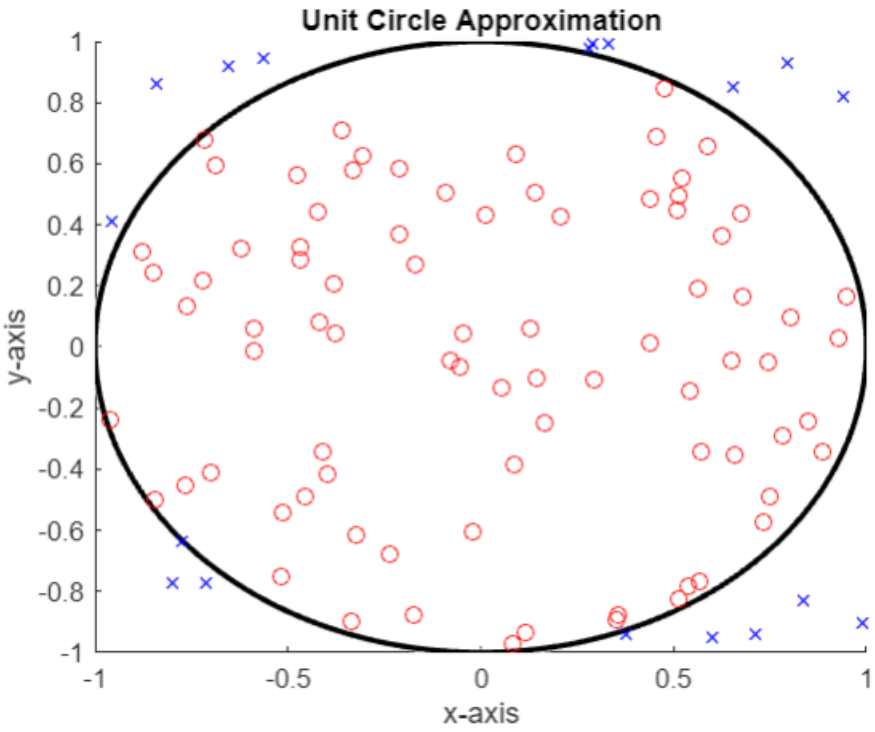
MLA - 007

```
%Use function command as requested
function [estimate,pcerror] = EstimatePi(numPoints)
xvec = (2.*rand(1,numPoints) - 1);
yvec = (2.*rand(1,numPoints) - 1);
%distance from center stuff
r = sqrt(yvec.^2 + xvec.^2);
rUnitC = 1;
inUnitC = r<=rUnitC;
numberinC = sum(inUnitC);
percentin = numberinC/length(inUnitC);
estimate = percentin*4;
pcerror = abs((estimate-pi)/pi)*100;
%disp unit circle
viscircles([0 0],1,'color','k')
%set x and y in and out of circle
xin = xvec(inUnitC==1);
xout = xvec(inUnitC ==0);
yin = yvec(inUnitC==1);
yout = yvec(inUnitC ==0);
hold on
scatter (xin,yin,'ro')
scatter (xout,yout,'bx')
hold off
xlabel('x-axis')
ylabel('y-axis')
title('Unit Circle Approximation')
end
%Driver stuff
[estimate1,pcerror1] = EstimatePi (10^2);
figure
[estimate2,pcerror2] = EstimatePi (10^3);
figure
[estimate3,pcerror3] = EstimatePi (10^4);
figure
[estimate4,pcerror4] = EstimatePi (10^5);
figure
[estimate5,pcerror5] = EstimatePi (10^6);
figure
[estimate6,pcerror6] = EstimatePi (10^7);
figure
%Make Table
NumPoints = [10^2,10^3,10^4,10^5,10^6,10^7];
Pi_Estimate = [estimate1,estimate2,estimate3,estimate4,estimate5,estimate6];
Percent_Error = [pcerror1,pcerror2,pcerror3,pcerror4,pcerror5,pcerror6];
NumPoints = NumPoints';
Pi_Estimate = Pi_Estimate';
```

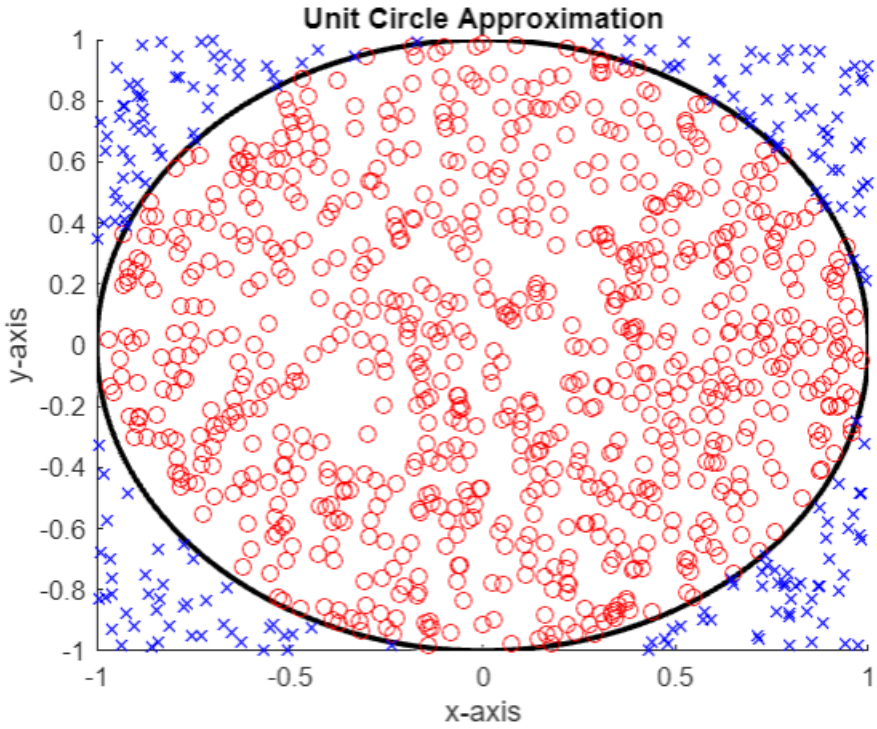
```
Percent_Error = Percent_Error';
table(NumPoints,Pi_Estimate,Percent_Error)
%{ Analysis
% Re-run your code numerous times. Why does the output change after each
% run?
% Because the rand function is used each approximation will differ slightly
% Explain the results of your statistical study. How does numPoints affect
% the π estimate and the percent error?
% As the number of points goes up the graph becomes more of a solid circle
% within a solid square, the processing of the code takes longer, and the
% estimations get more accurate
% Examine each plot (you should have 6, one for each value of numPoints).
% Are there any plots which appear to be flawed due to a lack of points?

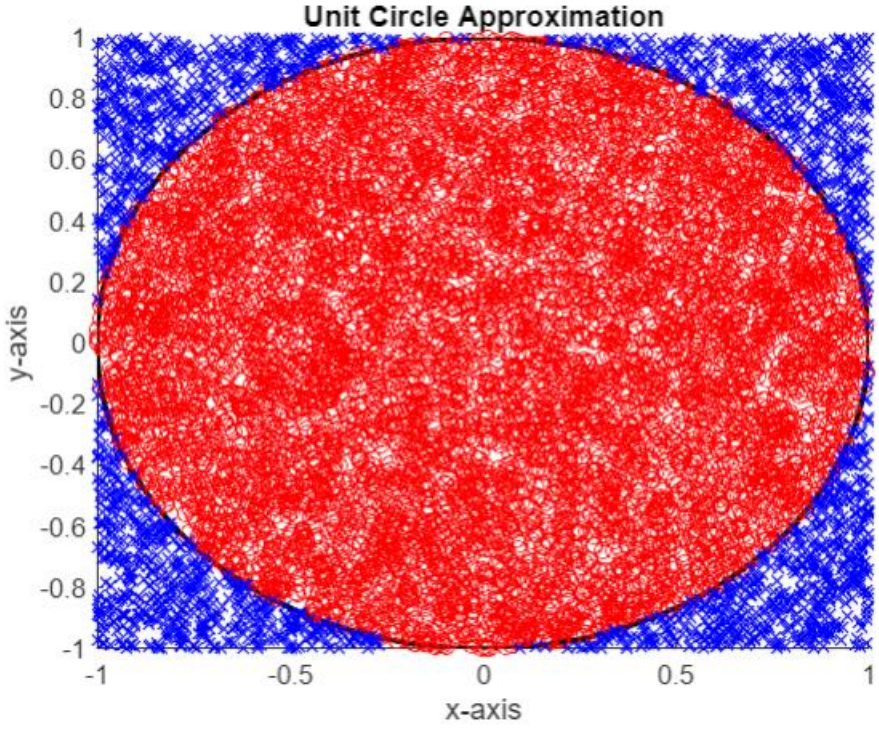

% Plot one appears to me to have too few points to be accurate

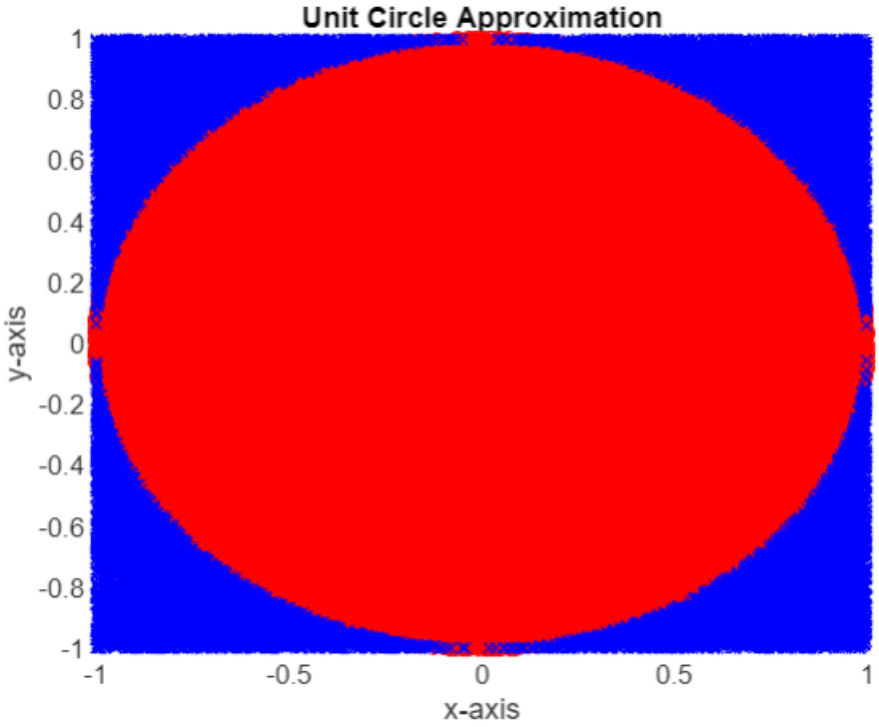

% Are there any plots which appear to contain too many points?
% Plots 5 and
% 6 offer little to no visual difference over 4 which to me suggests that
% they are simply taking up more computing power to acheive a similar
% result
% What’s the minimum number of points needed to estimate π without
% over-cluttering the plot?
% The first plot with a percent error of 1 or
% less was on average plot 3 but 4 offers good improvment without overly slowing
down
% processing.
%}
```

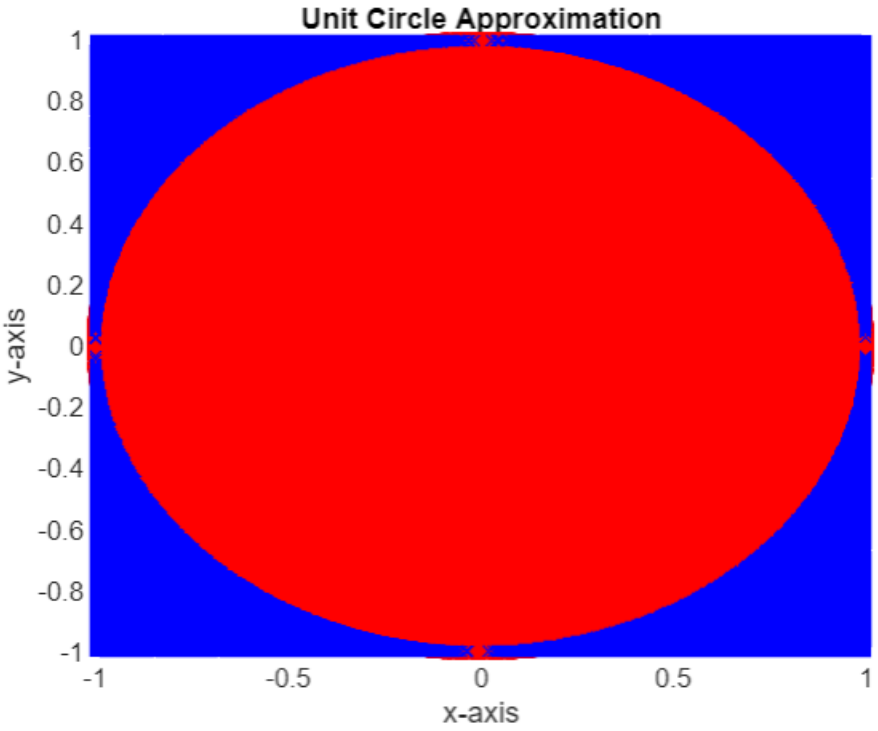

Unit Circle Approximation
y-axis
x-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

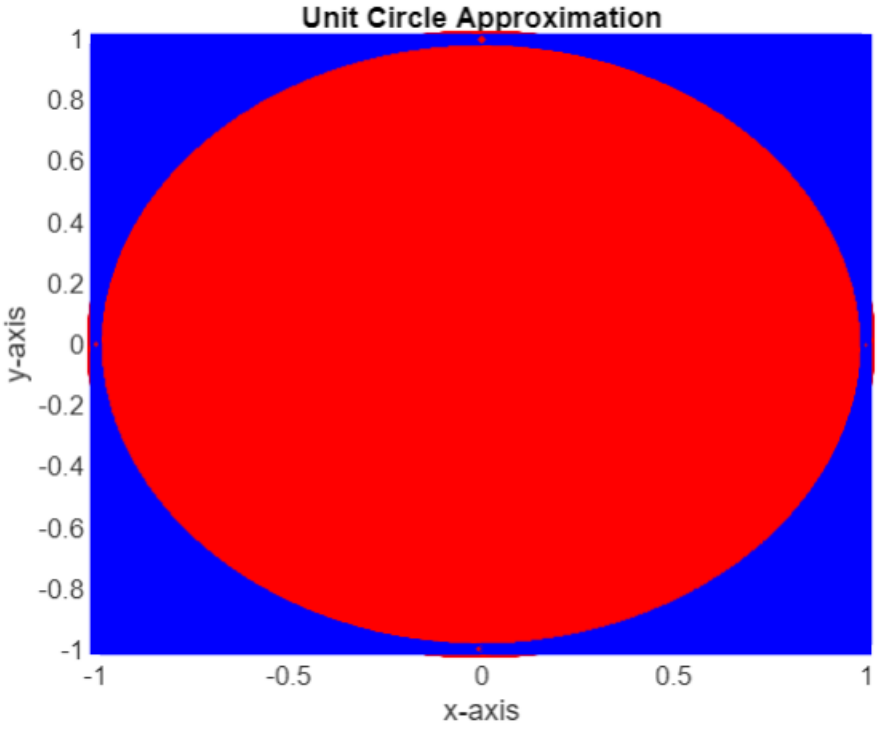
```
ans =
```


Unit Circle Approximation
y-axis
x-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Unit Circle Approximation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y-axis
-1
-0.5
0
0.5
1
x-axis


Unit Circle Approximation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y-axis
-1
-0.5
0
0.5
1
x-axis


Unit Circle Approximation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y-axis
-1
-0.5
0
0.5
1
x-axis


Unit Circle Approximation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
y-axis
-1
-0.5
0
0.5
1
x-axis

	NumPoints	Pi_Estimate	Percent_Error
1	100	3.2800	4.4056
2	1000	3.1640	0.7132
3	10000	3.1560	0.4586
4	100000	3.1377	0.1245
5	1000000	3.1441	0.0801
6	10000000	3.1420	0.0127

MLA - 009

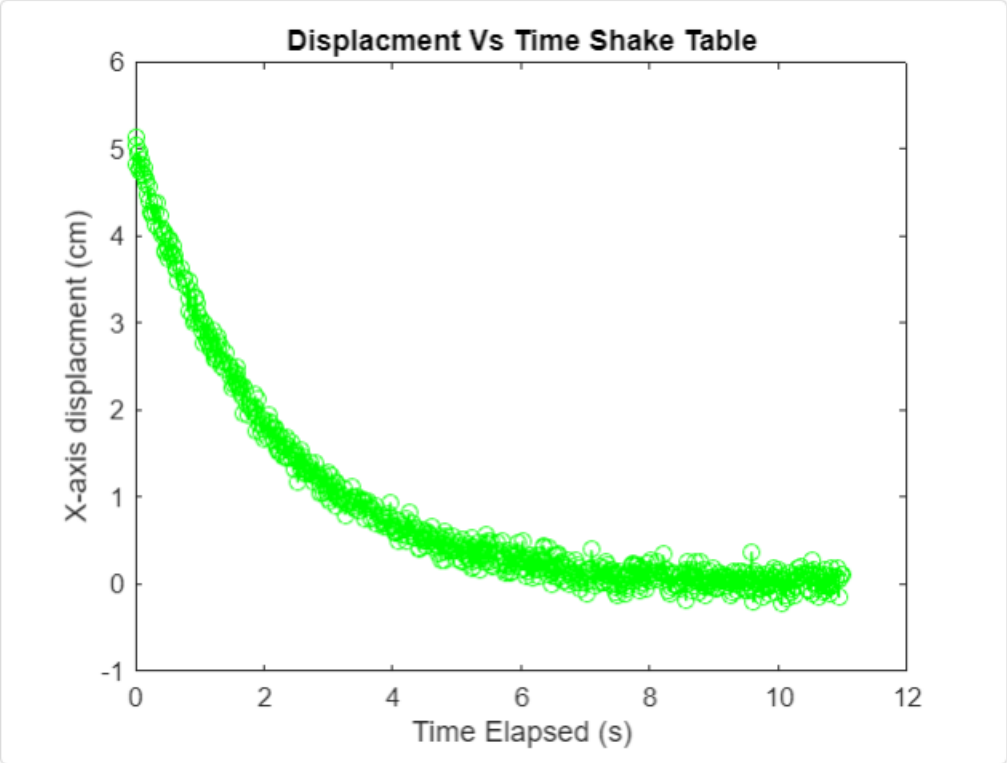
```
%Reset from exp 1
clc;
clear;
close all;
%load needed
load("shakedata.mat")
%Plot
plot(x,t,Color="g",Marker="o")
%% ShakeTable function
function [xOut] = ShakeTable(tIn, t, x)
% INPUT:
%   tIn: times for which x displacements should be returned (s) (vector)
%   t: data times (s) (vector)
%   x: data displacments (cm) (vector)
% OUTPUT:
%   xOut: displacements for tIn (cm) (vector)


tIn = (0:0.01:11);
isdatapoint = (tIn == t)

plot(x,tIn)
end


tIn = [0.01 5.39 9.04 10.21];
xOut = ShakeTable(t, x, tIn);


% 5. Plot
plot(tIn, xOut, 'r',Marker='diamond',LineStyle='none')
```



Code for the ShakeTable Function was non-functional

USF - 002

```
%Reset from exp 1 and 2
clc;
clear;
close all;
function y = BeamCantPoint(E,I,a,W,x)
% Specify the System Parameters
E = 2*10^7;          % Young's Modulus [psi]
I = 0.163;           % Moment of inertia [in^4]
L = 12;              % Beam length [in]
W = 1750;             % Point load magnitude [lb]
x = 0:0.01:L;        % Distance along beam [in]
a = 3; %Predefine A will be reset later
function1 = ((-1.*W.*x.^2)./(6*E*I)).*(-x+(3*a));
function2 = ((-1.*W.*a.^2)./(6*E*I)).*(-a+(3.*x));
y = (heaviside(x - 0) - heaviside(x - a)) .* function1 + (heaviside(x - a) -
heaviside(x - 12)) .* function2;
hold on
plot(x,y)
plot(a,0,'marker','V','Color','k',MarkerSize=8)
```

```
xlabel("location along beam (m)")
ylabel('Deflection(m)')
title ('Deflection of beam under load')
hold off

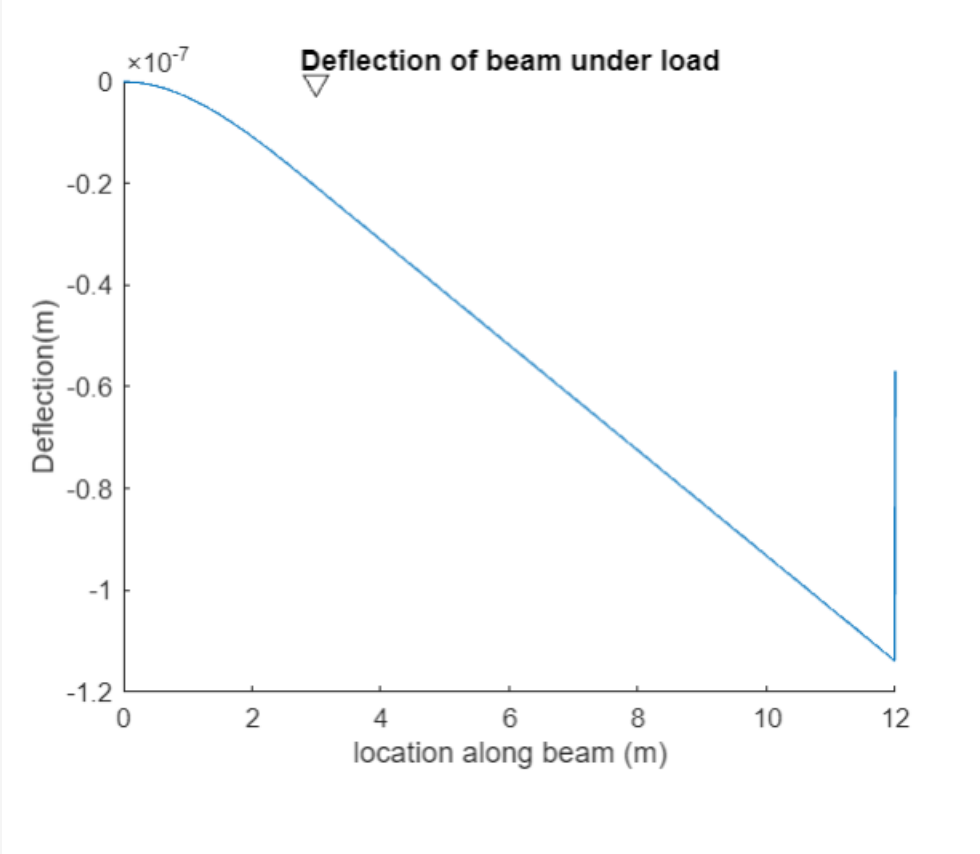

end
%Call function 1
a = 3;
y = BeamCantPoint;
%function with a as a vector
function y = BeamCantPoint2(E,I,A,W,x)
% Specify the System Parameters
E = 2*10^7;         % Young's Modulus [psi]
I = 0.163;          % Moment of inertia [in^4]
L = 12;             % Beam length [in]
W = 750;            % Point load magnitude [lb]
x = 0:0.01:L;       % Distance along beam [in]
A = 0:2:L; %Predefine A will be reset later
function1 = ((-1.*W.*x.^2)./(6*E*I)).*(-x+(3.*A))
function2 = ((-1.*W.*A.^2)./(6*E*I)).*(-A+(3.*x))
y = (heaviside(x - 0) - heaviside(x - A)) .* function1 + (heaviside(x - A) -
heaviside(x - L)) .* function2;
hold on
plot(x,y)
plot(A,0,'marker','V','Color','k',MarkerSize=8)
hold off


end
%Call function 2 and code breaks
L = 12;
A = 0:2:L;
n = A(1)
b=1
while (n<L)
    y = BeamCantPoint2
    b = b+1
    n = A(b)
end
%{
1. How does the deflection change as a increases?
Deflection will increase


2. For any value of a, where does the maximum deflection occur?
Max occurs at L as it is the farthest from the measuring point and as such
has the greatest distance to allow for deflection.
```

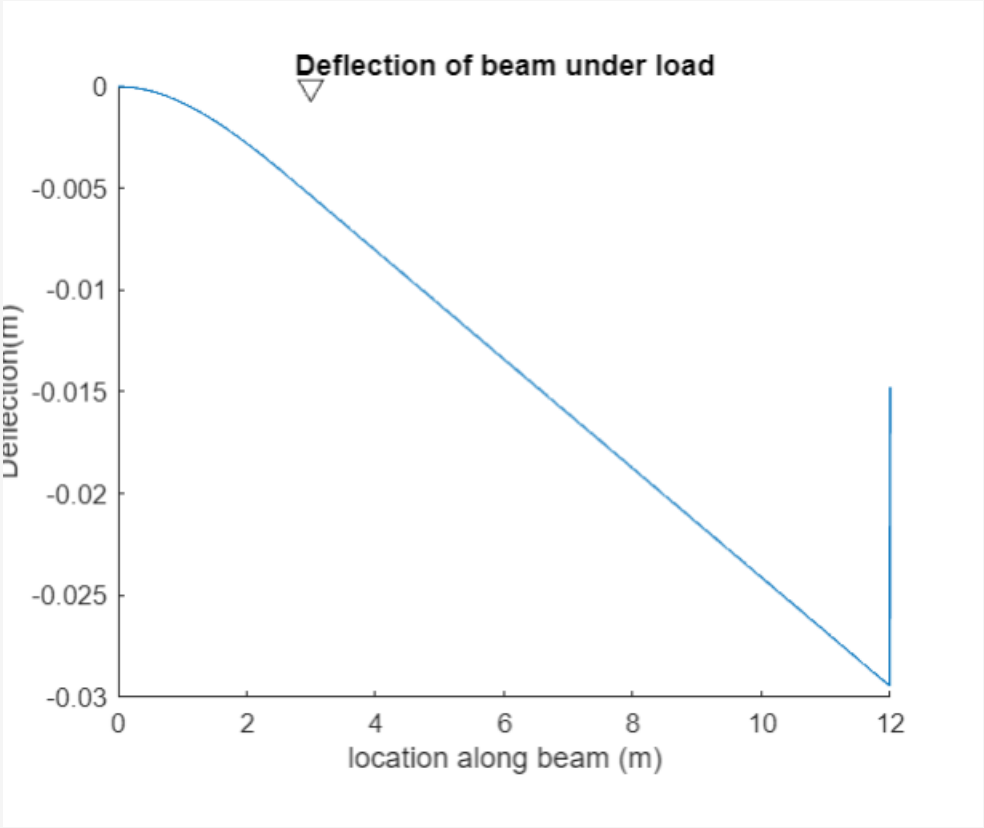
3. Briefly describe the difference between the overall shape of the beam deflection from the
wall to the point load location vs. from the point load location to the end of the beam.
Up to the point load the beam deflects exponentialy then past the load the curve is linear

4. Young's Modulus (E) is a material property representing the beam's stiffness. A higher
E represents a stiffer beam. Play around with a range of E values (of your choosing) and
hold the other parameters constant. How does the deflection change as you increase/decrease E? Provide a plot to support your explanation.
E has an inverse relationship
to deflection as stiffness is more or less a number quantifying ressistance to deflection. Plot shown had an increased E

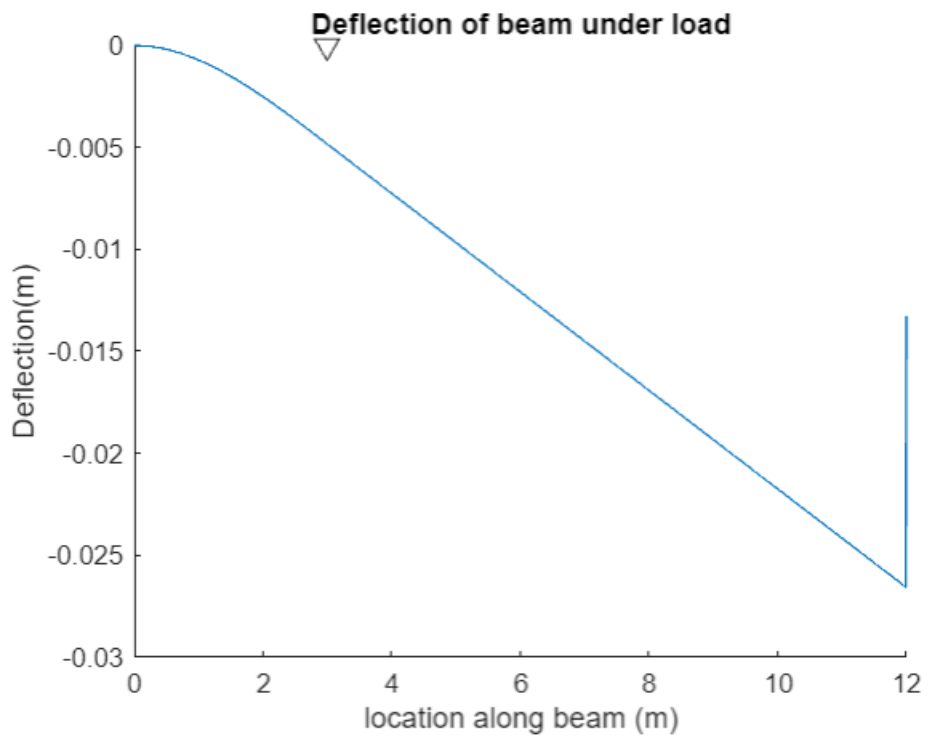


5. The area moment of inertia (I) is a geometric property representing the distribution of

matter relative to an axis. A high moment of inertia means the mass is concentrated away
from the centroid (think of an I-beam). Play around with a range of I values (of your
choosing) and hold the other parameters constant. How does the deflection change as you
increase/decrease I? Provide a plot to support your explanation.
I has an inverse relationship with the deflection similar to how E has an inverse relationship with deflection.Plot shown had a decreased I



6. Play around with a range of W values (of your choosing) and hold the other parameters
constant. How does the deflection change as you increase/decrease W? Provide a plot to
support your explanation.
W has a direct relationship to the deflection (ie it icreases deflection as W increases). Plot shown had an increased W



7. Are all of your observations consistent with elementary physics and/or common sense?
Yes all observations are consistant with what was taught in statics and past physics classes.

%}