

Jun 25, 14 11:24	evaluateFctBasesElemRef2D.m	Page 1/2
<pre> function FctBases= evaluateFctBasesElemRef2D(TypeEF) %UNTITLED2 Summary of this function goes here % Detailed explanation goes here %% % % dans le domaine % %% FctBases.TypeEF = TypeEF; % schema d'integration sur l'element de reference [x,y,poids] = schemaIntg2D(TypeEF.ordreIntg,TypeEF.triangle); nptg2 =length(x); FctBases.nptsGauss = nptg2; FctBases.ptsGauss = [x; y]; FctBases.poids = poids; % calcul des fonctions de bases aux noeuds de Gauss sur l'element de reference [FctBases.N,FctBases.dNdX,FctBases.dNdY] = FctBasesElementsFinis2D(x,y,TypeEF.type); FctBases.nnelem = size(FctBases.N,1); % calcul de la transformation geometrique aux noeuds de Gauss sur l'element de r eference [FctBases.NGeo,FctBases.dNGeodX,FctBases.dNGeodY] = FctBasesElementsFinis2D(x,y ,TypeEF.typeGeo); %% % % sur la frontiere % %% % schema d'integration sur l'element de reference if TypeEF.triangle intervalle = true; % intervalle = [0,1] else intervalle = false; % intervalle = [-1,1] end [x,poids] = schemaIntg1D(TypeEF.ordreIntgFace,intervalle); nptgf =length(x); FctBases.nptsGaussFace = nptgf; FctBases.ptsGaussFace = x; FctBases.poidsFace = poids; % calcul des fonctions de bases aux noeuds de Gauss sur l'element de reference [FctBases.NFace,FctBases.dNdXFace] = FctBasesElementsFinis1D(x,TypeEF.type); FctBases.nnelemFace = size(FctBases.NFace,1); % calcul de la transformation geometrique aux noeuds de Gauss sur l'element de r eference [FctBases.NGeoFace,FctBases.dNGeodXFace] = FctBasesElementsFinis1D(x,TypeEF.typ eGeo); </pre>		

Jun 25, 14 11:24	evaluateFctBasesElemRef2D.m	Page 2/2
<pre> end </pre>		

May 22, 15 13:54	evaluateFctBasesElemRef3D.m	Page 1/2
<pre> function FctBases= evaluateFctBasesElemRef3D(TypeEF) %UNTITLED2 Summary of this function goes here % Detailed explanation goes here %% % % dans le domaine 3D % %% FctBases.TypeEF = TypeEF; % schema d'integration sur l'element de reference [x,y,z,poids] = schemaIntg3D(TypeEF.ordreIntg, TypeEF.tetra); nptg3 =length(x); FctBases.nptsGauss = nptg3; FctBases.ptsGauss = [x; y; z]; FctBases.poids = poids; % calcul des fonctions de bases aux noeuds de Gauss sur l'element de reference [FctBases.N,FctBases.dNdX,FctBases.dNdY, FctBases.dNdZ] = FctBasesElementsFinis3 D(x,y,z,TypeEF.type); FctBases.nnelem = size(FctBases.N,1); % calcul de la transformation geometrique aux noeuds de Gauss sur l'element de r eference [FctBases.NGeo,FctBases.dNGeodX,FctBases.dNGeodY, FctBases.dNGeodZ] = FctBasesEl ementsFinis3D(x,y,z,TypeEF.typeGeo); FctBases.nnelemGeo = size(FctBases.NGeo,1); %% % % sur la frontiere % %% % schema d'integration sur l'element de reference if (TypeEF.tetra) FaceTriangle = true; % triangle else FaceTriangle = false; % quad end [x,y,poids] = schemaIntg2D(TypeEF.ordreIntgFace, FaceTriangle); nptg2f =length(x); FctBases.nptsGaussFace = nptg2f; FctBases.ptsGaussFace = [x; y]; FctBases.poidsFace = poids; % calcul des fonctions de bases aux noeuds de Gauss sur l'element de reference [FctBases.NFace,FctBases.dNdXFace,FctBases.dNdYFace] = FctBasesElementsFinis2D(x ,y,TypeEF.type); FctBases.nnelemFace = size(FctBases.NFace,1); % calcul de la transformation geometrique aux noeuds de Gauss sur l'element de r eference [FctBases.NGeoFace,FctBases.dNGeodXFace, FctBases.dNGeodYFace] = FctBasesElemen </pre>		

May 22, 15 13:54	evaluateFctBasesElemRef3D.m	Page 2/2
<pre> tsFinis2D(x,y,TypeEF.typeGeo); end </pre>		

May 22, 15 13:59

voirContraintes2D.m

Page 1/3

```

function [s11,s12,s22]= voirContraintes2D( Mail, FctBases, sol, param, TypeEF)
%UNTITLED Summary of this function goes here
% Detailed explanation goes here

% initialisation

% parametre de l'equation
eta = param.vis; % vis = E / 2*(1+nu) avec nu = coeff. de Poisson
lambda = param.lambda; % lambda = E*nu/ (1-2*nu)*(1+nu)

%nnoeud = Noeud.nnoeud;
nel = Mail.nel;
nse = Mail.nse;
nnelem=FctBases.nnelem;

% transfert de la solution sur les elements
ndle = Mail.connect;
coordX = (reshape(Mail.coord(1,ndle(:)),nnelem,nel))';
coordY = (reshape(Mail.coord(2,ndle(:)),nnelem,nel))';

u= sol(2*ndle-1)';
v= sol(2*ndle)';

% initialisation

switch TypeEF.degGeo

    case 1

x=[0 1 0];
y=[0 0 1];

dNdXSommet = zeros(3,3);
dNdXSommet(1,:) = -1;
dNdXSommet(2,:) = 1;
dNdXSommet(3,:) = 0;

dNdYSommet = zeros(3,3);
dNdYSommet(1,:) = -1;
dNdYSommet(2,:) = 0;
dNdYSommet(3,:) = 1;

    case 2

x=[0 1 0 0.5 0.5 0];
y=[0 0 1 0 0.5 0.5];

dNdXSommet = zeros(6,6);
dNdXSommet(1,:) = 1 - 4*(1 - x - y);
dNdXSommet(2,:) = 4*x - 1;
dNdXSommet(3,:) = 0;
dNdXSommet(4,:) = 4*(1 - 2*x - y);
dNdXSommet(5,:) = 4*y;
dNdXSommet(6,:) = -4*y;

dNdYSommet = zeros(6,6);
dNdYSommet(1,:) = 1 - 4*(1 - x - y);
dNdYSommet(2,:) = 0;
dNdYSommet(3,:) = 4*y - 1;
dNdYSommet(4,:) = -4*x;
dNdYSommet(5,:) = 4*x;
dNdYSommet(6,:) = 4*(1 - x - 2*y);

dNdX = zeros(6,6);
dNdX(1,:) = -1;

```

May 22, 15 13:59

voirContraintes2D.m

Page 2/3

```

dNdX(2,:) = 1;
dNdX(3,:) = 0;
dNdX(4,:) = 4.*(1-2.*x-y);
dNdX(5,:) = 4.*y;
dNdX(6,:) = -4.*y;

dNdY = zeros(6,6);
dNdY(1,:) = -1;
dNdY(2,:) = 0;
dNdY(3,:) = 1;
dNdY(4,:) = -4.*x;
dNdY(5,:) = 4.*x;
dNdY(6,:) = 4.*(1-x-2.*y);

end

% transformation geometrique

if FctBases.TypeEF.degGeo == 1;
% la matrice jacobienne
JX(:,1) = coordX(:,2) - coordX(:,1); % jac11
JX(:,2) = coordX(:,3) - coordX(:,1); % jac12
JY(:,1) = coordY(:,2) - coordY(:,1); % jac21
JY(:,2) = coordY(:,3) - coordY(:,1); % jac22

% le jacobien de taille n x 1
J = abs(JX(:,1).*JY(:,2) - JX(:,2).*JY(:,1));
invJ = 1.0./J;

% Jx = [ jac22, - jac21] = premiere colonne de la matrice inverse
invJx(:,1) = JY(:,2).*invJ; % jac22
invJx(:,2) = -JY(:,1).*invJ; % -jac21
% Jy = [ -jac12, jac11] = deuxieme colonne de la matrice inverse
invJy(:,1) = -JX(:,2).*invJ; % -jac12
invJy(:,2) = JX(:,1).*invJ; % jac11

end

% initialisation

sig11 = zeros(nel,nse);
sig12 = zeros(nel,nse);
sig22 = zeros(nel,nse);

% boucle sur les points de Gauss

for k=1:nse

if FctBases.TypeEF.degGeo == 2;

dNP2dX = dNdXSommet(:,k);
dNP2dY = dNdYSommet(:,k);

% la matrice jacobienne
JX(:,1) = coordX*dNP2dX; % jac11
JX(:,2) = coordX*dNP2dY; % jac12
JY(:,1) = coordY*dNP2dX; % jac21
JY(:,2) = coordY*dNP2dY; % jac22

% le jacobien de taille n x 1
J = abs(JX(:,1).*JY(:,2) - JX(:,2).*JY(:,1));
invJ = 1.0./J;

% Jx = [ jac22, - jac21] = premiere colonne de la matrice inverse
invJx(:,1) = JY(:,2).*invJ; % jac22
invJx(:,2) = -JY(:,1).*invJ; % -jac21
% Jy = [ -jac12, jac11] = deuxieme colonne de la matrice inverse
invJy(:,1) = -JX(:,2).*invJ; % -jac12
invJy(:,2) = JX(:,1).*invJ; % jac11

```

May 22, 15 13:59

voirContraintes2D.m

Page 3/3

end*% calcul de dudx et dudy aux points de Gauss***switch** TypeEF.degGeo**case** 1dNdXY = [dNdXSommet(:,k), dNdYSommet(:,k)]; *% les derivees des fcts de bases***case** 2dNdXY = [dNdX(:,k), dNdY(:,k)]; *% les derivees des fcts de bases***end***% transformation des derivees*dNdx = invJx*dNdXY'; *% taille nel x nnelem*dNdy = invJy*dNdXY'; *% taille nel x nnelem*

dudxk=zeros(nel,1);

dudyk=zeros(nel,1);

dvdxx=zeros(nel,1);

dvdyk=zeros(nel,1);

for ii=1:nnelem

dudxk = dudxk + u(:,ii).*dNdx(:,ii);

dudyk = dudyk + u(:,ii).*dNdy(:,ii);

dvdxx = dvdxx + v(:,ii).*dNdx(:,ii);

dvdyk = dvdyk + v(:,ii).*dNdy(:,ii);

end

sig11(:,k) = 2*eta*dudxk + lambda*(dudxk + dvdyk);

sig12(:,k) = eta*(dudyk + dvdxx);

sig22(:,k) = 2*eta*dvdyk + lambda*(dudxk + dvdyk);

*% sig11ek = 2*eta*u11 + lambda*(u11+u22);**% sig12ek = eta*(u12+u21);**% sig21ek = sig12ek;**% sig22ek = 2*eta*u22 + lambda*(u11+u22);***end**

sig11 = sig11';

sig12 = sig12';

sig22 = sig22';

totalVoisin = full(sparse(Mail.connect(:),1,ones(Mail.nse*Mail.nel,1),Mail.ns,1));

s11 = full(sparse(Mail.connect(:),1,sig11(:),Mail.ns,1));

s12 = full(sparse(Mail.connect(:),1,sig12(:),Mail.ns,1));

s22 = full(sparse(Mail.connect(:),1,sig22(:),Mail.ns,1));

s11=s11./totalVoisin;

s12=s12./totalVoisin;

s22=s22./totalVoisin;

end