1. Using computer software, compute the stiffness matrix for the triangular element at right. Use E= 20GPa and ν= 0.25.

E = 20;

nu = 0.25;

q = 4;

C = [1 1; 2 1; 1 2];

D = E/(1-nu^2)\*[1 nu 0;

nu 1 0;

0 0 (1-nu)/2];

K = compute\_k(C, D, q);

function Q = quadrature(q)

quadrature\_1\_pts = [1/3 1/3 1/2];

quadrature\_3\_pts = [1/6 1/6 1/6

2/3 1/6 1/6

1/6 2/3 1/6];

quadrature\_4\_pts = [1/3 1/3 -9/32

3/5 1/5 25/96

1/5 3/5 25/96

1/5 1/5 25/96];

if q == 1

Q = quadrature\_1\_pts;

elseif q == 3

Q = quadrature\_3\_pts;

elseif q == 4

Q = quadrature\_4\_pts;

else

Q = 0;

end

end

function K = compute\_k(C, D, q)

q = quadrature(q);

Npst = size(q, 1);

nnodes = size(C, 1);

ndof = 2;

K = zeros(nnodes\*ndof, nnodes\*ndof);

for i = 1:Npst

xi = q (i, 1);

eta = q (i, 2);

w = q (i, 3);

B = compute\_B(C, xi, eta);

dN = tri3\_derivs(xi, eta);

J = C'\*dN;

K = K + B'\*D\*B\*det(J)\*w;

end

end

function B = compute\_B(C, xi, eta)

nnodes = size(C, 1);

ndof = 2;

dN = tri3\_derivs(xi, eta);

J = C'\*dN;

dNdX = dN/J;

for i = 1: nnodes

c = (i-1) \* ndof;

B(1, c+1) = dNdX(i,1);

B(2, c+2) = dNdX(i,2);

B(3, c+1) = dNdX(i,2);

B(3, c+2) = dNdX(i,1);

end

end

function dN = tri3\_derivs (r, s)

syms xi eta

N = [ 1 - xi - eta

xi

eta];

dN = subs([diff(N, xi) diff(N, eta)], [xi eta], [r s]);

end

2. The two-element mesh at right was designed for a plane stress analysis. Each element has dimensions 1 m ×1 m and thikness equal to 0.2 m. Only self weight is considered in the analysis where γ= 25kN/m3.With the aid of computer software calculate nodal displacements and reaction forces. Consider full integration. The material properties are E= 20GPa and ν= 0.25.

E = 20e6; % Modulo de Young [Kpa]

nu = 0.25; % Poisson

q = 4; % Pontos de integração [1 4 9 16]

type = 4; % Nós por elemento [4]

h = 0.2; % Espessura [m]

D = E/(1-nu^2)\*[1 nu 0;

nu 1 0;

0 0 (1-nu)/2]; %Matriz D

Lx = 1; % Comprimento em x

Ly = 2; % Comprimento em y

Elemx = 1; % Divião da malha em x

Elemy = 2; % Divião da malha em y

[nos, elem] = quad\_mesh(Lx,Ly, Elemx,Elemy, type); % Coordenada de cada nó e numeração dos nos de cada elemento

Restr = [1 1 1; 2 0 1]; % NumNó, Rx, Ry [y=1 n=0]

gamma = -25; % Peso Própio [kN/m³]

NumElem = size(elem, 1); % Número de elementos

Ndof = 2; % Número de graus de liberdade

NoElem = size(elem, 2); % Número de Nós por elemento

NumNos = size(nos, 1); % Número de Nós

GL = size(nos, 1) \* Ndof; % Graus de liberdade

K = zeros(GL, GL); % Matriz de rigides com 0

U = zeros(GL, 1); % Vetor de deslocamentos com 0

F = zeros(GL, 1); % Vetor de forças com 0

Fr = zeros(GL, 1); % Vetor de reações com 0

coor = zeros(NumElem, NoElem\*Ndof); % ...

k=1;

for i = [linspace(1, NoElem\*Ndof-1, NoElem)]

coor(:,[i i+1]) = [elem(:, k)\*2-1, elem(:, k)\*2]; % Matriz de coordenadas

k=k+1;

end

for i = 1: NumElem

C = [nos(elem(i,:), :)]; % Matriz de coordenadas por elemento

Ke = compute\_K(C, D, q, h, NoElem); % Matriz de rigidez por elemtno

K(coor(i, :), coor(i, :)) = ...

K(coor(i, :), coor(i, :)) + ...

Ke; % Matriz de rigidez global

Fe = body\_forces(C, h, gamma, q, NoElem); % Forças de corpo por elemento

F(elem(i,:)\*2) = F(elem(i,:)\*2) + Fe; %vetor global de forças de corpo

end

NumGLR = sum(sum(Restr(:, [2 3]))); % Determinação do número de restrições

GLR = zeros(NumGLR, 1); % Vetor deslocamentos restringidos

i = 1;

for apoio = 1:(size(Restr, 1))

NoApoio = Restr(apoio, 1); % Nó de apoio

if Restr(apoio, 2) == 1

GLR(i, 1) = NoApoio \* 2 - 1; % Restrição em x

i = i + 1;

end

if Restr(apoio, 3) == 1

GLR(i, 1) = NoApoio \* 2 - 0; % Restrição em y

i = i + 1;

end

end

GLSR = setxor((1:GL)', GLR); % Grau de liberdade sem restrição

U(GLSR) = K(GLSR, GLSR) \ F(GLSR); % Deslocamentos

Fr(GLR) = K(GLR, :) \* U; % Força de reação

%% ................................................................

function F = body\_forces(C, h, gamma, q, NoElem)

q = quadrature(q);

Npst = size(q, 1);

nnodes = size(C, 1);

F = zeros(nnodes, 1);

for i = 1:Npst

xi = q (i, 1);

eta = q (i, 2);

w = q (i, 3);

[dN, N] = quad\_shape\_form(NoElem, xi, eta);

J = C'\*dN;

F = F + N\*gamma\*h\*det(J)\*w;

end

end

function K = compute\_K(C, D, q, h, NoElem)

q = quadrature(q);

Npst = size(q, 1);

nnodes = size(C, 1);

ndof = 2;

K = zeros(nnodes\*ndof, nnodes\*ndof);

for i = 1:Npst

xi = q (i, 1);

eta = q (i, 2);

w = q (i, 3);

B = compute\_B(C, xi, eta, NoElem);

[dN, N] = quad\_shape\_form(NoElem, xi, eta);

J = C'\*dN;

K = K + B'\*D\*B\*det(J)\*w\*h;

end

end

function B = compute\_B(C, xi, eta, NoElem)

nnodes = size(C, 1);

ndof = 2;

[dN, N] = quad\_shape\_form(NoElem, xi, eta);

J = C'\*dN;

dNdX = dN/J;

for i = 1: nnodes

c = (i-1) \* ndof;

B(1, c+1) = dNdX(i,1);

B(2, c+2) = dNdX(i,2);

B(3, c+1) = dNdX(i,2);

B(3, c+2) = dNdX(i,1);

end

end

function [dN, N] = quad4\_derivs (r, s)

syms xi eta

n = [1.0/4.0 \* (1 - xi) \* (1 - eta)

1.0/4.0 \* (1 + xi) \* (1 - eta)

1.0/4.0 \* (1 + xi) \* (1 + eta)

1.0/4.0 \* (1 - xi) \* (1 + eta)];

dN = subs([diff(n, xi) diff(n, eta)], [xi eta], [r s]);

N = subs(n, [xi eta], [r s]);

end

function [node, element] = ...

quad\_mesh(Lx, Ly, nelemX, nelemY, ...

elementType)

deltaX = Lx/nelemX;

deltaY = Ly/nelemY;

switch elementType

case 4

nodesX = nelemX+1;

nodesY = nelemY+1;

node = [];

for j = 1:nodesY

for i = 1:nodesX

x = (i-1)\*deltaX; y = (j-1)\*deltaY;

node = [node; x y];

end

end

element = [];

for j = 1:nelemY

for i = 1:nelemX

i1 = i+(j-1)\*nodesX;

i2 = i1+1;

i3 = i2+nodesX;

i4 = i1+nodesX;

element = [element; i1 i2 i3 i4];

end

end

end

function [dN, N] = quad\_shape\_form(x, xi, eta)

if x == 4

[dN, N] = quad4\_derivs (xi, eta);

elseif x == 8

[dN, N] = quad8\_derivs (xi, eta);

elseif x == 12

[dN, N] = quad12\_derivs (xi, eta);

else

end

end

function Q = quadrature(q)

quadrature\_1\_pts = [0.0 0.0 4.0];

quadrature\_4\_pts = [

-0.577350269189626 -0.577350269189626 1.0

0.577350269189626 -0.577350269189626 1.0

-0.577350269189626 0.577350269189626 1.0

0.577350269189626 0.577350269189626 1.0];

quadrature\_9\_pts = [

-0.774596669241483 -0.774596669241483 0.3086419753086419

0.0 -0.774596669241483 0.4938271604938271

0.774596669241483 -0.774596669241483 0.3086419753086419

-0.774596669241483 0.0 0.4938271604938271

0.0 0.0 0.7901234567901234

0.774596669241483 0.0 0.4938271604938271

-0.774596669241483 0.774596669241483 0.3086419753086419

0.0 0.774596669241483 0.4938271604938271

0.774596669241483 0.774596669241483 0.3086419753086419];

quadrature\_16\_pts = [

0.861136311594053 0.861136311594053 0.121002993285602

0.861136311594053 0.339981043584856 0.226851851851851

0.861136311594053 -0.339981043584856 0.226851851851851

0.861136311594053 -0.861136311594053 0.121002993285602

0.339981043584856 0.861136311594053 0.226851851851851

0.339981043584856 0.339981043584856 0.425293303010694

0.339981043584856 -0.339981043584856 0.425293303010694

0.339981043584856 -0.861136311594053 0.226851851851851

-0.339981043584856 0.861136311594053 0.226851851851851

-0.339981043584856 0.339981043584856 0.425293303010694

-0.339981043584856 -0.339981043584856 0.425293303010694

-0.339981043584856 -0.861136311594053 0.226851851851851

-0.861136311594053 0.861136311594053 0.121002993285602

-0.861136311594053 0.339981043584856 0.226851851851851

-0.861136311594053 -0.339981043584856 0.226851851851851

-0.861136311594053 -0.861136311594053 0.121002993285602];

if q == 1

Q = quadrature\_1\_pts;

elseif q == 4

Q = quadrature\_4\_pts;

elseif q == 9

Q = quadrature\_9\_pts;

elseif q == 16

Q = quadrature\_16\_pts;

else

Q = 0;

end

end

3. In the last exercise, calculate the strain and stress vectors at the integration points of element 1.

Tensor

[sig] = stress\_strain(nos, elem, U, D, q, coor);

%% ......................................................................

function [sig, eps] = stress\_strain(nodes,...

element,displacement,D,q,coordinateelem)

nodeselement= size(element, 2);

ndof = size(nodes, 2);

numelement = size(element, 1);

sig = zeros(numelement, q, 3);

for i = 1: numelement

C = [nodes(element(i,:), :)];

U = displacement(coordinateelem(i,:));

for j = 1: q

B = compute\_B(C, q, nodeselement, ndof);

sig(i,j,:) = (D\*B\*U);

eps = B\*U;

end

end

end

4. Verify if the analytical vertical stress γh at the bottom of the structure can be recovered from the stresses at integration points.

Não pode