Assignment 1: Continuum Damage Model

Jordi Parra Porcar
CIMNE
jordiparraporcar@gmail.com

1. Implementation of the numerical algorithm for a rate independent case

1.1. "Non-symmetric tension-compression" and "Tension only" damage models

To implement these models the subroutine *modelo_de_dano1.m* has been modified introducing the equations stated in the two points below. On the other hand the subroutine *dibujar_criterio_dano1.m* has been also modified in order to plot the two the elastic domains

- Non-symmetric tension-compression:

$$\bar{\sigma} = \epsilon \mathbf{C} \tag{1}$$

$$\bar{\sigma}^{+} = \langle \sigma \rangle \tag{2}$$

$$\theta = \frac{\sum \langle \bar{\sigma}_i \rangle}{\sum |\bar{\sigma} i|} \tag{3}$$

$$\tau = \sqrt{\bar{\sigma} \epsilon'} \tag{4}$$

- Tension only model:

$$\bar{\sigma} = \epsilon \mathbf{C} \tag{5}$$

$$\bar{\sigma}^{+} = \langle \sigma \rangle \tag{6}$$

$$\tau^{+} = \sqrt{\bar{\sigma}^{+} \epsilon'} \tag{7}$$

1.2. Exponential hardening/softening

In order to implement the exponential hardening law the following equations where introduced in the code:

$$H(r) = A \frac{q_\infty - r_0}{r_0} e^{A\left(1 - \frac{r}{r_0}\right)} \tag{8}$$

$$q_{n+1} = q_n + H \Delta t \tag{9}$$

1.3. Model paramenters

The following material parameters and simulations options will be used to conduct the tests: Young modulus = 20000MPa

Poisson coefficient = 0.3
Hardening/softening modulus = 0.5
Yield stress = 200MPa
Ratio compression/tension = 3
Total time = 10s
Load states = 3
Increment steps per each state = 5

2. Presentation of results with exponential hardening/softening law

The exponential law has two parameters that are indeed, material related: "A" is the velocity at which the function reaches the value of q_∞, this second parameter indicates the limit of evolution of q(r). In order to demonstrate the behaviour of the exponential hardening/softening law, a trial and error test with the values of stress used subsequent sections, has been conducted for obtaining the value of A=0.5. Likewise, $q_\infty = 2$ is chosen to ensure a reasonable extreme value of q(r) in comparison to the internal variable r which has a value of 1.4142. In order to show the results the first load path with non-symmetric domain is tested with the same parametric stress values used in subsequent sections, that is: $\alpha = 800, \beta = -1000, \gamma = 1400$. The effect of hardening will be explained in the following pictures:

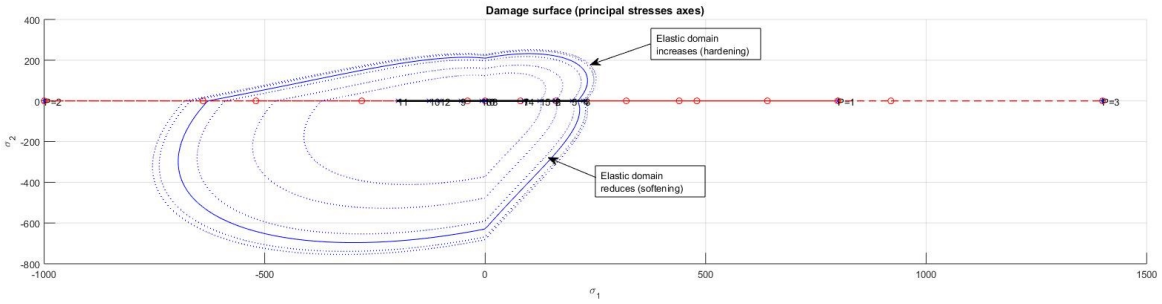

Figure 1: Elastic domain evolution

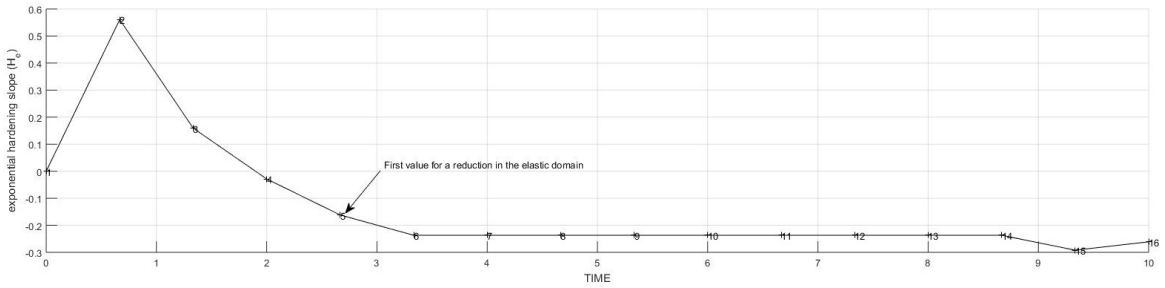

Figure 2: Hardening/softening slope H(r), see equation 8

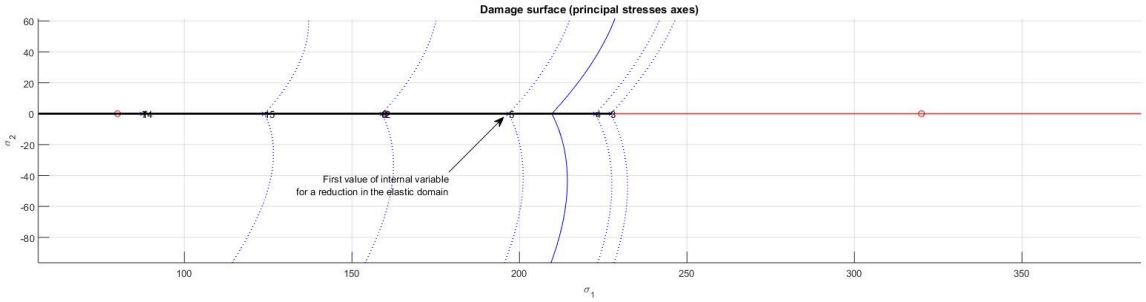

Figure 3: Detail of reduction in elastic domain due to softening

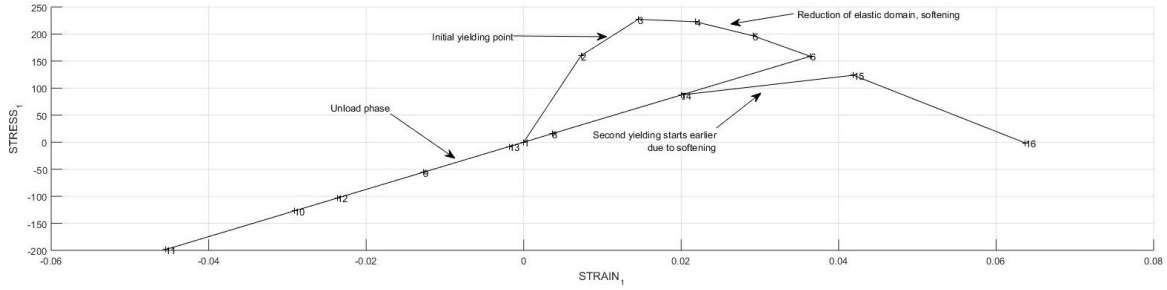

Figure 4: Stress/strain curve (first principal)

3. Assessing the correctness of the implementation

In order to assess the correctness of the implementation, 3 load paths in the stress space are going to be tested. The values of the parameters α, β, γ are chosen to be critical to show the various responses of the 3 different elastics domains; the values given are $\alpha = 800, \beta = 1000, \gamma = 1400$. The result will be presented by showing the linear hardening law with $q_{inf} = 2$, which is the limit value of evolution of the hardening variable q(r).

- First load path:
  $\Delta\bar{\sigma}_1^{(1)} = \alpha; \quad \Delta\bar{\sigma}_2^{(1)} = 0,$ (uniaxial tensile loading)
  $\Delta\bar{\sigma}_1^{(2)} = -\beta; \quad \Delta\bar{\sigma}_2^{(2)} = 0,$ (uniaxial tensile unloading/compressive loading)
  $\Delta\bar{\sigma}_1^{(3)} = \gamma; \quad \Delta\bar{\sigma}_2^{(3)} = 0,$ (uniaxial compressive unloading/ tensile loading)
- Second load path:
  $\Delta\bar{\sigma}_1^{(1)} = \alpha; \quad \Delta\bar{\sigma}_2^{(1)} = 0,$ (uniaxial tensile loading)
  $\Delta\bar{\sigma}_1^{(2)} = -\beta; \quad \Delta\bar{\sigma}_2^{(2)} = -\beta,$ (biaxial tensile unloading/compressive loading)
  $\Delta\bar{\sigma}_1^{(3)} = \gamma; \quad \Delta\bar{\sigma}_2^{(3)} = \gamma,$ (biaxial compressive unloading/ tensile loading)
- Third load path:
  $\Delta\bar{\sigma}_1^{(1)} = \alpha; \quad \Delta\bar{\sigma}_2^{(1)} = \alpha,$ (biaxial tensile loading)
  $\Delta\bar{\sigma}_1^{(2)} = -\beta; \quad \Delta\bar{\sigma}_2^{(2)} = -\beta,$ (biaxial tensile unloading/compressive loading)
  $\Delta\bar{\sigma}_1^{(3)} = \gamma; \quad \Delta\bar{\sigma}_2^{(3)} = \gamma,$ (biaxial compressive unloading/ tensile loading)

3.1. First load path: Symmetric domain

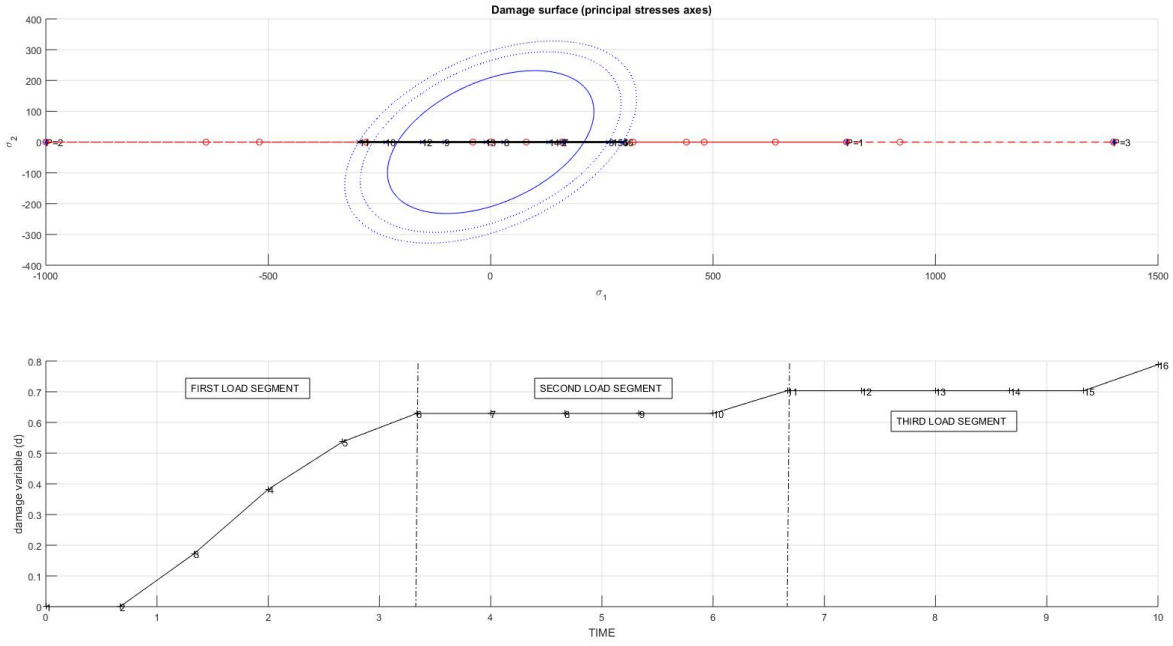

Figure 5: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.2. First load path:"Tension only" domain

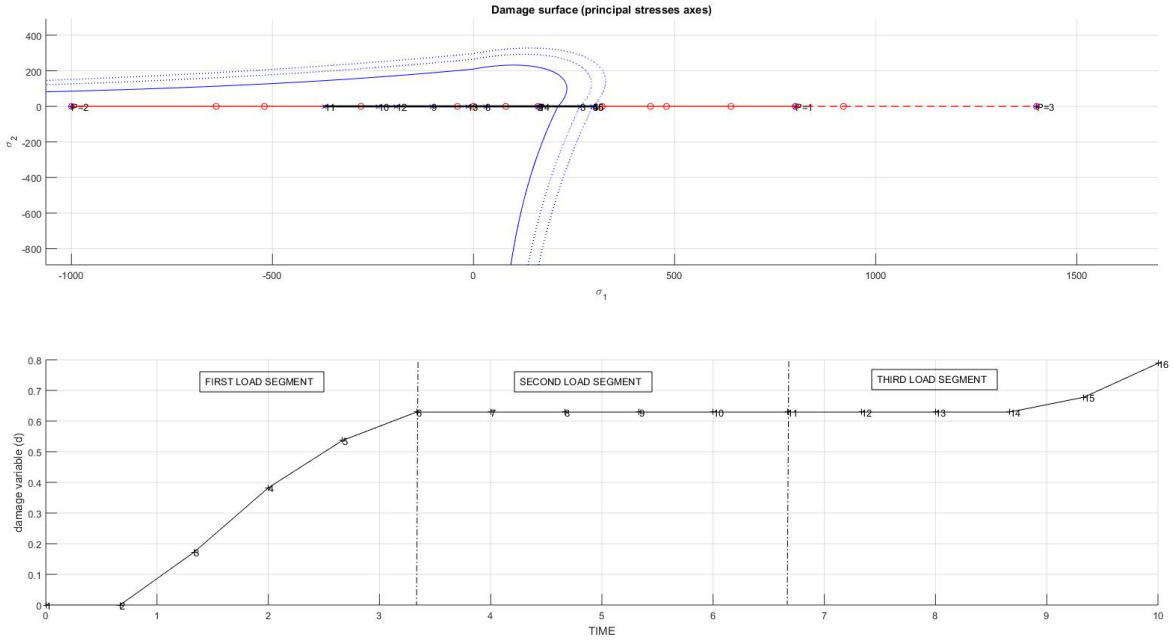

Figure 6: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.3. First load path: Non symmetric domain

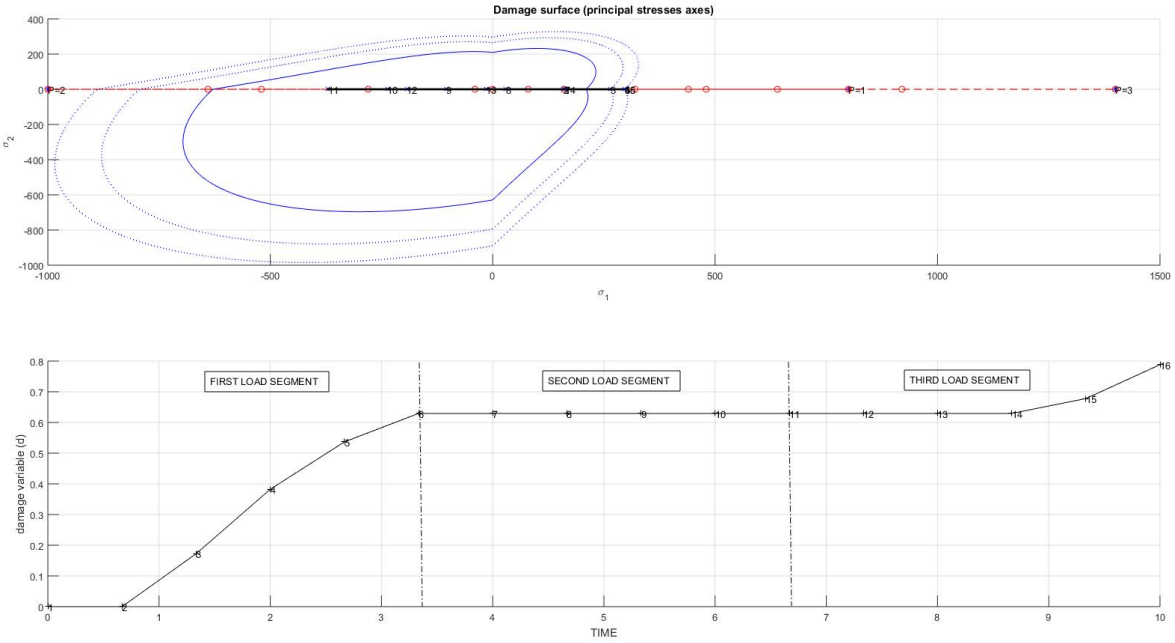

Figure 7: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.4. Second load path: Symmetric domain

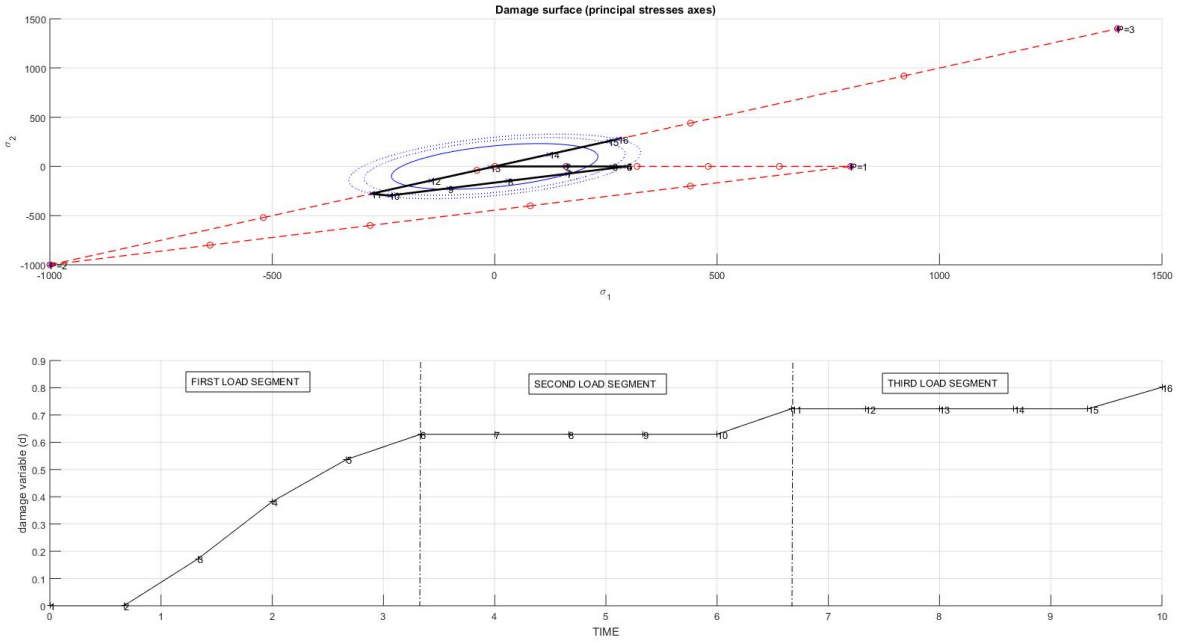

Figure 8: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.5. Second load path: ”Tension only” domain

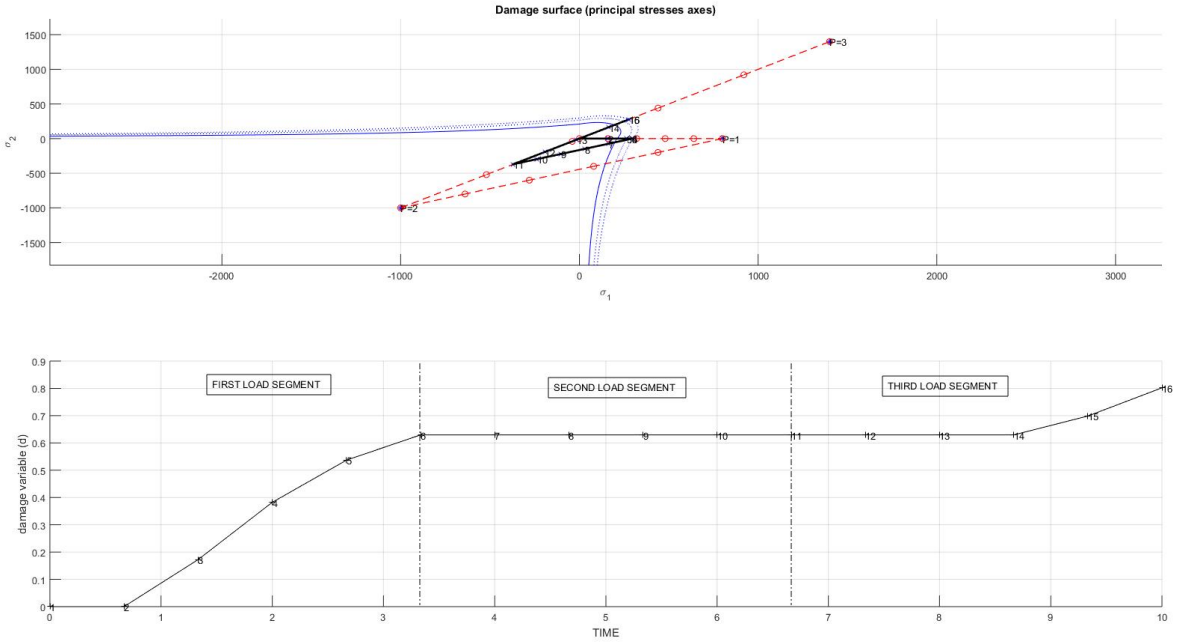

Figure 9: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.6. Second load path: Non symmetric domain

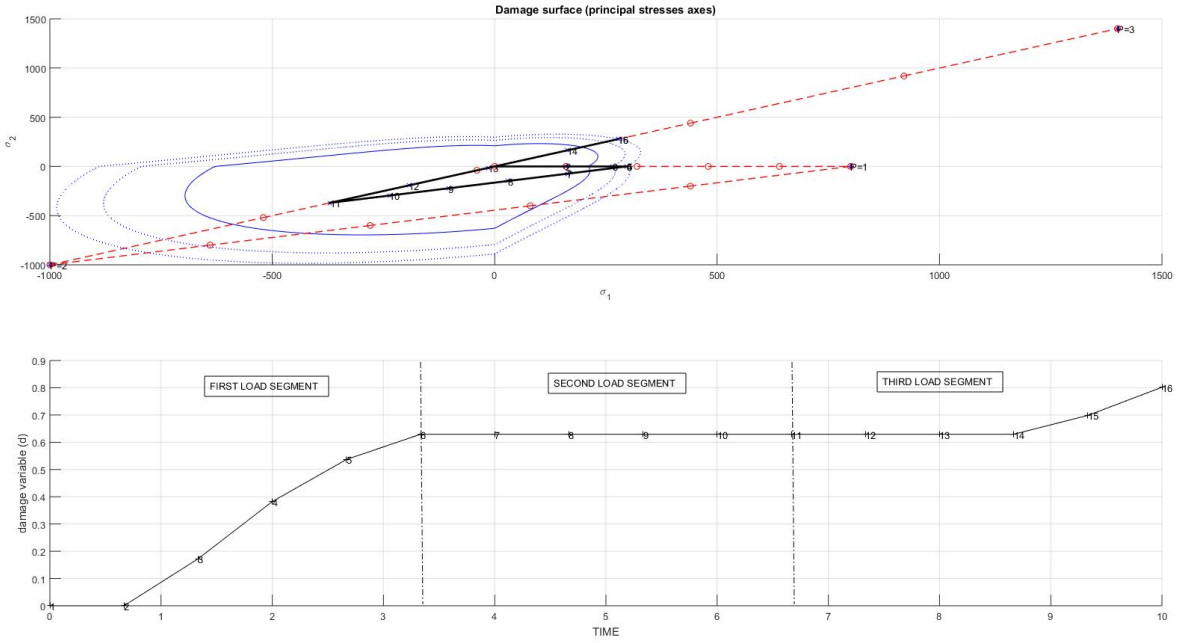

Figure 10: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.7. Third load path: Symmetric domain

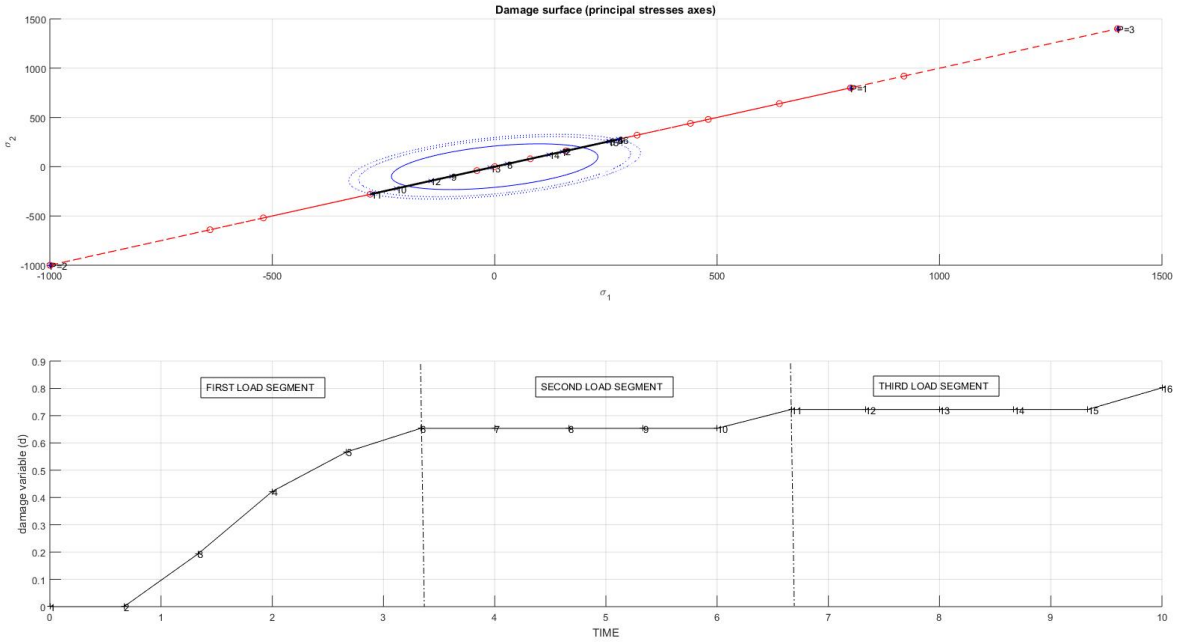

Figure 11: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.8. Third load path: ”Tension only” domain

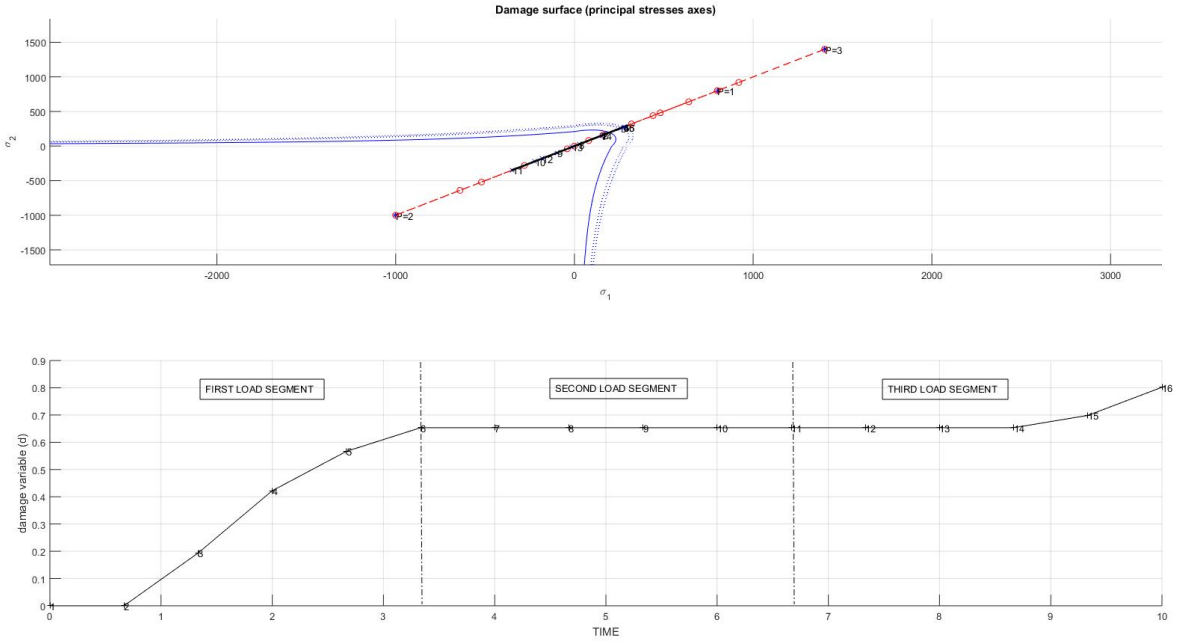

Figure 12: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

3.9. Third load path: Non symmetric domain

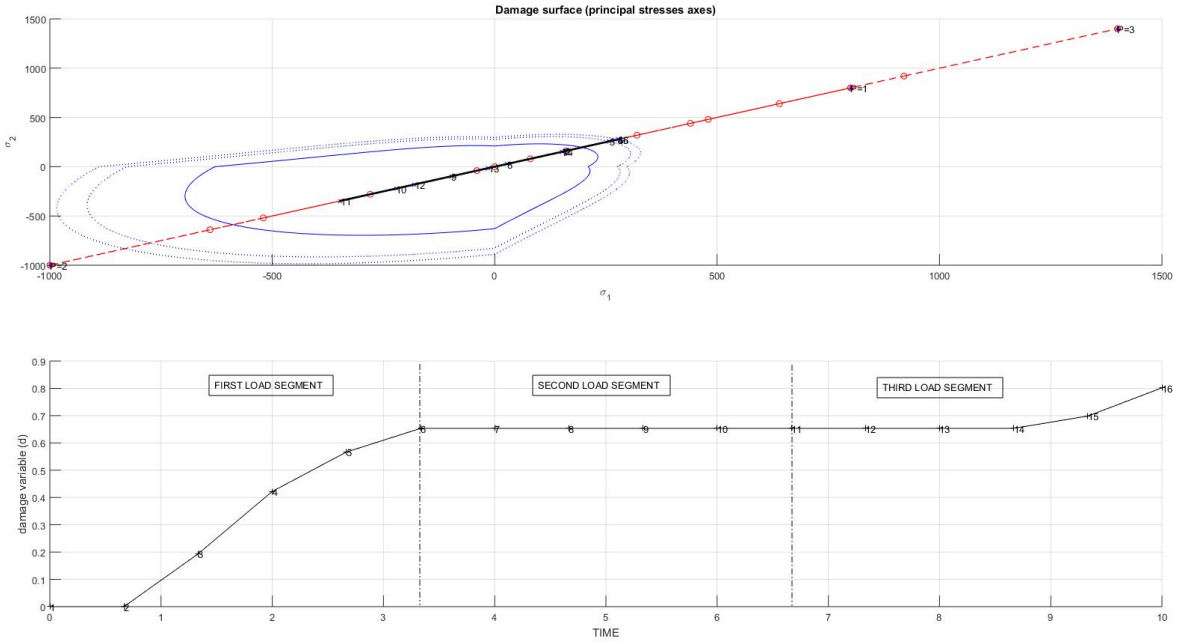

Figure 13: Elastic domain evolution (top), damage parameter (d) evolution (bottom)

4. Load paths: assessment of results

The evolution of the damage parameter shows to be very similar for the three elastic domains as well as for the three load paths. The hardening variable q(r) is reaching its maximum q_{inf} in all simulations during the first step, after this value has been reached the sizes of the domains does not evolve till the third load segment.

- First load segment (all paths)
  The evolution of the damage parameter is similar for all cases in the first load segment for the three load paths since the length of the domain is the same in the quadrant $\sigma^{1+}\sigma^{2+}$.

- Second load segment (all paths)
  The three domain shows similar results with minor differences, due to the domain shape differences. In this load segment the "Tension only" and non symmetrical domain cases are unloading so that the damage variable does not evolve at all, whereas the symmetrical one experiences a slight increase of the damage variable at the end of the segment.

- Third load segment (all paths)
  The symmetric domain presents a slightly smaller evolution in the end of the segments as it had already evolved in the second stage, whereas the symmetric and non symmetric domains evolves slightly due to the fact that in the previous segment were being unloaded.

5. Implementation of the numerical algorithm for a rate dependent case

For the implementation of the rate dependent case the subroutine *rmap_dano1.m* has been mainly modified. The following formulas has been implemented in order to calculate the evolution of the internal variable, in this case depending not only in strain but also in time. In order to calculate the evolution along time the so-called "Alpha method" is going to be used.

$$r_{n+1} = \frac{[\eta - \Delta t(1-\alpha)]}{\eta + \alpha\Delta t} r_n + \frac{\Delta}{\eta + \alpha\Delta t}\tau_{\epsilon+\alpha} \tag{10}$$

$$r_{n+\alpha} = (1-\alpha)r_n + \alpha r_{n+1} \tag{11}$$

6. Assessment of the effect in the stress/strain curves (first principals chosen for the assessment) of the variation of the following parameters: viscosity parameter, strain rate and alpha

The linear hardening law it has been chosen to present the results of this section. The following load path has been chosen in order to assess the effect of each parameter in the curves

$\Delta\bar{\sigma}_1^{(1)} = 200; \quad \Delta\bar{\sigma}_2^{(1)} = 200,$
$\Delta\bar{\sigma}_1^{(2)} = 400; \quad \Delta\bar{\sigma}_2^{(2)} = 400,$
$\Delta\bar{\sigma}_1^{(3)} = 400; \quad \Delta\bar{\sigma}_2^{(3)} = -600,$

6.1. Rate dependent model parameters

The following material parameters and simulations options will be used to conduct the tests:

Young modulus = 20000MPa
Poisson coefficient = 0.3
Hardening/softening modulus = 0.5
Yield stress = 200MPa
Total time = 10s
(will vary for strain rate tests) Load states = 3
Increment steps per each state = 5
Viscosity coefficient η=0.6

6.2. Effect of the variation of the viscosity parameter η

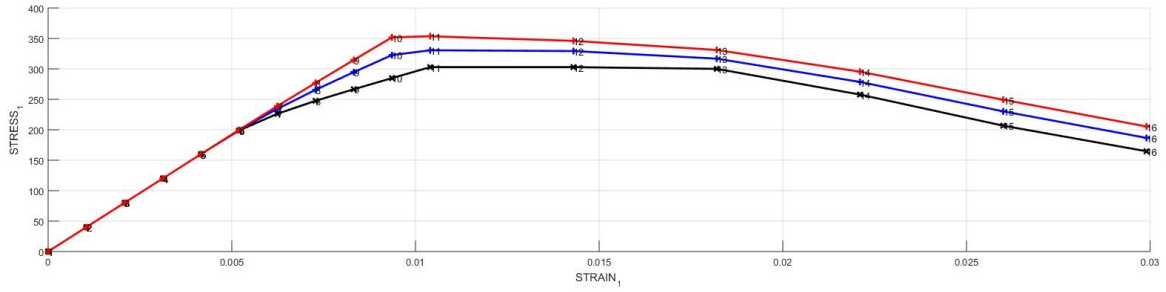

Figure 14: black $\eta = 0.3$, blue $\eta = 0.6$ and red $\eta = 0.9$

6.2.1. Assessment of results of viscosity variation

The effect of viscosity in the stress/strain curve (first principals) is the expected from a physical point of view. As the viscosity increases the stress increases due to the fact that we are imposing the same strain increments to each test, therefore a higher stress value is needed in higher viscosity in order to achieve the same strain increment.

6.3. Effect of the variation of the strain rate $\dot{\epsilon}$

In order to vary the strain rate, the total simulation time (T) is going to be reduced gradually, so that the lower the simulation time to apply the same load path, the higher the strain rate

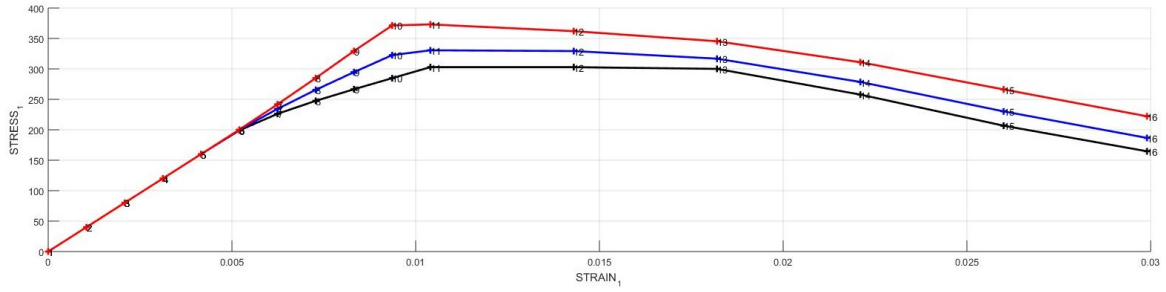

Figure 15: black T=10s, blue T=5s and red T=2.5s

6.3.1. Assessment of results of strain rate $\dot{\epsilon}$

It can be observed that maintaining the viscosity parameter constant, the variation in strain rate produce an analogous effect to the variation of η in the stress/strain (first principals) curves. That is, to achieve the same value of strain at a higher velocity is traduced in an increase of the stress values.

6.4. Effect of the variation of α parameters

The model tends to behave as inviscid as alpha tends to 1. When alpha reaches value 1 and the viscosity parameter η=0 then the inviscid model rate independent implicit scheme is recovered. The solution of the differential equation of the evolution of the internal value, only has second order accuracy for $\alpha = 0.5$, this is shown in the smoothness of the green curve compared with the remaining curves. The yellow path in the stress space shows how the solution is behaving as inviscid for $\mathtt{alpha} = 1$ and $\eta = 0$. It is posible to observe that the stress path points no longer remain outside the plastic domain.

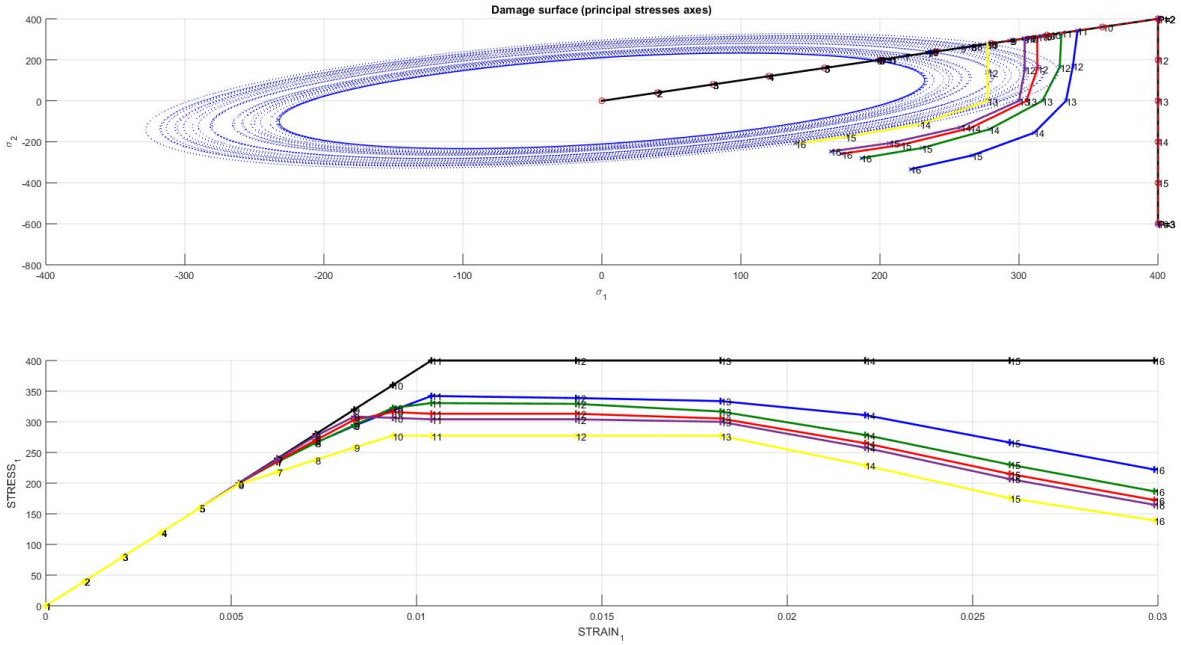

Figure 16: black $\alpha = 0$, blue $\alpha = 1/4$, green $\alpha = 1/2$, red $\alpha = 3/4$, purple $\alpha = 1$, yellow ($\alpha = 1$ and $\eta = 0$) - Top, evolution of elastic domain with α variation / bottom, stress/strain curve (first principals) variation with value of α

MAIN FILE: main_nointeractive.m

```
clc
clear all
addpath('AUX_SUBROUTINES');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program for modelling damage model
% (Elemental gauss point level)
% -----------------
% Developed by J.A. Hdez Ortega
% 20-May-2007, Universidad PolitÃ©cnica de CataluÃ±a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%profile on

% ------------------------
% ****************
% INPUTS
% ****************

% YOUNG's MODULUS
% ---------------
YOUNG_M = 20000 ;
% Poisson's coefficient
% ------------------------
POISSON = 0.3 ;
% Hardening/softening modulus
% ---------------------------
HARDSOFT_MOD = 0.5;
% Yield stress
% ------------
YIELD_STRESS = 200;
% Problem type  TP = {'PLANE STRESS','PLANE STRAIN','3D'}
% ------------------------ = 1            =2         =3
% ------------
ntype= 2 ;
% Model    PTC = {'SYMMETRIC','TENSION','NON-SYMMETRIC'} ;
%                     = 1         = 2         = 3
% ---------------------------------------------------
MDtype =1;
% Ratio compression strength / tension strength
% ---------------------------------------------
n = 3 ;
% SOFTENING/HARDENING TYPE
% ------------------------
HARDTYPE = 'LINEAR' ; %{LINEAR,EXPONENTIAL}
% VISCOUS/INVISCID
% ------------------------
VISCOUS = 'YES' ;
% Viscous coefficient ----
% ------------------------
eta = 0 ;
% TimeTotal (initial = 0) ----
% ------------------------
TimeTotal = 10 ;
% Integration coefficient ALPHA
% ------------------------
ALPHA_COEFF = 1;
% Points ---------------------------
% -----------------------------------
nloadstates = 3 ;
SIGMAP = zeros(nloadstates,2) ;
```

```
%PATHS NON VISCOUS CASE
%PATH_1
%SIGMAP(1,:) =[800 0];
%SIGMAP(2,:) =[-1000 0];
%SIGMAP(3,:) =[1400 0];
%PATH_2
%SIGMAP(1,:) =[800 0];
%SIGMAP(2,:) =[-1000 -1000];
%SIGMAP(3,:) =[1400 1400];
%PATH_3
%SIGMAP(1,:) =[800 800];
%SIGMAP(2,:) =[-1000 -1000];
%SIGMAP(3,:) =[1400 1400];

%PATH VISCOUS CASE
%PATH_3
SIGMAP(1,:) =[200 200];
SIGMAP(2,:) =[400 400];
SIGMAP(3,:) =[400 -600];

% Number of time increments for each load state
% ---------------------------------------
istep = 5*ones(1,nloadstates) ;

% VARIABLES TO PLOT
vpx = 'STRAIN_1'; % AVAILABLE OPTIONS: 'STRAIN_1', 'STRAIN_2'
%                   '|STRAIN_1|', '|STRAIN_2|'
% 'norm(STRAIN)', 'TIME'
vpy = 'STRESS_1';            % AVAILABLE OPTIONS: 'STRESS_1', 'STRESS_2'
%                   '|STRESS_1|', '|STRESS_2|'
% 'norm(STRESS)', 'TIME', 'DAMAGE VAR.','hardening variable (q)','damage
variable (d)'
%  'internal variable (r)'

%  3) LABELPLOT{ivar}              --> Cell array with the label string for
%                                    variables of "varplot"
%
LABELPLOT = {'hardening variable (q)','internal variable (r)','damage variable
(d)','exponential hardening slope (H_e)'};

%%%%%%%%%%%%%%%%%%55 END INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Plot Initial Damage Surface and effective stress path
strain_history =
PlotIniSurf(YOUNG_M,POISSON,YIELD_STRESS,SIGMAP,ntype,MDtype,n,istep);



E       = YOUNG_M      ;
nu      = POISSON      ;
sigma_u = YIELD_STRESS ;



switch  HARDTYPE
    case 'LINEAR'
        hard_type = 0  ;
    otherwise
        hard_type = 1  ;
end
switch  VISCOUS
    case 'YES'
        viscpr = 1     ;
    otherwise
```

```
        viscpr = 0     ;
end


Eprop   = [E nu HARDSOFT_MOD sigma_u hard_type viscpr eta ALPHA_COEFF]
;



% DAMAGE MODEL
% ------------
[sigma_v,vartoplot,LABELPLOT_out,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strai
n_history,MDtype,n,TimeTotal);



try; LABELPLOT;catch;LABELPLOT = LABELPLOT_out ; end ;

% PLOTTING
% -------

ncolores = 3 ;
colores =  ColoresMatrix(ncolores);
markers = MarkerMatrix(ncolores) ;
hplotLLL = [] ;

for i = 2:length(sigma_v)
    stress_eig  = sigma_v{i} ; %eigs(sigma_v{i}) ;
    tstress_eig = sigma_v{i-1}; %eigs(sigma_v{i-1}) ;
    hplotLLL(end+1) = plot([tstress_eig(1,1) stress_eig(1,1) ],[tstress_eig(2,2)
stress_eig(2,2)],'LineWidth',2,'color',colores(1,:),'Marker',markers{1},'MarkerS
ize',2);
    plot(stress_eig(1,1),stress_eig(2,2),'bx')
    text(stress_eig(1,1),stress_eig(2,2),num2str(i))


    % SURFACES
    % -----

end






% % SURFACES
% % -----
% if(aux_var(1)>0)
%     hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6), 'r:',MDtype,n );
%     set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1);
% end



DATA.sigma_v    = sigma_v     ;
DATA.vartoplot  = vartoplot   ;
DATA.LABELPLOT  = LABELPLOT   ;
DATA.TIMEVECTOR = TIMEVECTOR  ;
DATA.strain = strain_history ;



plotcurvesNEW(DATA,vpx,vpy,LABELPLOT,vartoplot) ;
```

FUNCTION: rmap_dano1.m

```
function [sigma_n1,hvar_n1,aux_var] = rmap_dano1
(eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t)

%*************************************************************************
*******
%*                                        *
%*           Integration Algorithm for a isotropic damage model
%*
%*
*
%*            [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n1,hvar_n,Eprop,ce)
*
%*
*
%* INPUTS              eps_n1(4)   strain (almansi)    step n+1
*
%*                                 vector R4    (exx eyy exy ezz)
*
%*                     hvar_n(6)   internal variables , step n
*
%*                                 hvar_n(1:4) (empty)
*
%*                                 hvar_n(5) = r  ; hvar_n(6)=q
*
%*                     Eprop(:)    Material parameters
*
%*
%*                     ce(4,4)     Constitutive elastic tensor
*
%*
*
%* OUTPUTS:            sigma_n1(4) Cauchy stress  , step n+1
*
%*                     hvar_n(6)   Internal variables , step n+1
*
%*                        Auxiliar variables for computing const. tangent tensor
*
%*************************************************************************
********


hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
H_e     = hvar_n(7);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5);
visc = Eprop(6);
eta = Eprop(7);
alpha = Eprop(8);
%*************************************************************************
******


%*************************************************************************
******
%*       initializing                                                  %*
```

```
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
% if(r_n<=0.d0)
%     r_n=r0;
%     q_n=r0;
% end
%*************************************************************************
******


%*************************************************************************
******
%*       Damage surface
%*
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
%*************************************************************************
******


%*************************************************************************
******
%*   Ver el Estado de Carga
%*
%*   --------->    fload=0 : elastic unload
%*
%*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)
%*
fload=0;
inf_q=2;%limiting the evolution of the elastic domain in the linear hardening
case
if visc==0
    if(rtrial > r_n)
        %*   Loading

        fload=1;
        delta_r=rtrial-r_n;
        r_n1=rtrial;
        if hard_type == 0
            %  Linear
            q_n1=q_n+H*delta_r;
            if (q_n1<zero_q)
                q_n1=zero_q;
            end
            if (q_n1>inf_q)
                q_n1=inf_q;
            end
        else
            A=0.5;
            H_e=A*(inf_q-r_n)/r_n*exp(A*(1-rtrial/r_n));
            q_n1=q_n+H_e*delta_r;
        end
    else
        %*     Elastic load/unload
        fload=0;
        r_n1= r_n  ;
        q_n1= q_n  ;
    end
else
    rtrial=(1-alpha)*r_n+alpha*rtrial;
    delta_r=rtrial-r_n;
    r_n1=(((eta-delta_t*(1-alpha))/(eta+alpha*delta_t))*r_n)+((delta_t/
(eta+alpha*delta_t))*rtrial);
    if rtrial > r_n
        fload=1;
```

```
        if hard_type == 0
            %  Linear
            q_n1=q_n+H*delta_r;
            if (q_n1<zero_q)
                q_n1=zero_q;
            end
            if (q_n1>inf_q)
                q_n1=inf_q;
            end
        else
            A=0.5;
            H_e=A*(inf_q-r_n)/r_n*exp(A*(1-rtrial/r_n));
            q_n1=q_n+H_e*delta_r;
        end
    else
        r_n1=r_n;
        q_n1=q_n;
    end
% Damage variable
% ---------
end

dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
%hold on
%plot(sigma_n1(1),sigma_n1(2),'bx')

%*************************************************************************
******


%*************************************************************************
******
%* Updating historic variables                                          %*
%  hvar_n1(1:4)  = eps_n1p;
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;
hvar_n1(7)= H_e ;
%*************************************************************************
******




%*************************************************************************
******
%* Auxiliar variables
%*
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
%*aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
%*************************************************************************
******
```

FUNCTION: damage_main.m

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strain,MD
type,n,TimeTotal)
global hplotSURF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% CONTINUUM DAMAGE MODEL
% ----------------------
% Given the almansi strain evolution ("strain(totalstep,mstrain)") and a set of
% parameters and properties, it returns the evolution of the cauchy stress and
other  variables
% that are listed below.
%
% INPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% ------------------------------------------------------------------
% Eprop(1) = Young's modulus  (E)
% Eprop(2) = Poisson's coefficient (nu)
% Eprop(3) = Hardening(+)/Softening(-) modulus (H)
% Eprop(4) = Yield stress (sigma_y)
% Eprop(5) = Type of Hardening/Softening law  (hard_type)
%            0 --> LINEAR
%            1 --> Exponential
% Eprop(6) = Rate behavior (viscpr)
%            0 --> Rate-independent (inviscid)
%            1 --> Rate-dependent   (viscous)
%
% Eprop(7) = Viscosity coefficient (eta)  (dummy if inviscid)
% Eprop(8) = ALPHA coefficient (for time integration), (ALPHA)
%             0<=ALPHA<=1 , ALPHA = 1.0 --> Implicit
%                           ALPHA = 0.0 --> Explicit
%            (dummy if inviscid)
%
% ntype    = PROBLEM TYPE
%            1 : plane stress
%            2 : plane strain
%            3 : 3D
%
% istep = steps for each load state (istep1,istep2,istep3)
%
% strain(i,j) = j-th component of the linearized strain vector at the i-th
%               step, i = 1:totalstep+1
%
% MDtype      = Damage surface criterion %
%            1 : SYMMETRIC
%            2 : ONLY-TENSION
%            3 : NON-SYMMETRIC
%
%
% n          = Ratio compression/tension strength (dummy if MDtype is different
from 3)
%
% TimeTotal  = Interval length
%
%  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%  ------------------------------------------------------------------
%  1) sigma_v{itime}(icomp,jcomp)  --> Component (icomp,jcomp) of the cauchy
%                                      stress tensor at step "itime"
%                                      REMARK: sigma_v is a type of
%                                      variable called "cell array".
%
%
```

```
%  2) vartoplot{itime}              --> Cell array containing variables one
wishes to plot
%                                    --------------------------------------
%   vartoplot{itime}(1) =   Hardening variable (q)
%   vartoplot{itime}(2) =   Internal variable (r)%

%
%  3) LABELPLOT{ivar}              --> Cell array with the label string for
%                                    variables of "varplot"
%
%          LABELPLOT{1} => 'hardening variable (q)'
%          LABELPLOT{2} => 'internal variable'
%
%
%  4) TIME VECTOR  - >
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%

% SET LABEL OF "vartoplot" variables  (it may be defined also outside this
function)
% ----------------------------------
 LABELPLOT = {'hardening variable (q)','internal variable'};

E      = Eprop(1) ; nu = Eprop(2) ;
viscpr = Eprop(6) ;
sigma_u = Eprop(4);



if ntype == 1
    menu('PLANE STRESS has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
else
    mstrain = 4    ;
    mhist   = 7    ;%enlarged +1 element to include exponential hardening slope
"H_e" in list of outputs
end

if viscpr == 1
    % Comment/delete lines below once you have implemented this case
    % *******************************************************
    %menu({'Viscous model has not been implemented yet. '; ...
    %    'Modify files "damage_main.m","rmap_dano1" ' ; ...
    %     'to include this option'},  ...
    %     'STOP');
    %error('OPTION NOT AVAILABLE')
else
end


totalstep = sum(istep) ;


% INITIALIZING GLOBAL CELL ARRAYS
% -------------------------------
sigma_v = cell(totalstep+1,1) ;
TIMEVECTOR = zeros(totalstep+1,1) ;
delta_t = TimeTotal./istep/length(istep) ;


% Elastic constitutive tensor
```

```
% ---------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
% Historic variables
% hvar_n(1:4) --> empty
% hvar_n(5) = q --> Hardening variable
% hvar_n(6) = r --> Internal variable
hvar_n  = zeros(mhist,1)  ;

% INITIALIZING  (i = 1) !!!!
% ***********i*
i = 1 ;
r0 = sigma_u/sqrt(E);
H_e=0.5*((3-r0)/r0)*exp(0.5*(1-r0/r0));%initial exponential hardening slope
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_n
hvar_n(7) = H_e; % H_n
eps_n1 = strain(i,:) ;
sigma_n1 =ce*eps_n1'; % Elastic
sigma_v{i}  = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0
sigma_n1(4)];

nplot = 3 ;
vartoplot = cell(1,totalstep+1) ;
vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5)  ; %  Damage variable (d)

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;
        % Total strain at step "i"
        % ------------------------
        eps_n1 = strain(i,:) ;

%*************************************************************************
*******
        %*      DAMAGE MODEL

        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t(iload));
        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6),
'r:',MDtype,n );
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1)
;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
        %*************************************************************
        % GLOBAL VARIABLES
        % ***************
        % Stress
        % ------
```

```
        m_sigma=[sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0
sigma_n1(4)];
        sigma_v{i} =  m_sigma ;

        % VARIABLES TO PLOT (set label on cell array LABELPLOT)
        % ----------------
        vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
        vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
        vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5)  ; %  Damage variable (d)
        vartoplot{i}(4) = hvar_n(7) ; %Exponential hardening slope (H_e)
    end
end
end
```

FUNCTION: Modelos_de_dano1.m

```
function [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n)
%*************************************************************************
*******
%*          Defining damage criterion surface
%*
%*
%*
%*
%*                          MDtype=  1      : SYMMETRIC
%*
%*                          MDtype=  2      : ONLY TENSION
%*
%*                          MDtype=  3      : NON-SYMMETRIC
%*
%*
%*
%*
%*
%* OUTPUT:
%*
%*                          rtrial
%*
%*************************************************************************
*******



%*************************************************************************
*******
if (MDtype==1)      %* Symmetric
rtrial= sqrt(eps_n1*ce*eps_n1');

elseif (MDtype==2)  %* Only tension
sigma_v=eps_n1*ce;
for i=1:4
    if sigma_v(i)<0
        sigma_v(i)=0;
    end
end
rtrial=sqrt(sigma_v*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
sigma_v=eps_n1*ce;
sigma_v_mac=zeros(1,4);
for i=1:4
```

```
    sigma_v_mac(i)=sigma_v(i);
    if sigma_v(i)<0
        sigma_v_mac(i)=0;
    end
end
tetha=(sigma_v_mac(1)+sigma_v_mac(2)+sigma_v_mac(4))/(abs(sigma_v(1))
+abs(sigma_v(2))+abs(sigma_v(4)));
rtrial=(tetha+(1-tetha)/n)*sqrt(sigma_v*eps_n1');

end
%*************************************************************************
*******
return
```

FUNCTION: dibujar_criterio_dano1.m

```
function [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n)
%*************************************************************************
*******
%*          Defining damage criterion surface
%*
%*
%*
%*
%*                           MDtype=  1      : SYMMETRIC
%*
%*                           MDtype=  2      : ONLY TENSION
%*
%*                           MDtype=  3      : NON-SYMMETRIC
%*
%*
%*
%*
%*
%* OUTPUT:
%*
%*                           rtrial
%*
%*************************************************************************
*******



%*************************************************************************
*******
if (MDtype==1)      %* Symmetric
rtrial= sqrt(eps_n1*ce*eps_n1');

elseif (MDtype==2)  %* Only tension
sigma_v=eps_n1*ce;
for i=1:4
    if sigma_v(i)<0
        sigma_v(i)=0;
    end
end
rtrial=sqrt(sigma_v*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
sigma_v=eps_n1*ce;
sigma_v_mac=zeros(1,4);
for i=1:4
    sigma_v_mac(i)=sigma_v(i);
```

```
    if sigma_v(i)<0
        sigma_v_mac(i)=0;
    end
end
tetha=(sigma_v_mac(1)+sigma_v_mac(2)+sigma_v_mac(4))/(abs(sigma_v(1))
+abs(sigma_v(2))+abs(sigma_v(4)));
rtrial=(tetha+(1-tetha)/n)*sqrt(sigma_v*eps_n1');

end
%*************************************************************************
*******
return
```