```
%Calculates the embedded element effect on host element during
%InternalForce_explicit

% T_internal = zeros(FEM.mesh.n_dofs_elem,1);
% ielement = 1;    eelt = 3; %Global elt num of embedded
%     global_nodes    = FEM.mesh.connectivity(:,ielement);
%     xlocal          = GEOM.x(:,global_nodes);
%     x0local         = GEOM.x0(:,global_nodes);
%     QUADRATURE = QUADRATURE.element;

function [T_internal] = CorrectInternalForce_explicit(ielement,...
          T_internal,FEM,xlocal,x0local,QUADRATURE,CONSTANT,GEOM,...
          PLAST,KINEMATICS,MAT,DAMPING,eelt)


global explicit
dim=GEOM.ndime;

VolCorrect = false;


  %--------------------------------------------------------------------------
    % GATHER material properties of the host and embedded elt

  %--------------------------------------------------------------------------
    material_number   = MAT.matno(ielement);
    matyp_h           = MAT.matyp(material_number);
    properties_h      = MAT.props(:,material_number);
    Ve_h              = GEOM.Ve(ielement);

    material_number   = MAT.matno(eelt);
    matyp_e           = MAT.matyp(material_number);
    properties_e      = MAT.props(:,material_number);
    Ve_e              = GEOM.Ve(eelt);


    switch matyp_h
     case {5,7,17}
          [pressure_h,kappa_bar_h,DN_x_mean_h,ve_h] = ...

 mean_dilatation_pressure(FEM,dim,matyp_h,properties_h,Ve_h,...
                                     QUADRATURE,KINEMATICS);
     otherwise
          pressure_h = 0;
    end

    switch matyp_e
     case {5,7,17}
          [pressure_e,kappa_bar_e,DN_x_mean_e,ve_e] = ...

 mean_dilatation_pressure(FEM,dim,matyp_e,properties_e,Ve_e,...
                                     QUADRATURE_eh,KINEMATICS_eh);
```

```
    otherwise
         pressure_e = 0;
   end



 %-------------------------------------------------------------------------
   %Step A
   %Get the embedded element quadrature points in the host element
domain

 %-------------------------------------------------------------------------

   %Get the number of Gauss points in the element
   nGp = GEOM.embedded.HostTotals(ielement,2);

   %Get host element nodes
   h_connectivity = FEM.mesh.connectivity(:,ielement);
   x_h = GEOM.x0(:,h_connectivity);  %Host node global coordinates

   %Get embedded element information
   e_connectivity = FEM.mesh.connectivity(:,eelt);
   x_e = GEOM.x0(:,e_connectivity);
   xelocal  = GEOM.x(:,e_connectivity);
   e_nodes_zeta = GEOM.embedded.Embed_Zeta(:,e_connectivity);

   QUADRATURE_E = QUADRATURE; %Quadrature of embedded elt (assuming
same element type as host)
   QUADRATURE_EH = QUADRATURE; %Quadrature of embedded elt in the
host domain
   KINEMATICS_EH = KINEMATICS;

   gp_zeta = zeros(3,8);
   for gp = 1: nGp

       %Get the coordinates of the gauss point in the embedded space
(LE)
       gp_nu = QUADRATURE_E.Chi(gp,:);

       %Find the gauss point coordinates in the host space (LH) by
mapping
       %from LE to LH using the coordinates of the embedded nodes in
LH
       %Or you could actually just get it from
EM.interpolation.element.N(:,gp)
       N_nu_a = shape_function_values_at(gp_nu, 'hex'); %mapping with
shape functions

       %z(n) = [N][ze] = sum(N1*ze1 +
       %Doing this for all gauss points at once, so loop through nGp
times
       for i=1:nGp
```

```
            gp_zeta(:,gp) = gp_zeta(:,gp) +
 N_nu_a(i,1)*e_nodes_zeta(:,i);
        end
%         for i=1:8
%            gp_zeta(:,gp) = gp_zeta(:,gp) +
 FEM.interpolation.element.N(i,gp)*e_nodes_zeta(:,i);
%         end


        %Test accuracy of that conversion
            %Find gp_x from gp_nu and gp_zeta
            gp_xz = find_xyz_in_host(gp_zeta(:,gp), x_h);
            gp_xn = find_xyz_in_host(gp_nu, x_e);
            check = gp_xz - gp_xn;
            if abs(check(1))>1E-10 || abs(check(2))>1E-10 ||
 abs(check(3)) >1E-10
                fprintf("Space conversion failure: host %u, guest %u,
 gp %u\n",ielement, eelt, gp);
                fprintf("     Error amount: %d %d %d\n",
 abs(check(1)), abs(check(2)), abs(check(3)));
            end

%     figure(); hold on; view(3); grid on;
%      xh = x_h; xe = x_e; a_xz = gp_xz; a_xn = gp_xn;
%     %Plot host element xh
%         faces = zeros(3,3,12);
%         faces(:,:,1) = [xh(:,4) xh(:,3) xh(:,2)];
%         faces(:,:,2) = [xh(:,2) xh(:,1) xh(:,4)];
%         faces(:,:,3) = [xh(:,1) xh(:,2) xh(:,6)];
%         faces(:,:,4) = [xh(:,6) xh(:,5) xh(:,1)];
%         faces(:,:,5) = [xh(:,2) xh(:,3) xh(:,7)];
%         faces(:,:,6) = [xh(:,7) xh(:,6) xh(:,2)];
%         faces(:,:,7) = [xh(:,3) xh(:,4) xh(:,8)];
%         faces(:,:,8) = [xh(:,8) xh(:,7) xh(:,3)];
%         faces(:,:,9) = [xh(:,1) xh(:,5) xh(:,8)];
%         faces(:,:,10) = [xh(:,8) xh(:,4) xh(:,1)];
%         faces(:,:,11) = [xh(:,5) xh(:,6) xh(:,7)];
%         faces(:,:,12) = [xh(:,7) xh(:,8) xh(:,5)];
%
%
%
 X=[faces(1,:,1);faces(1,:,2);faces(1,:,3);faces(1,:,4);faces(1,:,5);faces(1,:,6)]
%
 Y=[faces(2,:,1);faces(2,:,2);faces(2,:,3);faces(2,:,4);faces(2,:,5);faces(2,:,6)]
%
 Z=[faces(3,:,1);faces(3,:,2);faces(3,:,3);faces(3,:,4);faces(3,:,5);faces(3,:,6)]
%
%
 XX=[faces(1,:,7);faces(1,:,8);faces(1,:,9);faces(1,:,10);faces(1,:,11);faces(1,:,
%
 YY=[faces(2,:,7);faces(2,:,8);faces(2,:,9);faces(2,:,10);faces(2,:,11);faces(2,:,
%
 ZZ=[faces(3,:,7);faces(3,:,8);faces(3,:,9);faces(3,:,10);faces(3,:,11);faces(3,:,
%
```

```
%
%         patch(X,Y,Z,'black','FaceAlpha', 0.2);
%         patch(XX,YY,ZZ,'black','FaceAlpha', 0.2);
%
%         % Plot embedded elt xe
%         faces = zeros(3,3,12);
%         faces(:,:,1) = [xe(:,4) xe(:,3) xe(:,2)];
%         faces(:,:,2) = [xe(:,2) xe(:,1) xe(:,4)];
%         faces(:,:,3) = [xe(:,1) xe(:,2) xe(:,6)];
%         faces(:,:,4) = [xe(:,6) xe(:,5) xe(:,1)];
%         faces(:,:,5) = [xe(:,2) xe(:,3) xe(:,7)];
%         faces(:,:,6) = [xe(:,7) xe(:,6) xe(:,2)];
%         faces(:,:,7) = [xe(:,3) xe(:,4) xe(:,8)];
%         faces(:,:,8) = [xe(:,8) xe(:,7) xe(:,3)];
%         faces(:,:,9) = [xe(:,1) xe(:,5) xe(:,8)];
%         faces(:,:,10) = [xe(:,8) xe(:,4) xe(:,1)];
%         faces(:,:,11) = [xe(:,5) xe(:,6) xe(:,7)];
%         faces(:,:,12) = [xe(:,7) xe(:,8) xe(:,5)];
%
%
 X=[faces(1,:,1);faces(1,:,2);faces(1,:,3);faces(1,:,4);faces(1,:,5);faces(1,:,6)]
%
 Y=[faces(2,:,1);faces(2,:,2);faces(2,:,3);faces(2,:,4);faces(2,:,5);faces(2,:,6)]
%
 Z=[faces(3,:,1);faces(3,:,2);faces(3,:,3);faces(3,:,4);faces(3,:,5);faces(3,:,6)]
%
%
 XX=[faces(1,:,7);faces(1,:,8);faces(1,:,9);faces(1,:,10);faces(1,:,11);faces(1,:,
%
 YY=[faces(2,:,7);faces(2,:,8);faces(2,:,9);faces(2,:,10);faces(2,:,11);faces(2,:,
%
 ZZ=[faces(3,:,7);faces(3,:,8);faces(3,:,9);faces(3,:,10);faces(3,:,11);faces(3,:,
%
%         patch(X,Y,Z,'blue','FaceAlpha', 0.2);
%         patch(XX,YY,ZZ,'blue','FaceAlpha', 0.2);
%
%         %Plot point a and and some labels
%         colors = ['r.' 'b.' 'k.' 'g.' 'm.' 'c.' 'r.' 'b.' 'k.' 'g.'
 'm.' 'c.'];
%         plot3(a_xn(1),a_xn(2),a_xn(3), 'ro');
%         plot3(a_xz(1),a_xz(2),a_xz(3), 'b.');
%         xlabel('x'); ylabel('y'); zlabel('z');
%         hold off;
    end



  %-----------------------------------------------------------------------
    %Step B
    %Compute deformation measures at quad points

  %-----------------------------------------------------------------------
```

```
    QUADRATURE_EH.Chi = gp_zeta';
    KINEMATICS_EH =
 gradients(xlocal,x0local,FEM.interpolation.element.DN_chi,...
            QUADRATURE_EH,KINEMATICS_EH);

    KINEMATICS =
 gradients(xlocal,x0local,FEM.interpolation.element.DN_chi,...
            QUADRATURE,KINEMATICS);

%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
% Gauss quadrature integration loop. Loop of embedded element Gauss
 Points
%-------------------------------------------------------------------------
    for igauss = 1:nGp

        %Step B

 %-------------------------------------------------------------------------
        % Extract kinematics at the particular Gauss point.

 %-------------------------------------------------------------------------
        kinematics_gauss =
 kinematics_gauss_point(KINEMATICS_EH,igauss);

        % Step C

 %-------------------------------------------------------------------------
        % Calculate embeddded element stress measure in the host elt
 system
        % using the embedded elt material model

 %-------------------------------------------------------------------------
        [Cauchy_eh,PLAST_EH,...
         plast_gauss] =
 Cauchy_type_selection(kinematics_gauss,properties_e,...

 CONSTANT,dim,matyp_e,PLAST,igauss);

 %-------------------------------------------------------------------------
        % Obtain elasticity tensor (for incompressible or nearly
 incompressible,
        % only deviatoric component).

 %-------------------------------------------------------------------------
        if(explicit==0)
            c =
 elasticity_modulus_selection(kinematics_gauss,properties_e,CONSTANT,...

 dim,matyp_e,PLAST_EH,plast_gauss,igauss);
        else
            c = 0;
```

```
        end

 %-----------------------------------------------------------------------
        % Add pressure contribution to stresses and elasticity tensor.

 %-----------------------------------------------------------------------
        [Cauchy_eh,c] =
mean_dilatation_pressure_addition(Cauchy_eh,c,CONSTANT,pressure_e,matyp_e);

 %-----------------------------------------------------------------------


 %-----------------------------------------------------------------------
        % Compute numerical integration multipliers.

 %-----------------------------------------------------------------------
        JW = kinematics_gauss.Jx_chi*QUADRATURE_EH.W(igauss)*...

thickness_plane_stress(properties_e,kinematics_gauss.J,matyp_e);

 %-----------------------------------------------------------------------
        % Compute contribution to (internal) force vector.

 %-----------------------------------------------------------------------
        T = Cauchy_eh*kinematics_gauss.DN_x;
        T_e = T(:)*JW;



        %Step D

 %-----------------------------------------------------------------------
        % Calculate embeddded element stress measure in the host elt
system
        % using the host elt material model (ie correction stress)

 %-----------------------------------------------------------------------
        [Cauchy_C,PLAST_h,...
         plast_gauss] =
Cauchy_type_selection(kinematics_gauss,properties_h,...

CONSTANT,dim,matyp_h,PLAST,igauss);

 %-----------------------------------------------------------------------
        % Obtain elasticity tensor (for incompressible or nearly
incompressible,
        % only deviatoric component).

 %-----------------------------------------------------------------------
        if(explicit==0)
            c =
elasticity_modulus_selection(kinematics_gauss,properties_h,CONSTANT,...

dim,matyp_h,PLAST_h,plast_gauss,igauss);
```

```
        else
            c = 0;
        end

 %--------------------------------------------------------------------
        % Add pressure contribution to stresses and elasticity tensor.

 %--------------------------------------------------------------------
        [Cauchy_C,c] =
mean_dilatation_pressure_addition(Cauchy_C,c,CONSTANT,pressure_h,matyp_h);

 %--------------------------------------------------------------------


 %--------------------------------------------------------------------
        % Compute numerical integration multipliers.

 %--------------------------------------------------------------------
        JW = kinematics_gauss.Jx_chi*QUADRATURE_EH.W(igauss)*...

thickness_plane_stress(properties_h,kinematics_gauss.J,matyp_h);

 %--------------------------------------------------------------------
        % Compute contribution to (internal) force vector.

 %--------------------------------------------------------------------
        T = Cauchy_C*kinematics_gauss.DN_x;
        T_C = T(:)*JW;

        %Step E

 %--------------------------------------------------------------------
        % Compute equivilant (internal) force vector of the host
element.

 %--------------------------------------------------------------------
        if VolCorrect
            T_internal = T_internal + (T_e - T_C);
        else
            T_internal = T_internal + (T_e);
        end

    end
%     Step_globalT_int =
 force_vectors_assembly(T_internal,global_nodes,...
%                     zeros(FEM.mesh.n_dofs,1),FEM.mesh.dof_nodes);

end

Not enough input arguments.

Error in CorrectInternalForce_explicit (line 17)
dim=GEOM.ndime;
```

Published with MATLAB® R2019b