To: D. W. Mueller, Jr., Ph.D., P.E.

From: Mason Averill

Subject: Special Problem 5

Date: 22 October 2020

PURDUE UNIVERSITY FORT WAYNE

Purpose:

The purpose of this memo is to communicate the methodology and results of Special Problem 5 for ME-544: Modeling and Simulation of Mechanical Engineering Systems, completed October 22nd 2020.

Purpose and Scope of Assignment:

The purpose of this assignment was to determine the optimum profile for a constant-thickness cantilever beam such that the deflection at the end of the beam was minimized when subjected to a transverse point load at its end. The maximum allowable weight of the beam and the thickness were given. In addition, it was specified that the deflection and slope of the beam should be solved for by utilizing the FEM method.

Mathematical Modeling of Problem:

In order to implement the FEM method it was necessary to first determine the optimum profile of the beam. In order to do so, a MATLAB script was written with the sole purpose of optimizing the profile of the beam. The 'Optimizer' MATLAB script works based on the following mathematical principles/processes:

1. All parameters for the system are established, including:
    a. The length of the beam=1m
    b. The thickness of the beam=0.05m
    c. Young's Modulus of the material comprising the beam=200GPa
    d. The density of the material comprising the beam=7800 kg/m^3
    e. Acceleration due to gravity=9.81 $^{m}/_{s^2}$

f. Magnitude of the transverse point load at the end of the beam=5000N

2. A function consisting of two unknown coefficients is accepted as an input
3. One of the coefficients is solved in terms of the other by utilizing Equation 1
4. Bounds are set regarding where to 'look' for the optimum coefficient value
5. Values within the bounds are selected for the unknown coefficient and the deflection at the end of the beam is determined for each selected value by utilizing Equation 2
6. The coefficient value resulting in the minimum value of deflection at the end of the beam is selected as being the optimum
7. The function h(x) is now determined and optimized

$$W_{Max} = \rho * g * t * \int_0^L h(x)dx \qquad \text{Equation 1}$$

With:

- $W_{Max} = Maximum\ allowable\ weight\ of\ the\ beam\ (N)$
- $\rho\ = density\ of\ the\ material\ comprising\ the\ beam\ (kg/m^3)$
- $g = acceleration\ due\ to\ gravity\ (^m/_{s^2})$
- $t = thickness\ of\ the\ beam\ (m)$
- $h(x) = function\ of\ x\ describing\ the\ height\ of\ the\ beam\ (m)$
- $L = length\ of\ the\ beam\ (m)$

$$(\delta_{end})_v = \frac{12 * P}{E * t} * \int_0^L \frac{x^2}{h'(x)^3} dx \qquad \text{Equation 2}$$

With:

- $(\delta_{end})_v = Deflection\ of\ the\ beam\ at\ the\ free\ end\ in\ the\ vertical\ direction\ (m)$
- $P\ = transverse\ point\ load\ applied\ to\ the\ beam\ at\ the\ free\ end\ (N)$
- $E = Young's\ modulus\ of\ the\ material\ comprising\ the\ beam\ (^N/_{m^2})$
- $t = thickness\ of\ the\ beam\ (m)$
- $h'(x) =$ function of x describing the height of the beam from the free end towards the wall (m)
- $L = length\ of\ the\ beam\ (m)$

**This equation was developed by utilizing Castigliano's Theorem

Values for Young's Modulus and density were determined by selecting a material, in this case A-36 Steel was selected.

Many different function types were tested to attempt to find the optimum profile of the beam, including, but not limited to:

1. Linear Case ($C_1 * x + C_2$)
2. Polynomial Order 2 Case ($C_1 * x^2 + C_2$)
3. Polynomial Order 4 Case ($C_1 * x^4 + C_2$)
4. Polynomial Order 6 Case ($C_1 * x^6 + C_2$)
5. Polynomial Order 12 Case ($C_1 * x^{12} + C_2$)
6. Exponential Case ($C_1 * e^{C_2 * x}$)

The results for the given parameters of the system were as follows:

1. Linear Case: The optimum linear case results in a deflection at the free end of the beam of 0.61618 times the deflection for the optimum constant cross section case
2. Polynomial Order 2 Case: The optimum polynomial order 2 case results in a deflection at the free end of the beam of 0.67739 times the deflection for the optimum constant cross section case
3. Polynomial Order 4 Case: The optimum polynomial order 4 case results in a deflection at the free end of the beam of 0.76206 times the deflection for the optimum constant cross section case
4. Polynomial Order 6 Case: The optimum polynomial order 6 case results in a deflection at the free end of the beam of 0.81262 times the deflection for the optimum constant cross section case
5. Polynomial Order 12 Case: The optimum polynomial order 12 case results in a deflection at the free end of the beam of 0.88603 times the deflection for the optimum constant cross section case
6. Exponential Case: The optimum exponential case results in a deflection at the free end of the beam of 0.64671 times the deflection for the optimum constant cross section case

With these results, the linear case was selected as the optimum profile of the beam, shown by Equation 3(note that the values of the coefficients were carried out to a much higher precision than shown):

$$h(x) = -0.0642 * x + 0.0844 \quad \text{Equation 3}$$

With:

- $h(x) = height\ of\ the\ beam\ as\ a\ function\ of\ x\ (m)$
- $x = distance\ from\ wall\ towards\ the\ free\ end\ of\ the\ beam\ (m)$

Development of Simulation:

A separate MATLAB script was written to implement the FEM solution method, shown by Equation 4, now that the optimum profile of the beam was determined. First, the local stiffness matrix, with general form as shown by Figure 1, was computed for each discretized element of the beam.

$$\boldsymbol{F} = \boldsymbol{K}_{Global} * \boldsymbol{U} \quad \text{Equation 4}$$

With:

- $\boldsymbol{F} = Load\ Column\ Vector\ (N)$
- $\boldsymbol{K}_{Global} = Global\ Stiffness\ Matrix\ (N/m)$
- $\boldsymbol{U} = displacement\ matrix\ (m)$

$$[k]^{(e)} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix},$$

Figure 1: Local Stiffness Matrix in General Form

Note that in order to easily assemble the global stiffness matrix in later steps of the solution process, the local stiffness matrices were set to be the same size as the global stiffness matrix and were populated with values in the position where they would end up in the global stiffness matrix.

Next, the global stiffness matrix was assembled and the load column vector was established. Finally, boundary conditions were applied (displacement in both x and y directions at the wall and the slope of the beam at the wall is 0) and the displacement column vector was solved for.

Verification/Validation of Model:

The results from the solution of the displacement matrix were compared to what would be analytically expected and the percent error was found to be ~0.5% after discretizing the beam into only 15 elements. Plots generated from the solution were also analyzed to ensure that the results matched what would be expected.

Execution of Simulation to Determine The Deflection and Slope at the Free End of the Beam, and More

The optimum height of the beam from the wall towards the free end is shown by Figure 2.

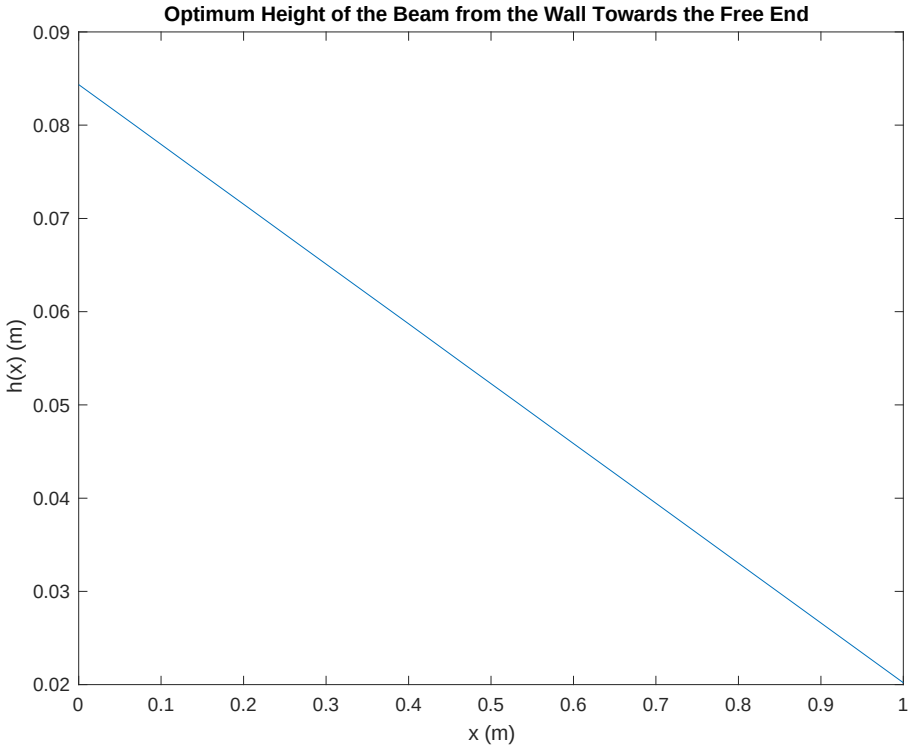


Figure 2: Height of the Beam vs Distance From the Wall

The deflection of the beam from the wall towards the free end is shown by Figure 3.

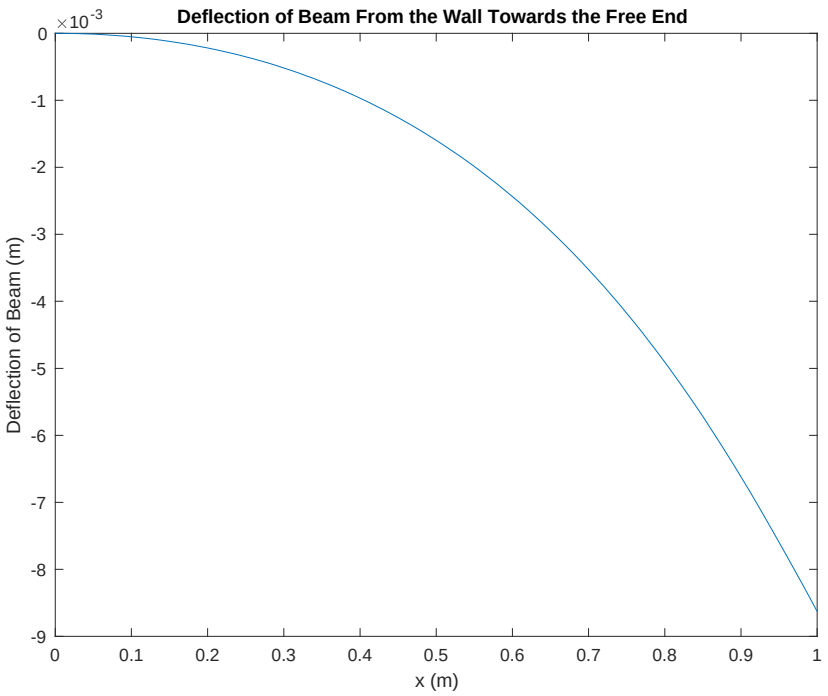


Figure 3: Deflection of the Beam vs Distance From the Wall

The slope of the beam from the wall towards the free end is shown by Figure 4.

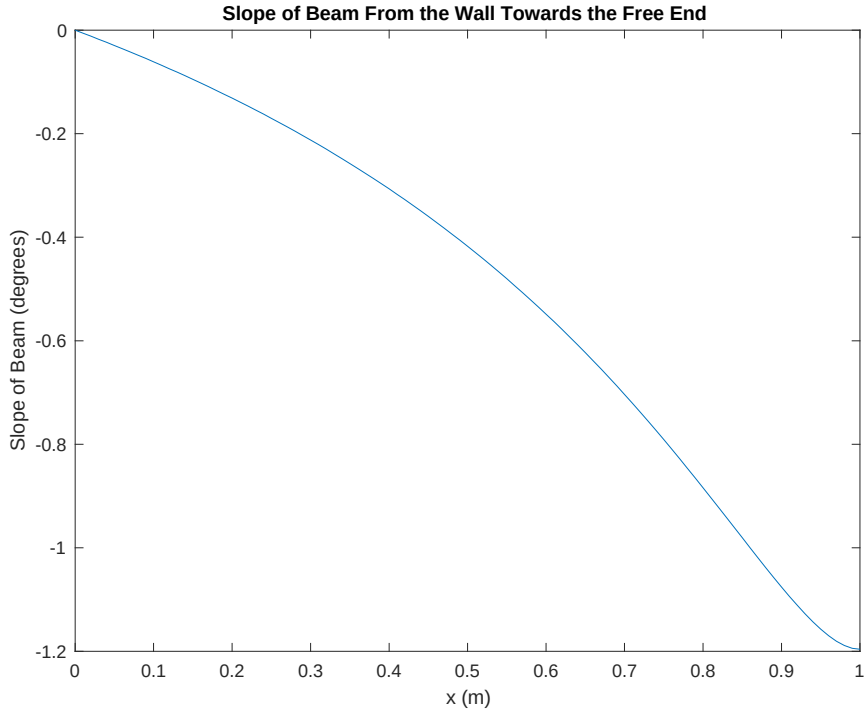


Figure 4: Slope of the Beam vs Distance From the Wall

Figure 5 shows the deflection at the free end of the beam vs the number of elements considered. It is evident by viewing this Figure that approximately 30 elements is enough to accurately determine the deflection at the free end of the beam.

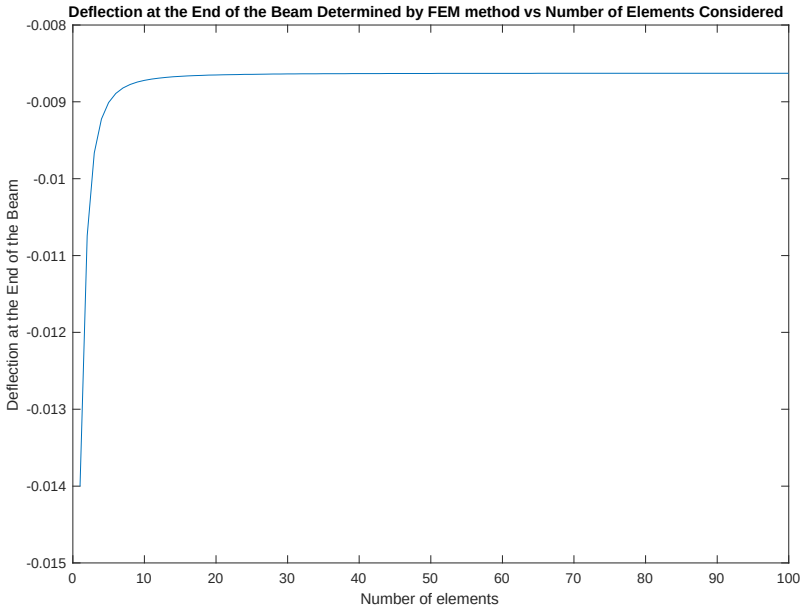


Figure 5: Deflection at the Free End of the Beam vs Number of Elements Considered

Figure 6 shows the slope at the free end of the beam vs the number of elements considered. It is evident by viewing this Figure that approximately 30 elements is enough to accurately determine the slope at the free end of the beam.

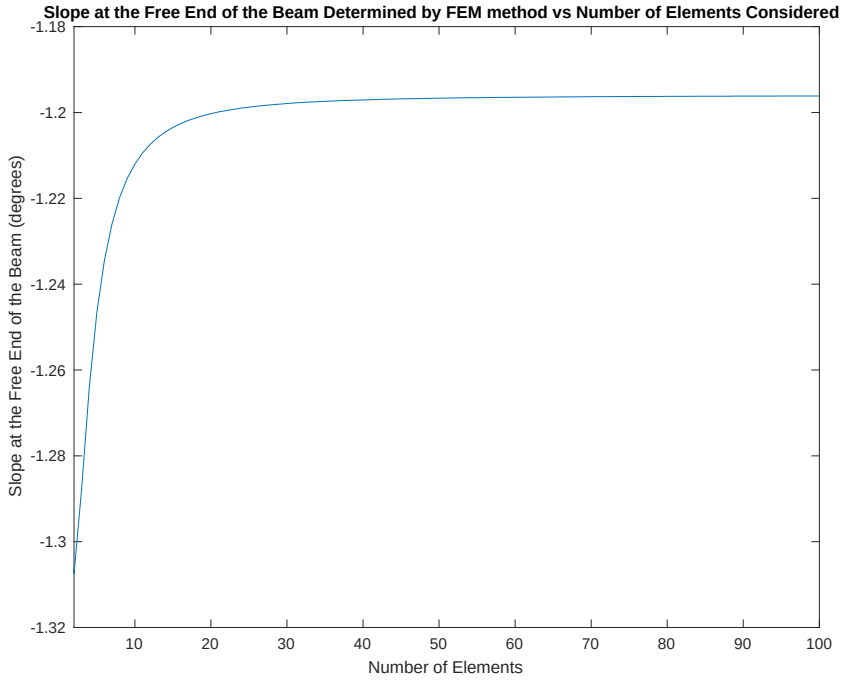


Figure 6: Slope at the Free End of the Beam vs Number of Elements Considered

Figure 7 shows the percent error between the analytical solution and the FEM solution method for the deflection at the free end of the beam vs the number of elements considered.

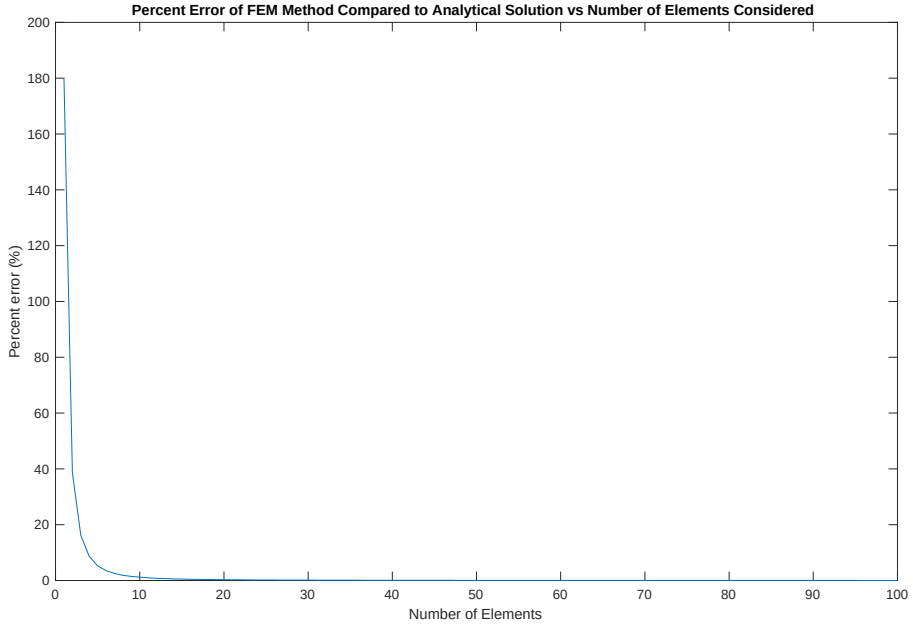


Figure 7: Percent Error vs Number of Elements Considered

Figure 8 shows a similar plot, but with different bounds for the number of elements considered.

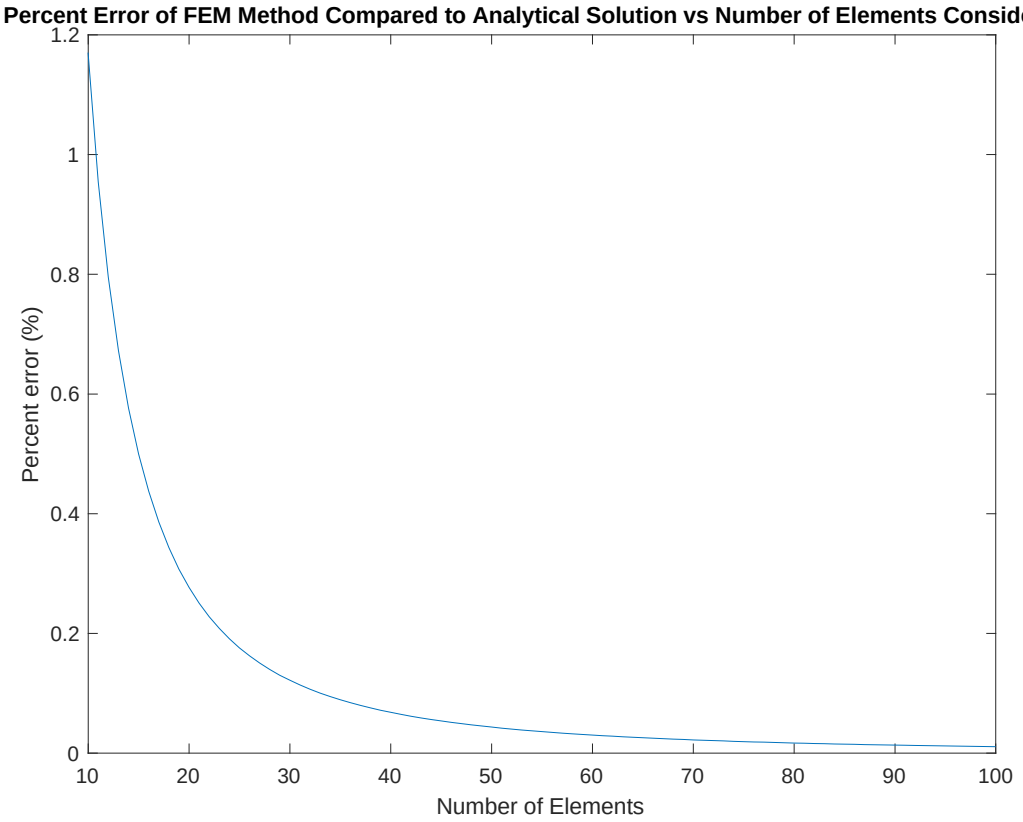


Figure 8: Percent Error vs Number of Elements Considered—Different Bounds

Conclusion:

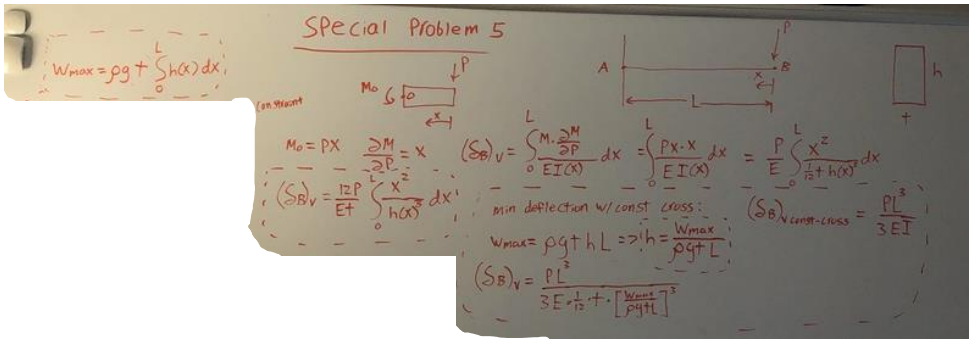
The objective of this project was completed by developing a mathematical model that could be implemented into a computational method in order to optimize the profile of the cantilever beam. Once the optimum profile was discovered, the FEM method was implemented to solve for the deflection and slope of the beam at numerous locations throughout its length. In addition, the FEM solution method was compared to the analytical solution for numerous discretizations considered. By reviewing Figure 8, it is evident that for a percent error of less than 0.2% the beam only had to be discretized into 24 elements.

Overall, this project helped to strengthen my ability to not only computationally optimize a system, but also solve beam problems by implementing the FEM method.

References:

N/A for this project. I am currently taking ME-480, Finite Element Analysis, and this was a great extension of some content we've been going over in there.

Supporting Calculations:



MATLAB Script-Optimizer:

```
%Mason Averill
%ME-544 Fall 2020, 10/15
%Special Problem 5--Optimizer

%%_____________________________________________________________________
%So, I first attempted to optimize this problem by strictly using the
%FEM method and hit some dead ends. Once I know all the parameters for the
%system SOLVING it is easy using the FEM method. Optimizing, not so much.
%If you have a method that works well to do so I would be very interested
%in viewing it.
%In order to optimize this problem, the following procedure was
%implemented. First, a function containing only two unknown coefficients is
%accepted as an input. One of the unknown coefficients is solved for in
%terms of the other utilizing the maximum beam weight constraint. The
%original expression inputed is now in terms of only one unknown
%coefficient. A symbolic expression is then developed and solved explicitly
%for the deflection at the end of the beam in terms of the unknown
coefficient.
%Appropriate bounds (user inputed) are set about where to look for this
%unknown optimum coefficient and values are incremented from the lower
%bound to the upper bound by the value associated with the variable
% 'step_size_const_solve'. The deflection at the end of the beam
%is found for each considered value of the unknown constant and stored in
% an array. The constant values are also stored in another array.
%Once all values have been considered, the deflection array is analyzed and
% the minimum value is selected. Using the location of the minimum
% deflection in this array, the corresponding value of the unknown optimum
% coefficient is extracted. The equation initially inputed is then updated
%with the newly found coefficients and then two different plots are
%generated. One plot shows the height of the beam as it varies in position
%FROM THE FREE END TOWARDS THE WALL and the other shows the deflection at
%the end of the beam vs the unknown coefficient. This plot is incredibly
%useful to ensure that the bounds set about where to look for the unknown
%coefficient is correct. It should show the deflection reaching a minimum
%at some point in the plot. If it doesn't, the bounds about where to look
%need adjusted (I was going to find a way to update these automatically but
%this has taken some time already and I am wanting to begin working on the
%main project). The ratio of the deflection for each considered function
%type vs the deflection for the ideal constant cross section type is
```

```
%written to the console. I have tried many different function types, more
%so than are now included in this script, mostly due to curiousity, and I
%am yet to find one that beats the linear case(for the given thickness),
%though the exponential case does merit an honorable mention.
%The optimum linear case results in a deflection of just ~62% of that
%for the constant cross section case. Note that these results are for the
%the given thickness. Changing the thickness will require setting
%new appropriate bounds about where to look for the optimum unknown
%coefficient.




%%_____________________________________________________________________
%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=7800; %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 Script-FEM Method:

```
%Mason Averill
%ME-544 Fall 2020, 10/15
%Special Problem 5--FEM Method

%I could make this quite a bit cleaner after developing the logic to solve
%using the FEM method, but this works and I need to start focusing on the
%main project



%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=7800; %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: 0.064150563266159596434825792623958*x1
+ 101/5000
%Now find h(x) from the wall towards the end of the beam

h_x=-0.064150563266159596434825792623958*x+0.084350563266160;

%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=0.064150563266159596434825792623958*x1 + 101/5000;

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
```