function Analyze_Truss

% Main code for solving 2D Truss problems using stiffness method. The code reads the geometry of the structure (nodal positions and connectivity),
% elastic modulus, member cross-section area, boundary conditions and 
% assembles the stiffness matrix, nodal force and displacement vectors. It subsequently performs the static condensation and finds the nodal 
% displacements, reaction forces and internal forces. It also plots the 
% deformed and undeformed trusses.
% cleaning the screen and closing all the figures (new start)
clc; close all;

% Read the truss input file using t
[nodes, elms, E, A, bcs, loads]=Input_Truss;
Nel = size(elms,1);  % Nel : number of truss elements
Nnodes = size(nodes,1);  % Nnode : number of nodal points

% decide degrees of freedom + Initiate Matrices
% Note: Degrees of freedom corresponding to node "i" are :
% [2*(i-1)+1 2*(i-1)+2]

% alldofs: total number of degrees of freedom = 2*Nnodes
alldofs = 1:2*Nnodes;

% K : global stiffness matrix
K = zeros(2*Nnodes,2*Nnodes);

% u : global displacement vector
u = zeros(2*Nnodes,1);

% f : global force vector
f = zeros(2*Nnodes,1);

% Create a list of all boundary conditions and relevant displacements
% dfspec : a vector containing specified dfs (Boundary Conditions:BC)
dfspec = [ ];
for ii = 1:size(bcs,1)
    % thisdof: the degree of freedom that pertains to ii BC
    thisdof = 2*(bcs(ii,1)-1)+bcs(ii,2);
    % adding thisdof to the list of all boundary conditions (dofspec)
    dofspec = [dofspec thisdof];
    % adding the value of the displacement in the BCs to the displacment vector
    u(thisdof)=bcs(ii,3);
end

% doffree : the list of all degree of freedom that are free to move
doffree = alldofs;
% Delete specified dfs from all dfs
doffree(dofspec) = [ ];

% Create the global load vector
for ii = 1: size(loads,1)
    % forcedof : the degree of freedom at which the force is implemented
    forcedof = 2*(loads(ii,1)-1)+loads(ii,2);
    % adding each force to the force vector
    f(forcedof)=loads(ii,3);
end

% Initialize the global stiffness matrix
for iel = 1:Nel
    % elnodes : the nodes at the either side of element iel
    elnodes = elms(iel, 1:2);
    % nodexy : coordinates of the nodes of element iel
    nodexy = ndces(elnodes,:);
    % Get the element global stiffness matrix for the current element
    [Kel] = TrussElement2D(nodexy, E(iel), A(iel));
% degrees of freedom for node 1 of the element
edofs = 2*(elnodes(1)-1)+1:2*elnodes(1);
% degrees of freedom for node 2 of the element
edofs = [eldofs 2*(elnodes(2)-1)+1:2*elnodes(2)];
% Assemble the element stiffness matrix into global stiffness matrix K
K(eldofs,eldofs) = K(eldofs,eldofs) + Kel;
end

% Solve the matrix equation by considering the boundary conditions
% Solving for the displacement unknowns
u(dofree) = K(dofree,dofree)\(K(dofree,dofree)^{-1}K(dofree,dofspec)\)\u(dofspec);
% Solving for the support forces
f(dofspec) = K(dofspec,:)*u;

% calculate the internal force in each member of the truss
F_int = zeros(Nel,1);
for iel = 1:Nel
    % elnodes : the nodes at the either side of element iel
elnodes = elms(iel,1:2);
    % nodexy : coordinates of the nodes of element iel
    nodexy = nodes(elnodes,:);
    % displacement in degrees of freedom for node 1 of the element
    u_el = u(2*(elnodes(1)-1)+1:2*elnodes(1),:);
    % displacement in degrees of freedom for node 2 of the element
    u_el = [u_el; u(2*(elnodes(2)-1)+1:2*elnodes(2),:)];
    % calculate the force in each element
    F_e = TrussForce2D(nodexy, E(iel), A(iel), u_el);
    F_int(iel) = F_e;
end

% display forces and displacements
disp(['Displacement : '])
disp(['Reactions Forces : '])
disp(['Internal Forces : '])
F_int

% Plot the undeformed shape of the truss structure
figure(1); hold on;
title('Deformed and Undeformed Truss')
plot(nodes(:,1),nodes(:,2),'k-') % plot nodes in undeformed truss
hold on; axis equal;
for iel = 1:Nel
    elnodes = elms(iel,1:2);
    nodexy = nodes(elnodes,:);
    plot(nodexy(:,1),nodexy(:,2),'k--') % plot elements in undeformed truss
end
hold on;

% Plot the deformed shape shape of the structure
% Magnification: magnifies deformation such that it can be seen in plots
Magnification = 5000;
xydisp = [u(1:2:end) u(2:2:end)];
nodesnew = nodes + Magnification*xydisp;
% plotting the nodes in undeformed truss
plot(nodesnew(:,1),nodesnew(:,2),'o','MarkerEdgeColor','b',... 'MarkerFaceColor','r','MarkerSize',10)
hold on; axis equal;
for iel = 1:Nel
    elnodes = elms(iel,1:2);
    nodexy = nodesnew(elnodes,:);
    plot(nodexy(:,1),nodexy(:,2),'b-') % plot elements in deformed truss
end
end
function [Kel] = TrussElement2D(nodeXy, E, A)
% This function must return a 4*4 element stiffness matrix.
% This matrix must be in global coordinate system. The inputs are:
% nodeXy : [ x1 y1; x2 y2]; E : Young's Modulus; A : Cross Section Area.

% DeltaXY: DeltaX and DeltaY of the two joint of the element
DeltaXY = (nodeXy(2,1)-nodeXy(1,1)) (nodeXy(2,2)-nodeXy(1,2));
% L : Length of the element
L = norm(DeltaXY);
% CosSin(1) = cos(theta); and CosSin(2) = sin (theta);
CosSin = DeltaXY / L;

% Kel_axial : local stiffness matrix
Kel_axial = E*A/L*[1 -1; -1 1];

% Tmatrix: Rotation matrix that transforms from local to global coordinates
Tmatrix = [CosSin(1) CosSin(2) 0 0; 0 0 CosSin(1) CosSin(2)];
% Kel : Global stiffness matrix of the element
Kel = Tmatrix'*Kel_axial*Tmatrix;
end
function [Fel]=TrussForce2D(nodexy, E, A, u)
% This function returns the internal force for the truss member

% DeltaXY: DeltaX and DeltaY of the two joint of the element
DeltaXY = [(nodexy(2,1)-nodexy(1,1)) (nodexy(2,2)-nodexy(1,2))];
% L : Length of the element
L = norm(DeltaXY);
% CosSin(1) = cos(theta); and CosSin(2) = sin(theta);
CosSin = DeltaXY / L;

% Tmatrix : Rotation matrix that transforms from local to global coordinates
Tmatrix = [-CosSin(1) -CosSin(2) CosSin(1) CosSin(2)];
% Fel : Internal force in member 1
Fel = E*A*L* Tmatrix* u;
end
function [nodes, elems, E, A, bcs, loads]=Input_TruSS
% Example of a determinate truss with five elements and 4 nodes.
% nodes matrix contains x and y coordinate of the truss nodes
% First Column : x coordinate; Second Column : Y coordinate
nodes=[0.0 0.0; ... 2.0 0.0; ... 2.0 2.0; ... 0.0 2.0; ];
% elements matrix contains the connectivity of the nodes
% First column : node i; Second column : node j; i and j are connected via an
% element
elems=[1 2; ... 2 3; ... 3 4; ... 4 1; ... 1 3];
% E matrix contains elastic modulus of each element
E = [2e11; 2e11; 2e11; 2e11; 2e11];
% A matrix contains cross-section area of each element
A = [1e-2; 1e-2; 1e-2; 1e-2; 1e-2];
% bcs matrix contains the specified boundary conditions
% First column : node, Second Column : DOF (1 for X and 2 for Y)
% Third column : the displacement in that DOF
bcs = [1 1 0; ... 1 2 0; ... 2 2 0];
% loads matrix contains nodal forces
% First Column : Nodes; Second Column : Forces
% Third Column : Forces
loads = [3 1 10^-4];
end