University of Colorado - Boulder
November 15, 2021

OEMP 3 Group Assignment 1

Nathan Evans — Dept. of Aerospace Engineering

Ian Faber — Dept. of Aerospace Engineering

Alex Herring — Dept. of Aerospace Engineering

Daniel Mascarenas — Dept. of Aerospace Engineering

Group 11, 10:40 Lab Section

Figure 1: The lab!

Part 1:

To see how a full whiffle tree is constructed, design a full whiffle tree with p = 4. This whiffle tree has three bars (seen in blue in Figure 3) that translate a single point force Fext pulling up on the whole system into the four point loads representing the lift on the wing. Following Figure 3 below, there are seven quantities needed to balance the whiffle tree: the external point force Fext and six lengths a-f.

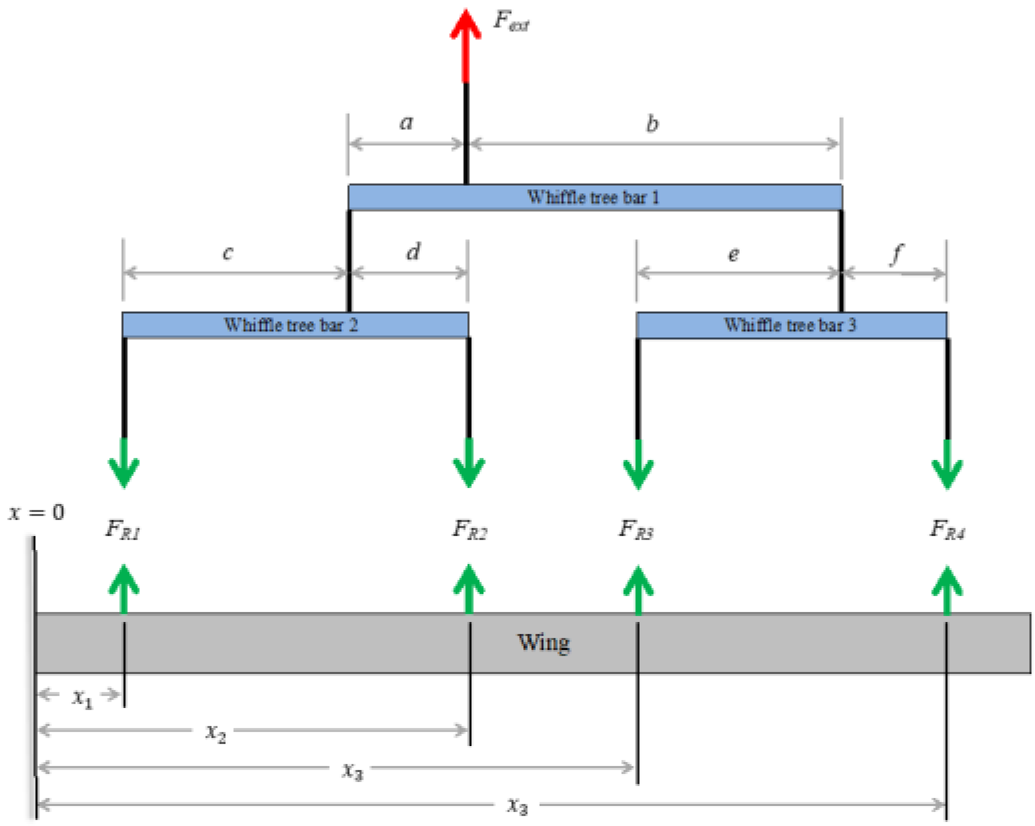


Figure 3. Diagram of the whiffle tree with p = 4. (Not necessarily to scale.)

First, use one of your group member's discretize_load functions to compute the value of the four resultant point forces F_{Ri} and the distance from each point load to the origin $\bar{x}_i$. Then, draw free body diagrams, solve equilibrium equations, and use other known information about the distances between the four point forces to compute the seven unknowns. Complete the chart to the right on the PDF you submit.

<u>**Unknown**</u>	<u>**Value**</u>	<u>**Units**</u>
F_{ext}	**27,264**	**lb**

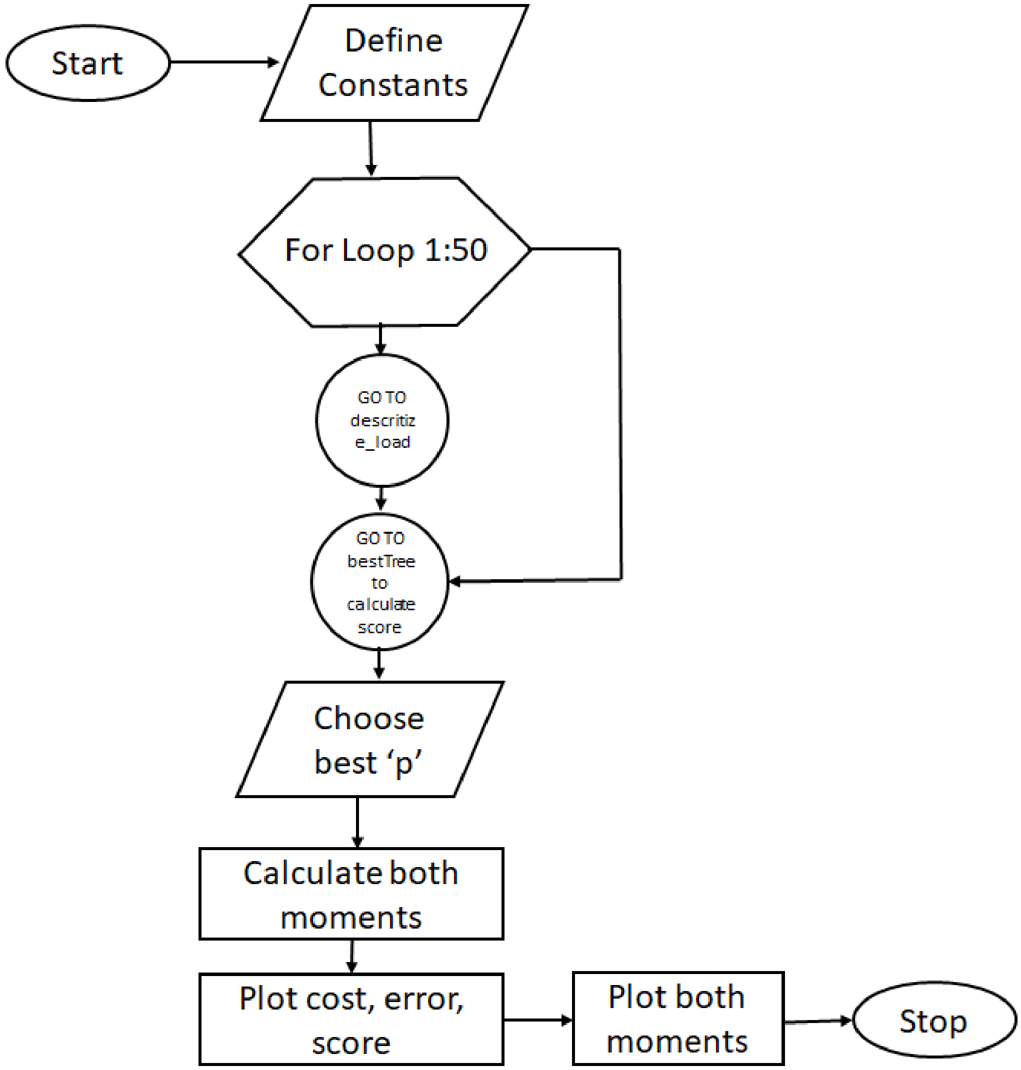
a	**3.245**	**ft**
b	**9.735**	**ft**
c	**2.758**	**ft**
d	**3.862**	**ft**
e	**1.135**	**ft**
f	**3.405**	**ft**

Part 2:
Before you start writing code, create a flow diagram that details how your code will be implemented. Show the different functions you're using and the inputs and outputs. Also describe what's happening inside of each function. You can see two example flow diagrams at the end of this assignment, along with a key for the different symbols you should use.

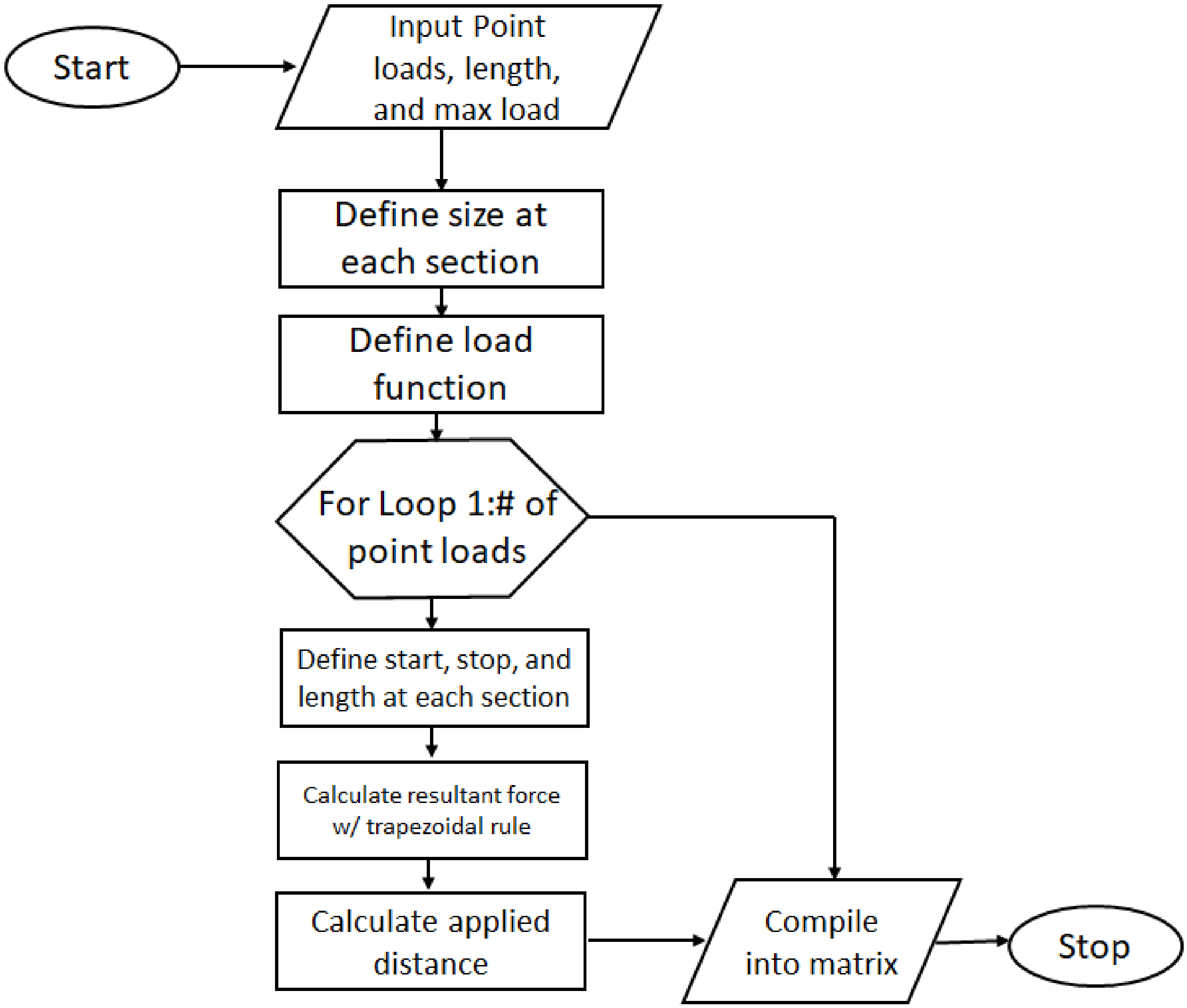
Once you have this flow diagram, you can start modifying the functions you wrote for the Individual Assignment and/or creating new functions and scripts. Your goal for the rest of this group assignment is to determine the best set of point loads. What do we mean by the "best" set of point loads? The exact definition is up to you! You should use the WT score as one criteria, but there may be other considerations.

While your group only has to submit one set of code, we recommend that your group develops multiple independent versions of the same code. This allows you to check your code to make sure it's working as you intend it to.

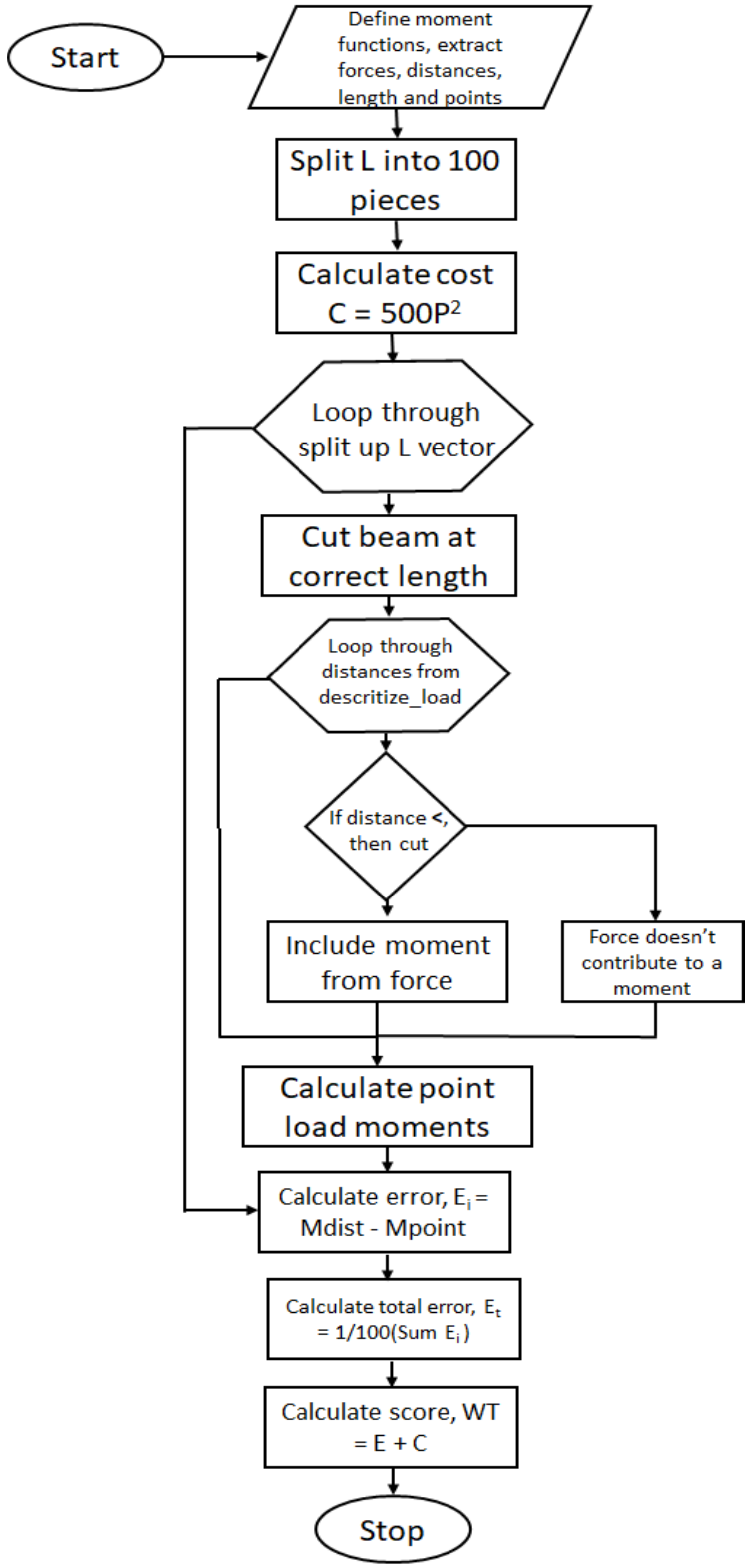
Main Flowchart:



Discretize Load Function Flowchart:



Best Tree Function Flowchart:



Main Code:

```
clear; clc; close all;

w0 = 2001;
L = 27.25;
maxP = 50;

score = zeros(1,maxP);
cost = zeros(1,maxP);
error = zeros(1,maxP);

for p = 1:maxP
    discretized = discretize_load(p, L, w0);
    [score(p), cost(p), error(p), ~, ~] = bestTree(p, discretized, w0, L);
end

chosenP = 15;

[~, ~, ~, Mdist, Mpoint] = bestTree(chosenP, discretize_load(chosenP, L, w0), w0, L);

figure
subplot(3,1,1)
hold on;
title("Wiffle Tree cost vs. p point loads");
plot(cost)
xline(chosenP);
xlabel("Number of point loads");
ylabel("Cost ($)");
legend("Cost","Chosen number of point loads: " + num2str(chosenP),'Location','best')
hold off;

subplot(3,1,2)
hold on;
title(" Wiffle Tree error vs. p point loads");
plot(error)
xline(chosenP);
xlabel("Number of point loads");
ylabel("Error (N*m)");
legend("Error","Chosen number of point loads: " + num2str(chosenP),'Location','best')
hold off;
```

```
subplot(3,1,3)
hold on;
title("Wiffle Tree score vs. p point loads");
plot(score)
xline(chosenP);
xlabel("Number of point loads");
ylabel("Wiffle Tree score");
legend("Score","Chosen number of point loads: " + num2str(chosenP),'Location','best')
hold off;

figure
subplot(2,1,1)
hold on;
title("Analytical beam moment and point beam moment, unzoomed");
plot(Mdist);
plot(Mpoint);
xlabel("Distance from origin (m)");
ylabel("Moment (N*m)");
legend("Analytical moment", "Point moment", 'Location', 'best');
hold off;

zoom = 10;

subplot(2,1,2)
hold on;
title("Analytical beam moment and point beam moment, zoomed in");
plot(Mdist(length(Mdist)/2-zoom:length(Mdist)/2+zoom));
plot(Mpoint(length(Mpoint)/2-zoom:length(Mpoint)/2+zoom));
xlabel("Distance from origin (m)");
ylabel("Moment (N*m)");
legend("Analytical moment", "Point moment", 'Location', 'best');
hold off;
```

Best Tree Function:

```
function [score, cost, error, Mdist, Mpoint] = bestTree(p, matrix, w0, L)

M_dist = @(x)(w0/2)*(((-x^3)/(3*L))+(x^2)+(L*x)-((L^2)/3));
M_shear = @(x)(w0*x)*((L/2) + x - ((x^2)/(2*L)));

forces = matrix(:,1);
distances = matrix(:,2);
[~, Ma] = wall_reactions(matrix);

x = linspace(0,L,100);
cost = 500*(p^2);

error = 0;

Mdist = zeros(1,length(x));
Mpoint = zeros(1,length(x));

for i = 1:length(x)
    d = x(i);
    M_forces = 0;

    for j = 1:length(distances)
        if(distances(j) < d)
            M_forces = M_forces + (forces(j)*distances(j));
        end
    end

    M_point = M_shear(d) - M_forces - Ma;

    error = error + (M_dist(d) - M_point)^2;

    Mdist(i) = M_dist(d);
    Mpoint(i) = M_point;
end

error = (1/100)*error;

score = cost + error;

end
```

Discretize Load Function:

```
function [matrix] = discretize_load(p, L, w0)
% Function that splits up a distributed load into p sections,
% calculating the resultant force and location of each section
%   Inputs: Number of sections (p), length of the beam (L),
%           force constant w0
%   Outputs: p x 2 matrix of resultant forces and locations
%
% For this function, I am using the trapezoidal rule since it is
% the exact area for this situation and generally a good option
% for approximating. I am calculating xBar with the 1-D centroid
% formula, i.e. integral(x*f(x)dx)/integral(f(x)dx), both integrals
% are calculated from x0 to xf, the start and stop of each section.
% I am assuming that the force distribution is acting according to
% a constant function, and does not change mathematically from section
% to section. If that assumption was not true, I would have another
% input that can accomodate different function handles.

sectionSize = L/p;

f = @(x) w0*(1-(x/L));

matrix = [];

for i = 1:p
    x0 = sectionSize*(i-1);
    xf = sectionSize*i;
    x = linspace(x0, xf, 1000);

    Fr(i) = trapz(x, f(x));
    xBar(i) = trapz(x, x.*f(x))/Fr(i);
    matrix = [matrix; [Fr(i), xBar(i)]];
end


end
```

```
function [Ay, Ma] = wall_reactions(matrix)
% Function that calculates wall reactions based on a discretized
% distributed load
%    Inputs: Matrix of resultant forces and applied distances after
%            discretizing a distributed load
%    Outputs: Vertical wall reaction (Ay) and moment (Ma)

forces = matrix(:,1);
distances = matrix(:,2);

Ay = sum(forces);
Ma = sum(distances.*forces);

end
```

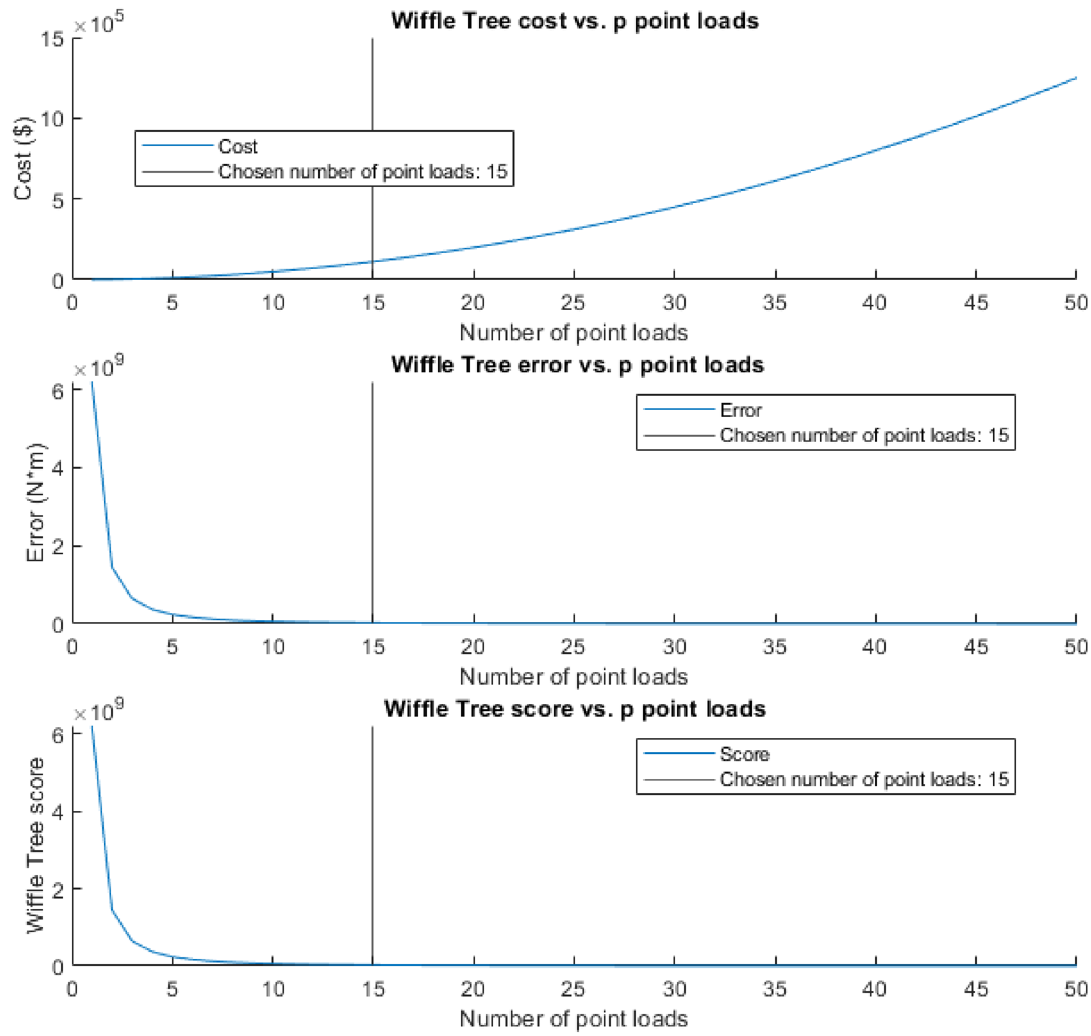
Part 3:
Describe all assumptions your group made in writing your code.

In writing our code, we first made assumptions for modeling the load in general. We assumed that the wing is static and rigid and that the mass of the wing can be ignored. We also assumed that a 2D rectangle is an effective representation of the wing. Finally, we assumed that the distributed load can be modeled by a single equation throughout the wing, which was given in the assignment.

For the coding part, we assumed the distributed load on the wing is a linear function and can be approximated by a series of point loads, which we calculated using the trapezoidal rule for an exact estimate of the resultant force. We also assumed that the cost can be approximated by $C = 500p^2$, where p is the number of point loads and is assumed to be less than 50, as given in the assignment. We also assumed that error is characterized by mean square error across 100 discrete sections of the wing, and that the overall score is characterized by WT = E + C, as given in the assignment.

Part 4:

Plot WT(p) vs. p. Clearly label what you determine to be the best value of p to use in a test.



Part 5:

Describe your group's chosen set of point loads by creating a table of the location and magnitude of each point load. Remember to include units! Note that you should not design the full whiffle tree like you did in Part 1 of this Group Assignment; you just need to tell us about the set of point loads that discretize the distributed load.

Position (x-distance) (ft)	Load (lb*ft)
0.8979	3514
2.714	3272
4.529	3029
6.345	2787
8.161	2545
9.976	2302
11.79	2060
13.60	1818
15.42	1575
17.23	1333
19.04	1090
20.85	848.2
22.65	605.9
24.42	363.5
26.04	121.2

Part 6:

Justify why your chosen set of point loads is the “best” by writing a paragraph convincing Prof. Schwartz and the Lab Assistants that thebesty should design and build this whiffle tree to test the Fire Boss wing.

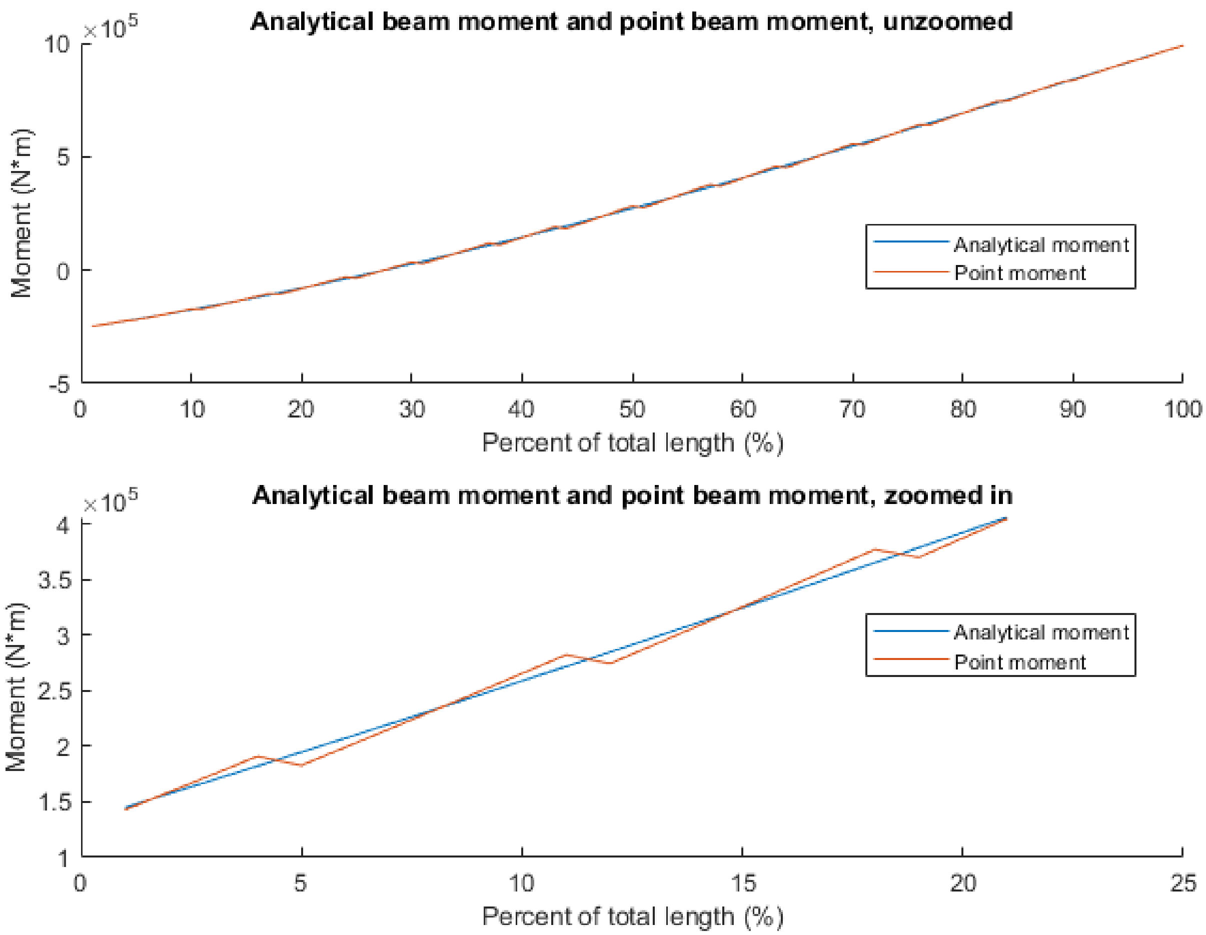
As a group, we decided to choose 15 as the “best” number of point loads for our Wiffle Tree. When looking at the trade-offs between cost and error, 15 point loads is a nice balance between the two. 15 loads has a cost of around $112,000, but it has an associated error that is almost zero. Our cost may seem expensive at first glance, but this number of point loads, in our opinion, is the minimum number needed to have a negligible error for our Wiffle Tree, so we feel this cost is justified in order to obtain the most accurate data.

Going for fewer point loads and a correspondingly less expensive Wiffle Tree would give results with a higher degree of error, and therefore could be unreliable. On the other hand, if we were to increase the number of point loads to the maximum of 50, the associated cost would be about $1.2 million, but the error would only be slightly less than the error at 15 point loads. In other words, we would get diminishing returns in accuracy if we used any more than 15 point loads, with unnecessary increasing costs. This reasoning is also backed up by the fact that our score function is minimized at 15 point loads, which was the goal in the first place!

Part 7/8:

On the same figure, plot two bending moment diagrams for your chosen value of p: $M_{dist}(x)$ and $M_{point}(x)$. Make sure that your graph shows the full range $0 \leq x \leq L$.

Now, show a zoomed-in version of the plot you created in Part 7. You should zoom in enough that you can see the difference between the two bending moment functions. Present this graph, describe the differences between the two bending moment functions, and explain how these differences relate to the error function E.



The difference between the analytical and point moments seem to oscillate between positive and negative, but the difference appears to decrease in magnitude as we move along the bar. This will cause the error to converge to 0 as the number of point loads gets unboundedly large, which is exactly what we see in our error graph!