COMPUTATIONAL SOLID MECHANICS

Assignment 1. Continuum Damage Models

Artemii Sattarov

1. INTRODUCTION

The paper is structured as follows: in Section 2 rate independent problem is considered. The tensile-only and non-symmetric tension-compression models and their assessment are considered. The linear and exponential hardening/softening law implementation is discussed. The Section 3 introduces rate-dependent problems.

2. PART I (RATE INDEPENDENT MODELS)

The following material properties were used in this section: Young's modulus $E = 20000$ Pa, Poisson ratio $\nu = 0.3$, yield stress $\sigma_y = 150$ MPa, time – 10 seconds hardening/softening modulus was chosen positive to represent a hardening $H = 0.3$

Initialization of the problem starts from defining initial r_0, based on the material parameters, such as uniaxial elastic limit and Young modulus. Afterwards the value of the hardening variable is imposed such that $q = r_0$, which corresponds to the undamaged material state.

2.1.1. Non-symmetric tension-compression damage model

The non-symmetric tension-compression model was implemented in the code. This model is used for the materials with different tensile and compressive strengths.

2.1.2. Tension-only damage model

In this model we impose that material can be inelastic in tension, but is always elastic in compression. In order to implement this model the McAuley bracket was used.It was also confirmed that for Poisson ratio $\nu = 0$ the linear dependency is recovered, see Figure 1.

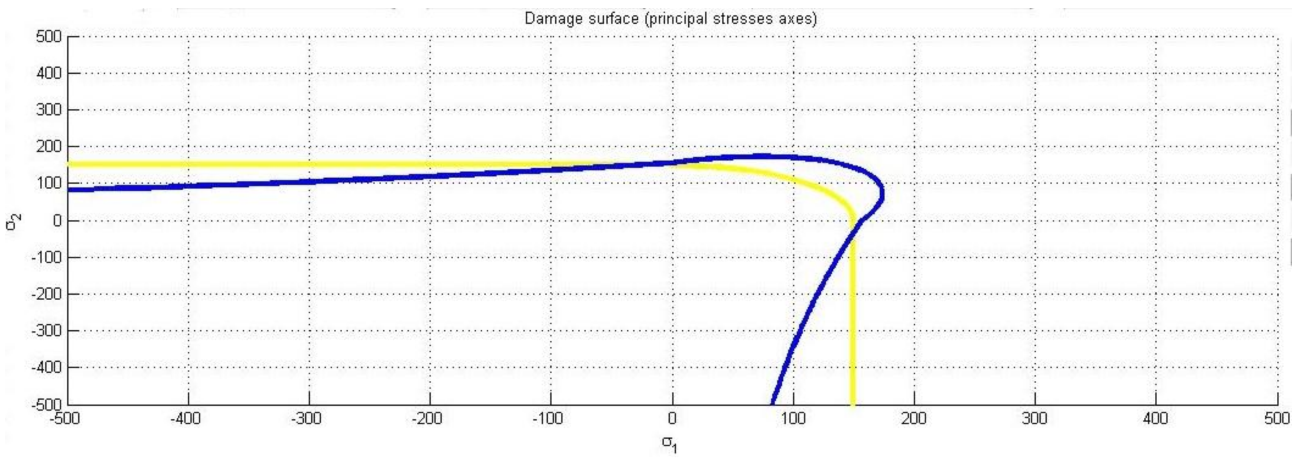


Figure 1. Tension-only damage model with Poisson ratio $\nu = 0$

2.2. Implementation of linear and exponential hardening and softening

The modifications were done in the file rmap_dano1.m, see Annex 1.

The formula for exponential lax was implemented with value of parameter $A = 1$. The softening variable q may not be negative. To avoid negative and zero q the asymptotic boundary of exponential hardening/softening law q_∞ was defined inside the function. The value for q_∞ was set to $10^{-6} \cdot q_0$. It could be also defined using r_0 as $q_0 = r_0$. As a result, we obtained bounded solution both for hardening and softening with their own values of q_∞.

In Figure 2 one can observe hardening/softening variable q as a linear and exponential functions of r. The exponential function is obtained as it had been expected.

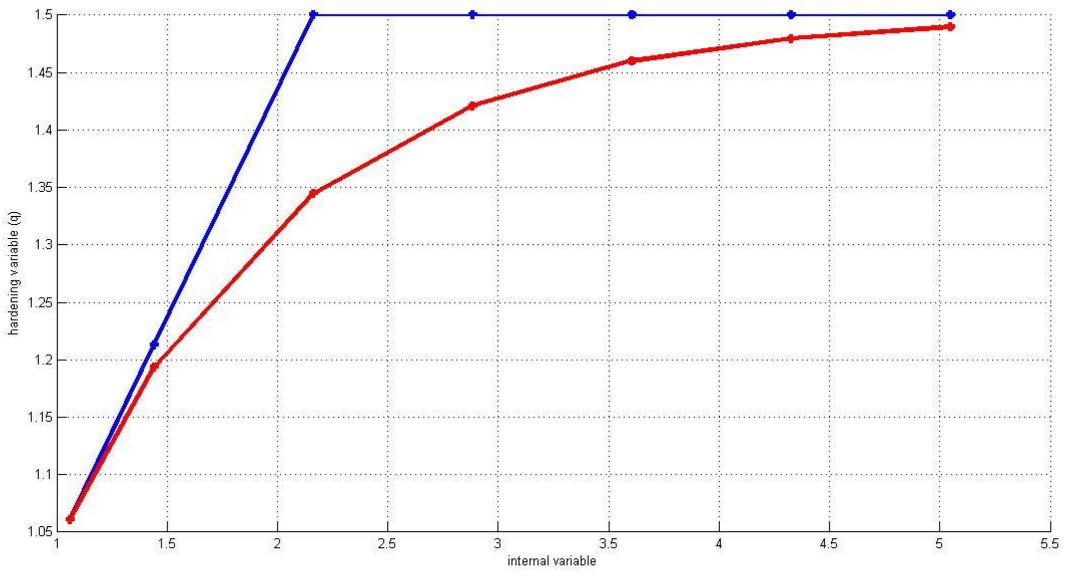


Figure 2. Hardening variable as a function of internal variable for linear (**blue**) and exponential (**red**) hardening/softening law

2.3.1 Assessment of the implementation correctness. Path 1.

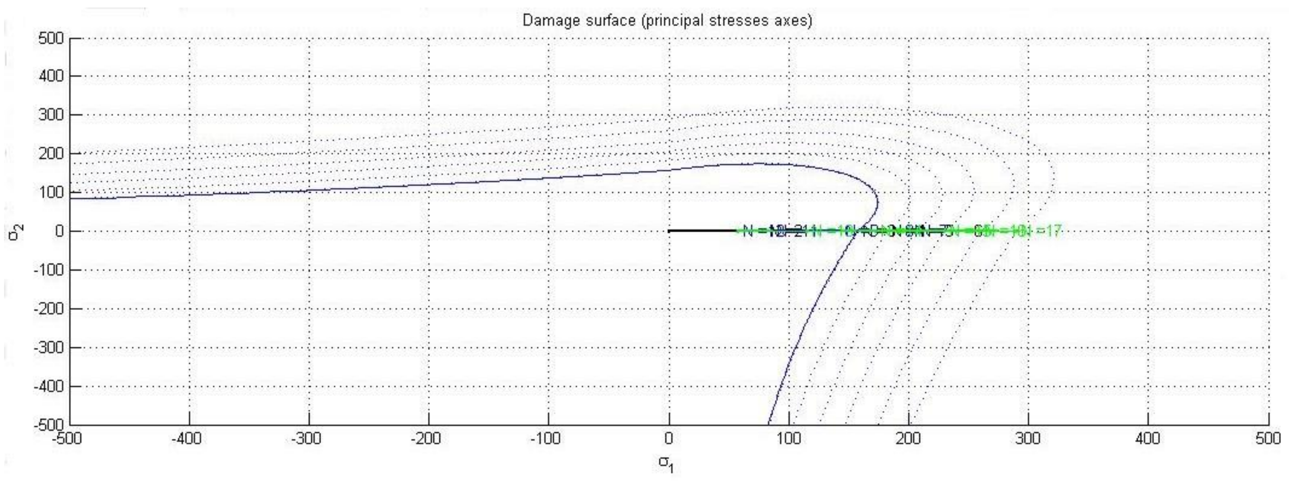
For the Path 1 the following stress segments were chosen:

$\Delta\sigma_1^{(1)} = 400$; $\Delta\sigma_2^{(1)} = 0$ (Segment 1, uniaxial tensile loading)

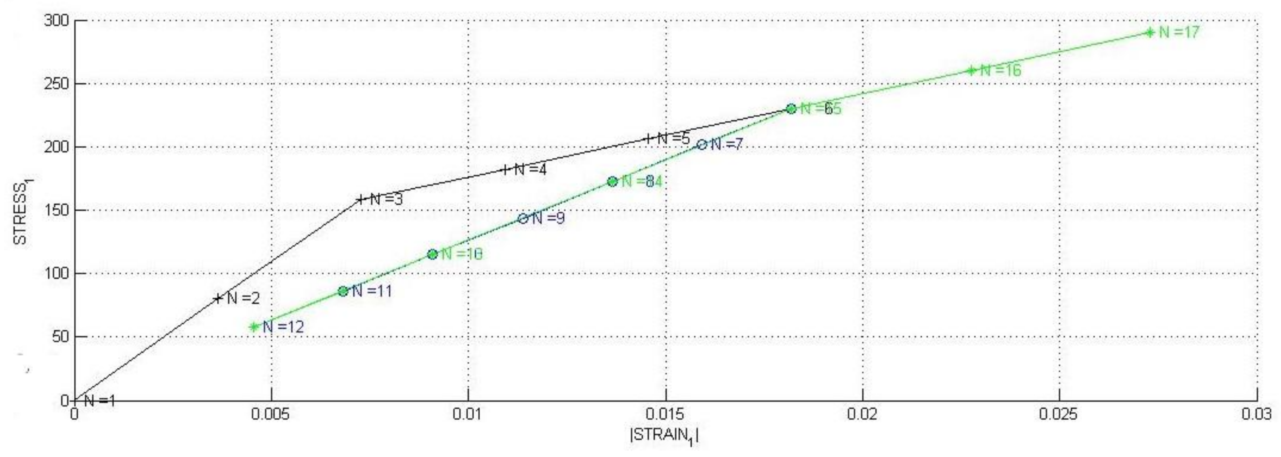
$\Delta\sigma_1^{(2)} = -300$; $\Delta\sigma_2^{(2)} = 0$ (Segment 2, uniaxial tensile unloading/compressive loading)

$\Delta\sigma_1^{(3)} = 500$; $\Delta\sigma_2^{(3)} = 0$ (Segment 3, uniaxial compressive unloading/tensile loading)

It is observed on the strain-stress plot, Figure 3, that the hardening begins after the yield stress (150 MPa) is reached. Unloading is directed to (0,0) point. That confirms validity of the model implementation.

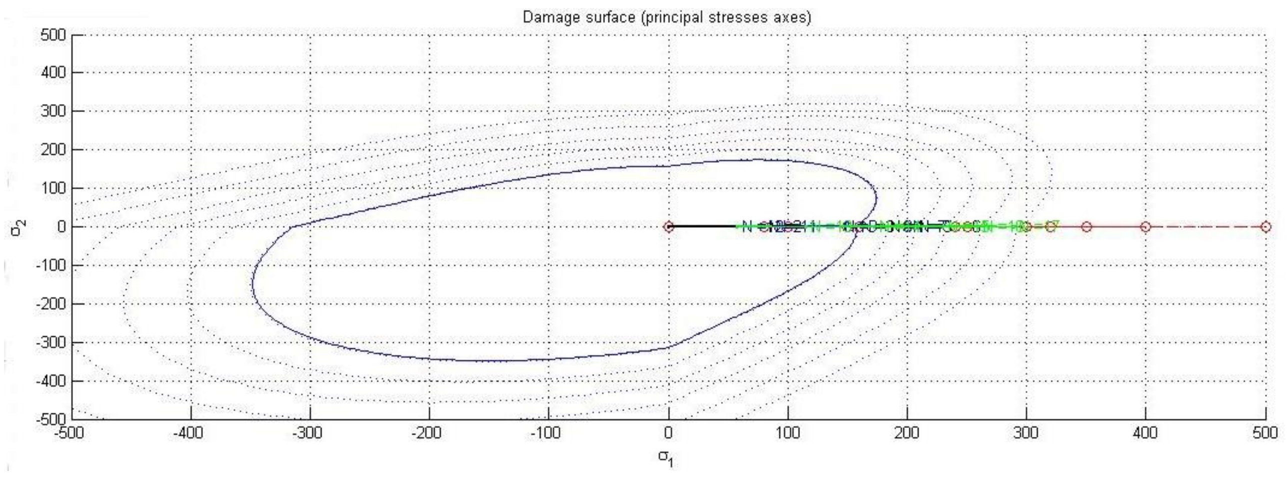


(a)

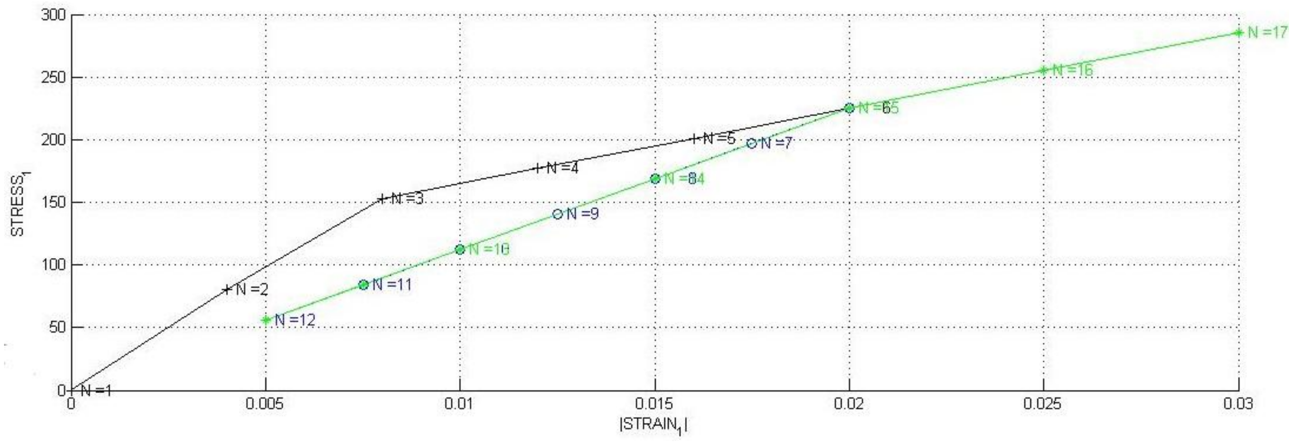


(b)

Figure 3. Tension-only model: (a) damage surface; (b) stress-strain curve



(a)



(b)

Figure 4. Non-symmetric tension-compression damage model: (a) damage surface; (b) stress-strain curve

The stress in direction 2 is equal to 0. The damage surface expands and never decreases. For both models the result is similar in this case.

2.3.2. Assessment of the implementation correctness. Path 2.

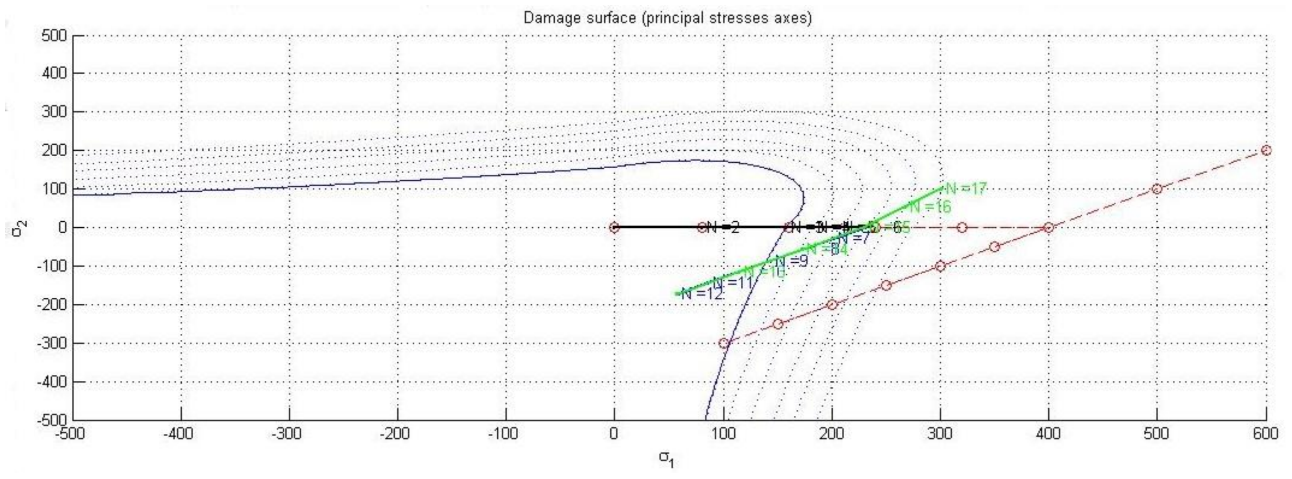
For the Test 2 the following stress paths were chosen:

$\Delta\sigma_1^{(1)} = 400$; $\Delta\sigma_2^{(1)} = 0$ (Segment 1, uniaxial tensile loading)

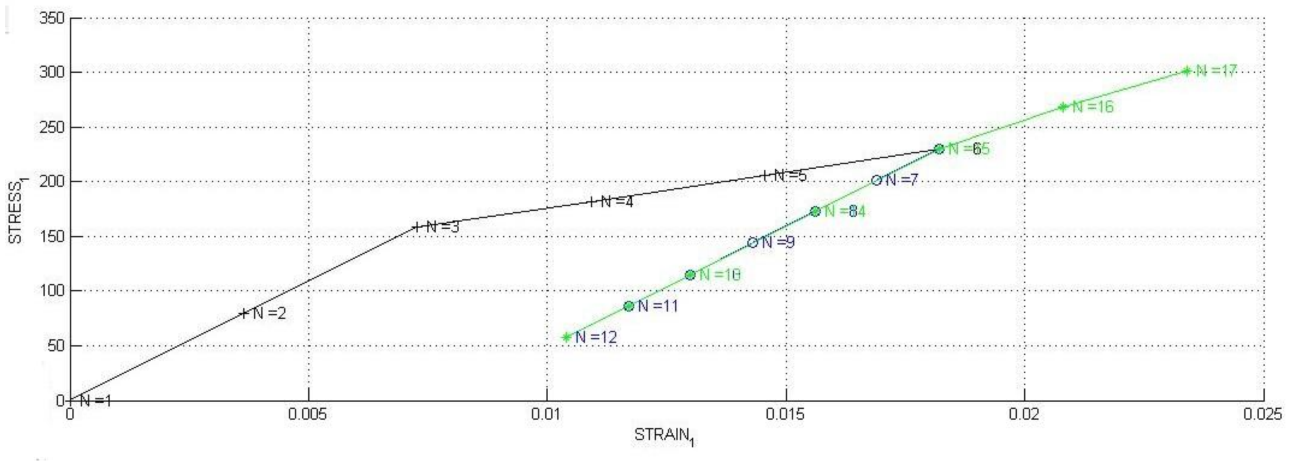
$\Delta\sigma_1^{(2)} = -300$; $\Delta\sigma_2^{(2)} = -300$ (Segment 2, biaxial tensile unloading/compressive loading)

$\Delta\sigma_1^{(3)} = 500$; $\Delta\sigma_2^{(3)} = 500$ (Segment 3, biaxial compressive unloading/tensile loading)

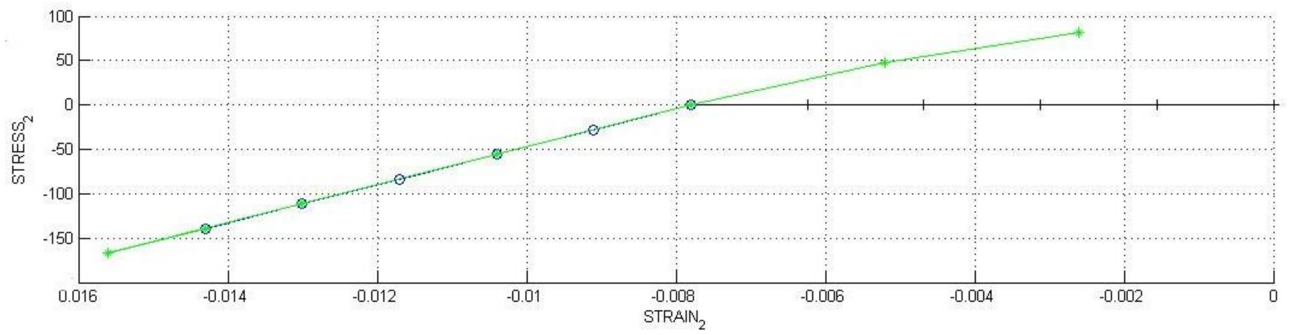
In this case the stress appeared also in the second direcion. The damage surface always expands and never decreases.



(a)

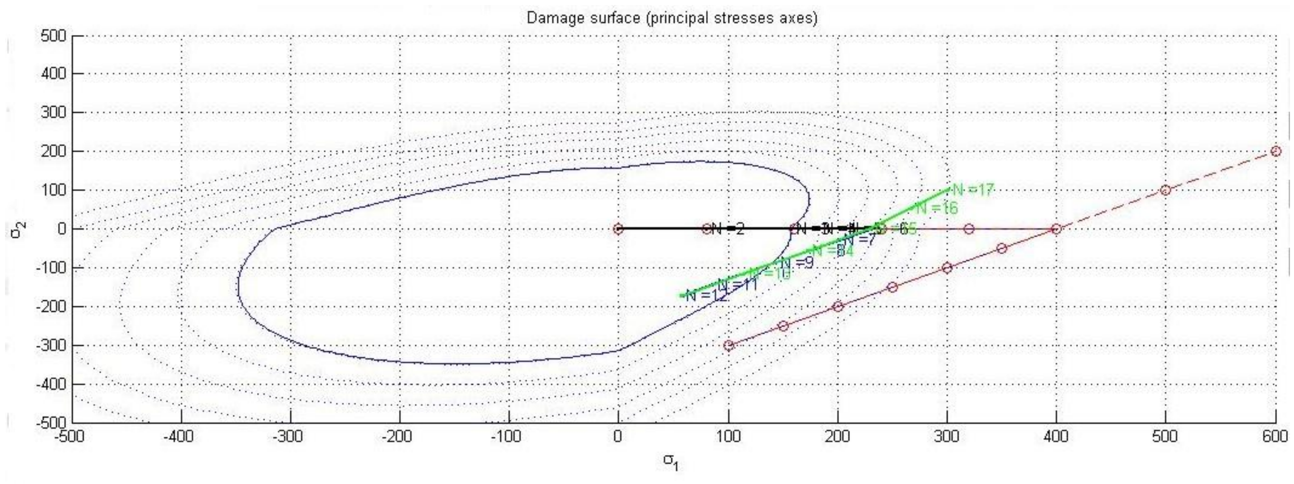


(b)

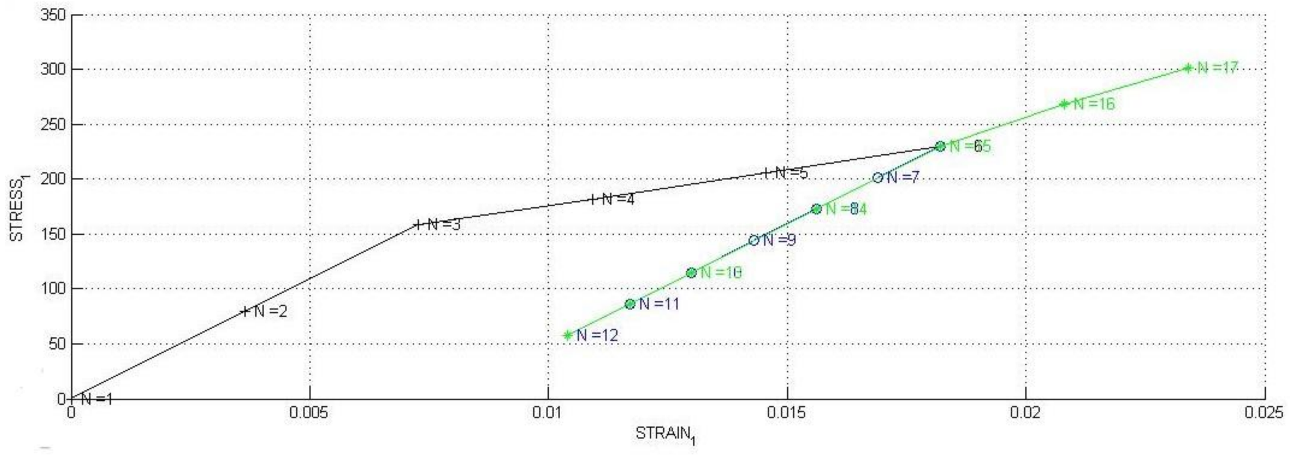


(c)

Figure 5. Tension-only damage model: (a) damage surface; (b) stress-strain curve in direction 1 (c) stress-strain curve in direction 2



(a)



(b)

Figure 6. Non-symmetric tension-compression damage model: (a) damage surface; (b) stress-strain curve

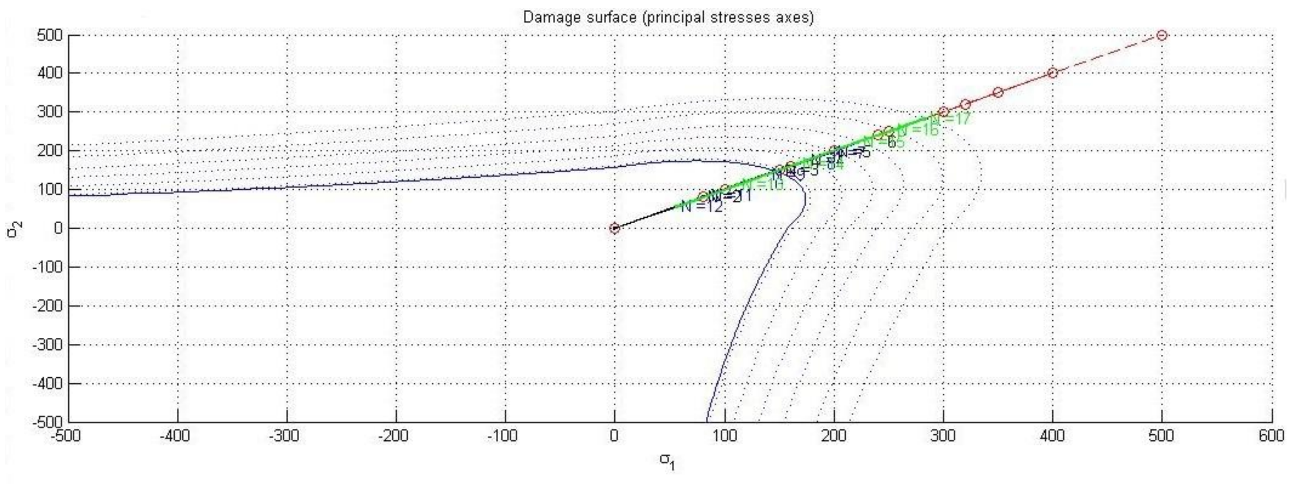
2.3.3. Assessment of the implementation correctness. Path 3.

For the Test 3 the following stress paths were chosen:

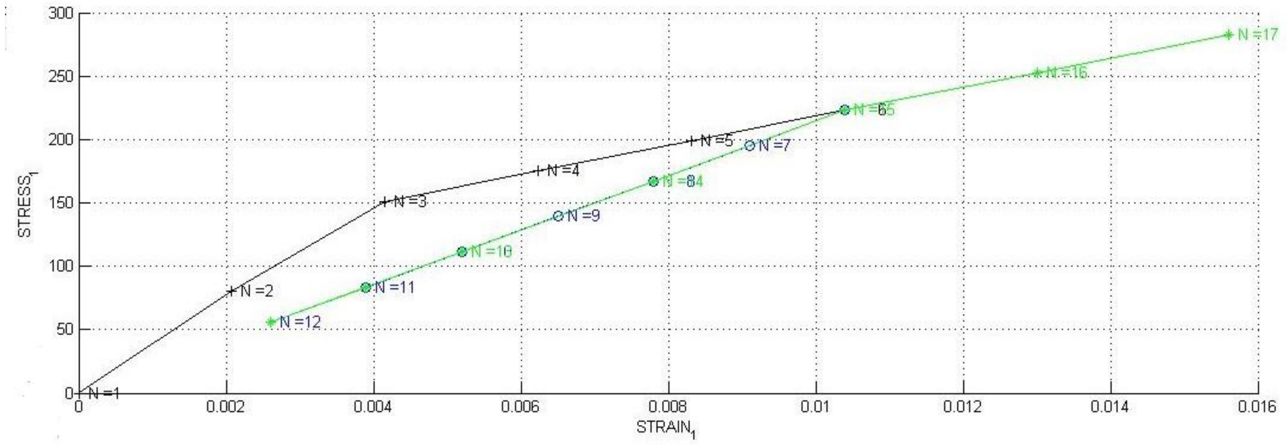
$\Delta\sigma_1^{(1)} = 400$; $\Delta\sigma_2^{(1)} = 400$ (biaxial tensile loading)

$\Delta\sigma_1^{(2)} = -300$; $\Delta\sigma_2^{(2)} = -300$ (biaxial tensile unloading/compressive loading)

$\Delta\sigma_1^{(3)} = 500$; $\Delta\sigma_2^{(3)} = 500$ (biaxial compressive unloading/tensile loading)

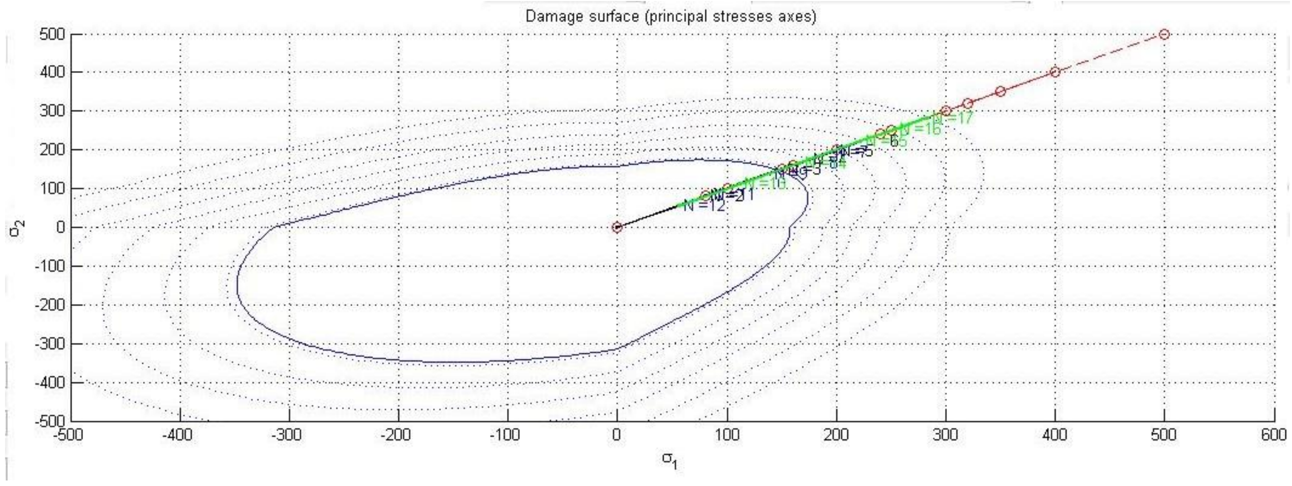


(a)

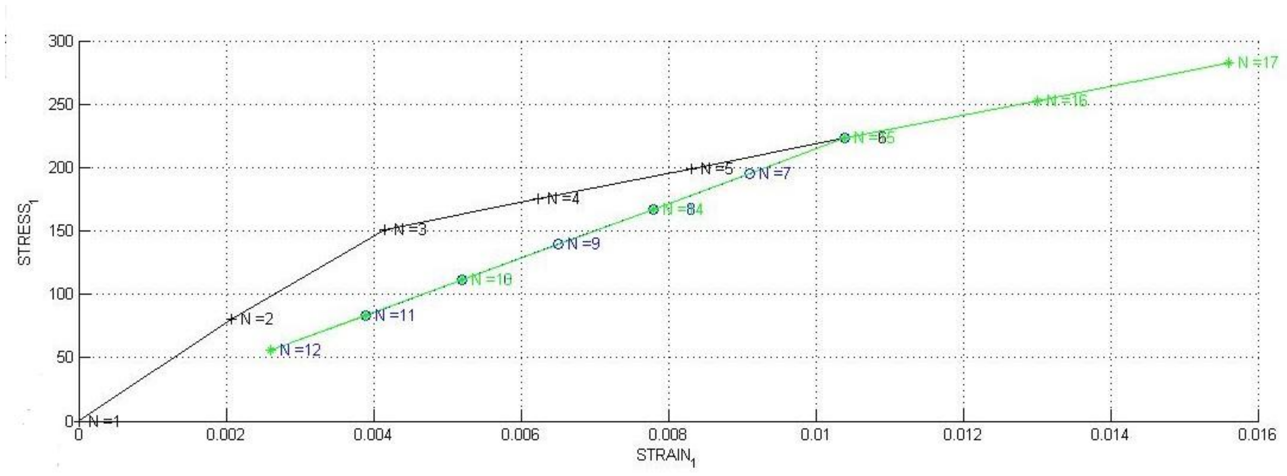


(b)

Figure 7. Tension-only damage model: (a) damage surface; (b) stress-strain curve



(a)



(b)

Figure 8. Non-symmetric tension-compression damage model: (a) damage surface; (b) stress-strain curve

3. PART II (RATE DEPENDENT MODELS)

In order to implement viscous model the files *rmap_dano1.m* and *damage_main.m* were modified, see Annex 3. The strain vector ε_n at the previous time step was transferred to *rmap_dano1.m* and used to determine inelastic loading.

The material parameters are taken from the previous section: Young's modulus $E = 20000$ Pa, Poisson ratio $\nu = 0.3$, yield stress $\sigma_y = 150$ MPa, time – 10 seconds hardening/softening modulus was chosen positive to represent a hardening $H = 0.3$

3.1. Assessment of the implementation correctness.

Now the stress value can lay outside the elastic domain, which corresponds to the theoretical foundations.

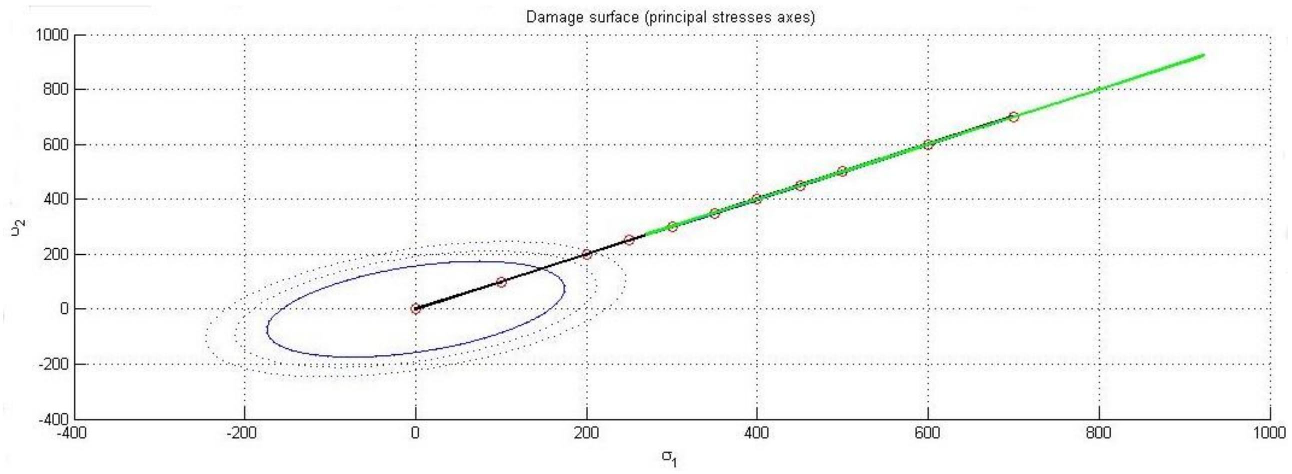


Figure 9. Damage surface for viscous case

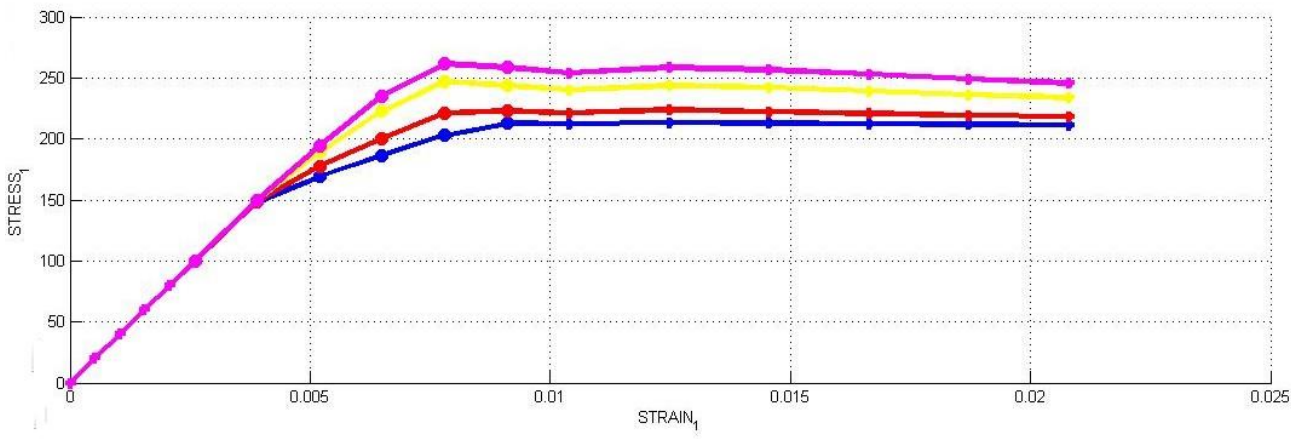


Figure 10. Strain stress curve for different viscosities: $\nu = 0.1$ (**blue**), $\nu = 0.3$ (**red**), $\nu = 0.7$ (yellow), $\nu = 1$ (**pink**).

4. CONCLUSION

In the rate independent section Non-symmetric tension-compression and Tensile-only models were implemented and tested for stress paths. The exponential law was implemented and compared to the linear one.

For rate dependent problem the assessment was conducted by means of strain stress curve for different viscosities. It was revealed that higher stresses correspond to higher viscosities. The code is stable for higher alpha coefficient values.

Annex 1.

Rmap_dano1

```
function [sigma_n1,hvar_n1,aux_var] = rmap_dano1
(eps_n0,eps_n1,hvar_n,Eprop,ce,MDtype,n,alpha,delta_t)
%*************************************************************************
*
%*            Integration Algorithm for a isotropic damage model
%*             [sigma_n1,hvar_n1,aux_var] = rmap_dano1
(eps_n1,hvar_n,Eprop,ce,MDtype)        *
%*
%* 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
*
%*                      aux_var(3)  Auxiliar variables for computing const.
tangent tensor  *
%*************************************************************************
********


hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5) ;
alpha = Eprop(8) ;
eta = Eprop(7) ;
visc_type=Eprop (6);
%*************************************************************************
%*       initializing
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
 q_inf=1.5;
 A=1;
% if(r_n<=0.d0)
%     r_n=r0;
%     q_n=r0;
% end
%*************************************************************************
******
%*       Damage surface
%*
[rtrial] = Modelos_de_dano1 (eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n0,alpha);
%*************************************************************************

%*************************************************************************
%*   Ver el Estado de Carga
%*
%*   --------->    fload=0 : elastic unload
%*
%*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)
%*
fload=0;
if(rtrial > r_n)
```

```
    %*   Loading
    % Viscous model
     fload=1;
    if  visc_type== 1
          r_n1 =(eta-delta_t*(1-
alpha))*r_n/(eta+alpha*delta_t)+(delta_t*rtrial)/(eta+alpha*delta_t);
    else
          r_n1 = rtrial;
    end
    delta_r=rtrial-r_n;
    if hard_type == 0
        if H<0
            q_inf=zero_q;
        end
        %  Linear
        q_n1= q_n+ H*delta_r;

    else
        q_n1= q_inf-(q_inf-r0)*exp(A*(1-rtrial/r0));

    end
    if(q_n1>q_inf)    % Hardening bound
        q_n1=q_inf;
    end
    if(q_n1<zero_q)   % Softening bound
        q_n1=zero_q;
    end


else

    %*     Elastic load/unload
    fload=0;
    r_n1= r_n  ;
    q_n1= q_n  ;


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

Annex 2
Modelos_de_dano1.m

```
function [rtrial] = Modelos_de_dano1
(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n0,alpha)
%*************************************************************************************
%*          Defining damage criterion surface
%*
%*                          MDtype=  1      : SYMMETRIC
%*
%*                          MDtype=  2      : ONLY TENSION
%*
%*                          MDtype=  3      : NON-SYMMETRIC
%* OUTPUT:
%*
%*                          rtrial
%*************************************************************************************



%*************************************************************************************
if (MDtype==1)      %* Symmetric

%Viscous case implementation
if Eprop(6) ==1
alpha = Eprop(8) ;
rtrial_n0 = sqrt(eps_n0 *ce*eps_n0');
rtrial_n1= sqrt(eps_n1*ce*eps_n1');
rtrial=(1 - alpha)* rtrial_n0 + alpha*rtrial_n1;

else

rtrial= sqrt(eps_n1*ce*eps_n1');

end

elseif (MDtype==2)  %* Tension-only
sig_tens=0.5*(abs(ce*eps_n1')+ce*eps_n1');
rtrial=sqrt(eps_n1*sig_tens);

elseif (MDtype==3)  %* Non-symmetric
sig_nonsym=ce*eps_n1';                            %
sig_nonsym_plus=0.5*(abs(sig_nonsym)+sig_nonsym);   %change
theta=(sum(sig_nonsym_plus))/(sum(abs(sig_nonsym_plus)));
rtrial=(theta+(1-theta)/n)*sqrt(eps_n1*ce*eps_n1');
end
%*************************************************************************************
*******
return
```

Annex 3.

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);
alpha =   Eprop(8) ;



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   = 6    ;
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);
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_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,:) ;
        % Strain for the viscous model%%%%%%%%%%%%%%%%%%%%%%%%%%
        if i >1
        eps_n0 = strain(i-1,:);
        end

%*******************************************************************************
*******
        %*      DAMAGE MODEL
        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n0,eps_n1,hvar_n,Eprop,ce,MDtype,n,alpha,delta_t);
        % 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)
    end
end
```