To: Professor Zhuming Bi

From: Kade Bontrager, Mason Averill, Kennedy Campbell, and Kellen Larsen

Subject: Finite Element Analysis Project

Date: December 7th, 2020

PURDUE UNIVERSITY FORT WAYNE

Purpose

The purpose of this memo is to communicate the results of the ME 480 Design Project. This project was conducted throughout the Fall 2020 semester by Mason Averill, Kade Bontrager, Kennedy Campbell, and Kellen Larsen.

Objective

The objective of this project was to optimize the profile of a cantilever beam subjected to various loading conditions. The optimum case should minimize deflection while also maintaining a factor of safety against failure of more than 1.

Introduction

In structural mechanics and analysis, minimizing weight while maximizing stiffness is a very common objective. While this may be a relatively simple task for simple loading situations, this is far from the case for more complex loading types. An analytical solution is typically not possible for these situations and one must resort to computational solution methods. This was demonstrated with this project by first considering a relatively simple problem consisting of the following:

A cantilever beam of constant thickness is subjected to a point load at the free end in the vertical direction. With a maximum beam weight constraint known, determine the optimum height of the beam as a function of its length.

This problem was first optimized using analytical solution methods implemented computationally in MATLAB. Next, with an optimum profile determined, the deflection of the beam was found utilizing the finite element method in MATLAB. The parameters for this problem consisted of the following:

- Constant thickness of 50 mm
- Length of 1 m
- A 5000 N point load at the free end
- A density of 7900 kg/m^3
- Young's Modulus of 200 GPa
- Maximum beam weight of 200 N

The following function types for the profile were considered:

1. Constant cross section case
2. Linear case
3. Polynomial Order 2 case
4. Polynomial Order 4 case
5. Polynomial Order 6 case
6. Polynomial Order 12 case
7. Exponential case

Each case was optimized in MATLAB and the results were as follows (note that all following results are a ratio of the deflection obtained for the specific case compared to the constant cross section case):

- Linear Case: 0.61618
- Polynomial Order 2 case: 0.67739
- Polynomial Order 4 case: 0.76206
- Polynomial Order 6 case: 0.81261
- Polynomial Order 12 case: 0.88603
- Exponential case: 0.64671

With this, it was evident that the linear case was the ideal function to describe the height of the beam as a function of its length to minimize the deflection at the free end. The function describing the height of the beam from the wall towards the free end is as follows:

$$h(x)= -0.063427x+0.083327 \quad \textbf{(1)}$$

This implies that the height at the fixed end is 0.083327 m and the height at the free end is 0.0199 m.

Next, a separate MATLAB script was written to implement the FEM method based on the results obtained from the optimizing script. The results were as shown by Figures 1-4.

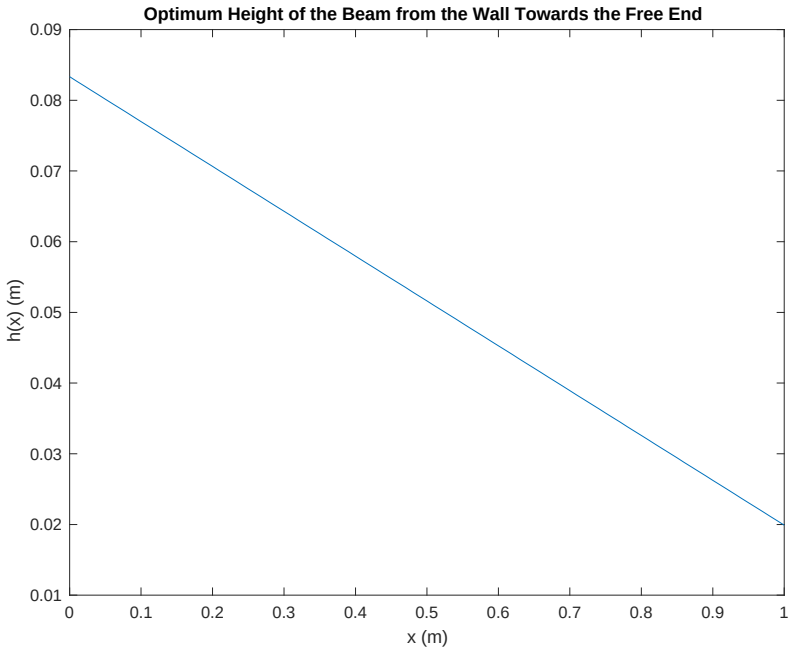


Figure 1: Plot displaying the optimum height of the beam as the distance from the wall increases.

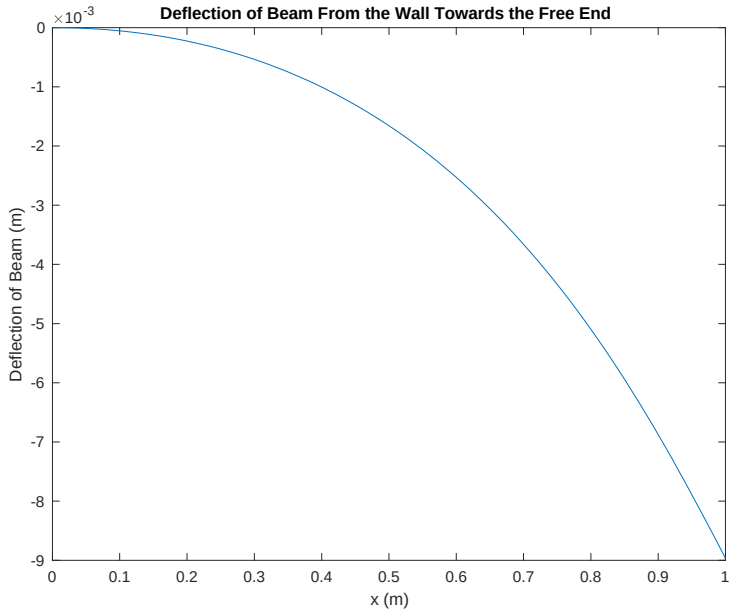


Figure 2: Plot of the deflection of the beam as the distance from the wall increases.

The deflection at the free end is 0.0089638 m or 8.9638 mm.

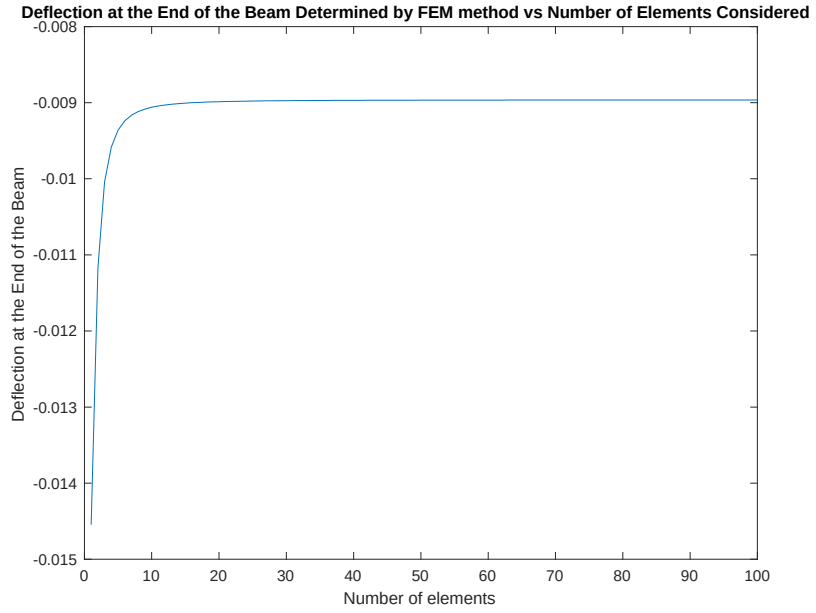


Figure 3: Plot of the beam deflection at the end versus the number of elements considered.

Figure 3 demonstrates that the results converge as more elements are considered.

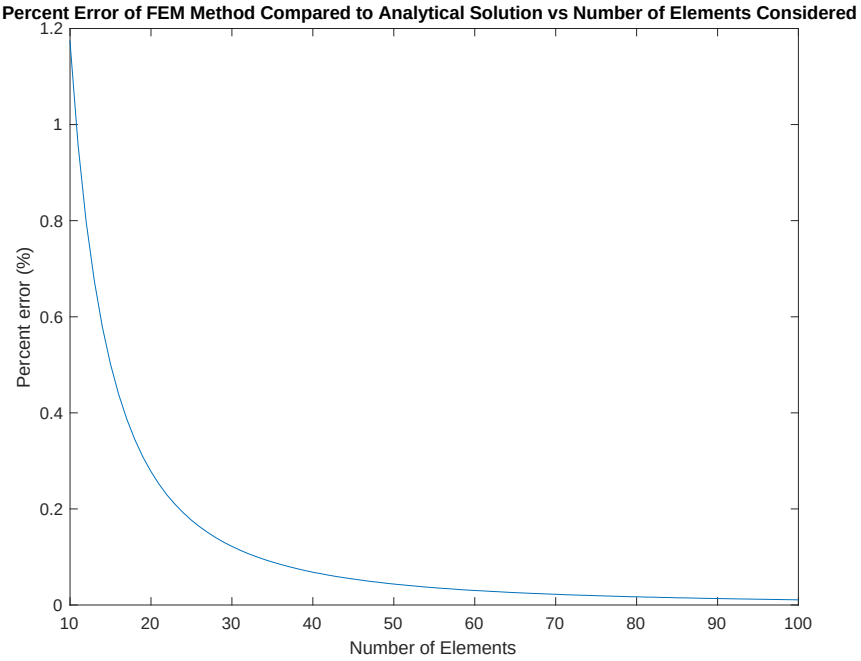


Figure 4: Plot of the percent error between FEM and analytical solution versus number of elements.

This Figure demonstrates that the results of the FEM solution method match very well with the analytical solution with enough elements considered.

Next, a SOLIDWORKS simulation was utilized to determine the ideal profile type for the same parameters to ensure that similar results were obtained—verifying both the MATLAB solution as well as the optimization capabilities of SOLIDWORKS. Figures 5-10 depict these results.

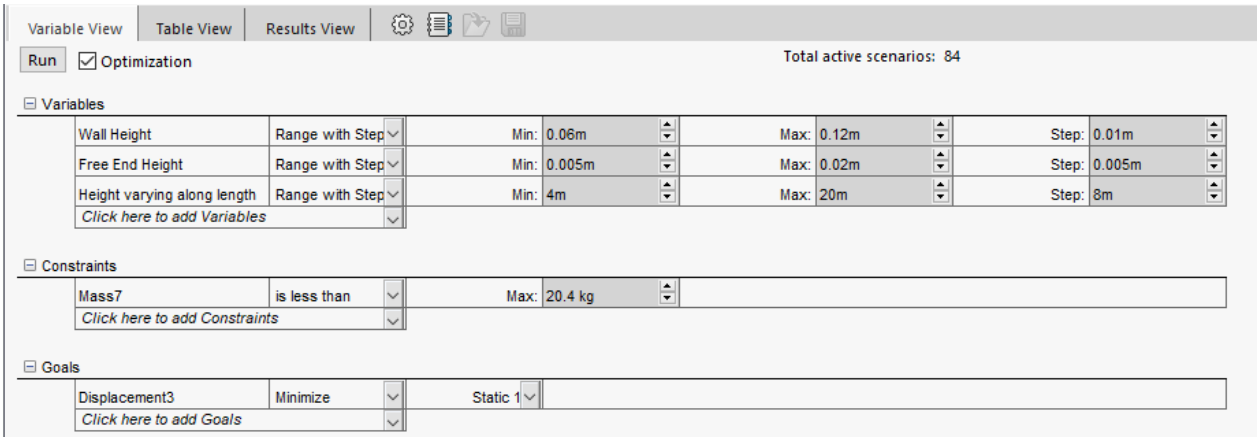


Figure 5: Optimization Confirmation Sweep 1 – Parameters

		Current	Initial	Optimal (75)
Wall Height		0.12m	0.12m	0.1m
Free End Height		0.02m	0.02m	0.015m
Height varying along length		20m	20m	20m
Mass7	< 20.4 kg	24.70013 kg	24.70013 kg	19.83052 kg
Displacement3	Minimize	5.68086mm	5.68086mm	11.53745mm

Figure 6: Optimization Confirmation Sweep 1 – Results

After sweep 1, the parameter ranges were decreased, and the resolutions were increased.

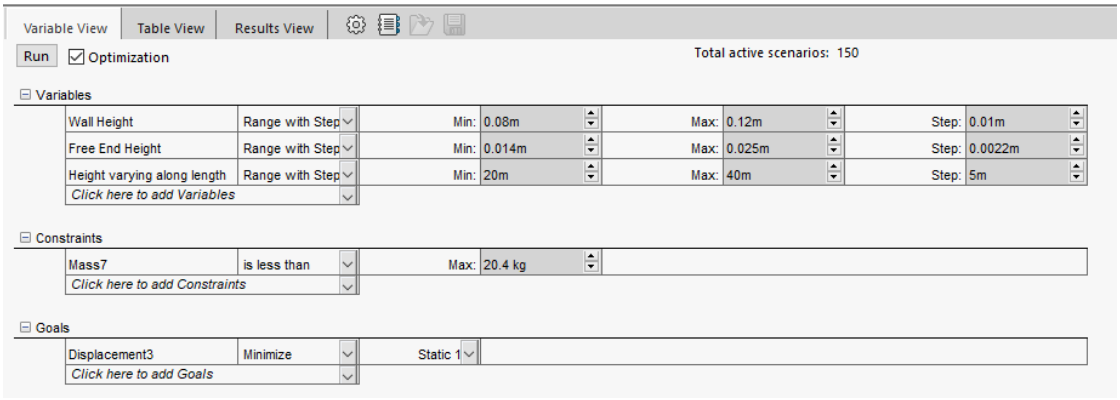


Figure 7: Optimization Confirmation Sweep 2 – Parameters

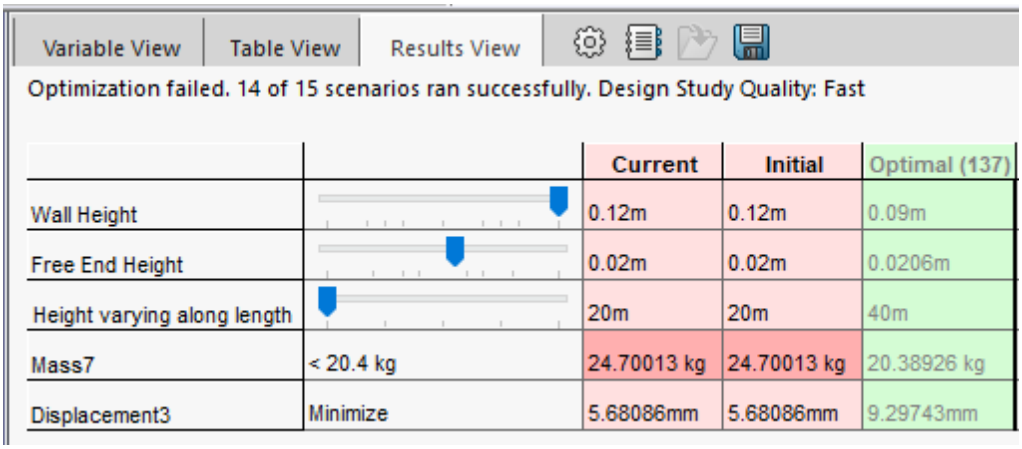


Figure 8: Optimization Confirmation Sweep 2 – Results

Again, the resolution was bumped up while narrowing in on the ideal case for the next sweep, as shown below.

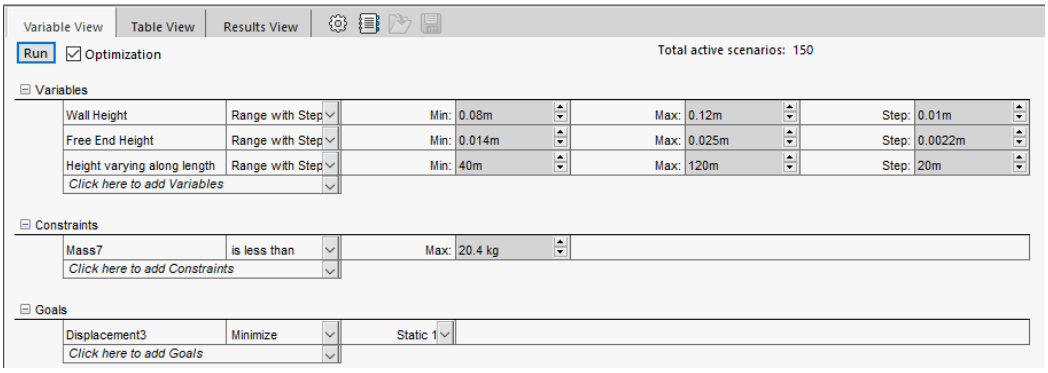


Figure 9: Optimization Confirmation Sweep 3 – Parameters

		Current	Initial	Optimal (146)
Wall Height		0.08m	0.08m	0.08m
Free End Height		0.025m	0.025m	0.025m
Height varying along length		40m	40m	120m
Mass7	< 20.4 kg	19.2815 kg	19.2815 kg	20.25196 kg
Displacement3	Minimize	10.98073mm	10.98073mm	9.27714mm

Figure 10: Optimization Confirmation Sweep 3 – Results

At this point, it was evident that SOLIDWORKS had also determined the ideal profile type to be linear, as the radius controlling the height of the beam as a function of its length took on the maximum allowable value for each sweep 1 through 3. This is because as the radius controlling the height of the beam as a function of its length increases, it tends towards a linear solution type. With the profile type now determined, the radius was assigned a constant large value (representative of a linear solution type) and the height at the free end, the height at the wall, and the total deflection was sought. Figures 11-18 depict these results.

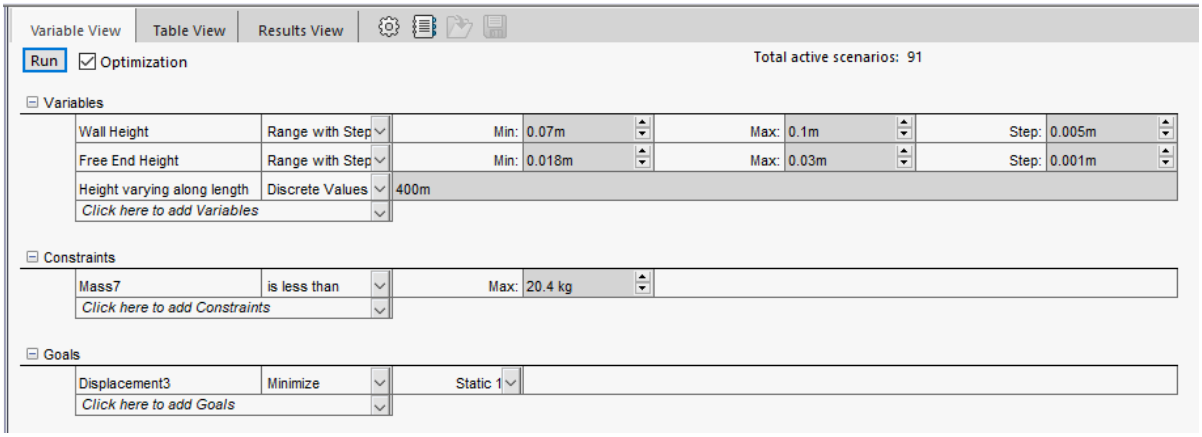


Figure 11: Optimization Confirmation Sweep 4 – Parameters

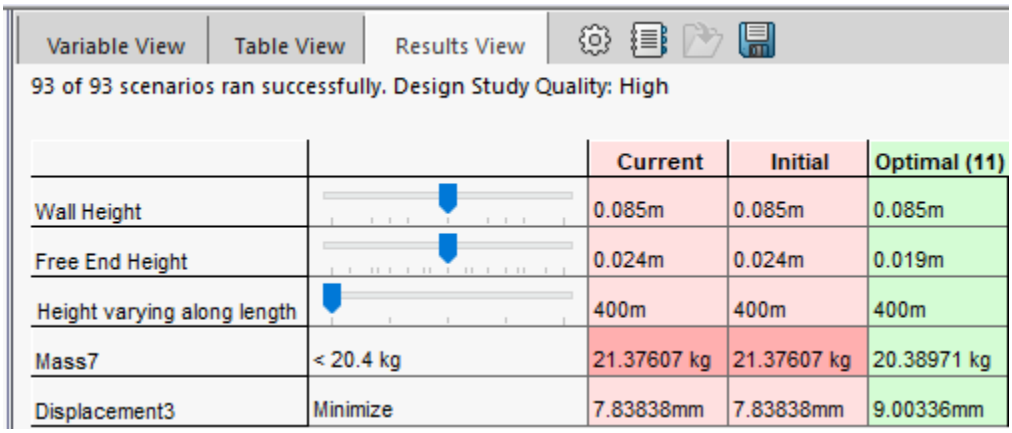


Figure 12: Optimization Confirmation Sweep 4 – Results

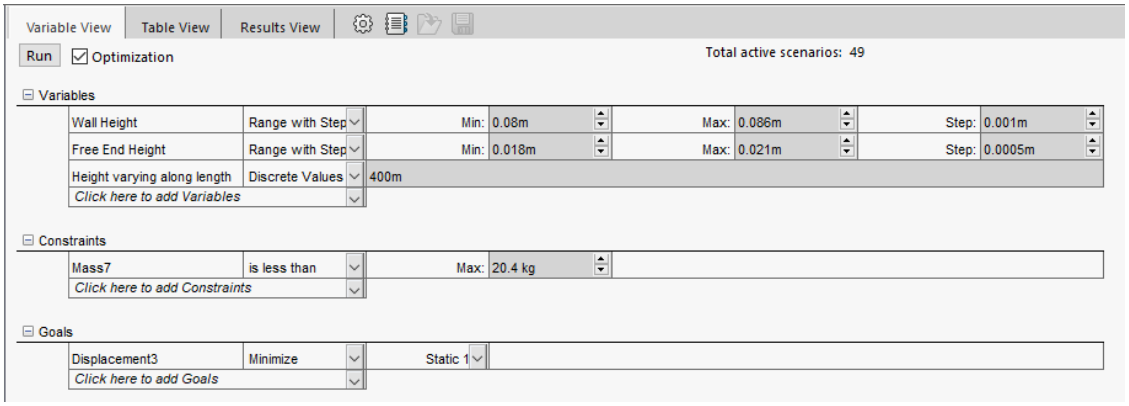


Figure 13: Optimization Confirmation Sweep 5 – Parameters

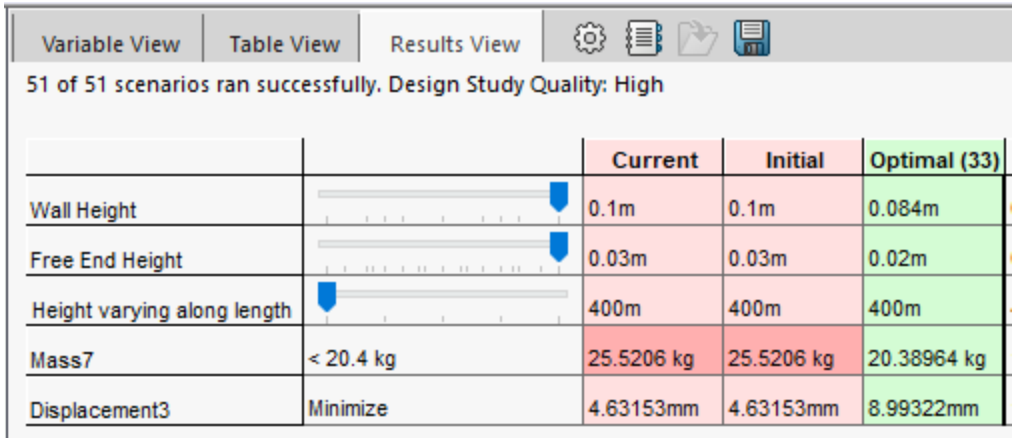


Figure 14: Optimization Confirmation Sweep 5 – Results

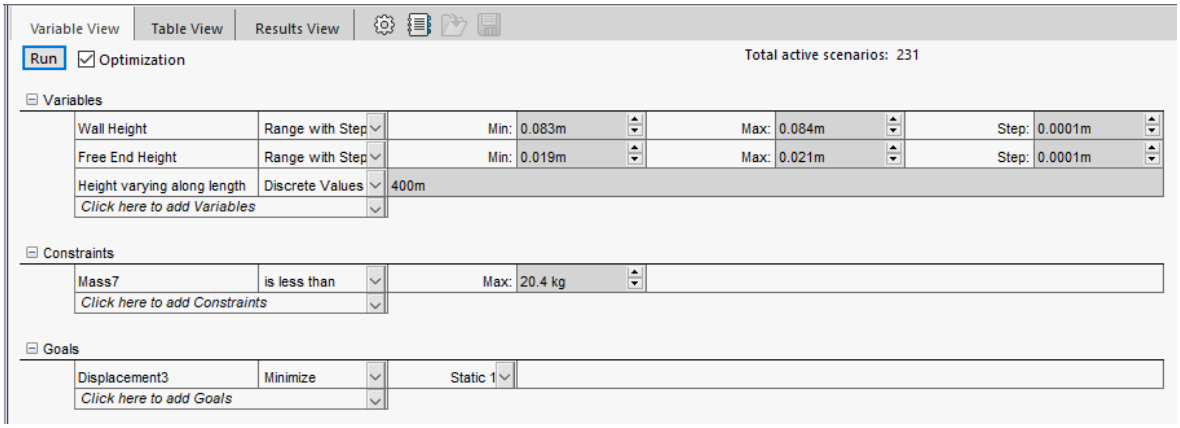


Figure 15: Optimization Confirmation Sweep 6 – Parameters

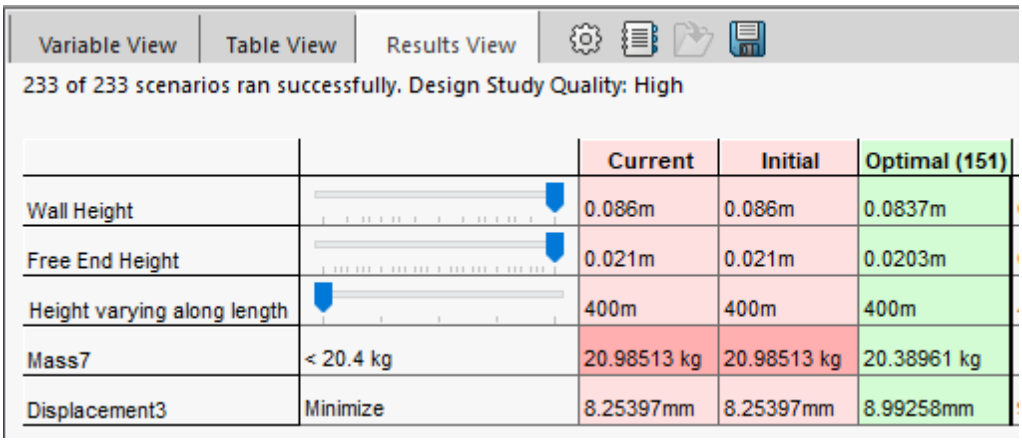


Figure 16: Optimization Confirmation Sweep 6 – Results

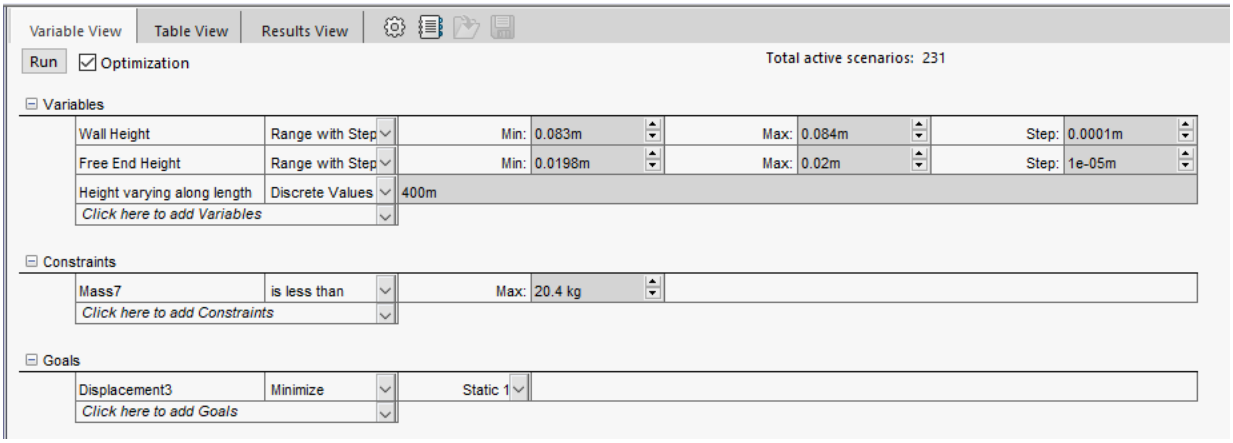


Figure 17: Optimization Confirmation Sweep 7 – Parameters

Variable View | Table View | Results View

233 of 233 scenarios ran successfully. Design Study Quality: High

		Current	Initial	Optimal (231)
Wall Height		0.0835m	0.0835m	0.084m
Free End Height		0.02m	0.02m	0.02m
Height varying along length		400m	400m	400m
Mass7	< 20.4 kg	20.29094 kg	20.29094 kg	20.38962 kg
Displacement3	Minimize	9.12367mm	9.12367mm	8.99236mm

Figure 18: Optimization Confirmation Sweep 7 – Results

Now that an optimum case had been determined, the resolution of the mesh was increased, and a static analysis was run. Figures 19-22 depict the results of this analysis.

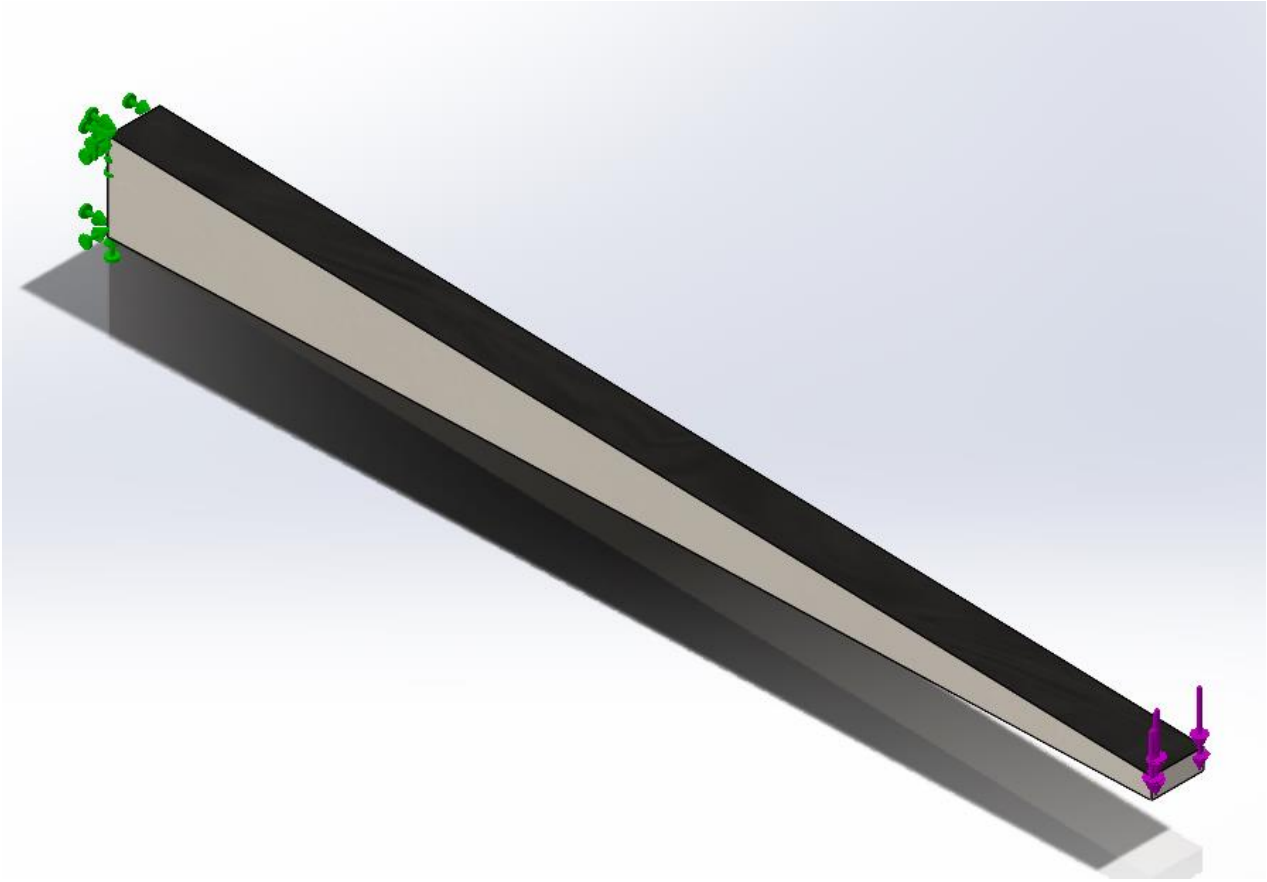


Figure 19: Optimum cantilever beam for a single point load on the free end.

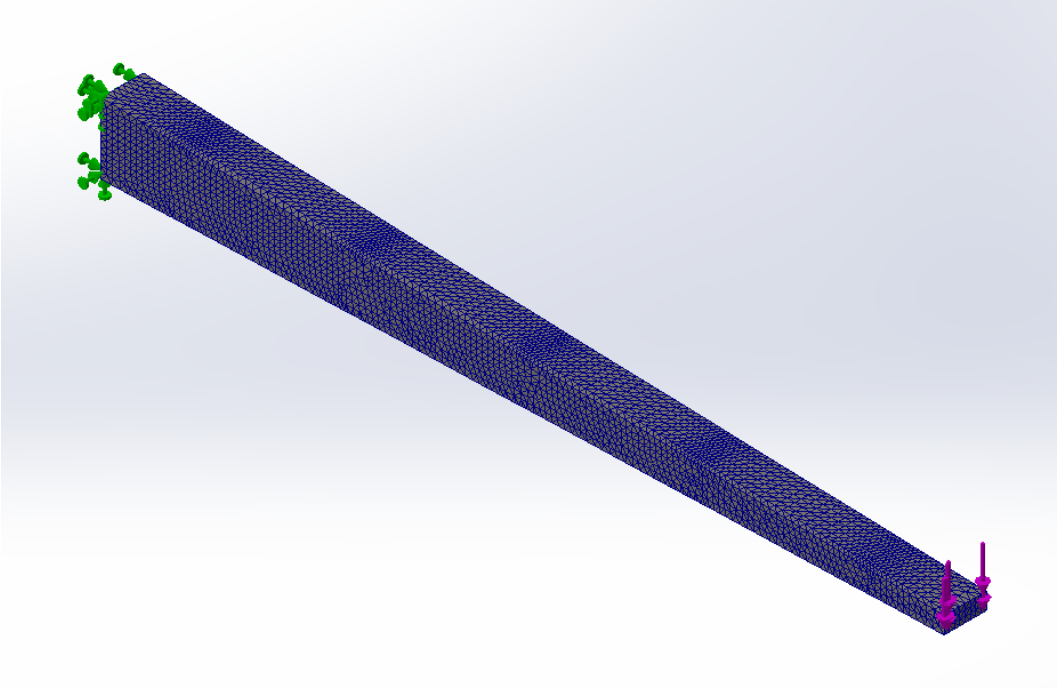


Figure 20: Optimum beam displaying the mesh elements used by SOLIDWORKS to analyze the beam.

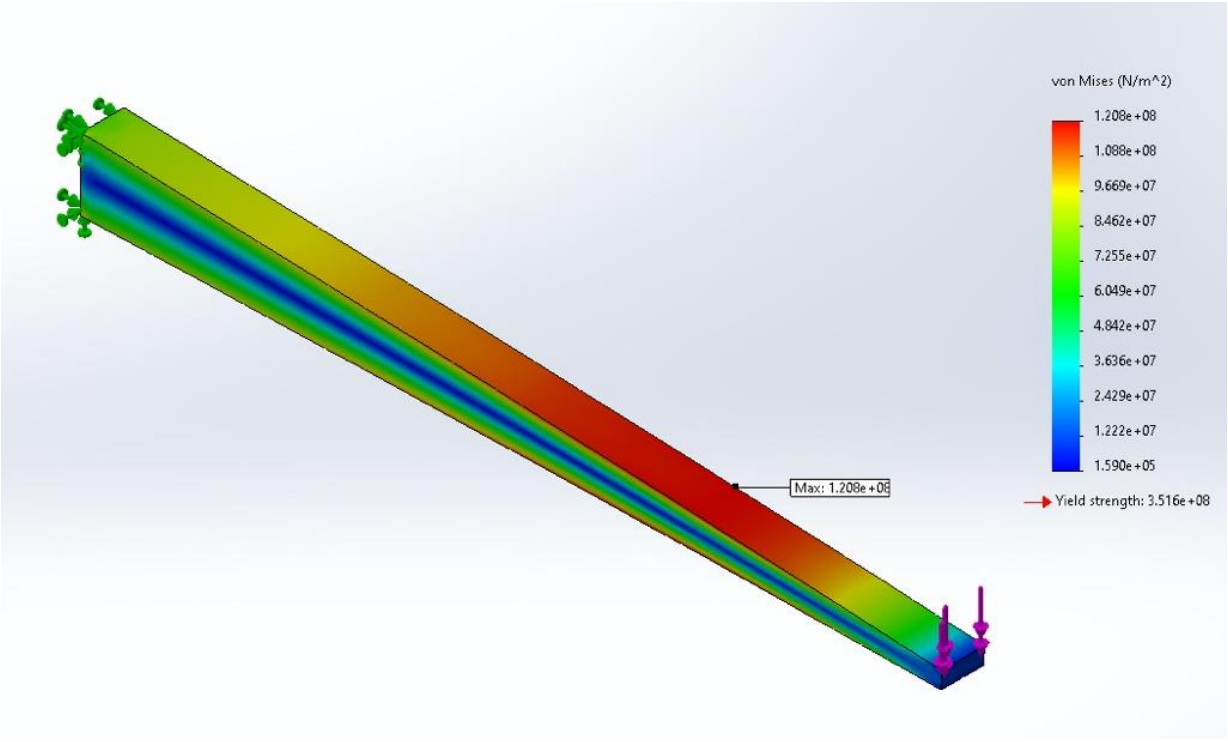


Figure 21: Von Mises stress intensity of the optimum beam with a single point load.

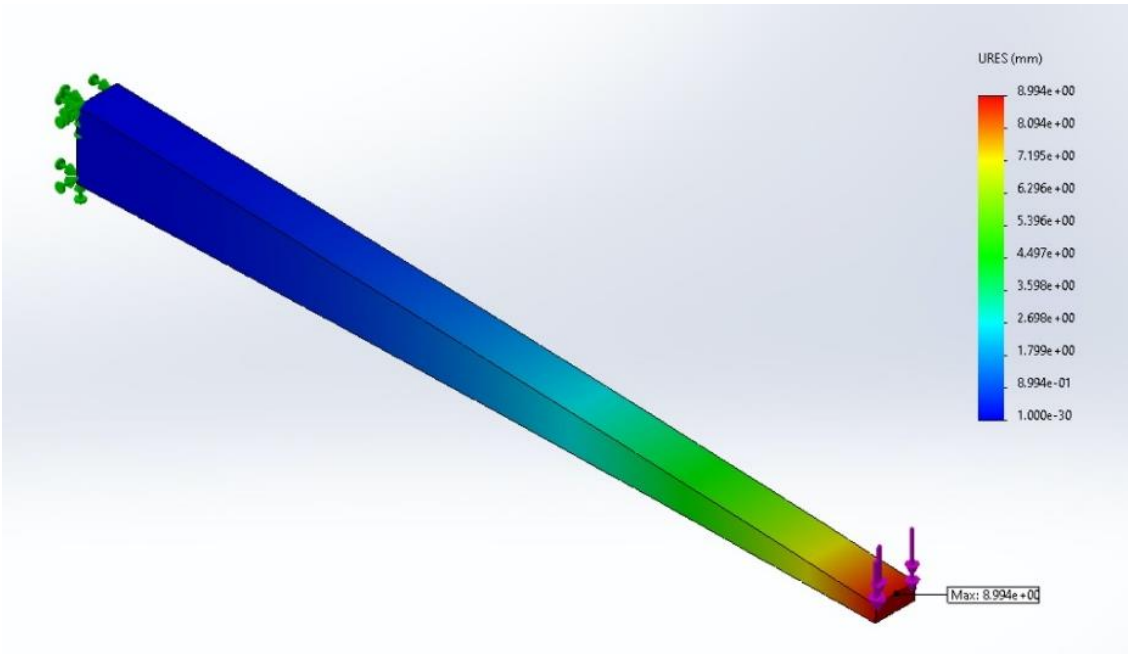


Figure 22: Deflection intensity of the optimum beam with a single point load.

The final sweep considered resulted in an optimum wall height of 0.084 m, an optimum free end height of 0.02 m, and a deflection of 8.994 millimeters. Compared to the MATLAB solution method, this is a percent error of 0.8%, 0.5%, and 0.3% for the wall height, free end height, and deflection, respectively.

This result demonstrated the optimization capabilities of SOLIDWORKS as well as confirming the accuracy of the MATLAB solution method. Now that the optimization capabilities of SOLIDWORKS had been verified, a complex loading situation could be considered and optimized in SOLIDWORKS alone.

Complex Loading SOLIDWORKS Simulation and Solution

In order to get the optimum beam profile for the more complex case, the cantilever beam was placed under numerous loading conditions. As shown in Figure 23, there are point loads acting in each direction on the free end of the beam. The axial and vertical loads, (blue and yellow in Figure 23) had magnitudes of 1000 pounds, while the horizontal load (black) had a magnitude of 2000 pounds. In addition to the point loads, a torsional load of 1000 inch-pounds acting on the end of the beam was also applied (red). The weight of the beam is also taken into consideration.

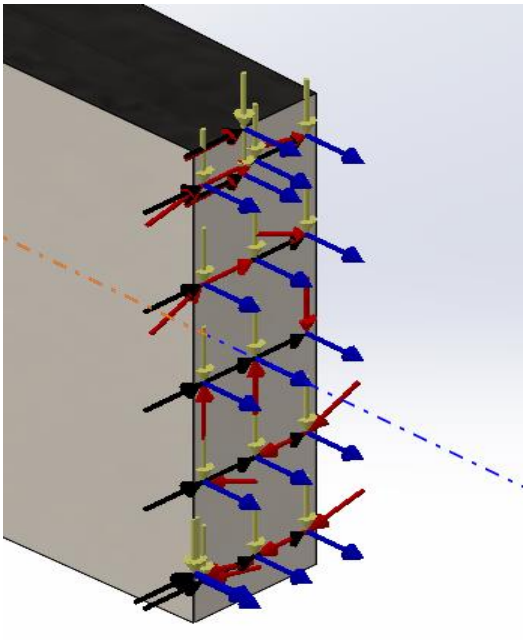


Figure 23: Representation of the loads applied on the cantilever beam.

Each of these different loading conditions creates a separate equivalent stiffness in each element. The axial, torsional, and bending loads all have unique stiffnesses. The axial load creates an equivalent stiffness which can be seen in **Equation 2**.

$$k_{eq\ (axial)} = \frac{EA}{L} \quad \textbf{(2)}$$

Where: E is Modulus of Elasticity
A is cross-sectional area of element
L is length of element

The torsional load's equivalent stiffness can be seen in **Equation 3**.

$$k_{eq\ (torsion)} = \frac{GJ}{L} \quad \textbf{(3)}$$

Where: G is Modulus of Rigidity
J is polar moment of inertia of element
L is length of element

Finally, the bending load's equivalent stiffness is given by **Equation 4**.

$$k_{eq\ (bending)} = \frac{3EI}{L^3} \quad \textbf{(4)}$$

Where: E is Modulus of Elasticity
I is moment of inertia of element
L is length of element

The beam being modeled has a total of six degrees of freedom. It is through a combination of the three equivalent stiffnesses and these six degrees of freedom that SOLIDWORKS is able to create a 12-by-12 stiffness matrix for each element. Once generated for each element in the case being analyzed, SOLIDWORKS can solve and generate an overall solution for the entire model using the FEM method.

With the loading conditions set, initial parameters and fixed values for the beam were set in SOLIDWORKS. This was done in order to ensure consistency throughout each trial. These fixed parameters include:

- Height of the beam at the wall to be 24 inches
- Width of the beam at the wall to be 6 inches
- Height of the beam at the end to be 4 inches
- Width of the beam at the end to be 1 inch
- Maximum beam weight of 1500 pounds
- Length of the beam to be 10 feet

From here, the optimization was ready to run. The goal of the design study was to minimize the deflection caused by the loadings by considering different combinations of two different dimensions on the beam. The section was constructed using two planes for the parameterized cross sections, distancing them 10 feet apart, equivalent to the length of the beam. The sections were then connected using a three-point arc on the top and bottom, and another for the two sides. The radii of these arcs were the two dimensions that were modified in the optimization design study. The parameters of the first study are shown in Figure 24, while the optimal results are displayed in Figure 25.

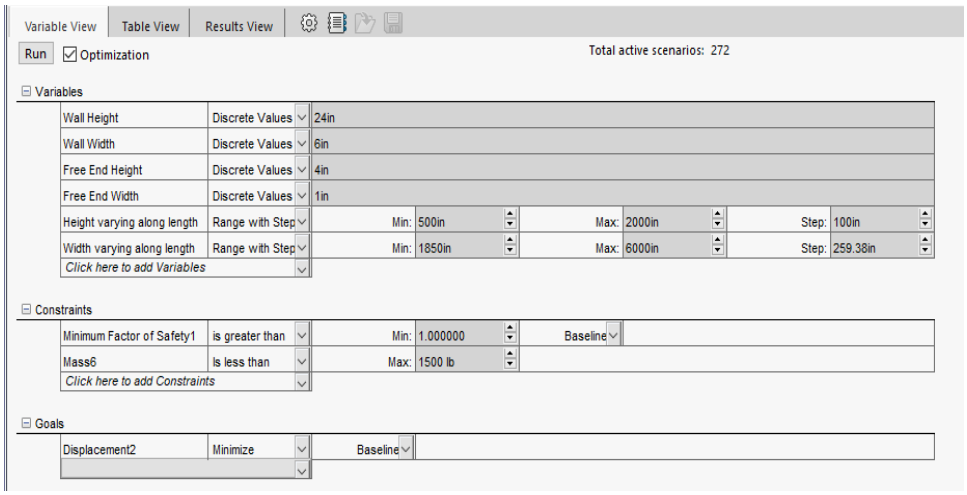


Figure 24: Sweep 1 – Parameters

274 of 274 scenarios ran successfully. Design Study Quality: High

		Current	Initial	Optimal (259)
Wall Height		24in	24in	24in
Wall Width		6in	6in	6in
Free End Height		4in	4in	4in
Free End Width		1in	1in	1in
Height varying along length		2000in	2000in	700in
Width varying along length		6000in	6000in	6000in
Minimum Factor of Safety1	> 1.000000	2.679438	2.679438	1.881103
Mass6	< 1500 lb	1712.56843 lb	1712.56843 lb	1469.50573 lb
Displacement2	Minimize	0.84262in	0.84262in	1.14169in

Figure 25: Sweep 1 – Results

It is observed in the optimal case above that the maximum radius that controls the width take the maximum bound, meaning that the width profile is attempting to become linear. Because of this, the maximum radius was increased as shown in the parameters for sweep 2 shown below in order to allow the simulation to continue to converge on where it is attempting to go.

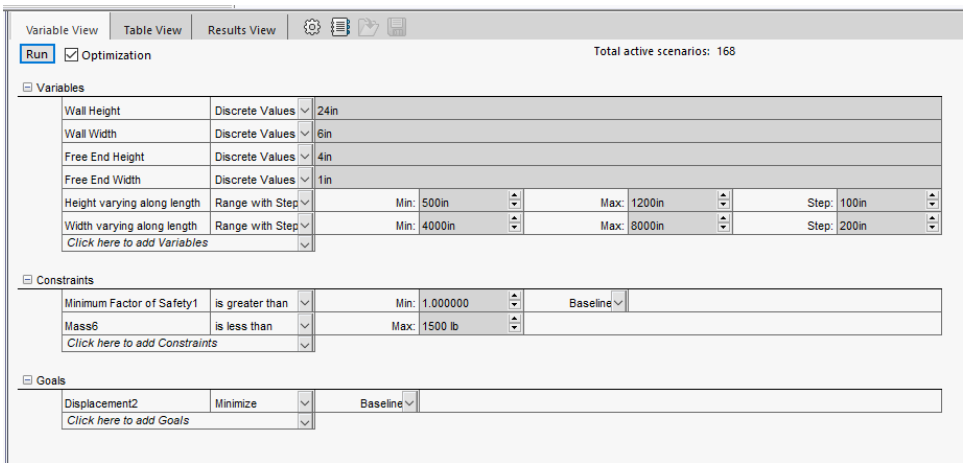


Figure 26: Sweep 2 – Parameters

		Current	Initial	Optimal (163)
Wall Height		24in	24in	24in
Wall Width		6in	6in	6in
Free End Height		4in	4in	4in
Free End Width		1in	1in	1in
Height varying along length		750in	750in	700in
Width varying along length		2000in	2000in	8000in
Minimum Factor of Safety1	> 1.000000	1.026946	1.026946	2.156970
Mass6	< 1500 lb	1313.12284 lb	1313.12284 lb	1493.90651 lb
Displacement2	Minimize	2.23584in	2.23584in	1.04096in

Figure 27: Sweep 2 – Results

Again, the optimal case shows a width profile taking the maximum bound, so the maximum was increased, as shown in Figures 28-29.

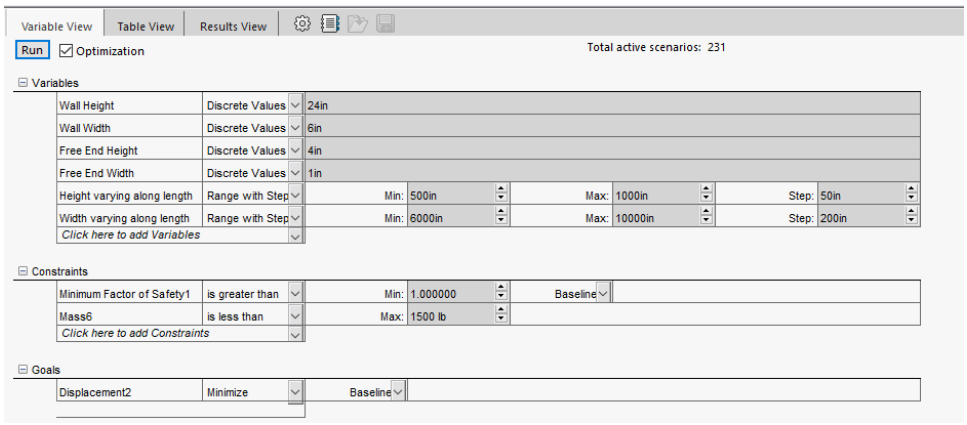
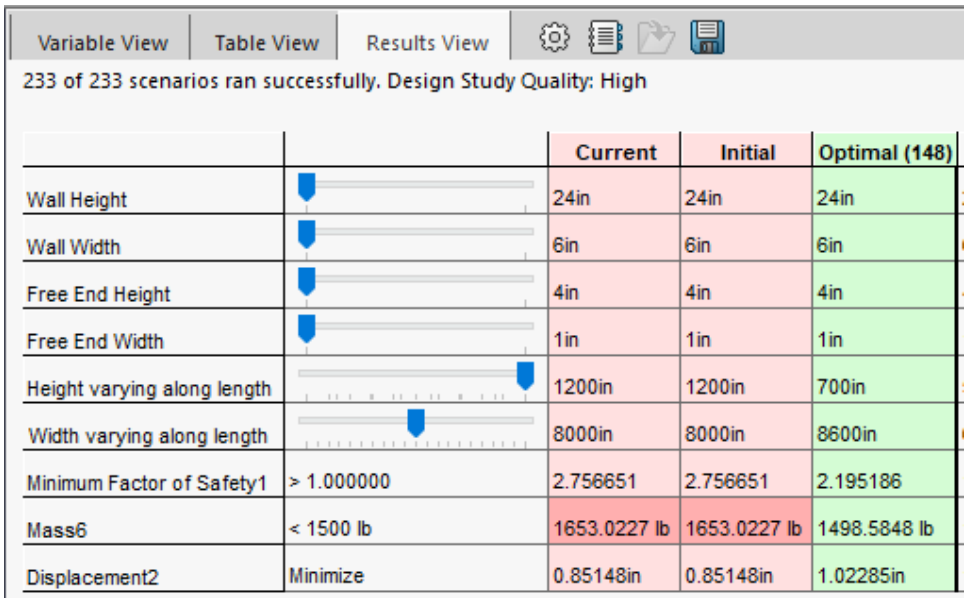


Figure 28: Sweep 3 – Parameters



		Current	Initial	Optimal (148)
Wall Height		24in	24in	24in
Wall Width		6in	6in	6in
Free End Height		4in	4in	4in
Free End Width		1in	1in	1in
Height varying along length		1200in	1200in	700in
Width varying along length		8000in	8000in	8600in
Minimum Factor of Safety1	> 1.000000	2.756651	2.756651	2.195186
Mass6	< 1500 lb	1653.0227 lb	1653.0227 lb	1498.5848 lb
Displacement2	Minimize	0.85148in	0.85148in	1.02285in

Figure 29*:* Sweep 3 – Results

In sweep 3, the optimal width was no longer the maximum bound, but to make sure this is the best case, the resolution of both dimensions was made finer, as shown in the sweep 4 parameters and results.

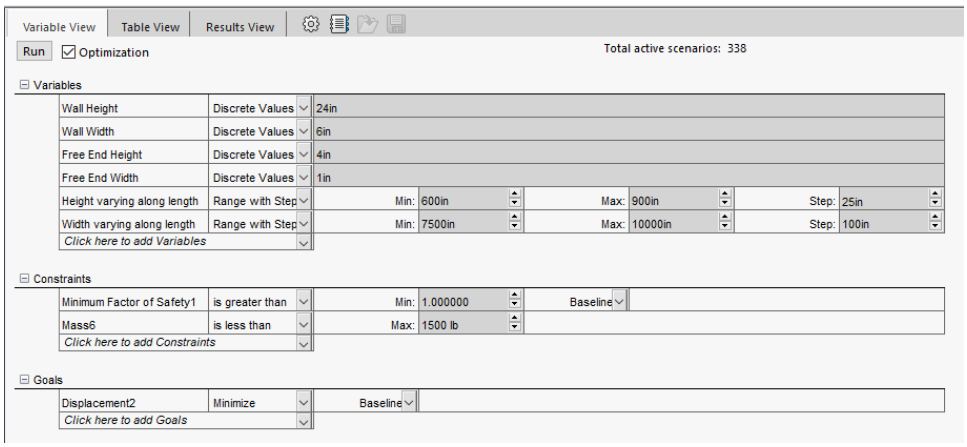


Figure 30: Sweep 4 – Parameters

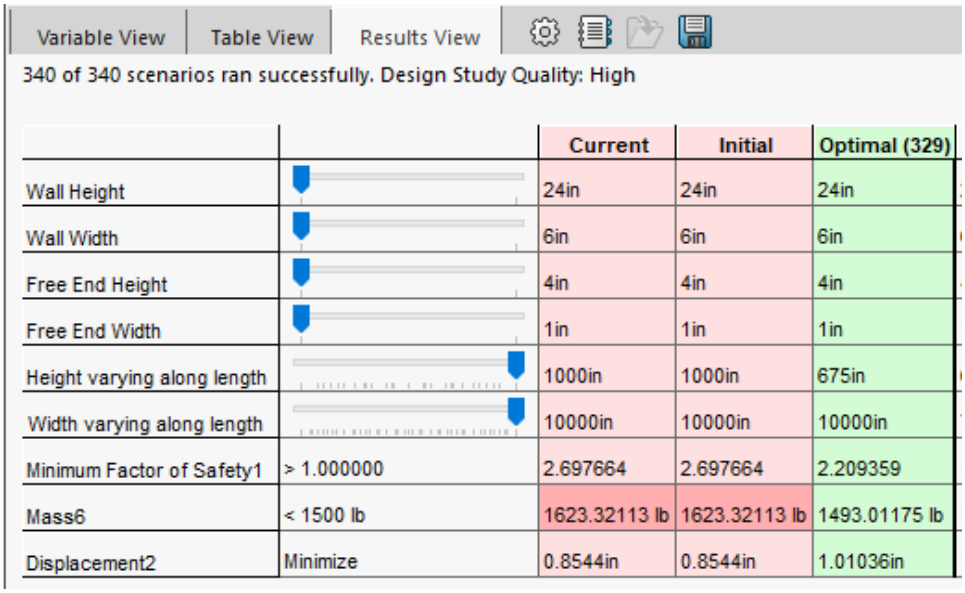


Figure 31: Sweep 4 – Results

Once again, the maximum bound for the width profile was taken by the design study, which means that the bound needed to be raised for sweep 5. Since it was apparent that the width was converging to a linear case, the radius of the arc was constrained to 20,000 inches. This way, the width profile modeled a linear case. The figures below display the parameters assigned to sweep 5 while the width was constrained to the linear case.

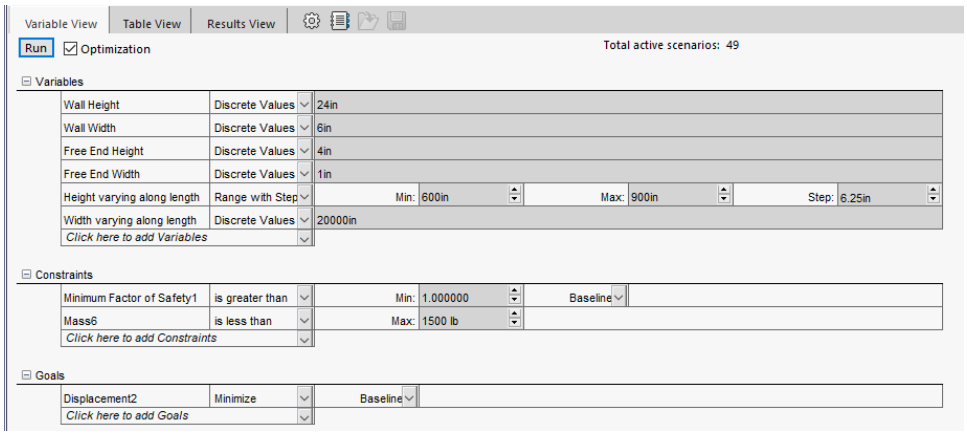
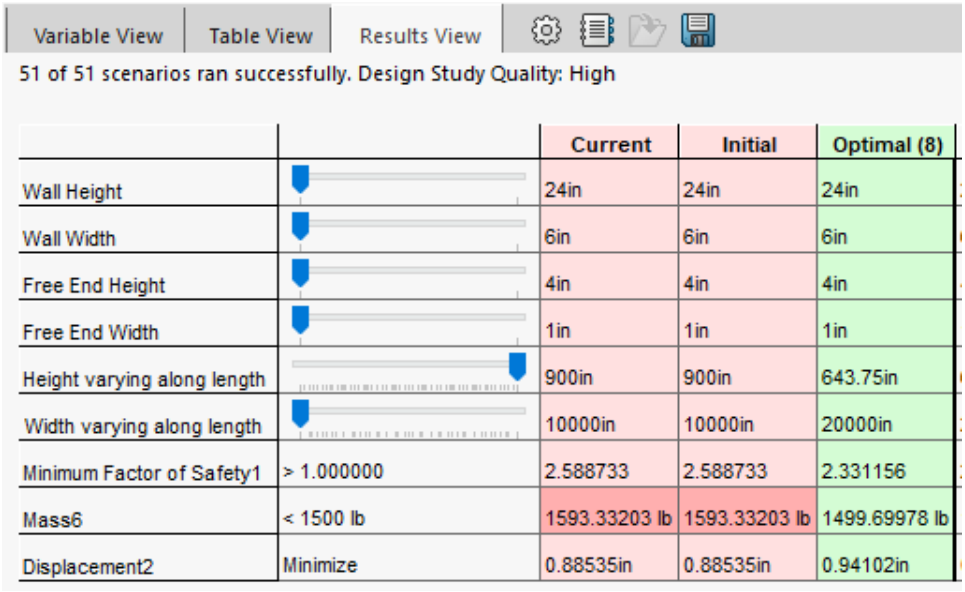


Figure 32: Sweep 5 – Parameters



		Current	Initial	Optimal (8)
Wall Height		24in	24in	24in
Wall Width		6in	6in	6in
Free End Height		4in	4in	4in
Free End Width		1in	1in	1in
Height varying along length		900in	900in	643.75in
Width varying along length		10000in	10000in	20000in
Minimum Factor of Safety1	> 1.000000	2.588733	2.588733	2.331156
Mass6	< 1500 lb	1593.33203 lb	1593.33203 lb	1499.69978 lb
Displacement2	Minimize	0.88535in	0.88535in	0.94102in

Figure 33: Sweep 5 – Results

The goal of sweep 5 was to narrow in on the optimal height value, so a lower resolution was applied to get a general idea of where the height would fall so that the range of the next test could be lowered, while the resolution could be increased. This technique was used to limit the time and data used on the simulations. While it was entirely possible to assign massive ranges and very fine resolutions to a single design study, it would take an extreme amount of time to fully run the simulation.

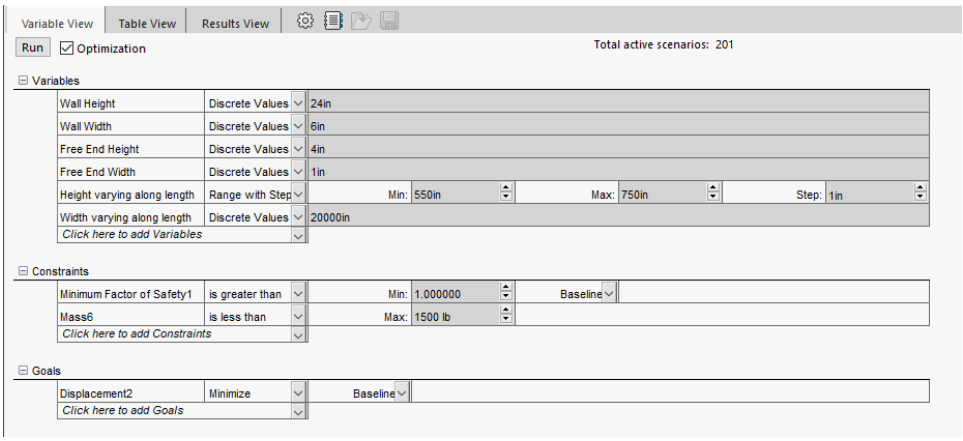


Figure 34: Sweep 6 – Parameters

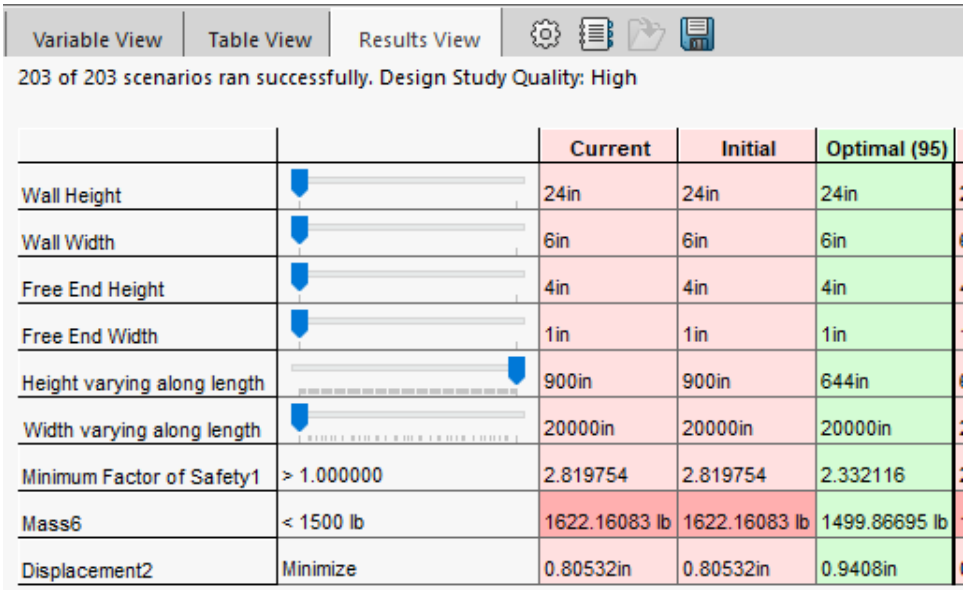


Figure 35: Sweep 6 - Results

Finally, sweep 6 was considered the optimal case. The height value fell between the allotted range, and the resolution was fine enough to consider this as the ideal case. By comparing the results of Figure 35 to those of Figure 25, it is clear that the factor of safety was increased from 1.88 to 2.33 and the deflection decreased from 1.14 in to 0.94 in. So not only did the optimized case minimize the deflection, but it also increased the factor of safety against failure. Next, the resolution of the mesh was increased, and a final static analysis was run. Figures 36-39 depict these results.

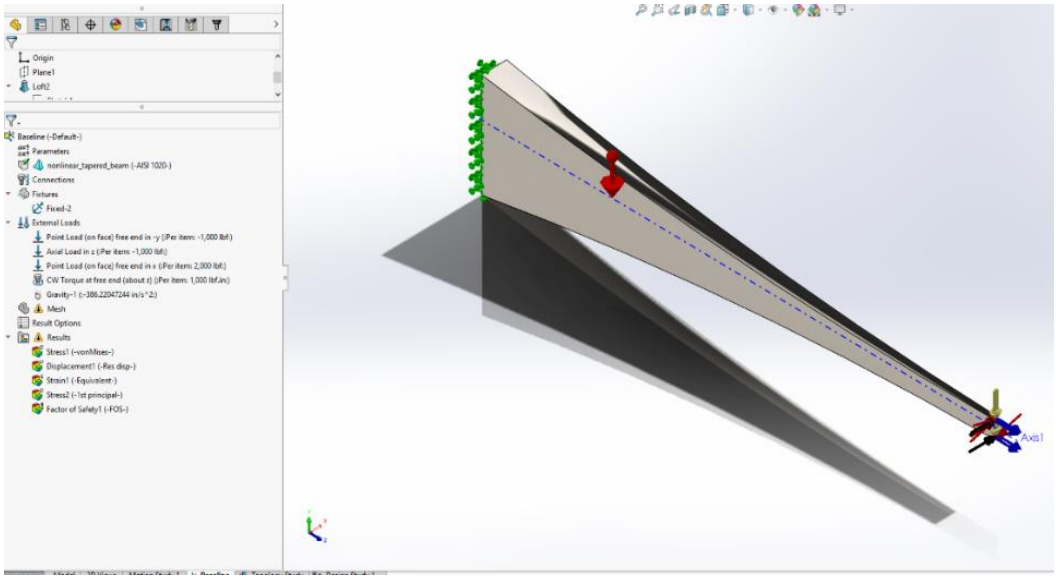


Figure 36: Optimized beam displaying the fixed and loading conditions

This beam would be very difficult to analyze by hand. However, with the finite element method, SolidWorks can calculate whatever is desired computationally. Figure 37 displays the mesh that the program uses to compute the deflection, stress intensities, von Mises stresses, and much more important values used to analyze the structural identity of the beam under the specific loading cases.

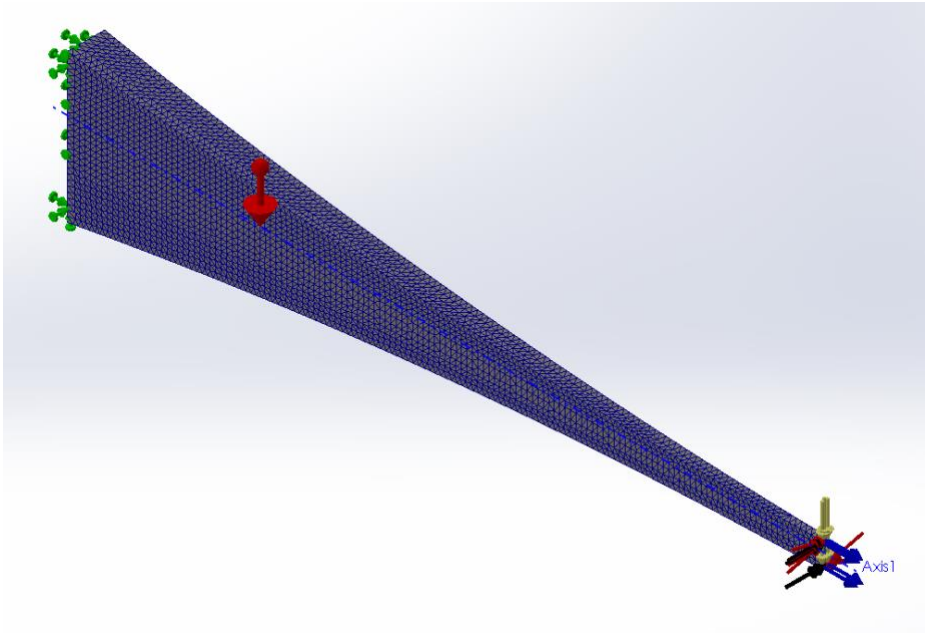


Figure 37: Optimized beam with the mesh and loading conditions

The program uses the finite element method, analyzing each element in the above figure in order to gather accurate results. Increasing the resolution is the equivalent to decreasing the size of each element, adding more elements to the beam for more precise results. Figure 38 below shows the deflection of the optimized beam after performing the simulation. The deflection is not to scale and is heavily exaggerated for visual aid. The maximum deflection of the optimal beam is 0.9804 inches.

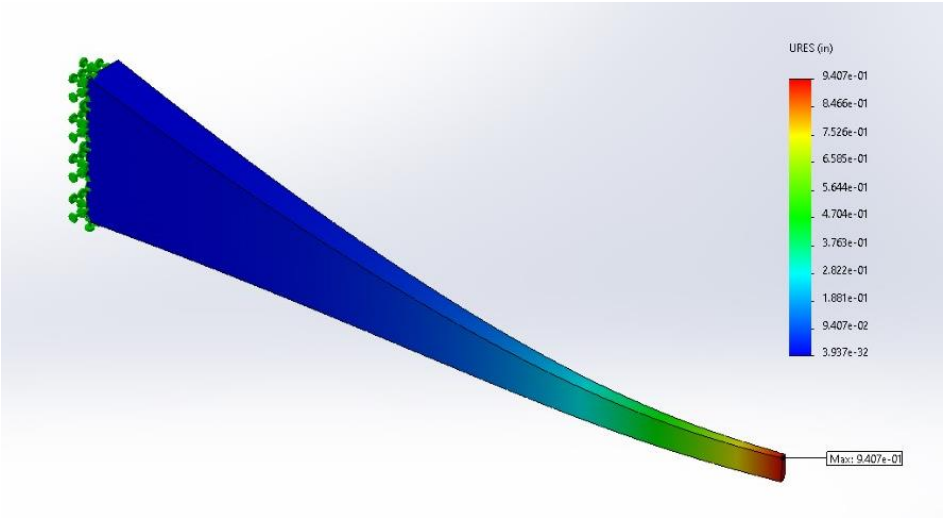


Figure 38: Optimized beam displaying the exaggerated displacement.

Finally, Figure 39 displays the von Mises stress intensity throughout the beam. These stresses were compared to the material properties when evaluating whether the beam supported the loads without yield failure, one of the constraints of the design study.

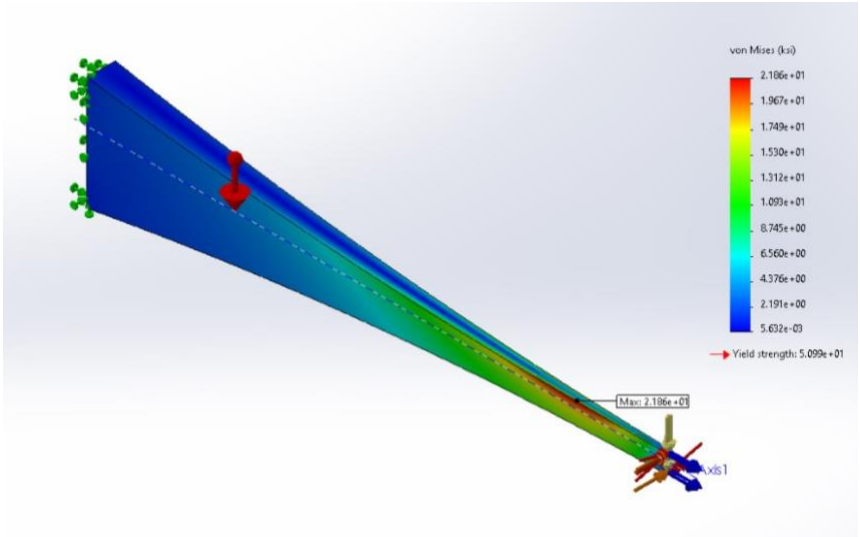


Figure 39: Optimized beam displaying the von Mises stresses.

Conclusions

This project first considered a simple loading condition on a cantilever beam. Optimization methods were implemented in MATLAB and SOLIDWORKS and the two were compared. The results of both optimization routines resulted in nearly identical results. This verified not only the accuracy of the MATLAB scripts written, consisting of both an optimizing script as well as an FEM implementation script, but also the optimization capabilities of SOLIDWORKS. With this, the loading situation on the cantilever beam was drastically increased in complexity and an optimum profile was determined in SOLIDWORKS.

Optimization is an important factor to consider when designing for a project. While it may be acceptable to "over engineer" a design in certain application, it is not always possible. Many fields require specific constraints that govern size, cost, weight, and factor of safety. Typically, as size increases, strength, weight, factor of safety and cost all increase. Certain fields of engineering such as aerospace rely on optimization of shapes and profiles to minimize weight and size while still maintaining strength, minimized deflection, and factor of safety. Aerospace is not the only field that utilizes optimization; it trickles down into other fields such as automotive, civil, architectural, structural, and many more. Optimization can be seen in everyday items such as a 3M® Command Strip. The hook used on the strip utilizes a tapered profile that retains its strength but minimized material and cost. Optimization plays a key role in engineering and design work when specific requirements and constraints are given in a problem.

Appendices

MATLAB Code-Optimizer

```
%Mason Averill
%ME-480 Fall 2020, 11/20
%Optimizer




%%_____________________________________________________________________
%input parameters

l=1; %length of the beam in meters
t=0.05; %width of the beam in meters
e=200*10^9; %Young's Modulus of the steel utilized in Pa
density=7900; %density of the steel utilized in kg/m^3
gravity=9.81; %acceleration due to gravity
Wmax=200; %maximum weight of beam in N
p=5000; %Point load applied at end of beam in N
number_of_elements=100; %specify number of elements to consider
step_size_const_solve=0.0001;

delta_x=l/number_of_elements;

x=0:delta_x:l;

%First lets find the deflection for a constant cross section beam

hmax=Wmax/(density*gravity*t*l); %maximum height permissible to still meet
max weight criteria

deflection_constant=(p*l^3)/(3*e*t*(1/12)*hmax^3); %deflection for the
constant cross-section beam




%_____________________________________________________________________
_____________________________
%Now lets find the ideal linear case
syms x1 m b

h_x_linear=m*x1+b; %general equation for linear h(x)

int_h_x_linear=vpa(int(h_x_linear,x1,[0 l])); %integrating h(x) symbolically
from 0 to l

m=solve(int_h_x_linear==Wmax/(density*gravity*t),m); %setting the integral
from 0 to l for h(x) equal to the max weight to eliminate one coefficient
```

```
h_x_linear=m*x1+b;%now have a function h(x) of only one unknown coefficient b

%find deflection in terms of b

deflection_linear=(12*p/(e*t))*vpaintegral((x1^2/h_x_linear^3),x1,[0 l]);
%this is the deflection of the beam at the free end in terms of b

height_max=0.023; %set an upper bound for the unknown coefficient
b_value=0.018; %initialize b value variable
deflection_b=[];
i=1;
b_store=[];
while(b_value<height_max)
    try
deflection_b(1,i)=double(subs(deflection_linear,b,b_value)); %adds the
deflection corresponding to a particular value of b to this array


b_store(1,i)=b_value; %stores the value of b for each of the values in the
above array


i=i+1;

    end
b_value=b_value+step_size_const_solve;

end


[min_value,column]=min(deflection_b); %returns the minimum value of
deflection at the free end and the location of b in b_store accompanying this
value

b_value=b_store(1,column);%selects the optimum b from all considered b values
figure (1)
plot(b_store,deflection_b);%shows a plot of b values vs the deflection at the
free end
xlabel('b values')
ylabel('deflection at free end')
title('Linear Case: deflection at free end vs b')
result=min_value/deflection_constant;

deflection_linear_store=min_value;


result=num2str(result);

string_out=strcat('The optimum linear case results in a deflection at the
free end of the beam of  ',  result , ' times the deflection for the constant
cross section case');
```

```
disp(string_out)

h_x_linear_sym=subs(h_x_linear,b,b_value);

h_x_linear=subs(h_x_linear,b,b_value);
h_x_linear=subs(h_x_linear,x1,x);
figure (2)
plot(x,h_x_linear)
xlabel('x')
ylabel('h(x)')
title('Linear Case: h(x) vs x')

%_____________________________________________________________________________
_____________________________________

%Now lets find the ideal polynomial order 2 case

syms x1 c1 c2

h_x_polynomial=c1*x1^2+c2; %general equation for polynomial order 2 h(x)

int_h_x_polynomial=vpa(int(h_x_polynomial,x1,[0 l])); %integrating h(x)
symbolically from 0 to l

c1=solve(int_h_x_polynomial==Wmax/(density*gravity*t),c1); %setting the
integral from 0 to l for h(x) equal to the max weight to eliminate one
coefficient


h_x_polynomial=c1*x1^2+c2;%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

deflection_polynomial=12*p/(e*t)*vpaintegral((x1^2/h_x_polynomial^3),x1,[0.00
1 l]); %this is the deflection of the beam at the free end in terms of b

height_max=.035; %set an upper bound for the height at the end of the beam
c2_value=.033; %initialize c2 value variable
deflection_c2=[];
i=1;
c2_store=[];
while(c2_value<height_max)
try
deflection_c2(1,i)=double(subs(deflection_polynomial,c2,c2_value)); %adds the
deflection corresponding to a particular value of c2 to this array


c2_store(1,i)=c2_value; %stores the value of c2 for each of the values in the
above array
```

```
i=i+1;
end

c2_value=c2_value+step_size_const_solve;

end


[min_value,column]=min(deflection_c2); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

c2_value=c2_store(1,column);%selects the optimum c2 from all considered c2
values
figure (3)
plot(c2_store,deflection_c2);%shows a plot of c2 values vs the deflection at
the free end
xlabel('c2 values')
ylabel('deflection at free end')
title('Polynomial order 2 Case: deflection at free end vs c2')
result=min_value/deflection_constant;

result=num2str(result);

string_out=strcat('The optimum polynomial order 2 case results in a
deflection at the free end of the beam of  ',  result , ' times the
deflection for the constant cross section case');

disp(string_out)

h_x_polynomial_sym=subs(h_x_polynomial,c2,c2_value);

h_x_polynomial=subs(h_x_polynomial,c2,c2_value);

h_x_polynomial=subs(h_x_polynomial,x1,x);
figure (4)
plot(x,h_x_polynomial)
xlabel('x')
ylabel('h(x)')
title('Polynomial order 2 case: h(x) vs x')

%_____________________________________________________________________________
_____________________________________

%Now lets find the ideal polynomial order 4 case

syms x1 c1 c2

h_x_polynomial_o4=c1*x1^4+c2; %general equation for polynomial order 4 h(x)

int_h_x_polynomial_o4=vpa(int(h_x_polynomial_o4,x1,[0 l])); %integrating h(x)
symbolically from 0 to l
```

```
c1=solve(int_h_x_polynomial_o4==Wmax/(density*gravity*t),c1); %setting the
integral from 0 to l for h(x) equal to the max weight to eliminate one
coefficient


h_x_polynomial_o4=c1*x1^4+c2;%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

deflection_polynomial_o4=12*p/(e*t)*vpaintegral((x1^2/h_x_polynomial_o4^3),x1
,[0 l]); %this is the deflection of the beam at the free end in terms of b

height_max=0.05; %set an upper bound for the height at the end of the beam
c2_value_o4=0.035; %initialize c2 value variable
deflection_c2_o4=[];
i=1;
c2_store_o4=[];
while(c2_value_o4<height_max)
try
deflection_c2_o4(1,i)=double(subs(deflection_polynomial_o4,c2,c2_value_o4));
%adds the deflection corresponding to a particular value of c2 to this array


c2_store_o4(1,i)=c2_value_o4; %stores the value of c2 for each of the values
in the above array


i=i+1;
end

c2_value_o4=c2_value_o4+step_size_const_solve;

end


[min_value,column]=min(deflection_c2_o4); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

c2_value_o4=c2_store_o4(1,column);%selects the optimum c2 from all considered
c2 values
figure (5)
plot(c2_store_o4,deflection_c2_o4);%shows a plot of c2 values vs the
deflection at the free end
xlabel('c2 values')
ylabel('deflection at free end')
title('Polynomial order 4 Case: deflection at free end vs c2')
result=min_value/deflection_constant;

result=num2str(result);
```

```
string_out=strcat('The optimum polynomial order 4 case results in a
deflection at the free end of the beam of  ',  result , ' times the
deflection for the constant cross section case');

disp(string_out)

h_x_polynomial_o4_sym=subs(h_x_polynomial_o4,c2,c2_value_o4);

h_x_polynomial_o4=subs(h_x_polynomial_o4,c2,c2_value_o4);

h_x_polynomial_o4=subs(h_x_polynomial_o4,x1,x);
figure (6)
plot(x,h_x_polynomial_o4)
xlabel('x')
ylabel('h(x)')
title('Polynomial order 4 Case: h(x) vs x')


%_____________________________________________________________________________
_____________________________________

%Now lets find the ideal polynomial order 6 case

syms x1 c1 c2

h_x_polynomial_o6=c1*x1^6+c2; %general equation for polynomial order 6 h(x)

int_h_x_polynomial_o6=vpa(int(h_x_polynomial_o6,x1,[0 l])); %integrating h(x)
symbolically from 0 to l

c1=solve(int_h_x_polynomial_o6==Wmax/(density*gravity*t),c1); %setting the
integral from 0 to l for h(x) equal to the max weight to eliminate one
coefficient


h_x_polynomial_o6=c1*x1^6+c2;%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

deflection_polynomial_o6=12*p/(e*t)*vpaintegral((x1^2/h_x_polynomial_o6^3),x1
,[0 l]); %this is the deflection of the beam at the free end in terms of b

height_max=0.0475; %set an upper bound for the height at the end of the beam
c2_value_o6=0.0445; %initialize c2 value variable
deflection_c2_o6=[];
i=1;
c2_store_o6=[];
while(c2_value_o6<height_max)
try
deflection_c2_o6(1,i)=double(subs(deflection_polynomial_o6,c2,c2_value_o6));
%adds the deflection corresponding to a particular value of c2 to this array
```

```
c2_store_o6(1,i)=c2_value_o6; %stores the value of c2 for each of the values
in the above array


i=i+1;

end
c2_value_o6=c2_value_o6+step_size_const_solve;

end


[min_value,column]=min(deflection_c2_o6); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

c2_value_o6=c2_store_o6(1,column);%selects the optimum c2 from all considered
c2 values
figure (7)
plot(c2_store_o6,deflection_c2_o6);%shows a plot of c2 values vs the
deflection at the free end
xlabel('c2 values')
ylabel('deflection at free end')
title('Polynomial order 6 Case: deflection at free end vs c2')
result=min_value/deflection_constant;

result=num2str(result);

string_out=strcat('The optimum polynomial order 6 case results in a
deflection at the free end of the beam of  ',  result , ' times the
deflection for the constant cross section case');

disp(string_out)

h_x_polynomial_o6_sym=subs(h_x_polynomial_o6,c2,c2_value_o6);

h_x_polynomial_o6=subs(h_x_polynomial_o6,c2,c2_value_o6);

h_x_polynomial_o6=subs(h_x_polynomial_o6,x1,x);
figure (8)
plot(x,h_x_polynomial_o6)
xlabel('x')
ylabel('h(x)')
title('Polynomial order 6 Case: h(x) vs x')

%_____________________________________________________________________________
_____________________________________


%Now lets find the ideal polynomial order 12 case
```

```
syms x1 c1 c2

h_x_polynomial_o12=c1*x1^12+c2; %general equation for polynomial order 12
h(x)

int_h_x_polynomial_o12=vpa(int(h_x_polynomial_o12,x1,[0 l])); %integrating
h(x) symbolically from 0 to l

c1=solve(int_h_x_polynomial_o12==Wmax/(density*gravity*t),c1); %setting the
integral from 0 to l for h(x) equal to the max weight to eliminate one
coefficient


h_x_polynomial_o12=c1*x1^12+c2;%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

deflection_polynomial_o12=12*p/(e*t)*vpaintegral((x1^2/h_x_polynomial_o12^3),
x1,[0 l]); %this is the deflection of the beam at the free end in terms of b

height_max=0.051; %set an upper bound for the height at the end of the beam
c2_value_o12=0.046; %initialize c2 value variable
deflection_c2_o12=[];
i=1;
c2_store_o12=[];
while(c2_value_o12<height_max)
try
deflection_c2_o12(1,i)=double(subs(deflection_polynomial_o12,c2,c2_value_o12)
); %adds the deflection corresponding to a particular value of c2 to this
array


c2_store_o12(1,i)=c2_value_o12; %stores the value of c2 for each of the
values in the above array


i=i+1;
end

c2_value_o12=c2_value_o12+step_size_const_solve;

end


[min_value,column]=min(deflection_c2_o12); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

c2_value_o12=c2_store_o12(1,column);%selects the optimum c2 from all
considered c2 values
figure (9)
```

```
plot(c2_store_o12,deflection_c2_o12);%shows a plot of c2 values vs the
deflection at the free end
xlabel('c2 values')
ylabel('deflection at free end')
title('Polynomial order 12 Case: deflection at free end vs c2')
result=min_value/deflection_constant;

result=num2str(result);

string_out=strcat('The optimum polynomial order 12 case results in a
deflection at the free end of the beam of  ',  result , ' times the
deflection for the constant cross section case');

disp(string_out)

h_x_polynomial_o12_sym=subs(h_x_polynomial_o12,c2,c2_value_o12);

h_x_polynomial_o12=subs(h_x_polynomial_o12,c2,c2_value_o12);

h_x_polynomial_o12=subs(h_x_polynomial_o12,x1,x);
figure (10)
plot(x,h_x_polynomial_o12)
xlabel('x')
ylabel('h(x)')
title('Polynomial order 12 Case: h(x) vs x')

%_____________________________________________________________________________
_____________________________________


%Now lets find the ideal exponential case

syms x1 c1 c2

h_x_exp=c1*exp(c2*x1); %general equation for exp h(x)

int_h_x_exp=vpa(int(h_x_exp,x1,[0 l])); %integrating h(x) symbolically from 0
to l

c1=solve(int_h_x_exp==Wmax/(density*gravity*t),c1); %setting the integral
from 0 to l for h(x) equal to the max weight to eliminate one coefficient


h_x_exp=c1*exp(c2*x1);%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

deflection_exp=12*p/(e*t)*vpaintegral((x1^2/h_x_exp^3),x1,[0 l]); %this is
the deflection of the beam at the free end in terms of b

height_max=1.14; %set an upper bound for the height at the end of the beam
c2_value_exp=1.115; %initialize c2 value variable
```

```
deflection_c2_exp=[];
i=1;
c2_store_exp=[];
while(c2_value_exp<height_max)
try
deflection_c2_exp(1,i)=double(subs(deflection_exp,c2,c2_value_exp)); %adds
the deflection corresponding to a particular value of c2 to this array


c2_store_exp(1,i)=c2_value_exp; %stores the value of c2 for each of the
values in the above array


i=i+1;
end

c2_value_exp=c2_value_exp+step_size_const_solve;

end

[min_value,column]=min(deflection_c2_exp); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

c2_value_exp=c2_store_exp(1,column);%selects the optimum c2 from all
considered c2 values
figure (11)
plot(c2_store_exp,deflection_c2_exp);%shows a plot of c2 values vs the
deflection at the free end
xlabel('c2 values')
ylabel('deflection at free end')
title('Exponential Case: deflection at free end vs c2')
result=min_value/deflection_constant;

result=num2str(result);

string_out=strcat('The optimum exponential case results in a deflection at
the free end of the beam of  ',  result , ' times the deflection for the
constant cross section case');

disp(string_out)

h_x_exp_sym=subs(h_x_exp,c2,c2_value_exp);

h_x_exp=subs(h_x_exp,c2,c2_value_exp);

h_x_exp=subs(h_x_exp,x1,x);
figure (12)
plot(x,h_x_exp)
xlabel('x')
ylabel('h(x)')
title('Exponential Case: h(x) vs x')
```

```
%_____
_____

%Now lets find the ideal cos case

%syms x1 c1 c2

%h_x_cos=c1*cos(c2*x1); %general equation for cos h(x)

%int_h_x_cos=vpa(int(h_x_cos,x1,[0 l])); %integrating h(x) symbolically from
0 to l

%c1=solve(int_h_x_cos==Wmax/(density*gravity*t),c1); %setting the integral
from 0 to l for h(x) equal to the max weight to eliminate one coefficient


%h_x_cos=c1*cos(c2*x1);%now have a function h(x) of only one unknown
coefficient c2

%find deflection in terms of c2

%deflection_cos=12*p/(e*t)*vpaintegral((x1^2/h_x_cos^3),x1,[0 l]); %this is
the deflection of the beam at the free end in terms of b

%height_max=0.04; %set an upper bound for the height at the end of the beam
%c2_value_cos=-0.1; %initialize c2 value variable
%deflection_c2_cos=[];
%i=1;
%c2_store_cos=[];
%while(c2_value_cos<height_max)
%try
%deflection_c2_cos(1,i)=double(subs(deflection_cos,c2,c2_value_cos)); %adds
the deflection corresponding to a particular value of c2 to this array


%c2_store_cos(1,i)=c2_value_cos; %stores the value of c2 for each of the
values in the above array


%i=i+1;
%end

%c2_value_cos=c2_value_cos+step_size_const_solve;

%end

%[min_value,column]=min(deflection_c2_cos); %returns the minimum value of
deflection at the free end and the location of c2 in c2_store accompanying
this value

%c2_value_cos=c2_store_cos(1,column);%selects the optimum c2 from all
considered c2 values
```

```
%figure (13)
%plot(c2_store_cos,deflection_c2_cos);%shows a plot of c2 values vs the
deflection at the free end
%xlabel('c2 values')
%ylabel('deflection at free end')
%title('Cosine Case: deflection at free end vs c2')
%result=min_value/deflection_constant;

%result=num2str(result);

%string_out=strcat('The optimum cosine case results in a deflection at the
free end of the beam of  ',  result , ' times the deflection for the constant
cross section case');

%disp(string_out)

%h_x_cos_sym=subs(h_x_cos,c2,c2_value_cos);

%h_x_cos=subs(h_x_cos,c2,c2_value_cos);

%h_x_cos=subs(h_x_cos,x1,x);
%figure (14)
%plot(x,h_x_cos)
%xlabel('x')
%ylabel('h(x)')
%title('Cosine Case: h(x) vs x')
```

MATLAB Code-FEM Solution Method

```
%Mason Averill
%ME-480 Fall 2020, 11/20
%FEM Method Solution




%Input Parameters

l=1; %length of the beam in meters
t=0.05; %width of the beam in meters
e=200*10^9; %Young's Modulus of the steel utilized in Pa
density=7900; %density of the steel utilized in kg/m^3
gravity=9.81; %acceleration due to gravity
p=5000; %Point load applied at end of beam in N
number_of_elements_max=100; %specify number of elements to consider


number_of_elements=1;
FEM_end_deflection_store=zeros(1,number_of_elements_max);
FEM_end_slope_store=zeros(1,number_of_elements_max);
percent_error_deflection_store=zeros(1,number_of_elements_max);
while(number_of_elements<=number_of_elements_max)
```

```
%Discretize x
delta_x=l/number_of_elements;
x=0:delta_x:l;

%Optimum h(x) determined by optimizer: 0.063427138414689221796410276261629*x1
+ 199/10000
%Now find h(x) from the wall towards the end of the beam

h_x=-
0.063427138414689221796410276261629*x+199/10000+0.063427138414689221796410276
261629;

%First lets find the area at each node

[rows,columns]=size(h_x);


A_x=zeros(rows,columns-1); %this will hold the area for each element
A_i=0;
A_j=0;
i=1;

while(i<columns)
 A_i=t*h_x(1,i);
 A_j=t*h_x(1,i+1);

 A_x(1,i)=(A_i+A_j)/2;

 i=i+1;

end

%Now lets find I for each element

I_x=zeros(rows,columns-1);

i=1;

while(i<columns)
 I_x(1,i)=(t*((h_x(1,i)+h_x(1,i+1))/2)^3)/12;

 i=i+1;
end


%Now lets assemble the keq for each element;


k_cell={};%create a cell array of all the k local matrices
k_local=zeros(6+3*(number_of_elements-1),6+3*(number_of_elements-
1));%initialize the size of k local, in this case
%we are setting the size of k local equal to that of k global to make
%assembly of k global much easier at the end
```

```
i=1;%used to index k local to be generated
A=0;
I=0;
start_row=0;
start_column=0;
l=delta_x;
while(i<=number_of_elements)
    start_row=(3*i)-2;
    start_column=start_row;

    A=A_x(1,i);%finds the area corresponding to the klocal in question
    I=I_x(1,i);%finds the second area moment of inertia for the klocal in
question

    %now lets populate the k_local

    %populate first row

    k_local(start_row,start_column)=A*e/l;
    k_local(start_row,start_column+3)=-A*e/l;

    %populate second row
    k_local(start_row+1,start_column+1)=12*e*I/l^3;
    k_local(start_row+1,start_column+2)=6*e*I/l^2;
    k_local(start_row+1,start_column+4)=-12*e*I/l^3;
    k_local(start_row+1,start_column+5)=6*e*I/l^2;


    %populate third row
    k_local(start_row+2,start_column+1)=6*e*I/l^2;
    k_local(start_row+2,start_column+2)=4*e*I/l;
    k_local(start_row+2,start_column+4)=-6*e*I/l^2;
    k_local(start_row+2,start_column+5)=2*e*I/l;


    %populate fourth row
    k_local(start_row+3,start_column)=-A*e/l;
    k_local(start_row+3,start_column+3)=A*e/l;

    %populate fifth row
    k_local(start_row+4,start_column+1)=-12*e*I/l^3;
    k_local(start_row+4,start_column+2)=-6*e*I/l^2;
    k_local(start_row+4,start_column+4)=12*e*I/l^3;
    k_local(start_row+4,start_column+5)=-6*e*I/l^2;


    %populate sixth row
    k_local(start_row+5,start_column+1)=6*e*I/l^2;
    k_local(start_row+5,start_column+2)=2*e*I/l;
    k_local(start_row+5,start_column+4)=-6*e*I/l^2;
    k_local(start_row+5,start_column+5)=4*e*I/l;
```

```
    k_cell{1,i}=k_local;

    k_local=zeros(6+3*(number_of_elements-1),6+3*(number_of_elements-1));
  i=i+1;
end


%now lets assemble k Global

k_global=zeros(6+3*(number_of_elements-1),6+3*(number_of_elements-1));

i=1;

while(i<=number_of_elements)
   k_global=k_global+k_cell{1,i};

   i=i+1;
end

%k_global is now assembled

k_global_solve=k_global;
%now lets apply bc's and simplify
k_global_solve(1,:)=0;
k_global_solve(2,:)=0;
k_global_solve(3,:)=0;
k_global_solve(:,1)=0;
k_global_solve(:,2)=0;
k_global_solve(:,3)=0;
k_global_solve(1,1)=1;
k_global_solve(2,2)=1;
k_global_solve(3,3)=1;

%now lets assemble the load matrix with b.c's applied

F=zeros((6+3*(number_of_elements-1)),1);

[rows, columns]=size(F);

F(rows-1,1)=-p;

u=k_global_solve^-1*F;

R=k_global*u-F;
l=1;

%Now lets find the analytical deflection
x1=0:delta_x:l;

h_x1=0.063427138414689221796410276261629*x1 + 199/10000;

analytical_deflection=-12*p/(e*t)*trapz(x1,x1.^2./h_x1.^3);
```

```
%Percent error between analytical and FEM method

percent_error=abs(analytical_deflection-u(rows-
1,1))/abs(analytical_deflection)*100;

FEM_end_deflection_store(1,number_of_elements)=u(rows-1,1);
FEM_end_slope_store(1,number_of_elements)=u(rows,1);
percent_error_deflection_store(1,number_of_elements)=percent_error;
number_of_elements=number_of_elements+1;
end

%grab the deflection for each delta x

[rows,columns]=size(u);
[rowsx,columnsx]=size(x);
i=2;
j=1;
u_x=zeros(1,columnsx);
while(i<=rows)
 u_x(1,j)=u(i,1);

 j=j+1;
 i=i+3;
end

%grab the slope for each delta x


i=3;
j=1;
theta_x=zeros(1,columnsx);
while(i<=rows)
 theta_x(1,j)=u(i,1);

 j=j+1;
 i=i+3;
end



%show ideal h(x) function
figure (1)
plot(x,h_x);
xlabel('x (m)')
ylabel('h(x) (m)')
title('Optimum Height of the Beam from the Wall Towards the Free End')
grid on

%show that the beam weight is equal to the maximum allowable
beam_weight=num2str(trapz(x,h_x)*density*gravity*t);

string_for_print=strcat('The weight of the beam is ',{' '},beam_weight,'
Newtons');
```

```
disp(string_for_print);

elements_for_plot=1:1:number_of_elements_max;


figure (2)

plot(elements_for_plot,FEM_end_deflection_store)
title('Deflection at the End of the Beam Determined by FEM method vs Number
of Elements Considered')
xlabel('Number of elements')
ylabel('Deflection at the End of the Beam')
grid on

FEM_end_slope_store=FEM_end_slope_store*180/pi();

figure (3)
plot(elements_for_plot,FEM_end_slope_store)
xlim([2,number_of_elements_max])
title('Slope at the Free End of the Beam Determined by FEM method vs Number
of Elements Considered')
xlabel('Number of Elements')
ylabel('Slope at the Free End of the Beam (degrees)')
grid on

figure (4)
plot(elements_for_plot,percent_error_deflection_store)
title('Percent Error of FEM Method Compared to Analytical Solution vs Number
of Elements Considered')
xlabel('Number of Elements')
ylabel('Percent error (%)')
xlim([10,100])
grid on

figure (5)
plot(x,u_x)
title('Deflection of Beam From the Wall Towards the Free End')
xlabel('x (m)')
ylabel('Deflection of Beam (m)')
grid on

figure (6)
plot(x,theta_x)
title('Slope of Beam vs Position in x')
xlabel('x (m)')
ylabel('Slope of Beam (rad)')
grid on

theta_x=theta_x*180/pi();

figure (7)
plot(x,theta_x)
title('Slope of Beam From the Wall Towards the Free End')
```

```
xlabel('x (m)')
ylabel('Slope of Beam (degrees)')
grid on
```