University of Colorado - Boulder

ASEN 3112: Structures

ASEN 3112 Lab 2

Author:
Parker Simmons - 107112741

Author:
Ryan Hughes - 107498312

Author:
Ansh Jerath - 107480973

Author:
Aidan Sesnic - 107477406

Author:
Chris Nylund - 108205918

Author:
Panitnan Yuvanondha - 104269949

Professors:
Dr. Aaron Johnson
Dr. Mahmoud Hussein

April 2nd, 2020

College of Engineering & Applied Science
UNIVERSITY OF COLORADO BOULDER

The goal of this lab is to experimentally measure the displacement of a 16-bay truss and compare it to two other methods, which are FEM method and equivalent beam. The total load of 50 lbs (222.4 N) is applied to the point of half-span by the increment of 10 lbs and also remove the load by 10 lbs. The data are collected during the lab and use them to find the vertical displacement. The FEM results come from an implementation in MATLAB. For equivalent beam method, the truss is treated as a beam and the area moment of inertia is computed using parallel axis theorem based on the horizontal members of the truss.

Contents

Nomenclature

A	=	Cross-Section Area [m^2]
t	=	thickness [m]
P	=	Applied Load [N]
d	=	Diameter [m]
E	=	Young's Modulus [GPa]
I_{zz}	=	moment of inertia [m^4]
v	=	vertical displacement [m]
L_b	=	joint-to-joint distance [m]

I. Experimental Results

Linear regression analysis on the displacement vs. load plots (Figures 1 and 2) showed that the two variables appeared to have a remarkably linear relationship, as expected.

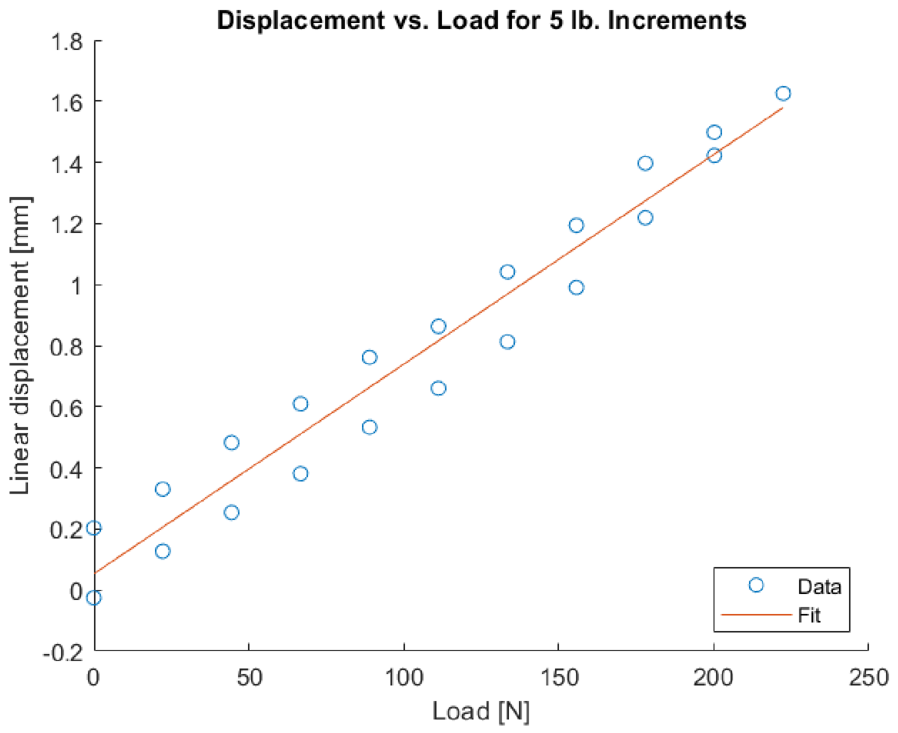


Fig. 1 Displacement vs. load for the 5 lb. increment data set

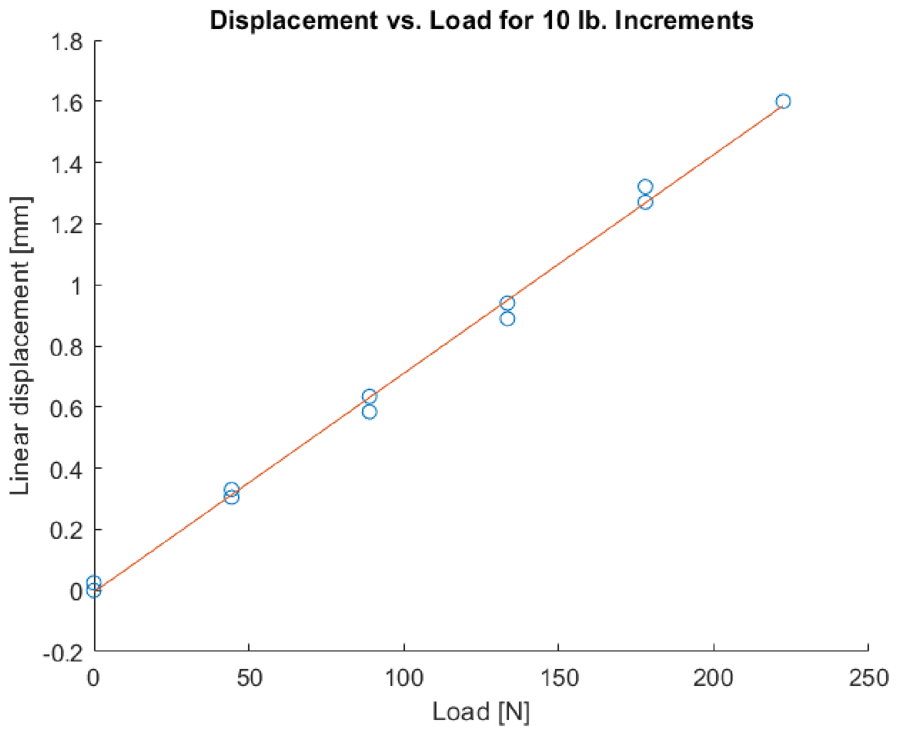


Fig. 2 Displacement vs. load for the 10 lb. increment data set

An important note to make in Figure 2 is that there was originally a negative offset to the data such that the LVDT returned a displacement of roughly $-1.3mm$ with no applied load. To account for this, the offset magnitude was added to the entire LVDT data set for 10 lb. increments. The F3D data set also showed this inconsistency for both 5 and 10 lb. increments, which was addressed by the same adjustment.
Below are the plots depicting internal and external reaction forces measured with the inline and button load cells, respectively, as a function of applied load. These correlations were also analyzed using linear regression.

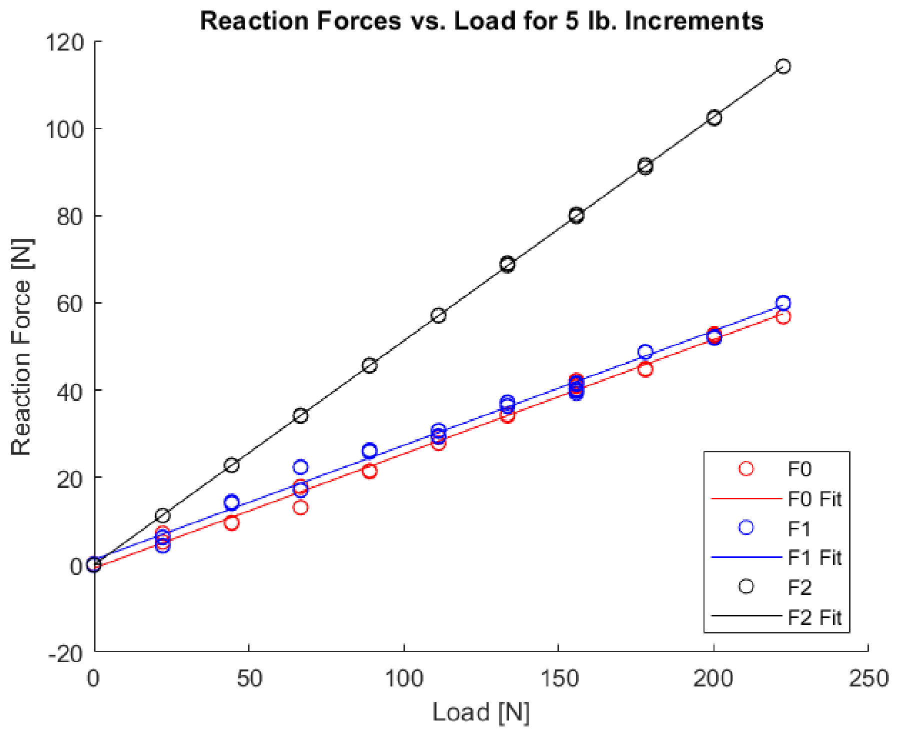


Fig. 3 External reaction forces vs. load for the 5 lb. increment data set

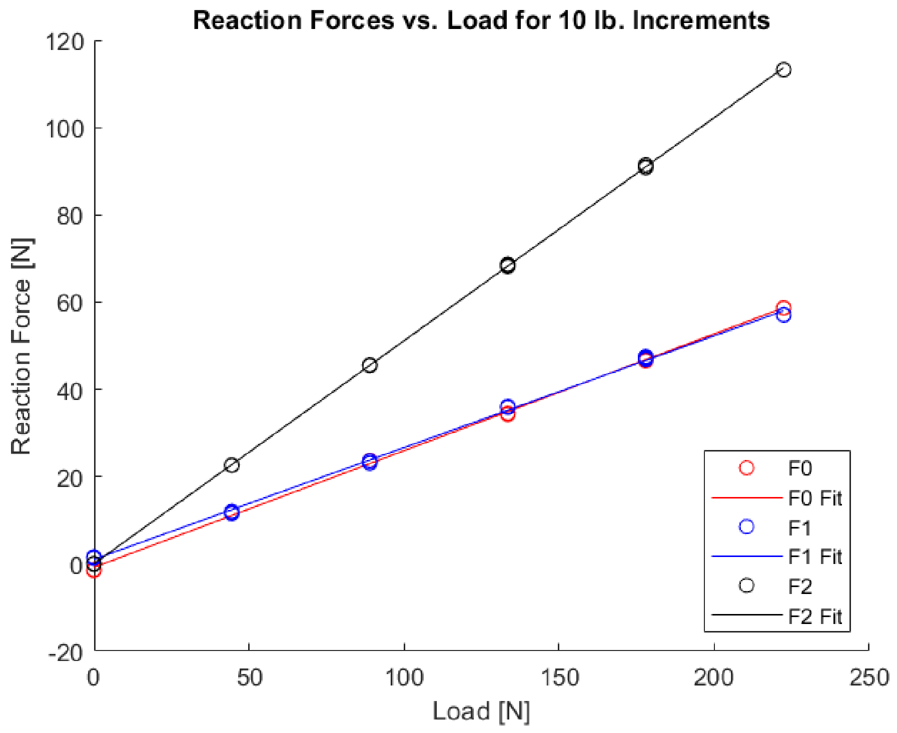


Fig. 4 External reaction forces vs. load for the 10 lb. increment data set

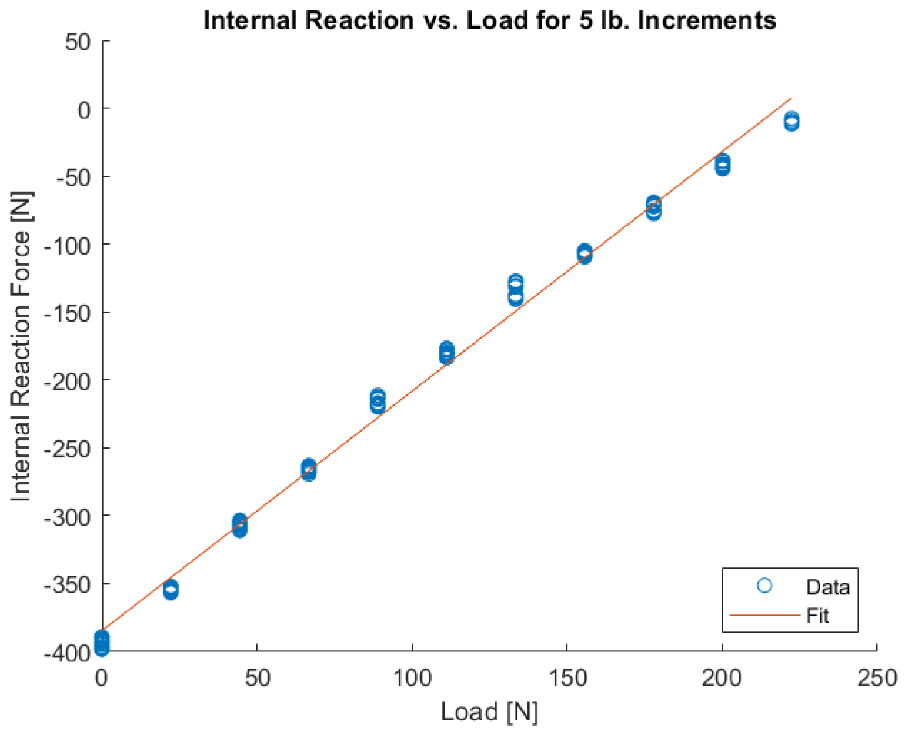


Fig. 5 Internal reaction vs. load for the 5 lb. increment data set

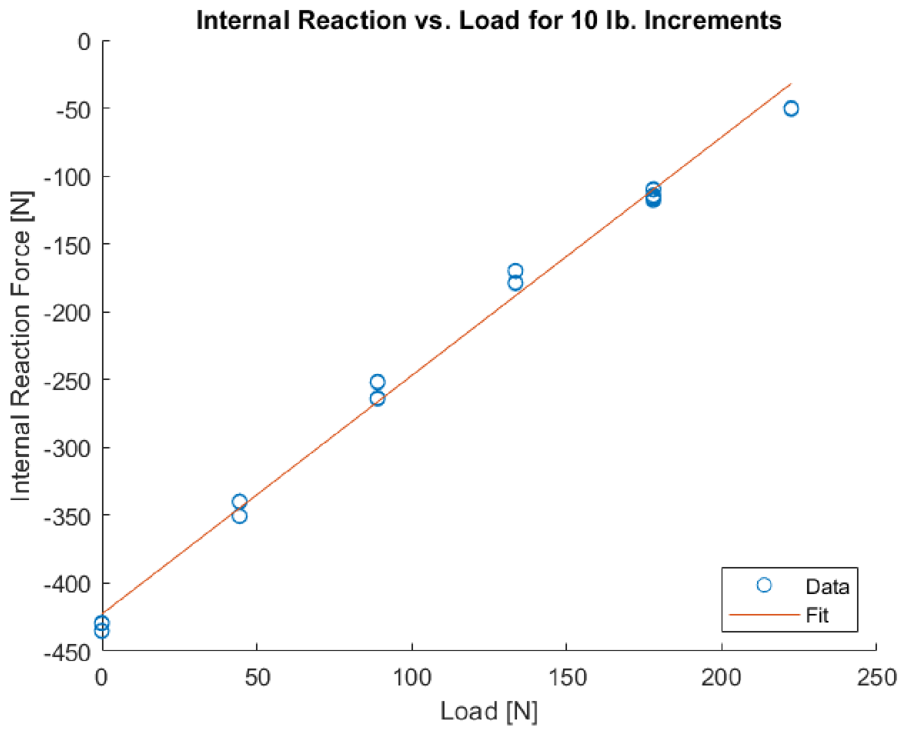


Fig. 6 Internal reaction vs. load for the 10 lb. increment data set

By observation of Figures 1 - 6, it is apparent that all of the analyzed values held a linear relationship with applied load. None of the individual data points were located far enough from the line of best fit to be deemed outliers, and the experimental data doesn't show any trends diverging it from the regression line.
To validate this observation, uncertainty analysis was performed on the linear regression by finding the square of the sample correlation coefficient, r^2, as a measure of correlation. An r^2 value of 1 signifies a perfectly linear relationship, and an r^2 value of 0 represents a lack of correlation altogether [2]. The r^2 values for experimental measurements made in this lab are tabulated below.

Measurement	r^2 (5 lb. increments)	r^2 (10 lb. increments)
Displacement	.9507	.9964
F_0	.9942	.9993
F_1	.9926	.9988
F_2	1	1
F_{3D}	1	1

Table 1 Squared correlation coefficient depicting the linearity of each measurement on a scale from 0 to 1

The r^2 values in Table 1 suggest that the measured experimental data does indeed have a linear correlation, as they are all very close to, or equal to, 1. These results reinforce the conclusion determined by observing the plots above.

II. Analytical and FEM Results

A. FEM Results

The FEM results were obtained from the MATLAB code provided by Dr. Johnson.
The truss shape was constructed iteravely based on the basic truss unit below:

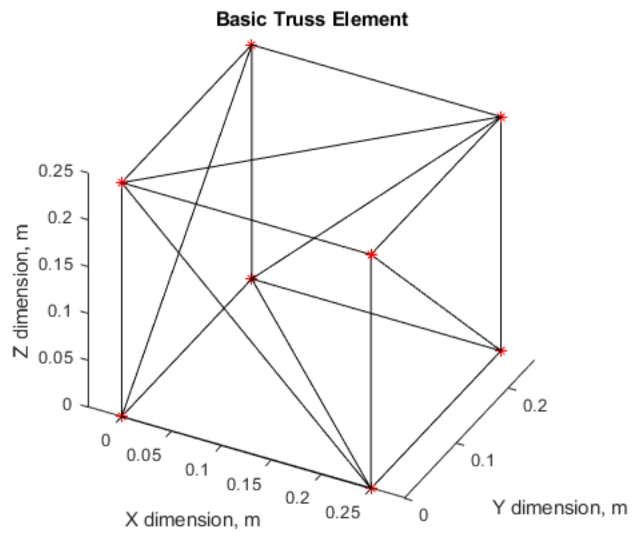


Fig. 7 Basic truss section

From the basic truss structure, the full 16-bay truss is constructed. The loads are also plotted:

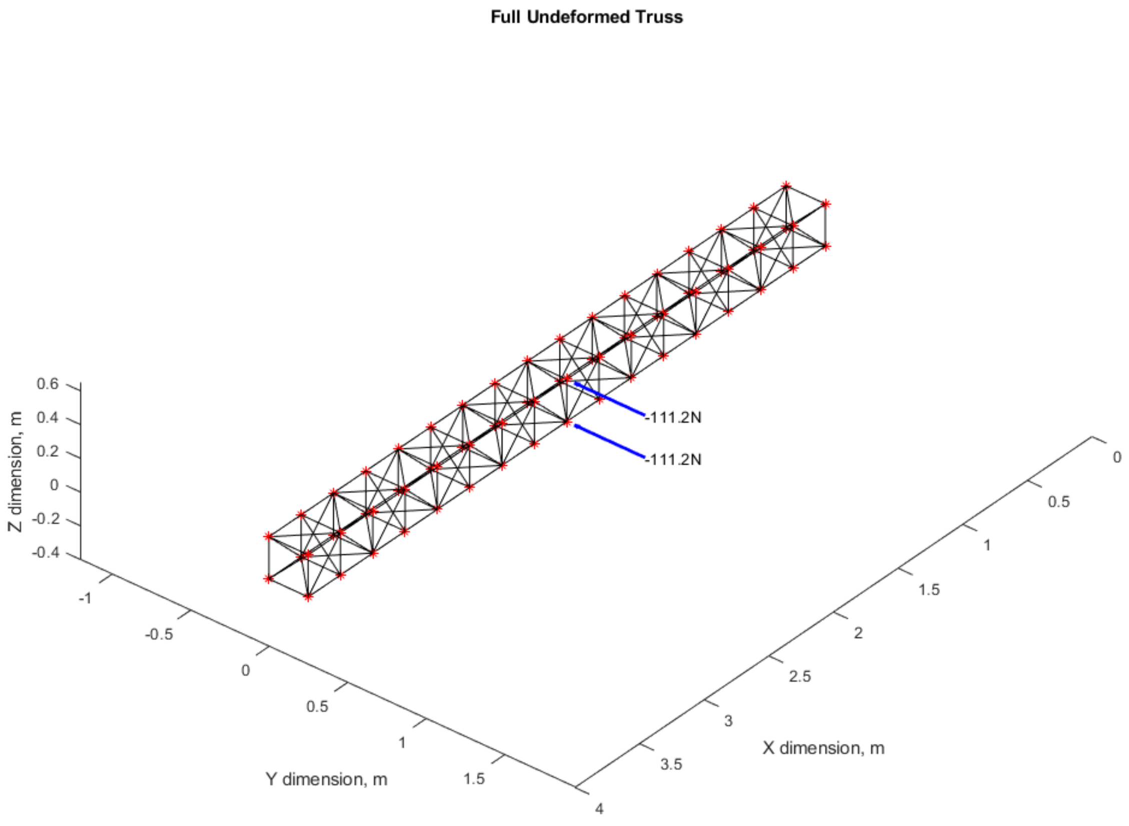


Fig. 8 Full truss

From the FEM results, key figures can be extracted, such as:

- Displacement of middle node in $\hat{y}$ direction: -0.00185 m.
- Force in longeron adjacent to midspan: -440.025 N
- Reaction force at left aft joint: $-1.129\hat{x} + 55.562\hat{y} + 1.087\hat{z}$ N
- Reaction force at left forward joint: $1.129\hat{x} + 55.638\hat{y} - 1.158\hat{z}$ N
- Reaction force at right aft joint: $0\hat{x} + 55.638\hat{y} + 55.638\hat{z}$ N
- Reaction force at right forward joint: $0\hat{x} + 55.562\hat{y} + 55.562\hat{z}$ N

B. Simplifications in FEM model

The FEM model differs from the physical system in several ways: in the loads supported by the joints, loads in the members, and boundary conditions.

The FEM model idealizes the joints in the truss as pinned joints; they support forces in each direction but no moments. In reality, since the members are threaded into the joints, the joints do exert moments. Further, the mechanics of the threads in the joint are not modeled. Threads will cause concentrations of stress.

Since the joints can support moments, the members will behave somewhat like beams in that they will support internal moments and shear forces. So internal forces will not be perfectly axial.

Finally, in the FEM model, the boundary conditions are to be such that an exact displacement or an exact force is known. However, in the physical system, this isn't perfectly realistic: on the right support of the truss, motion in the x direction isn't constrained but is still subject to friction. So, for example, if the magnitude of the $\hat{x}$ reaction force is less than the static frictional force, the displacement will be constrained to zero. However, if the reaction forces are sufficient to overcome friction, the truss displacement in the $\hat{x}$ direction will be nonzero. The FEM model simply can't capture these more complex boundary conditions, and in turn the boundary condition has to be simplified to allow motion in the $\hat{x}$ direction.

C. Results of FEM model

Finally, after using the FEM code, the displacements and stresses of each node and each member, respectively, are found. Note that in the plot below, the displacement is multiplied by a factor of 10 so that the deformation is more visible; the actual-scale deformation is very small. Further, the members are color-coded according to the magnitude of their internal stresses. Note that the black outline represents the undeformed truss, and the pink arrows represent the reaction forces. The plot is below:

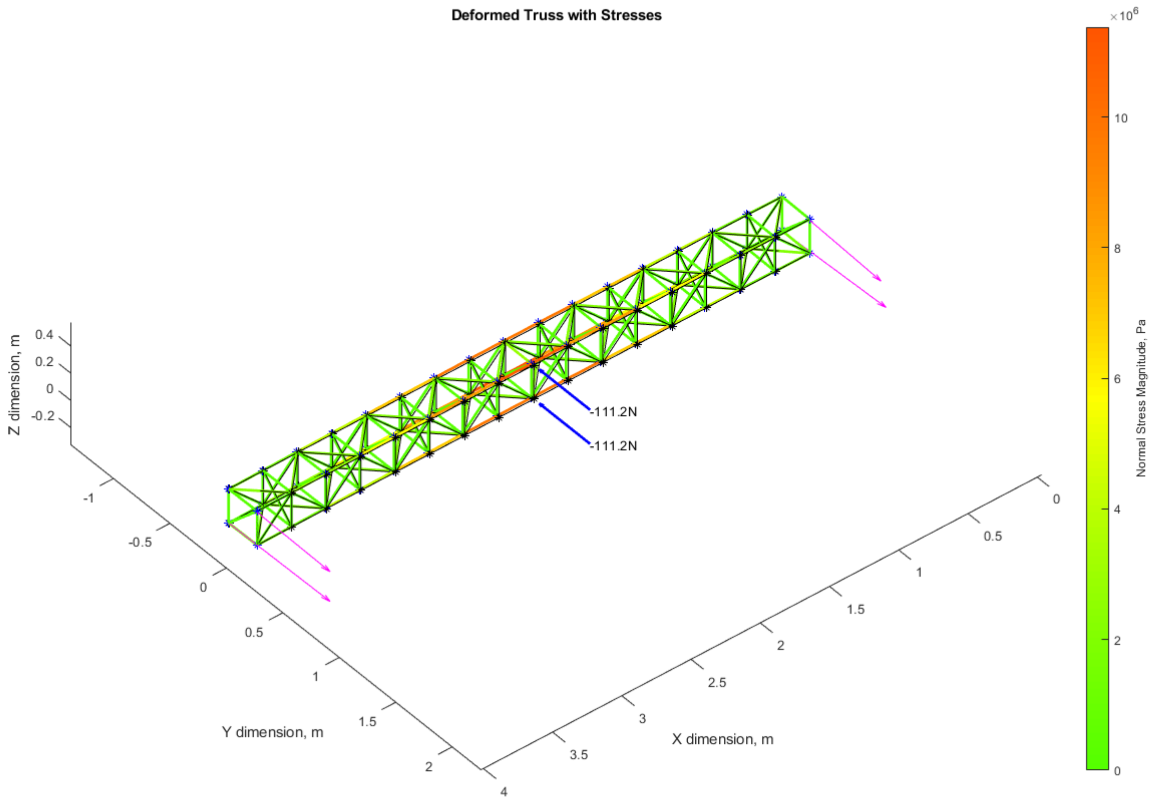


Fig. 9 Deformed truss with member stresses, factor of 10 exaggeration for displacements

Without any exaggeration in the displacements, the plot looks different, since the deformation is very subtle. Note that the plot is a 2D projection in the xy plane for simplicity of presentation. The two arrows representing the applied load are overlaid, and as such it appears that only half of the prescribed load is applied, but the full load is in fact present.

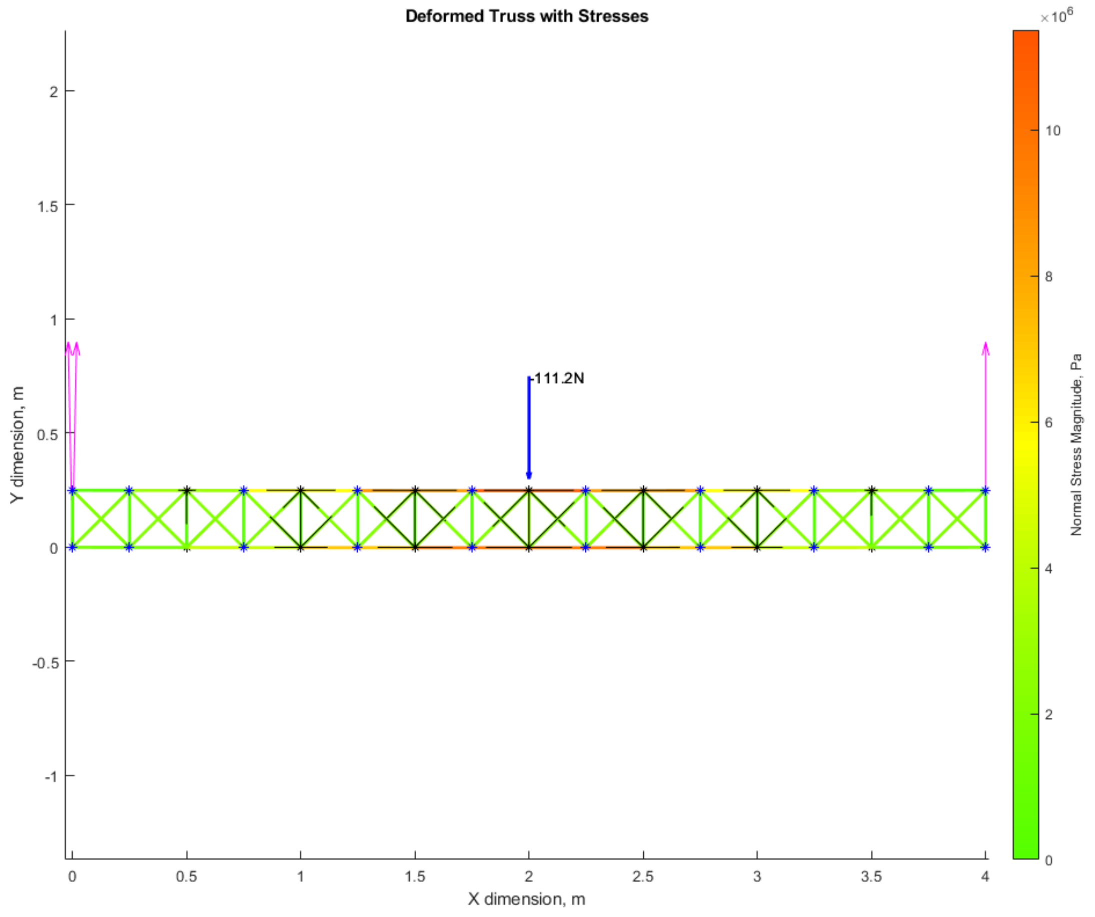
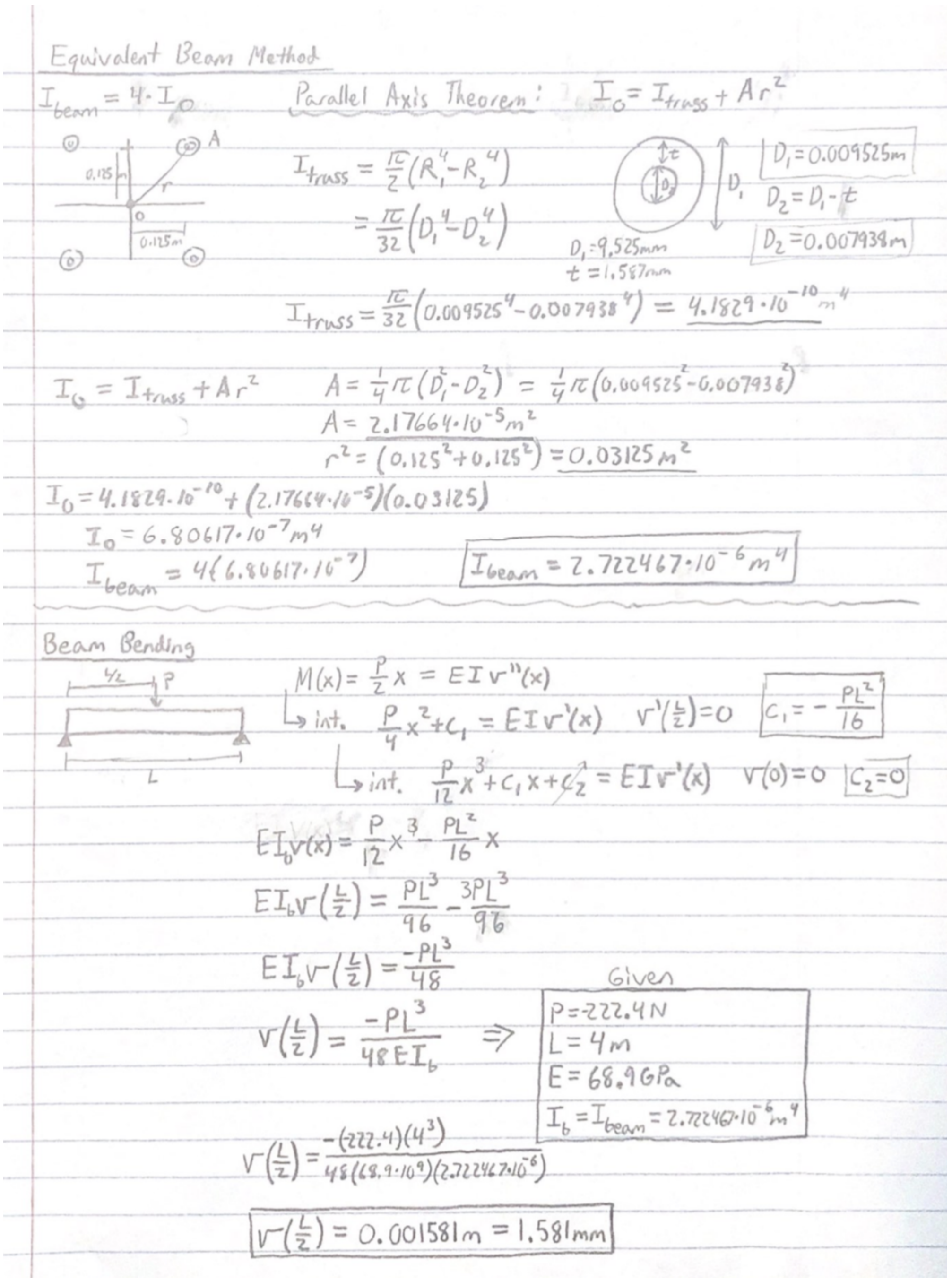


Fig. 10 Deformed truss with member stresses, actual scale

D. Comparison with equivalent beam model



Equivalent Beam Method

$I_{beam} = 4 \cdot I_0$ Parallel Axis Theorem: $I_0 = I_{truss} + Ar^2$

A
0.125m
r
0
0.125m
t
D_2
D_1

$$I_{truss} = \frac{\pi}{2}(R_1^4 - R_2^4) = \frac{\pi}{32}(D_1^4 - D_2^4)$$

$\boxed{D_1 = 0.009525m}$ $D_2 = D_1 - t$ $\boxed{D_2 = 0.007938m}$

$D_1 = 9.525mm$ $t = 1.587mm$

$$I_{truss} = \frac{\pi}{32}(0.009525^4 - 0.007938^4) = \underline{4.1829 \cdot 10^{-10} m^4}$$

$I_0 = I_{truss} + Ar^2$ $A = \frac{1}{4}\pi(D_1^2 - D_2^2) = \frac{1}{4}\pi(0.009525^2 - 0.007938^2)$

$A = \underline{2.17664 \cdot 10^{-5} m^2}$

$r^2 = (0.125^2 + 0.125^2) = \underline{0.03125 m^2}$

$I_0 = 4.1829 \cdot 10^{-10} + (2.17664 \cdot 10^{-5})(0.03125)$

$I_0 = 6.80617 \cdot 10^{-7} m^4$

$I_{beam} = 4(6.80617 \cdot 10^{-7})$ $\boxed{I_{beam} = 2.722467 \cdot 10^{-6} m^4}$

Beam Bending

$M(x) = \frac{P}{2}x = EIv''(x)$

$\hookrightarrow$ int. $\frac{P}{4}x^2 + C_1 = EIv'(x)$ $v'(\frac{L}{2}) = 0$ $\boxed{C_1 = -\frac{PL^2}{16}}$

$\hookrightarrow$ int. $\frac{P}{12}x^3 + C_1x + \cancel{C_2} = EIv(x)$ $v(0) = 0$ $\boxed{C_2 = 0}$

$$EI_bv(x) = \frac{P}{12}x^3 - \frac{PL^2}{16}x$$

$$EI_bv(\tfrac{L}{2}) = \frac{PL^3}{96} - \frac{3PL^3}{96}$$

$$EI_bv(\tfrac{L}{2}) = \frac{-PL^3}{48}$$

$$v(\tfrac{L}{2}) = \frac{-PL^3}{48EI_b} \Rightarrow$$

Given

$P = 222.4N$
$L = 4m$
$E = 68.9GPa$
$I_b = I_{beam} = 2.722467 \cdot 10^{-6} m^4$

$$v(\tfrac{L}{2}) = \frac{-(222.4)(4^3)}{48(68.9 \cdot 10^9)(2.722467 \cdot 10^{-6})}$$

$$\boxed{v(\tfrac{L}{2}) = 0.001581m = 1.581mm}$$

Fig. 11 Calculation of the Maximum Displacement Using the Equivalent Beam Method

Another method that can be used in finding the displacement of the midpoint of the truss is the Equivalent Beam Model. This method makes the assumption that the entire truss is instead a beam of the same elastic modulus. For this model, the second moment of the area was found at the center of the beam's cross section using the Parallel Axis Theorem on the four longeron strut cross sections. This can be seen in Figure 11 and was calculated to be $2.72246 * 10^{-6}$ km. Next, the displacement at the midpoint of the beam with a point load also at the center could be calculated using the Second Order Method of Beam Bending. This can also be seen in Figure 11.

Point Load (N)	Equivalent Beam Displacement (mm)	FEM Displacement (mm)	Difference (%)
-44.48	0.3162	0.3175	0.41
-88.96	0.6323	0.6096	3.72
-133.4	0.9485	0.9144	3.73
-177.9	1.2647	1.2954	2.37
-222.4	1.5809	1.6002	1.21

Table 2 Displacement with Different point loads at 10lb Increments

In Table 2, the displacement using the Equivalent Beam Model versus that of the FEM was calculated for the 10lb increment loads. The displacement found through the FEM was averaged between the two measurements found during load incrementing, one adding the loads and the other taking off loads. From this data it can be seen that the displacement is very similar for both methods with a maximum difference of only 3.73%. The reasons for this discrepancy is mainly due to the fact that the Equivalent Beam Model is making the large assumption that the truss, made of many small members, is now one large beam with a moment of inertia at the center of the truss. This simplified model also does not account for any diagonal members in the truss.

III. Uncertainty Analysis

A. Experimental Data comparison to FEM model

The FEM provides a significant amount of insight in determining the deflection of a truss. However, our FEM technique does not model everything that actually happens in the real-life structural system. Many assumptions simplify our FEM analysis and, in result, cause deviations from the true experimental results. Our group compared the FEM results computed in MATLAB to the experimental data. This was done by calculating the y-direction deflection at the middle of the beam from the FEM and comparing it to the y-direction displacement measured by the probe in the experiment as the load was incremented. Plots for both the 5lb and 10lb increments can be seen in Figure 12 and 13 respectively. Similarly, Tables 3 and 4 below shows the percent error of the FEM computation compared to the experiment for each applied load increment. Looking at the graph and the tables, it can be concluded that there is significant deviation between the FEM model and the experiment. Once again, these differences are most definitely due to the underlying assumptions that go into the FEM modelling. These assumptions are the sources of error which are discussed below.

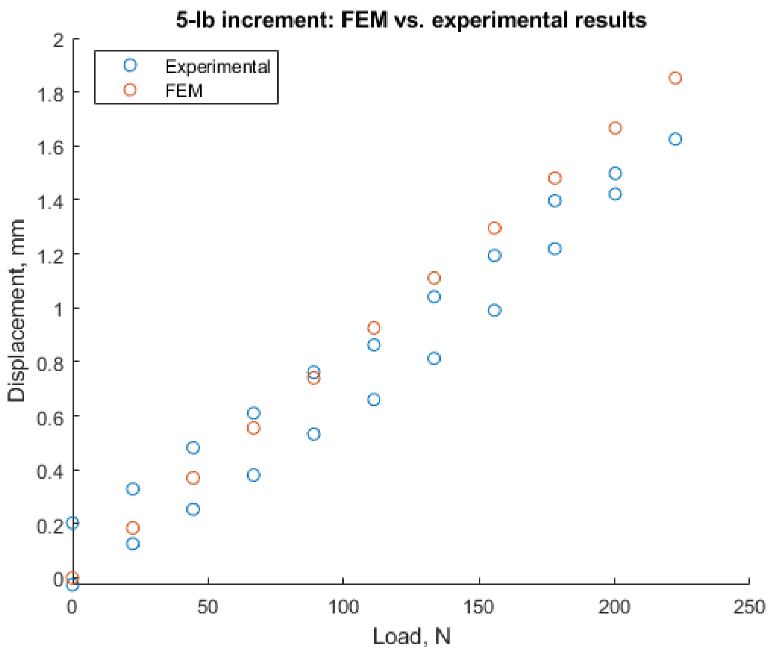


Fig. 12 Displacement vs. load for the 5 lb. increment data set with the FEM and Experimental Data

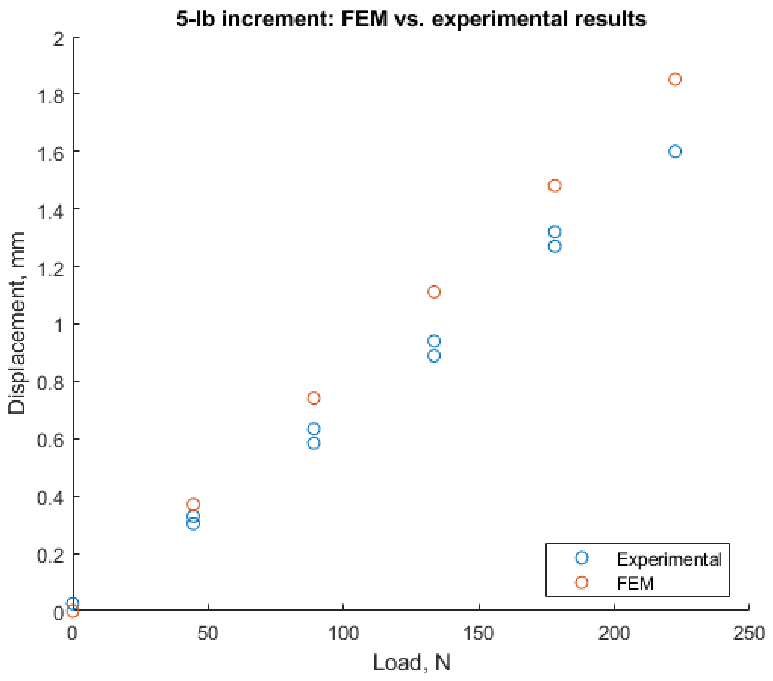


Fig. 13 Displacement vs. load for the 10 lb. increment data set with the FEM and Experimental Data

Load Increment [lbs.]	Percent Error [%]
5	38.9
10	46.8
20	50.8
30	52.1
40	52.8
50	53.2

Table 3 5 lb Load increment percent error (error shown for every 10 lbs)

Load Increment [lbs.]	Percent Error [%]
10	72.4
20	62.6
30	59.3
40	57.7
50	56.7

Table 4 10 lb Load increments percent error

B. Source Error/Assumption Analysis

Since there are several assumptions that were made when analyzing the truss using the FEM, there will be errors associated with it. In Table 5 below, we can see how a decrease in the Young's Modulus (Stiffness) impacts the percent error compared to the experimental data. The lower bound of the Young's Modulus for the Aluminium 6061-T6 material was chosen to compare against the true 69-70 GPa value for comparison. Looking at Table 3, we can quantitatively see these differences. The percent error due to decreasing the Young's Modulus increases the percent error meaning that a less stiffer bar will result in a larger deflection. We can observe this error by comparing the 50 lb load in Table 3. This value also corresponds to a stress of 2.07 MPa. Similarly, when only the bar in-line with the load cell is changed, we see a very small increase in percent error. This is likely because the entire 16-section truss is not changed rather just the single bar. This change is not nearly as significant as changing the stiffness for all bars. Cross sectional area also plays a significant factor in accuracy of the FEM. Similar to changing the Young's Modulus, arbitrary cross sectional areas above and below the given cross sectional area were selected and can be seen in Table 5. Changing these values significantly changed the deflection. A smaller cross sectional area results in a deflection closer to the experimental deflection. Furthermore, the bars are assumed to be weightless which means the FEM simulation does not account for additional loading due to weight. Also the load is assumed to be point load in the MATLAB script instead of distributed load, but this assumption wouldn't make much of a contribution to error because the average variation of the distributed load was negligible.

In this experiment, there was likely a small amount of friction at both supported ends in resulting from deformation, which potentially invalidates the boundary conditions assumed in the FEM model. For example, the button load cells at each support are allowed to rotate, but realistically there is friction inside these load cells which means that there are potentially non-zero moment reactions. As long as the devices are well maintained, however, these frictional moments have a nominal effect on the truss's displacement. Similarly, the bar model assumes there are no bending moments in the joints, however if we consider each strut as a beam with "fixed" support reactions, we note that there are non-zero moment reactions at either end of each strut.

Source	Percent Error [%]
Imperfect Joints (E = 69 GPa)	53.5
In-Line bar Imperfection (E = 69 GPa)	53.91
Cross Section (A = 3 * 10^-5 m^2)	67.73
Cross Section (A = 1.5 * 10^-5 m^2)	35.47

Table 5 Percent Error relative to experimental data for different sources of error at P = 222.4 N

In order to analyze the impact of each of the potential source of uncertainly, additional FEM simulations were run.

- Lower effective stiffness in bars. Note that the FEM model predicts, on average, slightly more displacement than the experiment. If the area of the bars is reduced to 90% of the actual value, the magnitude of the displacement is increased to 2.05mm (using the full 50 lb load). This worsens the problem. If the effective stiffness is greater, - let's say 5% more area - the magnitude of the displacement is reduced to 1.68mm, which essentially matches the experimental data.
- This result is generalize: the stiffness of the bars is underestimated by the FEM model, so if the member with the load cell as reduced effective stiffness, using a lower stiffness in the model will worsen the error between experimental and FEM results.
- In order to study the effects of differing boundary conditions, the right boundary conditions are changed to constrain motion in the $\hat{x}$ direction, while keeping the actual area of the members. This results in significantly less displacement; the magnitude of the displacement is reduced to 1.17mm, which is less displacement than was measured. However, despite the slightly better results for displacement, the reaction forces are far from realistic. The magnitude of the $\hat{x}$ reactions at each node is on the order of 230N. This isn't physical. This suggests that the displacement results are quite sensitive to the boundary conditions. As was discussed previously in the report, the right hand boundary conditions lie somewhere between the $\hat{x}$ displacement being zero and the $\hat{x}$ force being zero. Unsurprisingly, the displacement values with each of these boundary conditions bound the experimental displacement. So, the FEM results might be improved if more complex boundary conditions can be incorporated.
- Manufacturing error, resulting in joint misplacement, should be relatively small, if the truss is made on a device such as a mill, the tolerance should be below 0.005 inches. In order to be conservative, each joint will have a nominal error of 0 with normally distribution centered at 0 and with a standard deviation of 1mm. Ten thousand iterations are run in order to assess whether or not node position error is to blame for the difference in displacement from FEM and experiment. The result is that the mean displacement is -1.852mm with a standard deviation of 0.007mm. This does not come close to making up the gap between theoretical and experimental values of displacement of the middle node of the truss.
- Modeling members as beams (i.e. that they can support a bending moment) is not possible with the FEM model, since it assumes the member forces are purely axial. So, quantitative results for this analysis are not possible while keeping within the scope of the lab.

C. Factor of Safety Determination

In determining the factor of safety for our 16-bay structure, we must analyze the assumptions as well as the material properties. Aluminium 6061-T6 has an Ultimate Tensile Strength of 310 MPa. This value will result in complete failure of at least one bar in the truss, if not more depending on the load or truss property. In the Table below, we can see the maximum stresses in the truss when the sources discussed in Table 5 are implemented in the FEM. It is important to note that the cross section areas that were selected ranged from 0.1 to 0.5 μm^2.

Source	Maximum Stress [MPa]
Imperfect Joints (E = 69 GPa)	20.675
In-Line bar Imperfection (E = 69 GPa)	37.25
Cross Section (A = 5 * 10^-5 m^2)	45.014
Cross Section (A = 1 * 10^-5 m^2)	9.0028

Table 6 Maximum Stress in the truss caused by a source of potential error

Looking at this table, we can obtain a factor of safety based on FEM accuracy as well as maximum stress. Since extreme cases were selected for cross sectional area and we can assume values for these, we can eliminate the possibility of using a 0.5 μm^2 cross section as our bar area. This will also result in a much larger deflection compared to the experimental values which is not ideal. Our results suggest that smaller cross sectional areas provide a more accurate model. The next and main constraint is the in-line bar imperfection which results in a maximum stress in the in-line bar of 37.35 MPa. This value serves as our upper bound for the maximum stress that can be used to determine the Factor of Safety. The Factor of Safety can be defined by the following formula:

$$FOS = \frac{\sigma_{ult}}{\sigma_{working}} \tag{1}$$

Using this formula, the Factor of Safety was determined to be 8.32.

IV. Conclusion

From the results, ANSYS analysis and equivalent beam analysis seem to match very well due to the similar assumptions that were made for both methods. On the other hand, the experimental results didn't match the ANSYS results nor equivalent beam results since the assumptions that were made for experiment and ANSYS are fairly different such as the manufacture of the strut (Hollow vs solid) etc.

Acknowledgements

Thank you to Dr. Johnson for clarifying the use of his FEM code (during spring break, no less).

V. Appendix

A. Participation Report

Team Member	Experiment	FEM Modeling	2.1	2.2	2.3	Report	Total Score
Ryan Hughes:	1	1	2	1	1	1	100
Ansh Jerath:	1	1	1	1	2	1	100
Aidan Sesnic:	1	2	1	1	1	1	100
Parker Simmons:	1	1	1	2	1	1	100
Chris Nylund:	1	1	1	1	1	1	95
Panitnan Yuvanondha:	1	1	1	1	1	1	95

Table 7 Participation table is scored as follows: 2 - Led Section, 1 - Worked On, 0 - Not Responsible For. Total Score is a numerical reflection of team members' contributions out of 100.

B. Matlab Code

Listing 1 Script for experimental data analysis

```
%% Truss Experimental Data Analysis
%
% TO DO:
% - error percentages
%   - use yFit (line of fit) vector?
%   - do it to error before avg/max values taken
% - re-do uncertainty analysis
%   - polyval error is standard error of the fit line

% Housekeeping
clear; close all; clc
```

```
%% Data
% load data for both weight increments
data5lb = load("ASEN 3112 - Sp20 - Lab 2 Data - 5 lb increments");
data10lb = load("ASEN 3112 - Sp20 - Lab 2 Data - 10 lb increments");

% sort data
load5 = data5lb(:,1); load10 = data10lb(:,1); %lbs
F0_5 = data5lb(:,2); F0_10 = data10lb(:,2); %lbf
F1_5 = data5lb(:,3); F1_10 = data10lb(:,3); %lbf
F2_5 = data5lb(:,4); F2_10 = data10lb(:,4); %lbf
F3D_5 = data5lb(:,5); F3D_10 = data10lb(:,5); %lbf
LVDT5 = data5lb(:,6); LVDT10 = data10lb(:,6); %in

% conversions
% 50 lbs = 222.4 N, use to convert lbs -> N
load5 = load5*222.4/50; load10 = load10*222.4/50;
F0_5 = F0_5*222.4/50; F0_10 = F0_10*222.4/50;
F1_5 = F1_5*222.4/50; F1_10 = F1_10*222.4/50;
F2_5 = F2_5*222.4/50; F2_10 = F2_10*222.4/50;
F3D_5 = F3D_5*222.4/50; F3D_10 = F3D_10*222.4/50;
% convert displacement from in to mm
LVDT5 = LVDT5 * 25.4; LVDT10 = LVDT10 * 25.4;
% adjust LVDT10 data for negative offset
% due to miscalibration? or something else?
LVDT10 = LVDT10 + abs(min(LVDT10));
% repeat for F3D data, same issue
% unless 0 load -> large - force (compression) and inversely proportional?
F3D_5 = F3D_5 + abs(min(F3D_5)); F3D_10 = F3D_10 + abs(min(F3D_10));

%% Analysis
% linear regression
xFit5 = [0:5:50]*222.4/50; %cut to ascending half of load data
xFit10 = [0:10:50]*222.4/50;

[pDisp5,sDisp5] = polyfit(load5,LVDT5,1);
[pF0_5,sF0_5] = polyfit(load5,F0_5,1);
[pF1_5,sF1_5] = polyfit(load5,F1_5,1);
[pF2_5,sF2_5] = polyfit(load5,F2_5,1);
[pF3D_5,sF3D_5] = polyfit(load5,F3D_5,1);
[yDispFit5,errorDisp5] = polyval(pDisp5, xFit5, sDisp5);
[yF0Fit5,errorF0_5] = polyval(pF0_5, xFit5, sF0_5);
[yF1Fit5,errorF1_5] = polyval(pF1_5, xFit5, sF1_5);
[yF2Fit5,errorF2_5] = polyval(pF2_5, xFit5, sF2_5);
[yF3DFit5,errorF3D_5] = polyval(pF3D_5, xFit5, sF3D_5);

[pDisp10,sDisp10] = polyfit(load10,LVDT10,1);
[pF0_10,sF0_10] = polyfit(load10,F0_10,1);
[pF1_10,sF1_10] = polyfit(load10,F1_10,1);
[pF2_10,sF2_10] = polyfit(load10,F2_10,1);
[pF3D_10,sF3D_10] = polyfit(load10,F3D_10,1);
[yDispFit10,errorDisp10] = polyval(pDisp10, xFit10, sDisp10);
[yF0Fit10,errorF0_10] = polyval(pF0_10, xFit10, sF0_10);
[yF1Fit10,errorF1_10] = polyval(pF1_10, xFit10, sF1_10);
[yF2Fit10,errorF2_10] = polyval(pF2_10, xFit10, sF2_10);
[yF3DFit10,errorF3D_10] = polyval(pF3D_10, xFit10, sF3D_10);


% max and average error values
maxErrorDisp5 = max(errorDisp5);
meanErrorDisp5 = mean(errorDisp5);
maxErrorF0_5 = max(errorF0_5);
meanErrorF0_5 = mean(errorF0_5);
maxErrorF1_5 = max(errorF1_5);
meanErrorF1_5 = mean(errorF1_5);
maxErrorF2_5 = max(errorF2_5);
meanErrorF2_5 = mean(errorF2_5);
maxErrorF3D_5 = max(errorF3D_5);
meanErrorF3D_5 = mean(errorF3D_5);
```

```

maxErrorDisp10 = max(errorDisp10);
meanErrorDisp10 = mean(errorDisp10);
maxErrorF0_10 = max(errorF0_10);
meanErrorF0_10 = mean(errorF0_10);
maxErrorF1_10 = max(errorF1_10);
meanErrorF1_10 = mean(errorF1_10);
maxErrorF2_10 = max(errorF2_10);
meanErrorF2_10 = mean(errorF2_10);
maxErrorF3D_10 = max(errorF3D_10);
meanErrorF3D_10 = mean(errorF3D_10);

%% Plots
% plot displacement vs. load
figure
hold on
plot(load5,LVDT5,'o', xFit5,yDispFit5)
xlabel('Load [N]'); ylabel('Linear displacement [mm]');
title('Displacement vs. Load for 5 lb. Increments');
legend('Data','Fit','Location','southeast');

figure
hold on
plot(load10,LVDT10,'o', xFit10,yDispFit10)
xlabel('Load [N]'); ylabel('Linear displacement [mm]');
title('Displacement vs. Load for 10 lb. Increments');
legend('Data','Fit','Location','southeast');

% plot reaction forces vs. load
figure
hold on
plot(load5,F0_5,'or',xFit5,yF0Fit5,'r')
plot(load5,F1_5,'ob',xFit5,yF1Fit5,'b')
plot(load5,F2_5,'ok',xFit5,yF2Fit5,'k')
xlabel('Load [N]'); ylabel('Reaction Force [N]');
title('Reaction Forces vs. Load for 5 lb. Increments');
legend('F0','F0 Fit','F1','F1 Fit','F2','F2 Fit','Location','southeast');

figure
hold on
plot(load10,F0_10,'or',xFit10,yF0Fit10,'r')
plot(load10,F1_10,'ob',xFit10,yF1Fit10,'b')
plot(load10,F2_10,'ok',xFit10,yF2Fit10,'k')
xlabel('Load [N]'); ylabel('Reaction Force [N]');
title('Reaction Forces vs. Load for 10 lb. Increments');
legend('F0','F0 Fit','F1','F1 Fit','F2','F2 Fit','Location','southeast');

% plot internal reaction vs. load
figure
hold on
plot(load5,F3D_5,'o',xFit5,yF3DFit5);
xlabel('Load [N]'); ylabel('Internal Reaction Force [N]');
title('Internal Reaction vs. Load for 5 lb. Increments');
legend('Data','Fit','Location','southeast');

figure
hold on
plot(load10,F3D_10,'o',xFit10,yF3DFit10);
xlabel('Load [N]'); ylabel('Internal Reaction Force [N]');
title('Internal Reaction vs. Load for 10 lb. Increments');
legend('Data','Fit','Location','southeast');

%% Plot FEM vs. Experimental
u_0 = 0.00185225*1000;
% 5-lb data
weight = 0:5:50;
disp = nan(1, length(weight));
for i = 1:length(weight)
```

```

    disp(i) = (weight(i)/50)*u_0;

end

% 5-lb increment
figure
hold on
plot(load5,LVDT5,'o')
plot(weight/0.22482, disp, 'o');
xlabel('Load, N');
ylabel('Displacement, mm');
title('5-lb increment: FEM vs. experimental results');
legend('Experimental', 'FEM', 'location', 'best');

% 10-lb data
weight = 0:10:50;
disp = nan(1, length(weight));
for i = 1:length(weight)

    disp(i) = (weight(i)/50)*u_0;

end

% 10-lb increment
figure
hold on
plot(load10,LVDT10,'o')
plot(weight/0.22482, disp, 'o');
xlabel('Load, N');
ylabel('Displacement, mm');
title('5-lb increment: FEM vs. experimental results');
legend('Experimental', 'FEM', 'location', 'best');
```

Listing 2 Main Script for FEM

```
%% Main script for FEM analysis of ASEN 3112 lab 3

% Aidan Sesnic
% 22 March 2020

% Constructs input matricies describing truss, boundary conditions, and
% loading. Solves for nodal displacements and axial member forces using
% finite element method per Dr. Johnson's code. Plots resulting
% deformation.

% Housekeeping
clear
clc
close all

%% Defining input parameters

% Basic truss details
d_o = 9.525; %Outer diameter, mm
t = 1.587; %Thickness, mm
d_o = d_o / 1000; %Convert to m
t = t / 1000; %Convert to m
r_o = d_o / 2; %Outer radius
A_o = pi*r_o^2; %Outside area
r_i = r_o - t; %Inner radius
A_i = pi*r_i^2; %Inner area
A = A_o - A_i; %Find area of member
L_b = 0.25; %Length between bays of truss, m
E = 69.5E9; %Elastic modulous, Pa
P = 222.4; %Load, N
```

```
bays = 16; %Size of truss
numNodes = 4 + (4*bays); %Find number of nodes
numMems = 5 + (13*bays); %Find number of members
nodetableFull = nan(numNodes, 10); %Pre-allocate node table
connecttableFull = nan(numMems, 7); %Pre-allocate member table

% Degrees of freedom
dof = 3;

% Construct basic node table
nodetable_1 = nan(8, 10);
nodetable_1(1:8, 1) = (1:8)'; %Node numbers
nodetable_1(1:8, 2) = [0; 0; 0; 0; L_b; L_b; L_b; L_b]; %X-location
nodetable_1(1:8, 3) = [0; L_b; L_b; 0; 0; L_b; L_b; 0]; %Y-location
nodetable_1(1:8, 4) = [0; 0; L_b; L_b; 0; 0; L_b; L_b]; %Z-location

% Construct basic connection table
connecttable_1 = nan(18, 7);
connecttable_1(1:18, 1) = (1:18)'; %Element numbers
connecttable_1(1:18, 2) = [1; 1; 1; 1; 2; 2; 2; 2; 3; 3; 4; 4; 4; 5; 5; 6; 6; 7]; %Node A
connecttable_1(1:18, 3) = [2; 3; 4; 5; 3; 5; 6; 7; 4; 7; 5; 7; 8; 6; 8; 7; 8; 8]; %Node B
connecttable_1(1:18, 4) = A*ones(18, 1); %Area of rods
connecttable_1(1:18, 5) = E*ones(18, 1); %Elastic modulous of rods
connecttable_1(1:18, 6:7) = zeros(18, 2); %Thermal stuff

% Append node table
nodetableFull(1:8, :) = nodetable_1;
for i = 2:bays %For each additional bay
    n = i - 1; %Let n represent number of additional bays

    % Find new row numbers of each node
    r_1 = 5 + (4*n);
    r_2 = 6 + (4*n);
    r_3 = 7 + (4*n);
    r_4 = 8 + (4*n);
    nodeID = [r_1; r_2; r_3; r_4];

    % Add node ID
    nodetableFull(r_1:r_4, 1) = nodeID;

    % Add node coordinates
    nodetableFull(r_1, 2:4) = [nodetableFull(r_1-4, 2) + L_b, nodetableFull(r_1-4, 3), ...
        nodetableFull(r_1-4, 4)];
    nodetableFull(r_2, 2:4) = [nodetableFull(r_2-4, 2) + L_b, nodetableFull(r_2-4, 3), ...
        nodetableFull(r_2-4, 4)];
    nodetableFull(r_3, 2:4) = [nodetableFull(r_3-4, 2) + L_b, nodetableFull(r_3-4, 3), ...
        nodetableFull(r_3-4, 4)];
    nodetableFull(r_4, 2:4) = [nodetableFull(r_4-4, 2) + L_b, nodetableFull(r_4-4, 3), ...
        nodetableFull(r_4-4, 4)];

end

% Append connection table
connecttableFull(1:18, :) = connecttable_1;
for i = 2:bays
    n = i - 1; %Let n represent number of additional bays

    % Find new row numbers of each member
    r_1 = 6 + (13*n);
    r = nan(13, 1);
    for j = 1:13
        r(j) = r_1 + (j - 1);
    end

    % Add member ID
    connecttableFull(r, 1) = r;

    % Use letters as stand-in for node identifiers
    a = 1 + (4*n);
    b = 2 + (4*n);
    c = 3 + (4*n);
    d = 4 + (4*n);
    e = 5 + (4*n);
    f = 6 + (4*n);
    g = 7 + (4*n);
    h = 8 + (4*n);

    % Add member connections - straight segments
    connecttableFull(r(1), 2:3) = [b f];
    connecttableFull(r(2), 2:3) = [c g];
    connecttableFull(r(3), 2:3) = [d h];
    connecttableFull(r(4), 2:3) = [a e];
    connecttableFull(r(5), 2:3) = [e f];
    connecttableFull(r(6), 2:3) = [f g];
    connecttableFull(r(7), 2:3) = [g h];
    connecttableFull(r(8), 2:3) = [e h];

    % Add member connections - diagonal segments
    if mod(n, 2) == 1 %Odd bay number
        connecttableFull(r(9), 2:3) = [c f];
        connecttableFull(r(10), 2:3) = [c h];
        connecttableFull(r(11), 2:3) = [a h];
        connecttableFull(r(12), 2:3) = [a f];
        connecttableFull(r(13), 2:3) = [e g];
    else %Even bay number
        connecttableFull(r(9), 2:3) = [b g];
        connecttableFull(r(10), 2:3) = [d g];
        connecttableFull(r(11), 2:3) = [d e];
        connecttableFull(r(12), 2:3) = [b e];
        connecttableFull(r(13), 2:3) = [f h];
    end

    % Add cross-sectional area
    connecttableFull(r, 4) = A*ones(13, 1);

    % Add modulous of elasticity
    connecttableFull(r, 5) = E*ones(13, 1);

    % Add thermal stuff
    connecttableFull(r, 6:7) = zeros(13, 2);

end

%% Add boundary conditions to node table
% Known displacements:
    % Left supports: zero in all three directions; nodes 1 and 4
    % Right supports: zero in y, z directions; nodes (end) and (end-3)
% Known forces:
    % Top middle nodes: load P/2 downward (-x) direction; nodes at end of
    % bay (bays/2); nodes at position given by -2 + 4n and -1 + 4n where n
    % = ciel((bays + 1)/2)
    % Other nodes: f=0

% Keep track of which joints BCs have been applied to; all others are 0
% force and unknown displacement
BCapplied = [];

% First apply load BC
n = ceil((bays+1)/2); %Proxy for joint identifier
nodeID = [-2 + (4*n), -1 + (4*n)]; %Joint identifier
nodetableFull(nodeID, 5:10) = [NaN NaN NaN 0 -P/2 0; NaN NaN NaN 0 -P/2 0]; %Load
BCapplied = [BCapplied, nodeID]; %Append to list of BCs applied

% Apply support BC - left
nodetableFull(1, 5:7) = [0 0 0];
nodetableFull(4, 5:7) = [0 0 0];
```

```
BCapplied = [BCapplied, 1, 4];

% Apply support BC - right
nodetableFull(numNodes-3, 5:10) = [NaN 0 0 0 NaN NaN];
nodetableFull(numNodes, 5:10) = [NaN 0 0 0 NaN NaN];
BCapplied = [BCapplied, numNodes, numNodes-3];

% Apply other BCs - zero force, unknown displacement
nodes = 1:numNodes;
nodes(BCapplied) = []; %Vector of nodes w/o BCs yet is returned
for i = nodes
    nodetableFull(i, 5:10) = [NaN NaN NaN 0 0 0];
end

%% Plot the truss

% Individual bay of truss
plot_truss(nodetable_1, connecttable_1, 'Basic Truss Element');

% Full truss
plot_truss(nodetableFull, connecttableFull, 'Full Undeformed Truss');

%% Solve for nodal displacements, member forces/stresses

% Use function from Dr. Johnson
[u, F, sigma] = johnson_FEM_code(nodetableFull, connecttableFull, dof);

%% Process member stresses

% Bin each to 100 bins
sigma_plot = abs(sigma); %Care only about magnitude of stress
sigma_bins = linspace(min(sigma_plot), max(sigma_plot), 100); %Find bin edges
sigma_binned = discretize(sigma_plot, sigma_bins); %Put stress values in bins

% Assign RGB value to stress values
sigma_color = nan(length(sigma), 3);
for i = 1:length(sigma)
    sigma_color(i, :) = RYGfade(sigma_binned(i));
end

%% Plot results of simulation

% Deformation of middle segment - reference with nodeID variable
u_middle = u([nodeID(2)+1, nodeID(1)-1], 2); %Need only y position
fprintf('Deflection of middle of truss\n');
fprintf('  Forward node: %.8f m\n', u_middle(1));
fprintf('  Aft node: %.8f m\n', u_middle(2));

% Force in middle member
nodeA = nodeID(2) - 4;
nodeB = nodeID(2);
for i = 1:numMems %Find member with appropriate indexes
    if connecttableFull(i, 2) == nodeA && connecttableFull(i, 3) == nodeB
        memberID = i;
    end
end
fprintf('Force in longeron adjacent to midspan\n');
fprintf('  Force in member: %.3f N\n', sigma(memberID)*A);

% Reaction forces
fprintf('Reaction forces \n');
fprintf('  Left aft joint: (%.3f [x], %.3f [y], %.3f [z]) N \n', F(1, 1), F(1, 2), F(1, 3));
fprintf('  Left fwd joint: (%.3f [x], %.3f [y], %.3f [z]) N \n', F(4, 1), F(4, 2), F(4, 3));
fprintf('  Right aft joint: (%.3f [x], %.3f [y], %.3f [z]) N \n', F(end-3, 1), F(end-3, ...
    2), F(end-3, 2));
fprintf('  Right fwd joint: (%.3f [x], %.3f [y], %.3f [z]) N \n', F(end, 1), F(end, 2), ...
    F(end, 2));

```

```
% Make new vector of node positions after deformation
deformationScale = 10; %Exaggerate small deformations
nodetable_deformed = nodetableFull(:, 1:4); %Don't need BCs now
nodetable_deformed(:, 2:4) = nodetable_deformed(:, 2:4) + (deformationScale * u); %Add ...
    displacements

% Deformed truss with stresses
plot_truss_deformed_stress(nodetableFull, nodetable_deformed, connecttableFull, 'Deformed ...
    Truss with Stresses', sigma_color, sigma_plot, F);
```

Listing 3 FEM Solving Code

```
%Dr. Johnson's FEM Code
%Author: Aaron Johnson
%
%FUNCTION INPUTS
%
%connecttable describes the ELEMENTS
%   1: element number
%   2: node A
%   3: node B
%   4: cross-sectional area
%   5: elastic modulus
%   6: coefficient of thermal expansion (put 0 for all members)
%   7: temperature change (put 0 for all members)
%
%nodetable describes the NODES
%   1: node number
%   2: x-coordinate
%   3: y-coordinate
%   4: z-coodinate
%   5: x-displacement (number if known, NaN if unknown)
%   6: y-displacement (number if known, NaN if unknown)
%   7: z-displacement (number if known, NaN if unknown)
%   8: external force in x-direction (number if known, NaN if unknown)
%   9: external force in y-direction (number if known, NaN if unknown)
%   10: external force in z-direction (number if known, NaN if unknown)
%
%dof is the number of degrees of freedom in the truss (put 2 for ASEN 3112)
%
%
%FUNCTION OUTPUTS
%u = matrix of displacements for each node (size n x dof, where n is the number of nodes ...
    and dof is the number of degress of freedom)
%   For row n:
%   Column 1: x-displacement of node n
%   Column 2: y-displacement of node n
%   Column 3: z-displacement of node n
%
%F = matrix of external force on each node (size n x dof)
%   For row n:
%   Column 1: x-component of external force on node n
%   Column 2: y-component of external force on node n
%   Column 3: z-component of external force on node n
%
%sigma = vector of axial stress in each element (size m x 1, where m is the number of ...
    elements)
%   For row m:
%   Column 1: axial stress in element m

function [u, F, sigma] = johnson_FEM_code(nodetable, connecttable, dof)
%%%%%%%%%%%PREPROCESSING%%%%%%%%%%
% coordinate matrix for each node with the specified dof
co = nodetable(:,2:dof+1);

% external force matrix for each node with the specified dof
```

```
F = nodetable(:,8:dof+7);

% displacement matrix for each node with the specified dof
u = nodetable(:,5:dof+4);

Nel = size(connecttable,1); % number of elements
Nnodes = size(co,1);        % number of nodes

%%% initialize global stiffness matrix 'K' and force vector 'F'
K = zeros(Nnodes*dof,Nnodes*dof);
%Ftemp = zeros(Nnodes*dof,1);    % Column vector
%Fthermal = Ftemp;               % Column vector
Ftemp = reshape(F',Nnodes*dof,1);   % Column vector
Fthermal = zeros(Nnodes*dof,1);     % Column vector
sigma = nan(Nel,1);

%%% Assemble global K
% For each ELEMENT
for A = 1:Nel                       % for each element
    n = (co(connecttable(A,3),:) - co(connecttable(A,2),:));  % vector from node 2 to node 1
    L = norm(n);                    % length of element
    n = n./L;                       % unit vector
    Area = connecttable(A,4);       % area of element
    E = connecttable(A,5);          % elastic modulus of element
    alpha = connecttable(A,6);      % coefficient of thermal expansion
    delT = connecttable(A,7);       % temperature change

    k11 = (E*Area/L)*(n'*n);        % khat matrix, always 3x3

    % local stiffness matrix and force vector
    % localstiffness = [k11 -k11;-k11 k11];

    n1end = connecttable(A,2)*dof;        % Node 1 index end
    n1start = n1end-dof+1;                % Node 1 index start
    n2end = connecttable(A,3)*dof;        % Node 2 index end
    n2start = n2end-dof+1;                % Node 2 index start

    K(n1start:n1end, n1start:n1end) = K(n1start:n1end, n1start:n1end) + k11;
    K(n2start:n2end, n2start:n2end) = K(n2start:n2end, n2start:n2end) + k11;
    K(n1start:n1end, n2start:n2end) = K(n1start:n1end, n2start:n2end) - k11;
    K(n2start:n2end, n1start:n1end) = K(n2start:n2end, n1start:n1end) - k11;

    Fthermal(n1start:n1end) = Fthermal(n1start:n1end) + alpha*delT*E*Area*n';
    Fthermal(n2start:n2end) = Fthermal(n2start:n2end) - alpha*delT*E*Area*n';
end

Kfull = K;

%%% Incorporate boundary conditions
% For each NODE
for A=Nnodes:-1:1                   % Work backwards so as to not mess up indexing
    nend = A*dof;                   % Last index for this node
    for B=dof:-1:1                  % Check each dof
        if isnan(F(A,B))            % If the force is NaN (unknown) at this node in this dof
            K(nend-dof+B,:) = [];                % remove row of K
            Ftemp(nend-dof+B,:) = [];            % remove row of Ftemp
            Kcolumn = K(:,nend-dof+B).*u(A,B);   % store column of K multiplied by known ...
                displacement     W19
            K(:,nend-dof+B) = [];                % remove column of K
            Ftemp = Ftemp - Kcolumn;             % modify Ftemp by the column of K x ...
                displacement         W19
        else                        % if the force is known at this node in this dof
            Ftemp(nend-dof+B) = Ftemp(nend-dof+B) - Fthermal(nend-dof+B); % add thermal ...
                effects
        end
    end
end
%
```

```
%     if isnan(F(A,1))                 % Nodes where force is Nan (unknown)
%         K(nstart:nend,:) =[];            % remove rows of K
%         K(:,nstart:nend) =[];            % remove columns of K
%         Ftemp(nstart:nend,:) = [];       % remove rows of Ftemp
%     else                             % Nodes where force is known
%         Ftemp(nstart:nend) = F(A,:)';    % populate rows of Ftemp
%     end
% end

% Solve for unknown displacements u
% utemp = K\Ftemp;
utemp = K^(-1)*Ftemp;

% Reassemble u vector
index = 1;                           % Start an index to pick off elements of the utemp vector
for A=1:Nnodes                       % For each node
    for B=1:dof
        if isnan(u(A,B))             % If the displacement of this node is unknown
            u(A,B) = utemp(index);   % Pick off the next elements of the utemp vector
            index = index+1;         % Advance the index
        end
    end
end

% Calculate reaction forces
u2 = u'; u2 = u2(:);                         % Reshape the u vector back into a column vector
Ftemp2 = Kfull*u2 + Fthermal;                % Calculate the column vector of all external ...
    forces
F = reshape(Ftemp2,[dof,Nnodes]); F = F';    % Reshape the F vector

% Calculate internal forces
for A = 1:Nel                        % for each element
    n = (co(connecttable(A,3),:) - co(connecttable(A,2),:));    % vector from node 2 to ...
        node 1
    Δ = u(connecttable(A,3),:) - u(connecttable(A,2),:);   % elongation of bar (node 2 - ...
        node 1)
    L = norm(n);                     % length of element
    n = n./L;                        % unit vector
    Area = connecttable(A,4);        % area of element
    E = connecttable(A,5);           % elastic modulus of element
    alpha = connecttable(A,6);       % coefficient of thermal expansion
    delT = connecttable(A,7);        % temperature change
    P = (E*Area/L)*dot(Δ,n) - alpha*delT*E*Area;
    sigma(A) = P/Area;
end
```

Listing 4 Function to Plot Truss

```
%% Function to plot truss for ASEN3112 lab 2

% Originally by Dr. Johnson, modified by Aidan Sesnic
% 21 March 2020

function plot_truss(nodetable, connecttable, titletext)

figure
hold on

% Plot members
for m=1:size(connecttable,1)
    plot3([nodetable(connecttable(m,2),2);nodetable(connecttable(m,3),2)],...
          [nodetable(connecttable(m,2),3);nodetable(connecttable(m,3),3)],...
          [nodetable(connecttable(m,2),4);nodetable(connecttable(m,3),4)],'k');
end

% Plot joints
```

```
for m=1:size(connecttable,1)
    plot3([nodetable(connecttable(m,2),2);nodetable(connecttable(m,3),2)],...
          [nodetable(connecttable(m,2),3);nodetable(connecttable(m,3),3)],...
          [nodetable(connecttable(m,2),4);nodetable(connecttable(m,3),4)],'*r');
end

% Plot loads - look for nonzero/non-NAN entries in nodetable
nodeLoads = [nodetable(:, 1), nodetable(:, 8:10)];
[rows, ¬] = size(nodeLoads);
for i = 1:rows
    % X direction
    if ¬isnan(nodeLoads(i, 2)) && nodeLoads(i, 2) ≠ 0
        load = 0.5*sign(nodeLoads(i, 2));
        quiver3(nodetable(i, 2)-load, nodetable(i, 2), nodetable(i, 3), load, 0, 0, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2)-load, nodetable(i, 3), nodetable(i, 4), [num2str(nodeLoads(i, ...
            3)), 'N']);
    end

    % Y direction
    if ¬isnan(nodeLoads(i, 3)) && nodeLoads(i, 3) ≠ 0
        load = 0.5*sign(nodeLoads(i, 3));
        quiver3(nodetable(i, 2), nodetable(i, 3)-load, nodetable(i, 4), 0, load, 0, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2), nodetable(i, 3)-load, nodetable(i, 4), [num2str(nodeLoads(i, ...
            3)), 'N']);
    end

    % Z direction
    if ¬isnan(nodeLoads(i, 4)) && nodeLoads(i, 4) ≠ 0
        load = 0.5*sign(nodeLoads(i, 4));
        quiver3(nodetable(i, 2), nodetable(i, 3), nodetable(i, 4)-load, 0, 0, load, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2), nodetable(i, 3), nodetable(i, 4)-load, [num2str(nodeLoads(i, ...
            3)), 'N']);
    end
end

% Label plot
xlabel('X dimension, m');
ylabel('Y dimension, m');
zlabel('Z dimension, m');
title(titletext);
axis equal

hold off
```

Listing 5   Function to Plot Deformed Truss and Stresses

```
%% Function to plot truss for ASEN3112 lab 2

% Originally by Dr. Johnson, modified by Aidan Sesnic
% 21 March 2020

function plot_truss_deformed_stress(nodetable, nodetable_deformed, connecttable, ...
    titletext, sigma_color, sigma, F)

figure
hold on

% Plot members - undeformed
for m=1:size(connecttable,1)
    plot3([nodetable(connecttable(m,2),2);nodetable(connecttable(m,3),2)],...
          [nodetable(connecttable(m,2),3);nodetable(connecttable(m,3),3)],...
          [nodetable(connecttable(m,2),4);nodetable(connecttable(m,3),4)],'k');
end
```

```matlab

% Plot joints - undeformed
for m=1:size(connecttable,1)
    plot3([nodetable(connecttable(m,2),2);nodetable(connecttable(m,3),2)],...
          [nodetable(connecttable(m,2),3);nodetable(connecttable(m,3),3)],...
          [nodetable(connecttable(m,2),4);nodetable(connecttable(m,3),4)],'*k');
end

% Plot loads - look for nonzero/non-NAN entries in nodetable
nodeLoads = [nodetable(:, 1), nodetable(:, 8:10)];
[rows, ¬] = size(nodeLoads);
for i = 1:rows
    % X direction
    if ¬isnan(nodeLoads(i, 2)) && nodeLoads(i, 2) ≠ 0
        load = 0.5*sign(nodeLoads(i, 2));
        quiver3(nodetable(i, 2)-load, nodetable(i, 2), nodetable(i, 3), load, 0, 0, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2)-load, nodetable(i, 3), nodetable(i, 4), [num2str(nodeLoads(i, ...
            3)), 'N']);
    end

    % Y direction
    if ¬isnan(nodeLoads(i, 3)) && nodeLoads(i, 3) ≠ 0
        load = 0.5*sign(nodeLoads(i, 3));
        quiver3(nodetable(i, 2), nodetable(i, 3)-load, nodetable(i, 4), 0, load, 0, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2), nodetable(i, 3)-load, nodetable(i, 4), [num2str(nodeLoads(i, ...
            3)), 'N']);
    end

    % Z direction
    if ¬isnan(nodeLoads(i, 4)) && nodeLoads(i, 4) ≠ 0
        load = 0.5*sign(nodeLoads(i, 4));
        quiver3(nodetable(i, 2), nodetable(i, 3), nodetable(i, 4)-load, 0, 0, load, 'b', ...
            'linewidth', 2);
        text(nodetable(i, 2), nodetable(i, 3), nodetable(i, 4)-load, [num2str(nodeLoads(i, ...
            3)), 'N']);
    end
end

% Plot members with colors corresponding to stress
for m=1:size(connecttable,1)

    % Find nodes which the members connect
    nodeA = connecttable(m, 2);
    nodeB = connecttable(m, 3);

    % Find coordinates of nodes
    r_A = nodetable_deformed(nodeA, 2:4);
    r_B = nodetable_deformed(nodeB, 2:4);

    % Plot it
    plot3([r_A(1), r_B(1)], [r_A(2), r_B(2)], [r_A(3), r_B(3)], 'color', sigma_color(m, ...
        :), 'linewidth', 2);

end

% Plot joints - deformed
for m=1:size(connecttable,1)
    plot3([nodetable_deformed(connecttable(m,2),2);nodetable_deformed(connecttable(m,3),2)],...
          [nodetable_deformed(connecttable(m,2),3);nodetable_deformed(connecttable(m,3),3)],...
          [nodetable_deformed(connecttable(m,2),4);nodetable_deformed(connecttable(m,3),4)],'*b');
end

% Add color scale for stresses
color = nan(101, 3);
for i = 0:100
    color(i+1, :) = RYGfade(i);
```

```
end
colormap(color);
set(gca, 'CLim', [min(sigma), max(sigma)]);
c = colorbar;

% Reaction forces - external forces on joints 1, 4, (end), (end-3)
jointID = [1, 4, rows, rows-3]; %Joints where reactions take place
for i = 1:length(jointID)
    r = nodetable_deformed(jointID(i), 2:4); %Find position vector of joint
    f = F(jointID(i), :); %Find force vector at joint
    f = f/norm(f); %Resize force to make it fit with the scale of the plotted truss
    quiver3(r(1), r(2), r(3), f(1), f(2), f(3), 'm');
end

% Label plot
xlabel('X dimension, m');
ylabel('Y dimension, m');
zlabel('Z dimension, m');
title(titletext);
c.Label.String = 'Normal Stress Magnitude, Pa';
axis equal

hold off
```

Listing 6 Function to Generate Red-Yellow-Green Fade

```
%% Function to produce a color fade

% Aidan Sesnic
% 23 March 2020

% 100 point fade from red to green via yellow. 100 is red, 0 is green

function [RGB] = RYGfade(bin)

bin = (0.5*bin) + 25; %Adjust bin - with 'raw' bin number, colors at extremes are hard to ...
    distinguish

RGB = zeros(1, 3);

RGB(1) = bin/100; %Set red value
RGB(2) = (-0.01*bin) + 1; %Set green value
RGB(3) = 0; % Set blue value

% Ensure max value is 1 to avoid dark stuff in the middle
if max(RGB) ≠ 1
    factor = 1 / max(RGB);
    index = find(RGB == max(RGB), 1);
    RGB(index) = RGB(index) * factor;
    if index == 1
        index2 = 2;
    else
        index2 = 1;
    end

    RGB(index2) = RGB(index2) * factor;

end

end
```

Listing 7 Monte Carlo Simulation for Truss Node Positioning Error

```
%% Main script for FEM analysis of ASEN 3112 lab 3

```

```
% Aidan Sesnic
% 2 April 2020

% Constructs input matricies describing truss, boundary conditions, and
% loading. Solves for nodal displacements and axial member forces using
% finite element method per Dr. Johnson's code. Plots resulting
% deformation. Runs with 100k normally-distributed error values for joint
% positions.

% Housekeeping
clear
clc
close all

%% Defining input parameters

% Basic truss details
d_o = 9.525; %Outer diameter, mm
t = 1.587; %Thickness, mm
d_o = d_o / 1000; %Convert to m
t = t / 1000; %Convert to m
r_o = d_o / 2; %Outer radius
A_o = pi*r_o^2; %Outside area
r_i = r_o - t; %Inner radius
A_i = pi*r_i^2; %Inner area
A = A_o - A_i; %Find area of member
L_b = 0.25; %Length between bays of truss, m
E = 69.5E9; %Elastic modulous, Pa
P = 222.4; %Load, N
bays = 16; %Size of truss
numNodes = 4 + (4*bays); %Find number of nodes
numMems = 5 + (13*bays); %Find number of members
nodetableFull = nan(numNodes, 10); %Pre-allocate node table
connecttableFull = nan(numMems, 7); %Pre-allocate member table

% Degrees of freedom
dof = 3;

% Construct basic node table
nodetable_1 = nan(8, 10);
nodetable_1(1:8, 1) = (1:8)'; %Node numbers
nodetable_1(1:8, 2) = [0; 0; 0; 0; L_b; L_b; L_b; L_b]; %X-location
nodetable_1(1:8, 3) = [0; L_b; L_b; 0; 0; L_b; L_b; 0]; %Y-location
nodetable_1(1:8, 4) = [0; 0; L_b; L_b; 0; 0; L_b; L_b]; %Z-location

% Construct basic connection table
connecttable_1 = nan(18, 7);
connecttable_1(1:18, 1) = (1:18)'; %Element numbers
connecttable_1(1:18, 2) = [1; 1; 1; 1; 2; 2; 2; 2; 3; 3; 4; 4; 4; 5; 5; 6; 6; 7]; %Node A
connecttable_1(1:18, 3) = [2; 3; 4; 5; 3; 5; 6; 7; 4; 7; 5; 7; 8; 6; 8; 7; 8; 8]; %Node B
connecttable_1(1:18, 4) = A*ones(18, 1); %Area of rods
connecttable_1(1:18, 5) = E*ones(18, 1); %Elastic modulous of rods
connecttable_1(1:18, 6:7) = zeros(18, 2); %Thermal stuff

% Append node table
nodetableFull(1:8, :) = nodetable_1;
for i = 2:bays %For each additional bay
    n = i - 1; %Let n represent number of additional bays

    % Find new row numbers of each node
    r_1 = 5 + (4*n);
    r_2 = 6 + (4*n);
    r_3 = 7 + (4*n);
    r_4 = 8 + (4*n);
    nodeID = [r_1; r_2; r_3; r_4];

    % Add node ID
    nodetableFull(r_1:r_4, 1) = nodeID;

    % Add node coordinates
    nodetableFull(r_1, 2:4) = [nodetableFull(r_1-4, 2) + L_b, nodetableFull(r_1-4, 3), ...
        nodetableFull(r_1-4, 4)];
    nodetableFull(r_2, 2:4) = [nodetableFull(r_2-4, 2) + L_b, nodetableFull(r_2-4, 3), ...
        nodetableFull(r_2-4, 4)];
    nodetableFull(r_3, 2:4) = [nodetableFull(r_3-4, 2) + L_b, nodetableFull(r_3-4, 3), ...
        nodetableFull(r_3-4, 4)];
    nodetableFull(r_4, 2:4) = [nodetableFull(r_4-4, 2) + L_b, nodetableFull(r_4-4, 3), ...
        nodetableFull(r_4-4, 4)];

end

% Append connection table
connecttableFull(1:18, :) = connecttable_1;
for i = 2:bays
    n = i - 1; %Let n represent number of additional bays

    % Find new row numbers of each member
    r_1 = 6 + (13*n);
    r = nan(13, 1);
    for j = 1:13
        r(j) = r_1 + (j - 1);
    end

    % Add member ID
    connecttableFull(r, 1) = r;

    % Use letters as stand-in for node identifiers
    a = 1 + (4*n);
    b = 2 + (4*n);
    c = 3 + (4*n);
    d = 4 + (4*n);
    e = 5 + (4*n);
    f = 6 + (4*n);
    g = 7 + (4*n);
    h = 8 + (4*n);

    % Add member connections - straight segments
    connecttableFull(r(1), 2:3) = [b f];
    connecttableFull(r(2), 2:3) = [c g];
    connecttableFull(r(3), 2:3) = [d h];
    connecttableFull(r(4), 2:3) = [a e];
    connecttableFull(r(5), 2:3) = [e f];
    connecttableFull(r(6), 2:3) = [f g];
    connecttableFull(r(7), 2:3) = [g h];
    connecttableFull(r(8), 2:3) = [e h];

    % Add member connections - diagonal segments
    if mod(n, 2) == 1 %Odd bay number
        connecttableFull(r(9), 2:3) = [c f];
        connecttableFull(r(10), 2:3) = [c h];
        connecttableFull(r(11), 2:3) = [a h];
        connecttableFull(r(12), 2:3) = [a f];
        connecttableFull(r(13), 2:3) = [e g];
    else %Even bay number
        connecttableFull(r(9), 2:3) = [b g];
        connecttableFull(r(10), 2:3) = [d g];
        connecttableFull(r(11), 2:3) = [d e];
        connecttableFull(r(12), 2:3) = [b e];
        connecttableFull(r(13), 2:3) = [f h];
    end

    % Add cross-sectional area
    connecttableFull(r, 4) = A*ones(13, 1);

    % Add modulous of elasticity
    connecttableFull(r, 5) = E*ones(13, 1);
    % Add thermal stuff
    connecttableFull(r, 6:7) = zeros(13, 2);

end

%% Add boundary conditions to node table
% Known displacements:
    % Left supports: zero in all three directions; nodes 1 and 4
    % Right supports: zero in y, z directions; nodes (end) and (end-3)
% Known forces:
    % Top middle nodes: load P/2 downward (-x) direction; nodes at end of
    % bay (bays/2); nodes at position given by -2 + 4n and -1 + 4n where n
    % = ciel((bays + 1)/2)
    % Other nodes: f=0

% Keep track of which joints BCs have been applied to; all others are 0
% force and unknown displacement
BCapplied = [];

% First apply load BC
n = ceil((bays+1)/2); %Proxy for joint identifier
nodeID = [-2 + (4*n), -1 + (4*n)]; %Joint identifier
nodetableFull(nodeID, 5:10) = [NaN NaN NaN 0 -P/2 0; NaN NaN NaN 0 -P/2 0]; %Load
BCapplied = [BCapplied, nodeID]; %Append to list of BCs applied

% Apply support BC - left
nodetableFull(1, 5:7) = [0 0 0];
nodetableFull(4, 5:7) = [0 0 0];
BCapplied = [BCapplied, 1, 4];

% Apply support BC - right
nodetableFull(numNodes-3, 5:10) = [NaN 0 0 0 NaN NaN];
nodetableFull(numNodes, 5:10) = [NaN 0 0 0 NaN NaN];
BCapplied = [BCapplied, numNodes, numNodes-3];

% Apply other BCs - zero force, unknown displacement
nodes = 1:numNodes;
nodes(BCapplied) = []; %Vector of nodes w/o BCs yet is returned
for i = nodes
    nodetableFull(i, 5:10) = [NaN NaN NaN 0 0 0];
end


%% MCS
stddev = 0.001; %Maximum displacement
q = 10000; %Number of iterations

nodetable_test = nodetableFull; %Create placeholder node table to send to FEM

disp_MCS = nan(1, q); %Initialize displacement matrix

for i = 1:q

    % Add random displacement to nodes
    [num_nodes, ¬] = size(nodetableFull);
    for j = 1:num_nodes

        nodetable_test(j, 2) = nodetableFull(j, 2) + normrnd(0, stddev);
        nodetable_test(j, 3) = nodetableFull(j, 3) + normrnd(0, stddev);
        nodetable_test(j, 4) = nodetableFull(j, 4) + normrnd(0, stddev);

    end

    % Use function from Dr. Johnson
    [u, F, sigma] = johnson_FEM_code(nodetable_test, connecttableFull, dof);

    % Deformation of middle segment - reference with nodeID variable
    u_middle = u([nodeID(2)+1, nodeID(1)-1], 2); %Need only y position
    disp_MCS(i) = u_middle(1);

end

% Plot results
fprintf('Displacement of middle member: \n');
fprintf('%.8f mm +/- %.8f mm \n', mean(disp_MCS)*1000, std(disp_MCS)*1000);
```

References

[1] Johnson, Aaron "ASEN 3112 Lab 2" published on Feb. 14, 2020, pp.1-10.

[2] “Pearson Correlation Coefficient.” Wikipedia, Wikimedia Foundation, 7 Mar. 2020