vector pointing away from joint i
       vec_ij = joint_j - joint_i;
       uvec   = vec_ij/norm(vec_ij);

       % add unit vector into Amat
       Amat([idx idy idz],barid)=uvec; %changed to idx idy idz for 3D
   end
end

% build contribution of support reactions
for i=1:numreact

    % get joint id at which reaction force acts
    jid=reacjoints(i);

    % equation id numbers
    idx = (3*jid)-2;
    idy = (3*jid)-1;
    idz = (3*jid);

    % add unit vector into Amat
    Amat([idx idy idz],numbars+i)=reacvecs(i,:); %changed to idx idy idz for 3D
end

% build load vector
for i=1:numloads

    % get joint id at which external force acts
    jid=loadjoints(i);

    % equation id numbers
    idx = (3*jid)-2;
    idy = (3*jid)-1;
    idz = (3*jid);

    % add unit vector into bvec (sign change)
    bvec([idx idy idz])=-loadvecs(i,:); %changed to idx idy idz for 3D
end

% check for invertability of Amat
%if rank(Amat) ~= numeqns
%    error('Amat is rank defficient: %d < %d\n',rank(Amat),numeqns);
%end
```

```
103
104 % solve system
105 xvec=Amat\bvec;
106
107 % extract forces in bars and reaction forces
108 barforces=xvec(1:numbars);
109 reacforces=xvec(numbars+1:end);
110
111 end
```

Listing 3 Force analysis function

wirteoutput.m

```
1  function writeoutput(outputfile,inputfile,barforces,reacforces,joints,connectivity,reacjoints,
       reacvecs,loadjoints,loadvecs)
2  % writeoutput(outputfile,inputfile,barforces,reacforces,joints,connectivity,reacjoints,reacvecs,
       loadjoints,loadvecs);
3  %
4  % output analysis results
5  %
6  % input:  outputfile   - name of output file
7  %         inputfile    - name of input file
8  %         barforces    - force magnitude in bars
9  %         reacforces   - reaction forces
10 %         joints       - coordinates of joints
11 %         connectivity - connectivity
12 %         reacjoints   - joint id where reaction acts on
13 %         reacvecs     - unit vector associated with reaction force
14 %         loadjoints   - joint id where external load acts on
15 %         loadvecs     - load vector
16 %
17 %
18 % Author: Kurt Maute, Sept 21 2011
19
20 % open output file
21 fid=fopen(outputfile,'w');
22
23 % write header
24 fprintf(fid,'3-D Truss analysis\n');
25 fprintf(fid,'--------------------\n\n');
26 fprintf(fid,'Date: %s\n\n',datestr(now));
27
28 % write name of input file
29 fprintf(fid,'Input file: %s\n\n',inputfile);
30
31 % write coordinates of joints
32 fprintf(fid,'Joints:          Joint-id  x-coordinate y-coordinate z-coordinate\n');
33 for i=1:size(joints,1)
34     fprintf(fid,'%17d %12.2f %12.2f %12.2f\n',i,joints(i,1),joints(i,2),joints(i,3));
35 end
36 fprintf(fid,'\n\n');
37
38 % write external loads
39 fprintf(fid,'External loads: Joint-id  Force-x      Force-y      Force-z\n');
40 for i=1:size(loadjoints,1)
41     fprintf(fid,'%17d %12.2f %12.2f %12.2f\n',loadjoints(i),loadvecs(i,1),loadvecs(i,2),loadvecs(
       i,3));
42 end
43 fprintf(fid,'\n');
44
45 % write connectivity and forces
46 fprintf(fid,'Bars:            Bar-id    Joint-i      Joint-j     Force    (T,C)\n');
47 for i=1:size(connectivity,1)
48     if barforces(i)>0;tc='T';else tc='C';end
49     fprintf(fid,'%17d   %7d %12d    %12.3f     (%s)\n',...
50         i,connectivity(i,1),connectivity(i,2),abs(barforces(i)),tc);
```

```
end
fprintf(fid,'\n');

% write connectivity and forces
fprintf(fid,'Reactions:      Joint-id  Uvec-x       Uvec-y       Uvec-z      Force\n');
for i=1:size(reacjoints,1)
    fprintf(fid,'%17d %12.2f %12.2f %12.2f %12.3f\n',reacjoints(i),reacvecs(i,1),reacvecs(i,2),
    reacvecs(i,3),reacforces(i));
end

% close output file
fclose(fid);

end
```

Listing 4 Force analysis function

MonteCarls.m

```
function [probfail] = MonteCarls(inputfile)
%
% Stochastic analysis of 3-D statically determinate truss by
% Monte Carlo Simulation. Only positions and strength of joints
% treated as random variables
%
% Assumption: variation of joint strength and positions described
%             via Gaussian distributions
%
%             joint strength : mean = 4.8
%                              coefficient of variation = 0.4
%             joint position :
%                              coefficient of variation = 0.01
%                              (defined wrt to maximum dimension of truss)
%
%             number of samples is set to 1e5
%
% Input:  inputfile  - name of input file
%
% Author: Kurt Maute for ASEN 2001, Oct 13 2012

% parameters
jstrmean   = 4.8;   % mean of joint strength 4.8 N
jstrcov    = 0.08;  % coefficient of variation (sigma/u) of joint strength = 0.4/4.8 N
jposcov    = 0.01;  % coefficient of variation of joint position percent of length of truss (ext)
numsamples = 1e5;   % number of samples



% read input file
[numbers,cord_joints,connectivity,Reactions_forces,External_Loads] = ExtractTruss(inputfile);


%Prepare inputs:
[ r c ] = size(cord_joints);
joints = cord_joints(:,(2:end));

[r c] = size(connectivity);
connectivity = connectivity(:,2:c);

%throw error for sizes c

[r c] = size(Reactions_forces);
reacjoints = Reactions_forces(:,1);
reacvecs = Reactions_forces(:,2:c);

[r c] = size(External_Loads);
loadjoints = External_Loads(:,1);
```

```
loadvecs = External_Loads(:,2:c);

% determine extension of truss
ext_x=max(joints(:,1))-min(joints(:,1));   % extension in x-direction
ext_y=max(joints(:,2))-min(joints(:,2));% extension in y-direction
ext_z=max(joints(:,3))-min(joints(:,3)); %extension in z-direction
ext  =max([ext_x,ext_y,ext_z]);

% loop overall samples
numjoints=size(joints,1);       % number of joints
maxforces=zeros(numsamples,1);  % maximum bar forces for all samples
maxreact=zeros(numsamples,1);   % maximum support reactions for all samples
failure=zeros(numsamples,1);    % failure of truss

for is=1:numsamples

    % generate random joint strength limit
    varstrength = (jstrcov*jstrmean)*randn(1,1);

    jstrength = jstrmean + varstrength;

    % generate random samples
    varjoints = (jposcov*ext)*randn(numjoints,3);

    % perturb joint positions
    randjoints = joints + varjoints;

    % compute forces in bars and reactions
LinDensity = 31.13 / 1000 ; % Bars linear density kg / m
slevemass = (5.35/1000); % mass in kg
jointmass = 0.00845; % in kg
magnetmass = 0.0017; % in kg.

%location of sleves if there's any, go back and read TrussInNOutDesign.m if
%you're confused about this
[r1 c1] = size(connectivity);
sleve = zeros(1,r1);


[loadvecs_weighted,loadjoints_weighted]=addweight(connectivity,joints,loadjoints,loadvecs,
    LinDensity,sleve,slevemass,jointmass,magnetmass);
[barforces,reacforces]=FAA(randjoints,connectivity,reacjoints,reacvecs,loadjoints_weighted,
    loadvecs_weighted);

    % determine maximum force magnitude in bars and supports
    maxforces(is) = max(abs(barforces));
    maxreact(is)  = max(abs(reacforces));

    % determine whether truss failed
    failure(is) = maxforces(is) > jstrength || maxreact(is) > jstrength;
end

figure(2);
subplot(1,2,1);
histogram(maxforces,30);
title('Histogram of maximum bar forces');
xlabel('Magnitude of bar forces');
ylabel('Frequency');

subplot(1,2,2);
histogram(maxreact,30);
title('Histogram of maximum support reactions');
xlabel('Magnitude of reaction forces');
ylabel('Frequency');

%fprintf('\nFailure probability : %e \n\n',sum(failure)/numsamples);

probfail = sum(failure)/numsamples;
```

```
end
```

Listing 5 Monte Carlo simulation function

plottruss.m

```
function plottruss(xyz,topo,eforce,fbc,rads,pltflags)
%function plottruss(xyz,topo,eforce,fbc,rads,pltflags)
%-------------------------------------------------
%
%  plot 3D-truss, bars colored accoring to force
%  supported nodes are plotted in red
%  non-supported nodes are plotted in black
%
%  Input:   xyz -  x,y,and z coordinates of nodes
%                 ( number of nodes x 3 )
%
%                   Node 1 - [ x1 y1 z1;
%                   Node 2 -   x2 y2 z2;
%                     ...        ...
%                   Node n -   xn yn zn ];
%
%           topo -  topology of truss
%                  ( number of bars x 2 )
%
%                   Member 1 - [ NodeA NodeB;
%                   Member 2 -   NodeD NodeC;
%                     ...           ...
%                   Member m -   NodeX NodeK ];
%
%           eforce - internal bar forces
%                   ( number of bars x 1 )
%
%                   Member 1 - [ InternalForce1;
%                   Member 2 -   InternalForce2;
%                     ...           ...
%                   Member m -   InternalForceM ];
%
%           fbc -    id numbers of nodes which are supported
%                   ( number of supported nodes x 1 )
%
%           rads -   plot radius of bars and joints
%                       1 x 1 : automatic ratio (Recommend: 0.01)
%                       3 x 1 : [ bar, node, suported node]
%
%           pltflags - flags for printing annotation
%                     (3 x 1)
%                     1. component: 0/1 - plot node id numbers
%                     2. component: 0/1 - plot bar id numbers
%                     3. component: 0/1 - plot force value
%                     4. component: 0/1 - 2D(0) or 3D(1) view
%
%-------------------------------------------------------
% Kurt Maute for ASEN 2001 Oct. 2006
%   Revised Oct. 2007 by Sungeun Jeon
%   Revised Sep. 2010 by Kurt Maute
%-------------------------------------------------------

figure(1);
clf;

% check input

if nargin < 6; error('routine requires 6 input parameters'); end

% extract

```

```
[numnode, dim]=size(xyz);
numelem = size(topo,1);

if dim ~= 3
   display('Error in plottruss: 3 coordinates are needed for array xyz');
   return;
end

% extract min and max force values

minfrc=floor(min(eforce));
maxfrc=ceil(max(eforce));

% define radius of bars

if length(rads) == 1
    radb = rads(1);
    radj = 1.75*radb;
    radk = 1.5*radj;
else
    radb = rads(1);
    radj = rads(2);
    radk = rads(3);
end

% plot bars

figure(1)

for i=1:numelem
    na=topo(i,1);
    nb=topo(i,2);
    [xc,yc,zc]=plotbar(xyz(na,:),xyz(nb,:),radb);
    cc=eforce(i)*ones(size(xc));
    surf(xc,yc,zc,cc,'EdgeColor','none');
    if pltflags(2) > 0
        text((xyz(na,1)+xyz(nb,1))/2+1.8*radb,(xyz(na,2)+xyz(nb,2))/2+1.8*radb,(xyz(na,3)+xyz(nb
    ,3))/2+1.8*radb,num2str(i));
    end
    if pltflags(3) > 0
        text((xyz(na,1)+xyz(nb,1))/2+1.8*radb,(xyz(na,2)+xyz(nb,2))/2+1.8*radb,(xyz(na,3)+xyz(nb
    ,3))/2+1.8*radb,sprintf('%.2f',eforce(i)));
    end
    hold on;
end

% plot node id numbers

if pltflags(1) > 0
    for i=1:numnode
            text(xyz(i,1)+1.8*radj,xyz(i,2)+1.5*radj,xyz(i,3)+1.3*radj,num2str(i));
    end
end

colorbar;
caxis([minfrc maxfrc])

% plot ball joints

for i=1:numnode
    [sx,sy,sz]=plotnode(xyz(i,:),radj);
    surf(sx,sy,sz,'EdgeColor','none','FaceColor','black');
    hold on;
end

% plot supported nodes

for i=1:length(fbc)
```

```
    na=fbc(i);
    [sx,sy,sz]=plotnode(xyz(na,:),radk);
    surf(sx,sy,sz,'EdgeColor','none','FaceColor','red');
    hold on;
end

% plot parameters

axis('equal');
title('Member forces');

lightangle(-45,30)
set(gcf,'Renderer','openGl')
set(findobj(gca,'type','surface'),...
    'FaceLighting','phong',...
    'AmbientStrength',.3,'DiffuseStrength',.8,...
    'SpecularStrength',.9,'SpecularExponent',25,...
    'BackFaceLighting','unlit')

if pltflags(4)
    % use default
else
    % set view point to 0,0,1
    view([0 0 1]);
end

return

function [sx,sy,sz]=plotnode(xyz,rads)
%-------------------------------------------------------
% returns sphere of radius rads at position xyz (3x1)
%-------------------------------------------------------
% Kurt Maute for ASEN 2001 Oct. 2006
%-------------------------------------------------------
[sx,sy,sz]=sphere(30);
nl=length(sx);
omat=ones(size(sx));
sx=rads*sx+xyz(1)*omat;
sy=rads*sy+xyz(2)*omat;
sz=rads*sz+xyz(3)*omat;

return

function [xc,yc,zc]=plotbar(xa,xb,rads)
%-------------------------------------------------------
% returns cylinder of radius rads
% endpoints defined by xa (3x1) and xb (3x1)
%-------------------------------------------------------
% Kurt Maute for ASEN 2001 Oct. 2006
%-------------------------------------------------------

xba=xb-xa;
len=norm(xba);

% create basic cylinder

[xc,yc,zc]=cylinder([rads rads],20);

% stretch cylinder

cmat=[len*zc' xc' yc'];

% create transformation matrix

ev1=1/len*xba';
eva=[ev1(2); -ev1(1); 0];

if (norm(eva) == 0)
```

```
    eva=[ev1(3); 0;  -ev1(1)];
    if (norm(eva) == 0)
      eva=[0; ev1(3); -ev1(2)];
    end
end

eva=1/norm(eva)*eva;
ev2=cross(ev1,eva);
ev3=cross(ev1,ev2);

T=[ev1 ev2 ev3];

% rotate cylinder

cpmat(:,[1 3 5])=(T*cmat(:,[1 3 5])')';
cpmat(:,[2 4 6])=(T*cmat(:,[2 4 6])')';

nl=length(xc);
omat=ones(nl,1);

xc(1,:)=cpmat(:,1)+xa(1)*omat;
xc(2,:)=cpmat(:,2)+xa(1)*omat;
yc(1,:)=cpmat(:,3)+xa(2)*omat;
yc(2,:)=cpmat(:,4)+xa(2)*omat;
zc(1,:)=cpmat(:,5)+xa(3)*omat;
zc(2,:)=cpmat(:,6)+xa(3)*omat;

return
```

Listing 6 3D Plotting function

Main scripts/ changes per design

TrussInNOutDesign1.m

```
clear;
clc;
close all;

inputfile = 'Design1.inp';
outputfile = 'Design1Out.txt';
AssumedFail = 0.17 ;
LinDensity = 31.13 / 1000 ; % Bars linear density kg / m
slevemass = (5.35/1000); % mass in kg
jointmass = 0.00845; % in kg
magnetmass = 0.0017; % in kg.

[numbers,cord_joints,connectivity,Reactions_forces,External_Loads] = ExtractTruss(inputfile);


% HARD CODE, PUT 1 where'ever you have a sleeve somewhere in connectivity
% vector, all 0's means there's no sleves used any where, which is default.
%
[r1 c1] = size(connectivity);
sleve = zeros(1,r1);
sleve(35:42) = 1; %there's sleves from bar 35 until bar 42


%% Force Analysis

%Prepare inputs:
[ r c ] = size(cord_joints);
joints = cord_joints(:,(2:end));

[r c] = size(connectivity);
connectivity = connectivity(:,2:c);

%throw error for sizes c

[r c] = size(Reactions_forces);
reacjoints = Reactions_forces(:,1);
reacvecs = Reactions_forces(:,2:c);

[r c] = size(External_Loads);
loadjoints = External_Loads(:,1);
loadvecs = External_Loads(:,2:c);

%% add weight

[loadvecs_weighted,loadjoints_weighted]=addweight(connectivity,joints,loadjoints,loadvecs,
    LinDensity,sleve,slevemass,jointmass,magnetmass);

%%
[barforces,reacforces]=FAA(joints,connectivity,reacjoints,reacvecs,loadjoints_weighted,
    loadvecs_weighted);

%% Prepare writeoutput functions inputs

writeoutput(outputfile,inputfile,barforces,reacforces,joints,connectivity,reacjoints,reacvecs,
    loadjoints_weighted,loadvecs_weighted);

%% plot 3D Truss

joints3D=zeros(size(joints,1),3);
joints3D(:,1:3)=joints;
plottruss(joints3D,connectivity,barforces,loadjoints_weighted,3*[0.025,0.04,0.05],[1 1 0 0]);


```

```
%% Monte Carlo


jstrmean    = 4.8;   % mean of joint strength 4.8 N
jstrcov     = 0.08;  % coefficient of variation (sigma/u) of joint strength = 0.4/4.8 N
jposcov     = 0.01;  % coefficient of variation of joint position percent of length of truss (ext)
numsamples = 1e5;   % number of samples


ProbFaliure = MonteCarls(inputfile);

Fdsr = icdf('normal',AssumedFail,jstrmean,0.4);

% any given time your max tensile/compressive strength shouldn't exceed the
% the max force in the bars

% the safety factor.

Saf = 4.8 / Fdsr ;

%save the montecarlo result
saveas(gcf,['MonteCarlo.png'])

%% fprintf

fprintf('Assumed probability failure: %2.5f\n',AssumedFail);
fprintf('Monte-Carlo probability of failure: %2.5f\n',ProbFaliure);
fprintf('Safety factor: %2.5f\n',Saf);
```

Listing 7 Main script for design 1

TrussInNOutDesign2.m

```
clear;
clc;
close all;

inputfile = 'Design2.inp';
outputfile = 'Design2Out.txt';
AssumedFail = 0.45 ;
LinDensity = 31.13 / 1000 ; % Bars linear density kg / m
slevemass = (5.35/1000); % mass in kg
jointmass = 0.00845; % in kg
magnetmass = 0.0017; % in kg.

[numbers,cord_joints,connectivity,Reactions_forces,External_Loads] = ExtractTruss(inputfile);


% HARD CODE, PUT 1 where'ever you have a sleeve somewhere in connectivity
% vector, all 0's means there's no sleves used any where, which is default.
%
[r1 c1] = size(connectivity);
sleve = zeros(1,r1);



%% Force Analysis

%Prepare inputs:
[ r c ] = size(cord_joints);
joints = cord_joints(:,(2:end));

[r c] = size(connectivity);
connectivity = connectivity(:,2:c);

%throw error for sizes c

```

```
[r c] = size(Reactions_forces);
reacjoints = Reactions_forces(:,1);
reacvecs = Reactions_forces(:,2:c);

[r c] = size(External_Loads);
loadjoints = External_Loads(:,1);
loadvecs = External_Loads(:,2:c);

%% add weight

[loadvecs_weighted,loadjoints_weighted]=addweight(connectivity,joints,loadjoints,loadvecs,
    LinDensity,sleve,slevemass,jointmass,magnetmass);

%%
[barforces,reacforces]=FAA(joints,connectivity,reacjoints,reacvecs,loadjoints_weighted,
    loadvecs_weighted);

%% Prepare writeoutput functions inputs

writeoutput(outputfile,inputfile,barforces,reacforces,joints,connectivity,reacjoints,reacvecs,
    loadjoints_weighted,loadvecs_weighted);

%% plot 3D Truss

joints3D=zeros(size(joints,1),3);
joints3D(:,1:3)=joints;
plottruss(joints3D,connectivity,barforces,loadjoints_weighted,3*[0.025,0.04,0.05],[1 1 0 0]);


%% Monted Carlo


jstrmean   = 4.8;   % mean of joint strength 4.8 N
jstrcov    = 0.08;  % coefficient of variation (sigma/u) of joint strength = 0.4/4.8 N
jposcov    = 0.01;  % coefficient of variation of joint position percent of length of truss (ext)
numsamples = 1e5;   % number of samples


ProbFaliure = MonteCarls(inputfile);

Fdsr = icdf('normal',AssumedFail,jstrmean,0.4);

% any given time your max tensile/compressive strength shouldn't exceed the
% the max force in the bars

% the safety factor.

Saf = 4.8 / Fdsr ;

%save the montecarlo result
saveas(gcf,['MonteCarlo.png'])

%% fprintf

fprintf('Assumed probability failure: %2.5f\n',AssumedFail);
fprintf('Monte-Carlo probability of failure: %2.5f\n',ProbFaliure);
fprintf('Safety factor: %2.5f\n',Saf);
```

Listing 8 Main script for design 2