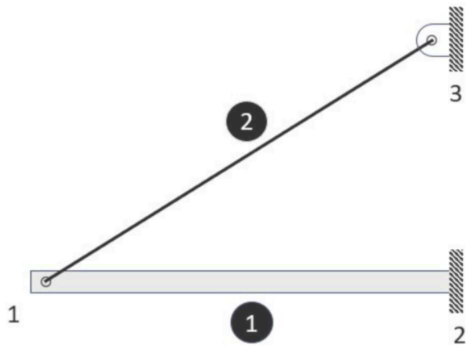
In the mesh shown below, element 1 is a beam element about which we will learn later. It has 3 dof's per node and two nodes. The three dof's are two translations and one rotation. On the other hand, element 2 is a truss element with two dof's per node (two translations) and two nodes. The stiffness matrices in the global coordinate system are given as `K1,K2`. The element connectivities are

element 1 1,2

element 2 1,3

Assemble the global stiffness `Kglobal`. Form the destination arrays `dest1,dest2`.



```
% problem 2b: assembly, nodes with different dofs from different
elements
nelem=2;
nnode=3;
Kglobal=zeros(8 ,8);
%for element 1
K1=250000*[10 0 0 -10 0 0;0 0.835 5 0 -0.0835 5;0 5 400 0 -5 200;...
   -10 0 0 10 0 0;0 -0.0835 -5 0 0.0835 -5;0 5 200 0 -5 400];
dest1=[1 2 3 4 5 6];
%for element 1
k=3.15e04;
theta=pi/4;
c=cos(theta);s=sin(theta);
K2=k*[c*c c*s -c*c -c*s;c*s s*s -c*s -s*s;c*s s*s c*c c*s;-c*s -s*s c*s
s*s];
dest2=[1  2 7 8];
for ii=1:nelem
   if ii == 1
       Klocal=K1;
       dest=dest1;
   end
   if ii == 2
       Klocal=K2;
```

```
        dest=dest2;
    end
%Assemble Kglobal here
c=dest;
l=Klocal;
Kglobal(c,c)=Kglobal(c,c)+l ;
end
```

```
function Ke = axisymmetric_quad4_stiffness(R_nodes, z_nodes, E, nu)
% R_nodes, z_nodes: 4x1 vectors [node1..node4]
% DOF order per node: [u (radial), w (axial)]
% returns Ke (8x8)

% material D (4x4) as given
coef = E / ((1+nu)*(1-2*nu));
D = coef * [1-nu, nu,   nu,   0;
            nu,   1-nu, nu,   0;
            nu,   nu,   1-nu, 0;
            0,    0,    0,    (1-2*nu)/2];

% initialize
Ke = zeros(8,8);
gp = 1/sqrt(3);
rGP = [-gp, gp];
sGP = [-gp, gp];
w = [1, 1];

for i = 1:2
 for j = 1:2
   r = rGP(i); s = sGP(j);
   weight = w(i)*w(j);

   % shape functions and their parametric derivatives (dr, ds)
   N = 0.25 * [(1-r)*(1-s);
               (1+r)*(1-s);
               (1+r)*(1+s);
               (1-r)*(1+s)];
   dNdr = 0.25 * [ -(1-s);
                    (1-s);
```

```
                     (1+s);
                    -(1+s) ];
   dNds = 0.25 * [ -(1-r);
                   -(1+r);
                    (1+r);
                    (1-r) ];

   % Jacobian J = [dR/dr dR/ds; dz/dr dz/ds]
   dRdr = dNdr' * R_nodes;
   dRds = dNds' * R_nodes;
   dzdr = dNdr' * z_nodes;
   dzds = dNds' * z_nodes;
   J = [dRdr, dRds; dzdr, dzds];
   detJ = det(J);
   invJ = inv(J);

   % physical derivatives dNi/dR and dNi/dz
   dN = invJ * [dNdr'; dNds'];   % 2 x 4 matrix
   dN_dR = dN(1,:)';   % 4x1
   dN_dz = dN(2,:)';   % 4x1

   % radial coordinate at gauss point
   Rgp = N' * R_nodes;   % scalar

   % assemble B (4x8)
   B = zeros(4,8);
   for a = 1:4
     col_u = 2*(a-1) + 1;  % radial DOF index
     col_w = col_u + 1;    % axial DOF index
     B(1, col_u) = dN_dR(a);
     B(2, col_w) = dN_dz(a);
     B(3, col_u) = dN_dz(a);
     B(3, col_w) = dN_dR(a);
     B(4, col_u) = N(a) / Rgp;
   end

   % integrand: B' * D * B * (2*pi*Rgp) * detJ * weight
   Ke = Ke + (B' * D * B) * (2*pi * Rgp) * detJ * weight;
 end
end

end
```