MATLAB-Code Demonstrations

Subroutine `PlaneTrussElementStiffness.m`

```
function y = PlaneTrussElementStiffness(E,A,node_i,node_j)
%-compute element stiffnes matrix-----------------------------------
L = sqrt((node_j(1) - node_i(1))^2 + (node_j(2) - node_i(2))^2);
if node_j(1) == node_i(1)
    if node_i(2) > node_j(2)
        theta = -pi/2;
    elseif node_i(2) < node_j(2)
        theta = pi/2;
    end
else
    theta = atan((node_j(2) - node_i(2))/(node_j(1) - node_i(1)));
end
C = cos(theta);
S = sin(theta);
y = E*A/L*[ C*C  C*S -C*C -C*S ;
            C*S  S*S -C*S -S*S ;
           -C*C -C*S  C*C  C*S ;
           -C*S -S*S  C*S  S*S ];
```

Hauptdatei `PlaneTruss.m`

```
% MAIN PROGRAM
%--------------------------------------------------------------------
% FEM for simple plane struss structures
%--------------------------------------------------------------------
clear all
close all
%-read general properties-------------------------------------------
num_ele   = input('give the total number of elements. num_ele = ');
num_nodes = input('give the total number of nodes.  num_nodes = ');
disp(' ');
%-define nodes, coordinates and stiffnes (E and A) for elements------
for i = 1:num_ele
    fprintf('element no.%g\n',i);
    connectivity(1,i) = input('  global node for 1st local node = ');
    connectivity(2,i) = input('  global node for 2nd local node = ');
    elementdata(i,:)  = input('  element data           [ E A ] = ');
```

```
    disp(' ');
end
for i = 1:num_nodes
    fprintf('node no.%g\n',i);
    X(:,i) = input('  global node coordinates [ x y ] = ')';
    disp(' ');
end
%-compute number of equations-------------------------------------
disp('give the total number of displacement boundary conditions')
num_bc = input('                                          num_bc = ');
num_eq = num_nodes*2 - num_bc;
%-assemble matrix-------------------------------------------------
assembleid = -ones(2,num_nodes);
disp('    input of global dofs with zero boundary displacements')
disp('                                    for i from 1 to num_bc')
for i = 1:num_bc
    node = ...
    input('        global node of constrained degree of freedom = ');
    dof = ...
    input('          degree of freedom to constrained (1=x,2=y) = ');
    assembleid(dof,node) = 0;
end
idof = 0;
for i = 1:num_nodes
    for j = 1:2
        if assembleid(j,i) ~= 0
            idof = idof + 1;
            assembleid(j,i) = idof;
        end
    end
end
%-compute global stiffnes matrix----------------------------------
K = zeros(num_eq,num_eq);
for i = 1:num_ele
    k = PlaneTrussElementStiffness(elementdata(i,1), ...
    elementdata(i,2),X(:,connectivity(1,i)),X(:,connectivity(2,i)));
    I = assembleid(:,connectivity(1,i));
    J = assembleid(:,connectivity(2,i));
    IJ = find(( [ I J ] ) ~= 0);
    K = PlaneTrussAssemble(K,k,I,J,IJ);
end
disp(' ');
%-global force vector---------------------------------------------
```

```
R = zeros(num_eq,1);
disp('give the total number of nodes with non-zero load vectors')
num_loads = ...
          input('                                        num_loads = ');
disp('             input of load vectors for i from 1 to num_loads')
for i = 1:num_loads
    lbc = input('  no. of global node with non-zero load vector = ');
    if assembleid(2*lbc-1) ~= 0 & assembleid(2*lbc) ~= 0
        R(assembleid(2*lbc-1)) = ...
        input('                    force in global x-direction = ');
        R(assembleid(2*lbc))   = ...
        input('                    force in global y-direction = ');
    elseif assembleid(2*lbc-1) ~= 0
        R(assembleid(2*lbc-1)) = ...
        input('                    force in global x-direction = ');
    elseif assembleid(2*lbc) ~= 0
        R(assembleid(2*lbc))   = ...
        input('                    force in global y-direction = ');
    elseif assembleid(2*lbc-1) == 0 & assembleid(2*lbc) == 0
        disp('all degrees of freedom in this node are constrained!');
        pause(1);
    end
end
%-displacement computed by Gaussian elimination----------------------
u = K\R;
%-print the data-----------------------------------------------------
disp(' ');
disp('element connectivity table');
disp(' ');
disp('  ============================')
disp('  |  e  |  Node i  |  Node j  |')
disp('  ============================')
for i = 1:num_ele
    fprintf('  |  %g  |     %g    |     %g    |', ...
    i,connectivity(1,i),connectivity(2,i))
    disp(' ');
end
disp('  ============================')
disp(' ');disp('global stiffness matrix'); K
disp(' ');disp('global force vector    '); R
disp(' ');disp('displacement vector    '); u
```

Subroutine `PlaneTrussAssemble.m`

```
function y = PlaneTrussAssemble(K,k,I,J,IJ)
%-------------------------------------------------------------------
%Input:  element stiffnes matrix
%Output: updated global stiffnes matrix
%-------------------------------------------------------------------
if size(find([ I' J' ] ~= 0),2)     == 1
    K = Assemble1(K,k,I,J,IJ);
elseif size(find([ I' J' ] ~= 0),2) == 2
    K = Assemble2(K,k,I,J,IJ);
elseif size(find([ I' J' ] ~= 0),2) == 3
    K = Assemble3(K,k,I,J,IJ);
elseif size(find([ I' J' ] ~= 0),2) == 4
    K = Assemble4(K,k,I,J,IJ);
end
y = K;
```

`Assemble1.m`

```
function y = Assemble1(K,k,I,J,IJ)
%-------------------------------------------------------------------
% Assembly routine for element with one active dof
i = find([ I' J' ] ~= 0);
switch i
    case 1,  a = I(1);
    case 2,  a = I(2);
    case 3,  a = J(1);
    case 4,  a = J(2);
end
K(a,a) = K(a,a) + k(IJ,IJ);
y = K;
```

`Assemble2.m`

```
function y = Assemble2(K,k,I,J,IJ)
%-------------------------------------------------------------------
% Assembly routine for element with two active dofs
if size(find(I ~= 0),1) == 2
    a = I(1);
```

```
    b = I(2);
elseif size(find(J ~= 0),1) == 2
    a = J(1);
    b = J(2);
elseif (size(find(I ~= 0),1) == 1) & (size(find(J ~= 0),1) == 1)
    a = I(find(I ~= 0));
    b = J(find(J ~= 0));
end
i = IJ(1);
j = IJ(2);
K(a,a) = K(a,a) + k(i,i);
K(a,b) = K(a,b) + k(i,j);
K(b,a) = K(b,a) + k(j,i);
K(b,b) = K(b,b) + k(j,j);
y = K;
```

Assemble3.m

```
function y = Assemble3(K,k,I,J,IJ)
%--------------------------------------------------------------------
% Assembly routine for element with three active dofs
if size(find(I ~= 0),1) == 1
    a = I(find(I ~= 0));
    b = J(1);
    c = J(2);
elseif size(find(J ~= 0),1) == 1
    a = I(1);
    b = I(2);
    c = J(find(J ~= 0));
end
K(a,a) = K(a,a) + k(IJ(1),IJ(1));
K(a,b) = K(a,b) + k(IJ(1),IJ(2));
K(a,c) = K(a,c) + k(IJ(1),IJ(3));
K(b,a) = K(b,a) + k(IJ(2),IJ(1));
K(b,b) = K(b,b) + k(IJ(2),IJ(2));
K(b,c) = K(b,c) + k(IJ(2),IJ(3));
K(c,a) = K(c,a) + k(IJ(3),IJ(1));
K(c,b) = K(c,b) + k(IJ(3),IJ(2));
K(c,c) = K(c,c) + k(IJ(3),IJ(3));
y = K;
```

Assemble4.m

```
function y = Assemble4(K,k,I,J,IJ)
%-------------------------------------------------------------------------
% Assembly routine for element with four active dofs
a = I(1);
b = I(2);
c = J(1);
d = J(2);
K(a,a) = K(a,a) + k(1,1);
K(a,b) = K(a,b) + k(1,2);
K(a,c) = K(a,c) + k(1,3);
K(a,d) = K(a,d) + k(1,4);
K(b,a) = K(b,a) + k(2,1);
K(b,b) = K(b,b) + k(2,2);
K(b,c) = K(b,c) + k(2,3);
K(b,d) = K(b,d) + k(2,4);
K(c,a) = K(c,a) + k(3,1);
K(c,b) = K(c,b) + k(3,2);
K(c,c) = K(c,c) + k(3,3);
K(c,d) = K(c,d) + k(3,4);
K(d,a) = K(d,a) + k(4,1);
K(d,b) = K(d,b) + k(4,2);
K(d,c) = K(d,c) + k(4,3);
K(d,d) = K(d,d) + k(4,4);
y = K;
```