%This is a MATLAB program that analyzes a conical beam using finite element

% method to compute stress and displacement along the beam

% Upon analysis you will notice that length, and radius at both ends can be

% customized. You can also add an unlimited number of forces, giving custom

% values for x, and the nodes for the finite element method will move

% along the beam

% Brian Loughran

% clc;

& clear all;

& close all;

% inputs for radius a, radius b, legnth

a=input('Enter a value for radius a (m): ');

b=input('Enter a value for radius b (m): ');

L=input('Enter a value for legnth L (m): ');

% constraints

fprintf('\n Left side constrained (a)? Enter 1 for yes, 0 for no \n');

LeftConstraint=input('Enter a value: ');

fprintf('\n Right side constrained (b)? Enter 1 for yes, 0 for no \n');

RightConstraint=input('Enter a value: ');

% initialize the force matrix and the displacement matrix

Sections=input('\n Enter a value for the number of sections: ');

L\_Section(Sections)=0;

Nodes=Sections+1;

X\_Node(1)=0;

X\_Node(Nodes)=L;

MidpointSection(Sections)=0;

for n=1:1:Sections;

L\_Section(n)=L/Sections;

X\_Node(n+1)=X\_Node(n)+(L/Sections);

MidpointSection(n)=((X\_Node(n+1)-X\_Node(n))/2)+X\_Node(n);

end

Force\_Matrix(Nodes,1)=0;

Displacement\_Matrix(Nodes)=0;

% enter forces into the force matrix and move nodes

NumberOfLoads=input('Enter the number of loads to be applied: ');

P(NumberOfLoads)=0;

ForceOnNode=0;

diffnodepoint(Sections)=0;

for n=1:1:NumberOfLoads;

fprintf('\n For Load % d:\n',n);

P(n)=input('Enter the force of the load (N): ');

X\_Indicator=input('\n Enter a distance x from the left hand \n side of the beam at which to apply this force (m): ');

for i=2:1:Nodes;

SectionSize=L/Sections;

NodeCoordinate=SectionSize\*(i-1);

if X\_Indicator==NodeCoordinate;

ForceOnNode=1;

NodeIndicator=i;

end

end

if ForceOnNode==1;

Force\_Matrix(NodeIndicator)=P(n);

else

bestdiff=99999999999999999;

for i=2:1:Sections;

diffnodepoint(i)=abs(X\_Indicator-X\_Node(i));

if diffnodepoint(i)<bestdiff;

bestdiff=diffnodepoint(i);

CloseNode=i;

diff=X\_Indicator-X\_Node(i);

end

end

X\_Node(CloseNode)=X\_Node(CloseNode)+diff;

L\_Section(CloseNode-1)=L\_Section(CloseNode-1)+diff;

L\_Section(CloseNode)=L\_Section(CloseNode)-diff;

Force\_Matrix(CloseNode)=P(n);

end

ForceOnNode=0;

end

% Create Matrix for Young's Modulus

fprintf('Is Youngs Modulus variable along the beam?\n');

Evar=input('Enter 1 for yes, 0 for no: ');

if Evar==0;

E=input('Enter a value for the Youngs Modulus (N/(m^2)): ');

YoungsModulus(Sections)=0;

for n=1:1:Sections;

YoungsModulus(n)=E;

end

elseif Evar==1;

YoungsModulus(Sections)=0;

C=70000000000;

Beta=(-.773);

for n=1:1:Sections;

YoungsModulus(n)=C\*(2.781^(Beta\*MidpointSection(n)));

end

end

% find the radius at all nodes and elements

Radius(1)=a;

Radius(Nodes)=b;

for n=2:1:Sections;

diffab=a-b;

Radius(n)=a-((X\_Node (n)/L)\*diffab);

end

AvgRadius(Sections)=0;

for n=1:1:Sections;

AvgRadius(n)=(Radius(n)+Radius(n+1))/2;

end

% filling the elemental stiffness matricies

K\_Global(Nodes,Nodes)=0;

K\_Elemental(Nodes,2,2)=0;

AE\_L(Sections)=0;

for n=1:1:Sections;

var(1)=((3.14\*(AvgRadius (n)^2)\*YoungsModulus(n))/(L\_Section(n)));

AE\_L(n)=var(1);

K\_Elemental(n,1,1)=AE\_L(n);

K\_Elemental(n,1,2)=(AE\_L(n))\*(-1);

K\_Elemental(n,2,1)=(AE\_L(n))\*(-1);

K\_Elemental(n,2,2)=AE\_L(n);

end

% filling and adjusting the global stiffness matrix

for n=1:1:Sections

K\_Global(n,n+1)=K\_Elemental(n,1,2);

K\_Global(n+1,n)=K\_Elemental(n,2,1);

end

if LeftConstraint==0;

K\_Global(1,1)=K\_Elemental(1,1,1);

elseif LeftConstraint==1;

K\_Global(1,1)=1;

K\_Global(2,1)=0;

K\_Global(1,2)=0;

end

for n=2:1:Sections;

K\_Global(n,n)=K\_Elemental(n-1,1,1)+K\_Elemental(n,1,1);

end

if RightConstraint==0;

K\_Global(Nodes,Nodes)=K\_Elemental(Sections,2,2);

elseif RightConstraint==1;

K\_Global(Nodes,Nodes)=1;

K\_Global(Nodes,Sections)=0;

K\_Global(Sections,Nodes)=0;

end

% Solving the system of equations

Displacement\_Matrix=linsolve(K\_Global,Force\_Matrix);

% Solving for stresses in their respective sections

Stress(Sections)=0;

for n=1:1:Sections;

Stress(n)=(YoungsModulus(n)\*(Displacement\_Matrix(n+1)-Displacement\_Matrix(n)));

end

% Plot the data

% Plot\_C

% plot(X\_Node,Displacement\_Matrix);

% xlabel('x from Left Hand Side of the Beam ');

% ylabel('Displacement');

% title('Displacement vs. x');

% hold on;

% Plot\_D

% scatter(Sections,Displacement\_Matrix(Closenode));

% Use .65001 for x

% xlabel('Sections');

% ylabel('Displacement at x=.65');

% title('Displacement at x=.65 vs. Sections');

% hold on;

% Plot\_E

% plot(MidpointSection,Stress);

% xlabel('x from Left Hand Side of the Beam ');

% ylabel('Stress');

% title('Stress vs. x');

% hold on;

% Plot\_F

plot(X\_Node,Displacement\_Matrix);

xlabel('x from Left Hand Side of the Beam ');

ylabel('Displacement');

title('Displacement vs. x (With Varying Youngs Modulus)');

hold on;

% Plot\_G

% plot(MidpointSection,Stress);

% xlabel('x from Left Hand Side of the Beam ');

% ylabel('Stress');

% title('Stress vs. x (With Varying Youngs Modulus)');

% hold on;