Bilioteca de funçoes

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)

% QUAD4\_DERIVS Função de elemetno quadrilateral com 4 nós que retorna a matriz de devivadas

% $\xi \;e\;\eta \;\;$e sua função de forma para $\xi \;e\;\eta \;\;$inseridos

%

% dN = derivadas

%

% N = funão de forma

%

% r = ponto avaliado em $\xi \;$

%

% s = ponto avaliado em $\eta \;$

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 [dN, N] = quad8\_derivs (r, s)

syms xi eta

rp1=1.0+xi; rm1=1.0-xi;sp1=1.0+eta; sm1=1.0-eta;

n = [0.25\*rm1\*sm1\*(rm1+sm1-3.0)

0.25\*rp1\*sm1\*(rp1+sm1-3.0)

0.25\*rp1\*sp1\*(rp1+sp1-3.0)

0.25\*rm1\*sp1\*(rm1+sp1-3.0)

0.50\*sm1\*(1.0-xi\*xi)

0.50\*rp1\*(1.0-eta\*eta)

0.50\*sp1\*(1.0-xi\*xi)

0.50\*rm1\*(1.0-eta\*eta)];

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

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

end

function [dN, N] = quad12\_derivs (r, s)

syms xi eta

n = [

(5\*(eta))/16 + (5\*(xi))/16 - (5\*(eta)\*(xi))/16 + (9\*(eta)^2)/32 - (9\*(eta)^3)/32 + (9\*(xi)^2)/32 - (9\*(xi)^3)/32 - (9\*(eta)\*(xi)^2)/32 - (9\*(eta)^2\*(xi))/32 + (9\*(eta)\*(xi)^3)/32 + (9\*(eta)^3\*(xi))/32 - 5/16

(5\*(eta))/16 - (5\*(xi))/16 + (5\*(eta)\*(xi))/16 + (9\*(eta)^2)/32 - (9\*(eta)^3)/32 + (9\*(xi)^2)/32 + (9\*(xi)^3)/32 - (9\*(eta)\*(xi)^2)/32 + (9\*(eta)^2\*(xi))/32 - (9\*(eta)\*(xi)^3)/32 - (9\*(eta)^3\*(xi))/32 - 5/16

(9\*(eta)^2)/32 - (5\*(xi))/16 - (5\*(eta)\*(xi))/16 - (5\*(eta))/16 + (9\*(eta)^3)/32 + (9\*(xi)^2)/32 + (9\*(xi)^3)/32 + (9\*(eta)\*(xi)^2)/32 + (9\*(eta)^2\*(xi))/32 + (9\*(eta)\*(xi)^3)/32 + (9\*(eta)^3\*(xi))/32 - 5/16

(5\*(xi))/16 - (5\*(eta))/16 + (5\*(eta)\*(xi))/16 + (9\*(eta)^2)/32 + (9\*(eta)^3)/32 + (9\*(xi)^2)/32 - (9\*(xi)^3)/32 + (9\*(eta)\*(xi)^2)/32 - (9\*(eta)^2\*(xi))/32 - (9\*(eta)\*(xi)^3)/32 - (9\*(eta)^3\*(xi))/32 - 5/16

(27\*(eta)\*(xi))/32 - (27\*(xi))/32 - (9\*(eta))/32 - (9\*(xi)^2)/32 + (27\*(xi)^3)/32 + (9\*(eta)\*(xi)^2)/32 - (27\*(eta)\*(xi)^3)/32 + 9/32

(27\*(xi))/32 - (9\*(eta))/32 - (27\*(eta)\*(xi))/32 - (9\*(xi)^2)/32 - (27\*(xi)^3)/32 + (9\*(eta)\*(xi)^2)/32 + (27\*(eta)\*(xi)^3)/32 + 9/32

(9\*(xi))/32 - (27\*(eta))/32 - (27\*(eta)\*(xi))/32 - (9\*(eta)^2)/32 + (27\*(eta)^3)/32 - (9\*(eta)^2\*(xi))/32 + (27\*(eta)^3\*(xi))/32 + 9/32

(27\*(eta))/32 + (9\*(xi))/32 + (27\*(eta)\*(xi))/32 - (9\*(eta)^2)/32 - (27\*(eta)^3)/32 - (9\*(eta)^2\*(xi))/32 - (27\*(eta)^3\*(xi))/32 + 9/32

(9\*(eta))/32 + (27\*(xi))/32 + (27\*(eta)\*(xi))/32 - (9\*(xi)^2)/32 - (27\*(xi)^3)/32 - (9\*(eta)\*(xi)^2)/32 - (27\*(eta)\*(xi)^3)/32 + 9/32

(9\*(eta))/32 - (27\*(xi))/32 - (27\*(eta)\*(xi))/32 - (9\*(xi)^2)/32 + (27\*(xi)^3)/32 - (9\*(eta)\*(xi)^2)/32 + (27\*(eta)\*(xi)^3)/32 + 9/32

(27\*(eta))/32 - (9\*(xi))/32 - (27\*(eta)\*(xi))/32 - (9\*(eta)^2)/32 - (27\*(eta)^3)/32 + (9\*(eta)^2\*(xi))/32 + (27\*(eta)^3\*(xi))/32 + 9/32

(27\*(eta)\*(xi))/32 - (9\*(xi))/32 - (27\*(eta))/32 - (9\*(eta)^2)/32 + (27\*(eta)^3)/32 + (9\*(eta)^2\*(xi))/32 - (27\*(eta)^3\*(xi))/32 + 9/32];

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

case 8

nodesX = nelemX\*2+1;

nodesY = nelemY\*2+1;

node = [];

for j = 1:nodesY

y = (j-1)\*deltaY/2;

if mod(j,2)

nodesX = nelemX\*2+1;

for i = 1:nodesX

x = (i-1)\*deltaX/2;

node = [node; x y];

end

else

nodesX = nelemX+1;

for i = 1:nodesX

x = (i-1)\*deltaX;

node = [node; x y];

end

end

end

element = [];

for j = 1:nelemY

for i = 1:nelemX

i1 = 1 + 2\*(i-1) + (j-1)\*round(1.5\*nodesX);

i2 = i1 + 2;

i3 = i2 + round(1.5\*nodesX);

i4 = i1 + round(1.5\*nodesX);

i5 = i1 + 1;

i6 = i1 + nodesX+2-i;

i7 = i4 + 1;

i8 = i6 - 1;

element = [element; i1 i2 i3 i4 i5 i6 i7 i8];

end

end

case 12

nodesX = nelemX\*3+1;

nodesY = nelemY\*3+1;

node = [];

k = 1;

for j = 1:nodesY

y = (j-1)\*deltaY/3;

if j == k

nodesX = nelemX\*3+1;

for i = 1:nodesX

x = (i-1)\*deltaX/3;

node = [node; x y];

k = k + (3/nodesX);

end

else

nodesX = nelemX+1;

for i = 1:nodesX

x = (i-1)\*deltaX;

node = [node; x y];

end

end

k = round(k);

end

element = [];

for j = 1:nelemY

for i = 1:nelemX

i1 = 1 + 3\*(i-1) + (j-1)\*(2\*nodesX) - 2\*(j-1);

i2 = i1 + 3;

i3 = i1 + nodesX + (nelemX\*2+2) - 2\*(i-2.5) + 2\*(i-1);

i4 = i3 - 3;

i5 = i1 + 1;

i6 = i5 + 1;

i7 = i1 + nodesX + 1 - 2\*(i-1);

i8 = i1 + nodesX + (nelemX+2) - 2\*(i-1);

i9 = i3 - 1;

i10 = i9 - 1;

i11 = i8 - 1;

i12 = i7 - 1;

element = [element; i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12];

end

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

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

function draw\_mesh(nodeCoordinates,elementNodes,elementType,lineType)

switch elementType

case 'Q4'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,:),1),nodeCoordinates(elementNodes(i,:),2),'w','FaceColor','none','LineStyle',lineType,'EdgeColor','k')

end

case 'Q8'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,1:4),1),nodeCoordinates(elementNodes(i,1:4),2),'w','FaceColor','none','LineStyle',lineType,'EdgeColor','k')

end

case 'Q12'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,1:4),1),nodeCoordinates(elementNodes(i,1:4),2),'w','FaceColor','none','LineStyle',lineType,'EdgeColor','k')

end

otherwise

disp('Element type not available')

end

end

function draw\_field(nodeCoordinates,elementNodes,...

elementType,field)

switch elementType

case 'Q4'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,:),1),nodeCoordinates(elementNodes(i,:),2),field(i,:));

end

case 'Q8'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,1:4),1),nodeCoordinates(elementNodes(i,1:4),2),field(i,:));

end

case 'Q12'

for i = 1:size(elementNodes,1)

patch(nodeCoordinates(elementNodes(i,1:4),1),nodeCoordinates(elementNodes(i,1:4),2),field(i,:));

end

otherwise

disp('Element type not available')

end

end