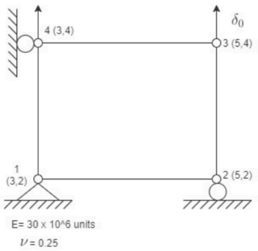
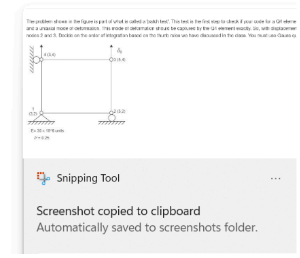
Stresses in a Q4 element

3 solutions submitted (max: 4) | View my solutions

Open Item in MATLAB Online

The problem shown in the figure is part of what is called a 'patch test'. This test is the first step to check if your code for a Q4 element is working. Assuming plane stress, you have a problem which leads to an uniform state of stress and a uniaxial mode of deformation. This mode of deformation should be captured by the Q4 element exactly. So, with displacements of $\delta_0 = 1 \times 10^{-6}$ applied at nodes 3 and 4 in the y direction, determine the x displacements of nodes 2 and 3. Decide on the order of integration based on the thunb rules we have discussed in the class. You must use Gauss quadrature and not try to determine the element stiffness using direct integration.






```matlab
% Q4 element patch test
X=[3 5 5 3]; % global x coords
Y=[2 2 4 4]; % global y coords
conn=[1 2 3 4]; % element connectivity
E=30e6; % E
nu=0.25; % nu
delta=1e-6; % delta_0

% global stiffness and global force initialised
Kg=zeros(8,8);
Fg=zeros(8,1);

% 2x2 Gauss points and weights
gp=[-1/sqrt(3) 1/sqrt(3)];
weights=[1 1];

for ielem=1:size(conn,1) % loop over elements starts
    Ke=zeros(8,8); % local stiffness matrix
    Fe=zeros(8,1); % local force vector

    % Plane stress elasticity matrix
    D = (E/(1-nu^2))*[1 nu 0; nu 1 0; 0 0 (1-nu)/2];

    % Nodal coordinates for this element
    xe = X(conn(ielem,:));
```

```matlab
ye = Y(conn(ielem,:));

% 2x2 Gauss quadrature
for i=1:length(gp)
    for j=1:length(gp)
        xi = gp(i);
        eta = gp(j);
        wtx = weights(i);
        wty = weights(j);

        % Shape function derivatives wrt natural coords
        dN_dxi = 0.25*[-(1-eta) (1-eta) (1+eta) -(1+eta)];
        dN_deta= 0.25*[-(1-xi) -(1+xi) (1+xi) (1-xi)];

        % Jacobian matrix
        J = [dN_dxi; dN_deta]*[xe' ye'];
        detJ = det(J);
        invJ = inv(J);

        % Derivatives wrt x,y
        dN = invJ*[dN_dxi; dN_deta];

        % Construct B matrix (3x8)
        B=zeros(3,8);
        B(1,1:2:8)=dN(1,:);
        B(2,2:2:8)=dN(2,:);
        B(3,1:2:8)=dN(2,:);
        B(3,2:2:8)=dN(1,:);

        % Element stiffness
        Ke = Ke + B'*D*B*detJ*wtx*wty;
    end
end

% Assembly
dest = zeros(1,8);
for n=1:4
    dest(2*n-1)=2*conn(ielem,n)-1;
    dest(2*n)=2*conn(ielem,n);
end

for idof=1:8
    for jdof=1:8
```

```
            Kg(dest(idof),dest(jdof)) = Kg(dest(idof),dest(jdof)) +
Ke(idof,jdof);
        end
    end
end

% Apply boundary conditions (patch test setup)
U = zeros(8,1);

% Known displacements:
% Nodes: 1(3,2), 2(5,2), 3(5,4), 4(3,4)
% Fixed x at 1,4; fixed y at 1,2
% uy(3)=uy(4)=delta

bc = [1 0; 2 0; 7 delta; 8 delta]; % [dof, value]
% Apply boundary conditions to Kg and Fg
free = setdiff(1:8, bc(:,1));
for i=1:size(bc,1)
    dof = bc(i,1);
    val = bc(i,2);
    Fg = Fg - Kg(:,dof)*val;
    U(dof) = val;
end

% Solve for free DOFs
U(free) = Kg(free,free)\Fg(free);

% Final displacement vector
U




%% ===================== ELEMENT ROUTINE ==========================
function [Ke, stresslocal, strainlocal] = elemstiffQ4(x, y, disp,
stresslocal, strainlocal, D)

    Ke = zeros(8);
    ngauss = 4;
    gp = [-1/sqrt(3), -1/sqrt(3);
```

```
         1/sqrt(3), -1/sqrt(3);
         1/sqrt(3),  1/sqrt(3);
        -1/sqrt(3),  1/sqrt(3)];

   for k = 1:ngauss
       r = gp(k,1);
       s = gp(k,2);

       % Natural derivatives
       dNr = 0.25*[-(1-s),  (1-s),  (1+s), -(1+s)];
       dNs = 0.25*[-(1-r), -(1+r),  (1+r),  (1-r)];

       % Jacobian
       J = [dNr; dNs] * [x(:), y(:)];
       detJ = det(J);
       if detJ <= 0, error('Negative Jacobian'); end
       invJ = inv(J);

       % Cartesian derivatives
       dN = invJ * [dNr; dNs];
       dNx = dN(1,:);
       dNy = dN(2,:);

       % B matrix
       B = zeros(3,8);
       B(1,1:2:end) = dNx;
       B(2,2:2:end) = dNy;
       B(3,1:2:end) = dNy;
       B(3,2:2:end) = dNx;

       % Stiffness
       Ke = Ke + B' * D * B * detJ;

       % Strain & stress
       strainlocal(k,:) = (B * disp)';
       stresslocal(k,:) = (D * strainlocal(k,:)')';
   end
end
```