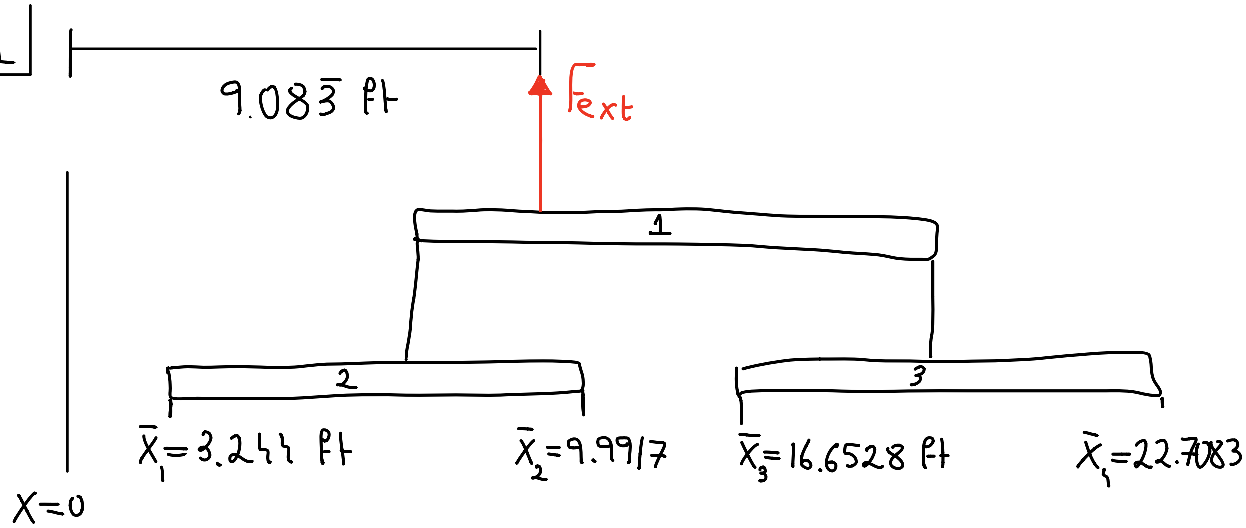
OEMP3:

Names: 1) Aziz Alwalban 2) Jim Austin
3) Andrew Logue 4) Ian Wong
5) Jack Foster

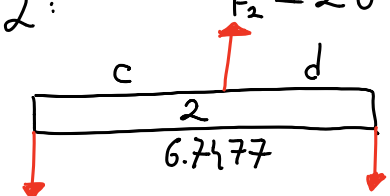
1



Results of $\bar{x}$ and F_{Ri} are found in MATLAB

- Whiffle 2:

$F_2 = 20447.9$ lb



$x_2 - x_1 = 6.7477$ ft

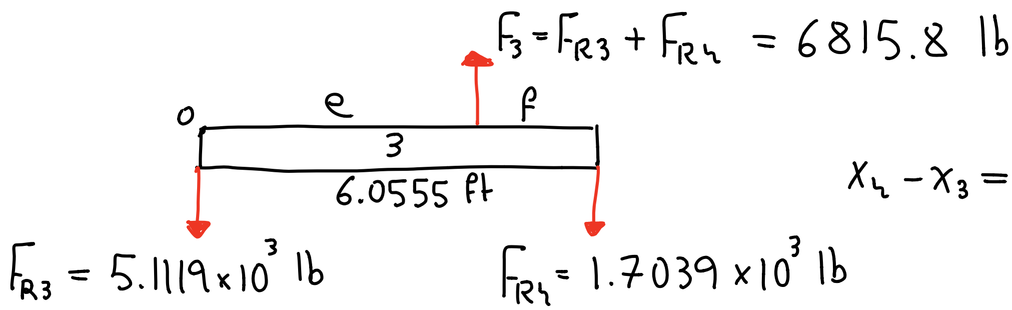
$F_{R1} = 1.1928 \times 10^4$ lb

8.5199×10^3 lb $= F_{R2}$

$$\sum M_o = 0: \quad (20447.9)(c) - (8.5199 \times 10^3)(6.7477) = 0$$

$\boxed{c = 2.81152 \text{ ft}}$ $\boxed{d = 3.9362 \text{ ft}}$

- **Whiffle 3:**

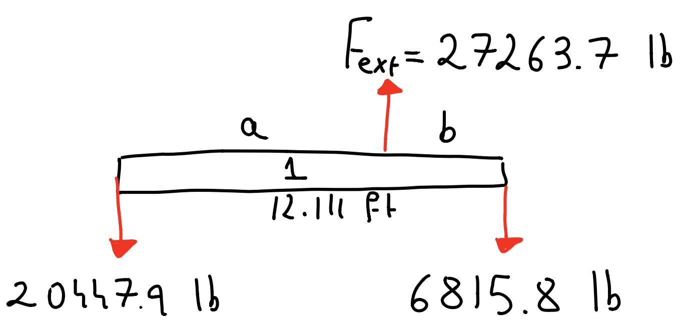


$x_4 - x_3 = 6.0555 \text{ ft} = L$

$$\sum M_o = 0: \quad (6815.8)(e) - (6.0555)(1.7039 \times 10^3) = 0$$

$\boxed{e = 1.5138 \text{ ft}}$ $\quad$ $\boxed{f = 4.5417 \text{ ft}}$

- **Whiffle 1:**



$$L = d + (\bar{x}_3 - \bar{x}_2) + e$$
$$= 12.111 \text{ ft}$$

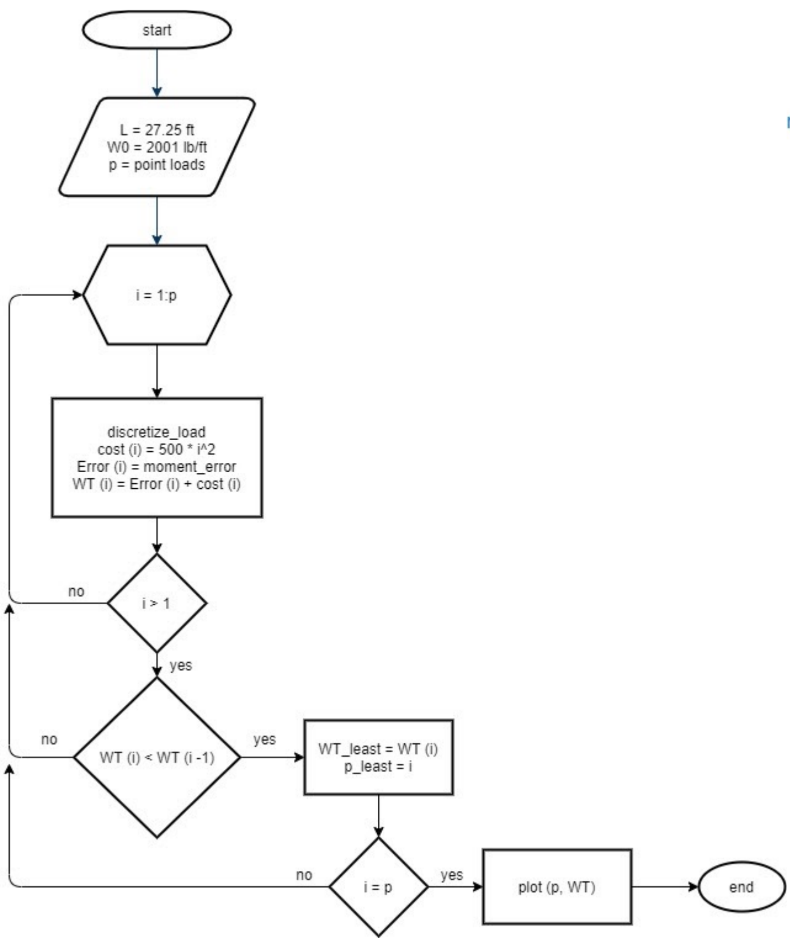
$$\sum M_o = 0: (27263.7)a - (6815.8)(12.111) = 0$$

$\boxed{a = 3.0277 \text{ ft}}$ $\quad$ $\boxed{b = 9.083 \text{ ft}}$

Unknown	Value	Units
F_{ext}	27263.7	lb
a	3.028	ft
b	9.083	ft
c	2.8115	ft
d	3.936	ft
e	1.514	ft
f	4.542	ft

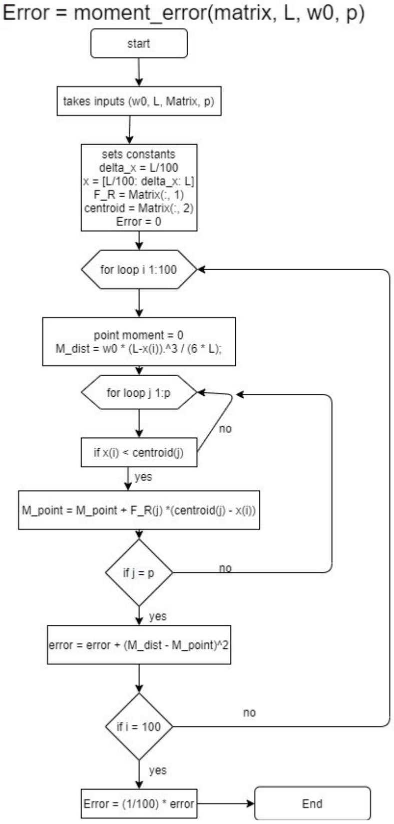
2

main script

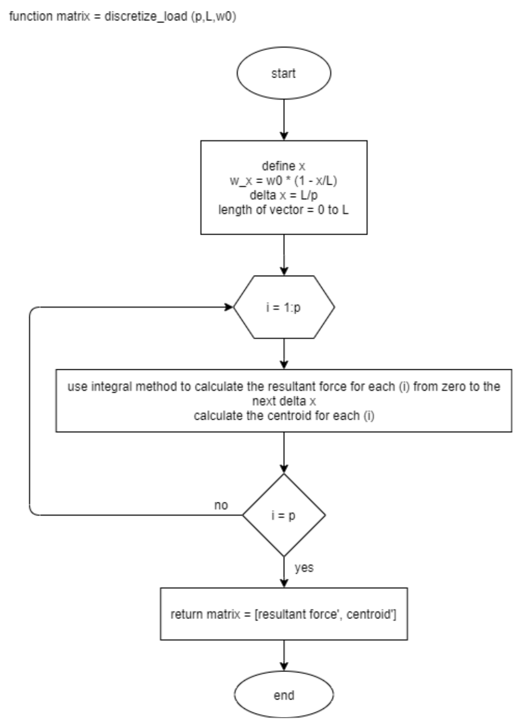


Error Function

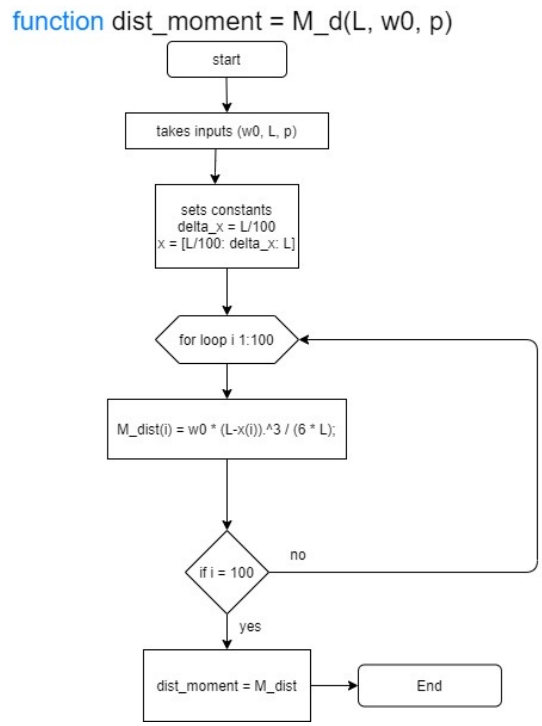
nction Error = moment_error(matrix, L, w0, p)



Discretize Function



Moment Function



4

Assumptions:

1) The wing is in a static equilibrium
2) No deformations experienced in the wing due to moments
3) No change in span length L from normal strain
4) p is less than or equal to 50 for the cost equation to be valid

6

Shown in page 6

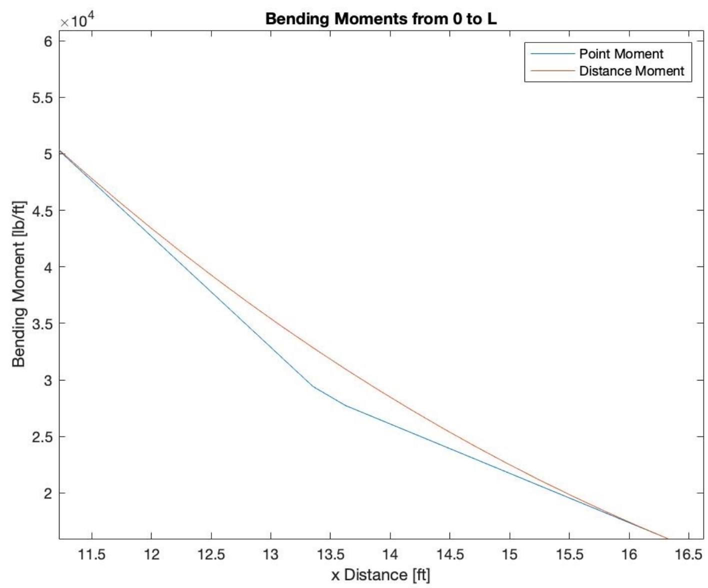
7

The optimum number number of point loads for this application is 14. Although each additional point decreases the error, it also adds to the cost of production. We were able to find the optimum number of points by assigning a whiffle tree score to each number of point loads, determined by the sum of a cost function and the moment error calculation. The two functions are as such:

WT = Whiffle Tree Score
C = Cost
P = number of points
E = error of bending moment

Plotting the whiffle tree score as a function of the number of points, it reaches its minimum at 14 points, meaning we are able to minimize the error at the most economic cost.

9



There is an obvious difference in the two moments only when we zoom in. This is because the point loads are not as accurate as the distributed load. This is also amplified from the error equation since its squared and can sense small errors.

Point Load	Magnitude	Units	Location	Units
1	3756	lb	0.961	ft
2	3478	lb	2.907	ft
3	3200	lb	4.852	ft
4	2921	lb	6.797	ft
5	2643	lb	8.742	ft
6	2365	lb	10.69	ft
7	2087	lb	12.63	ft
8	1808	lb	14.57	ft
9	1530	lb	16.51	ft
10	1252	lb	18.46	ft
11	973.7	lb	20.39	ft
12	695.5	lb	22.32	ft
13	417.3	lb	24.22	ft
14	139.1	lb	25.95	ft

Table of Contents

House Keeping

```
clc
clear all
close all
```

Part 1:

```
L = 27.25; %In ft
w0 = 2001; %lb/ft
p = 50;
delta_x = L/100;
x = [L/100: delta_x: L];

[result] = discretize_load(p, L, w0);
F_R = result(:,1);
centroid = result(:,2);

[reaction, moment] = wall_reaction(F_R, centroid);


for i = 1:p

[result] = discretize_load(i, L, w0);
cost(i) = 500 * i^2;
E(i) = moment_error(result, L, w0, i);
WT(i) = E(i) + cost(i);

if(i>1)
if (WT(i) < WT(i-1))
    WT_least = WT(i);
    p_leat = i;
    point_table = result;
    point_moment = M_p(i,L,w0,result);
    dist_moment = M_d(i,L,w0);
end
end

end

u = [1:p];
figure(1)
semilogy(u, WT);
```

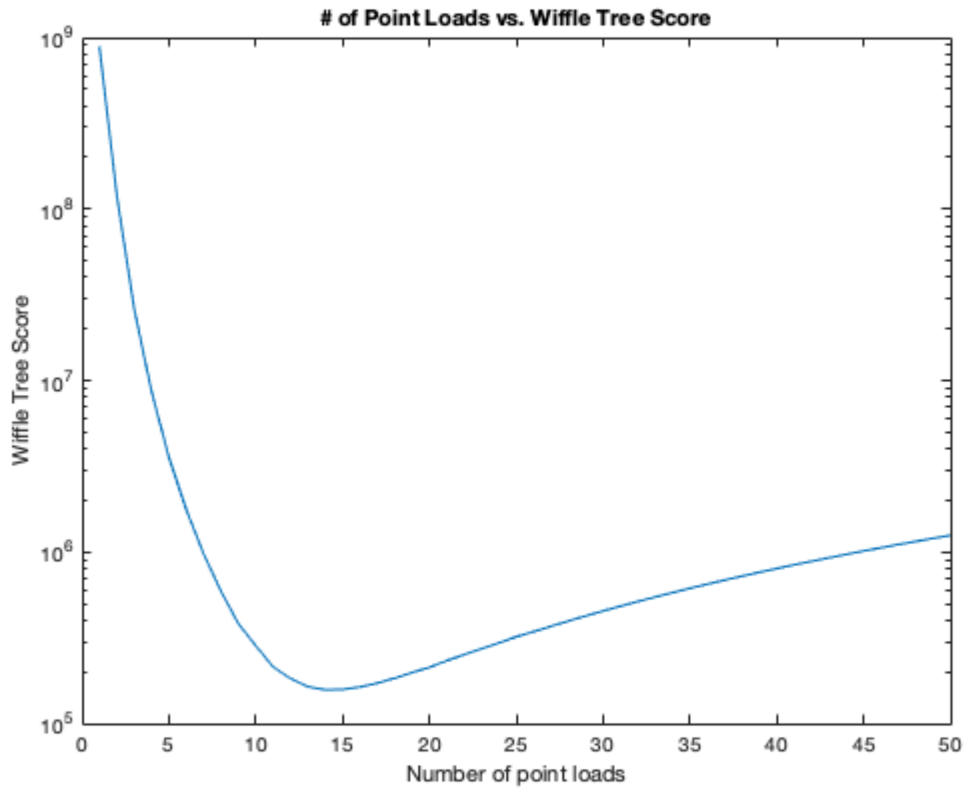
```
xlabel('Number of point loads');
ylabel('Wiffle Tree Score');
title('# of Point Loads vs. Wiffle Tree Score');
figure(2)
plot(x,point_moment);
hold on
plot(x,dist_moment);
xlabel('x Distance [ft]');
ylabel('Bending Moment [lb/ft]');
title('Bending Moments from 0 to L');
legend('Point Moment','Distance Moment');
hold off

figure(3);
plot(x, point_moment, [0 1], [0 2]);
```

Functions:

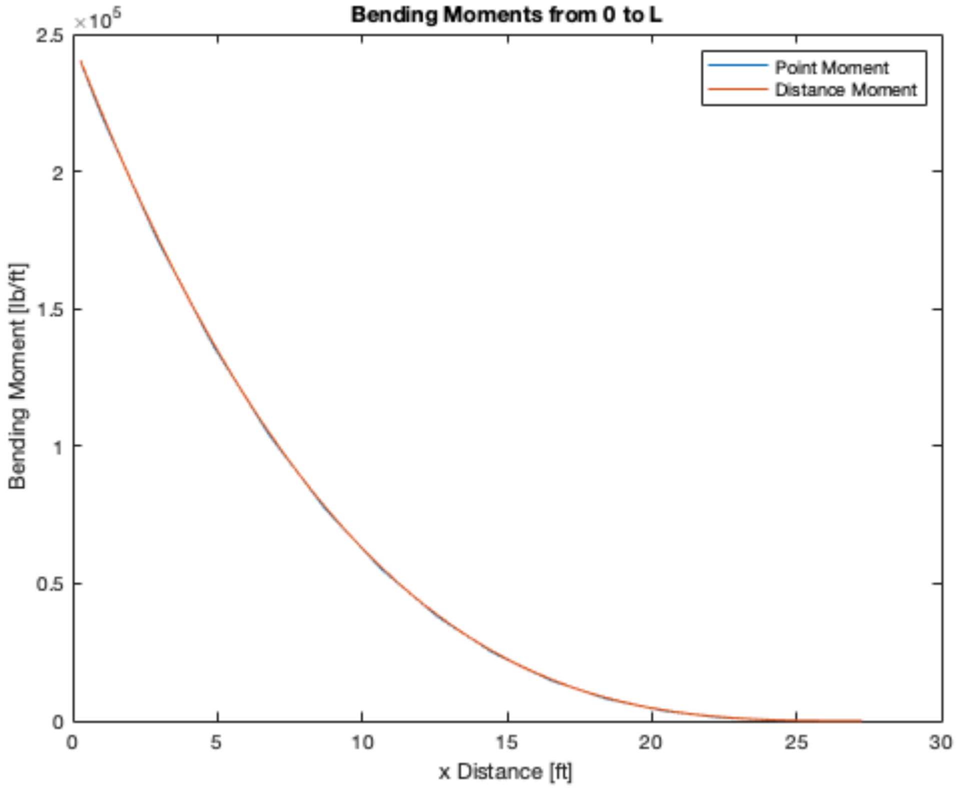
```
function Matrix = discretize_load(p, L, w0)
    %Function that calculates F_r and the centroid x
    syms x;

    w_x = w0 * (1 - x / L);
    delta_x = L / p;
    length_vector = [0:delta_x:L];

    for i = 1:p

        %Integral method to find resultant force where it starts
        %from zero to the next delta x for each integral
      F_R(i) = int(w_x, length_vector(i), length_vector(i+1));

      %The equation of the centroid is the integral of w(x)x dx
 divided by
      %the integral of w(x) dx.
      centroid(i) = int(w_x * x,length_vector(i), length_vector(i
+1))...
          / int(w_x, length_vector(i), length_vector(i+1));
    end

    F_R = double(F_R);
    centroid = double(centroid);
    Matrix = [F_R', centroid'];
end


function [Reaction, Moment] = wall_reaction(F_R, centroid)
    %Function that calculates the moment and reaction force

    Reaction = sum(F_R);
    M = F_R .* centroid;
    Moment = sum(M);
```

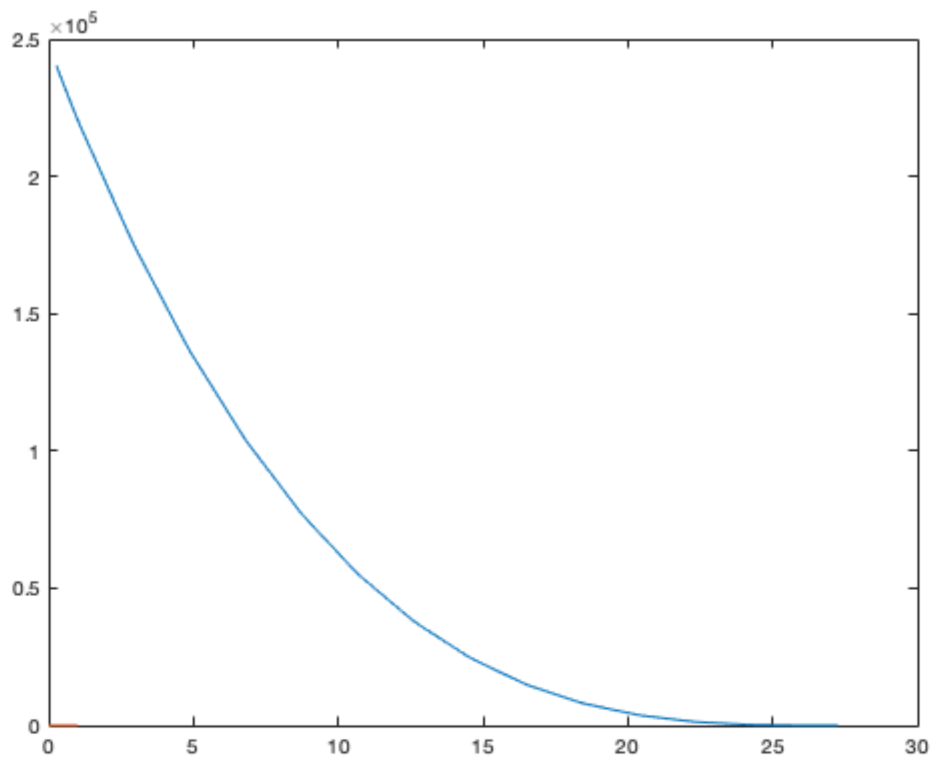
```
end


function E = moment_error(matrix, L, w0, p)

    delta_x = L/100;
    x = [L/100: delta_x: L];

    F_R = matrix(:,1);
    centroid = matrix(:,2);

    Error = 0;
    for i = 1:100
        M_point = 0;

         M_dist = w0 * (L-x(i)).^3 / (6 * L);
        for j = 1:p
        if(x(i) < centroid(j))
           M_point = M_point + F_R(j) * (centroid(j)-x(i));
        end

        end
        Error = Error  +  (M_dist - M_point)^2;
    end
    E = (1/100) * Error;
    M_dist = w0 * (L-(0.5*L)).^3 / (6 * L);

end

function point_moment = M_p(p_leat,L,w0,matrix)

    delta_x = L/100;
    x = [L/100: delta_x: L];
    F_R = matrix(:,1);
    centroid = matrix(:,2);
    for i = 1:100
        M_point(i) = 0;

        for j = 1:p_leat
            if(x(i) < centroid(j))
                M_point(i) = M_point(i) + F_R(j) * (centroid(j)-x(i));
            end
        end
    end
    point_moment = M_point;
end

function dist_moment = M_d(p_leat,L,w0)
    delta_x = L/100;
    x = [L/100: delta_x: L];
    for i = 1:100
         M_dist(i) = w0 * (L-x(i)).^3 / (6 * L);
    end
    dist_moment = M_dist;
```

```
end
```



# of Point Loads vs. Wiffle Tree Score
Wiffle Tree Score
Number of point loads
10^9
10^8
10^7
10^6
10^5
0
5
10
15
20
25
30
35
40
45
50


Bending Moments from 0 to L
Point Moment
Distance Moment
Bending Moment [lb/ft]
x Distance [ft]
×10^5
0
0.5
1
1.5
2
2.5
0
5
10
15
20
25
30


×10^5
0
0.5
1
1.5
2
2.5
0
5
10
15
20
25
30

Published with MATLAB® R2020a