Margarita Smolentseva
Master Program in Computational Mechanics

Assignment 1

Continuum Damage Models

Part I. Rate Independent Models

There are different models which describe material behavior. In current work two of them are considered: non-symmetric tension-compression and tension-only damage models.

The **non-symmetric tension-compression damage model** is useful for simulation of materials which have different tension and compression behavior. For description of material behavior and definition of damage surface this model uses the following norm:

$$\tau_\sigma = \left[\theta + \frac{\theta - 1}{n}\right]\sqrt{\boldsymbol{\sigma}: \mathbb{C}^{-1}: \boldsymbol{\sigma}}, \qquad \theta = \frac{\sum_1^3 < \bar{\sigma}_i >}{\sum_1^3 |\bar{\sigma}_i|}$$

Which means:

$$\begin{cases} \sigma_1, \sigma_2, \sigma_3 > 0 \ \rightarrow \ \tau_\sigma = \sqrt{\boldsymbol{\sigma}: \mathbb{C}^{-1}: \boldsymbol{\sigma}} \\ \sigma_1, \sigma_2, \sigma_3 < 0 \ \rightarrow \ \tau_\sigma = \dfrac{1}{n}\sqrt{\boldsymbol{\sigma}: \mathbb{C}^{-1}: \boldsymbol{\sigma}} \end{cases}$$

The **tension-only damage model** is useful for simulation of materials which can be damaged only by tension. In other words, compression is not considered in the model. This model uses the following norm:

$$\begin{cases} \boldsymbol{\sigma} = \boldsymbol{\sigma}^+ \ \rightarrow \ \tau_\sigma^+ = \sqrt{\boldsymbol{\sigma}^+: \mathbb{C}^{-1}: \boldsymbol{\sigma}} = \tau_\sigma \\ \boldsymbol{\sigma}^+ = 0 \ \rightarrow \ \tau_\sigma^+ = \sqrt{\boldsymbol{\sigma}^+: \mathbb{C}^{-1}: \boldsymbol{\sigma}} = 0 \end{cases}$$

In order to implement these damage models, Modelos_de_dano1.m, dibujar_criterio_dano1.m were changed in accordance with theory.

As for graphs plotting polar coordinate system is used, these norms take the following view:

1. Non-symmetric tension-compression damage model

$$\begin{cases} \boldsymbol{\sigma_\theta} > 0 \ \rightarrow \ r = \dfrac{q}{\sqrt{\boldsymbol{\sigma_\theta}: \mathbb{C}^{-1}: \boldsymbol{\sigma_\theta^T}}} \\ \boldsymbol{\sigma_\theta} < 0 \ \rightarrow \ r = n\dfrac{q}{\sqrt{\boldsymbol{\sigma_\theta}: \mathbb{C}^{-1}: \boldsymbol{\sigma_\theta^T}}} \end{cases}$$

2. Tension-only damage model

$$r = \frac{q}{\sqrt{\boldsymbol{\sigma_\theta^+}: \mathbb{C}^{-1}: \boldsymbol{\sigma_\theta^T}}}$$

Where $\boldsymbol{\sigma_\theta} = [\cos(\theta) \;\; \sin(\theta) \;\; 0 \;\; \vartheta(\cos(\theta) + \sin(\theta))]$, r – internal variable, q – hardening variable.

The exponential hardening/softening law

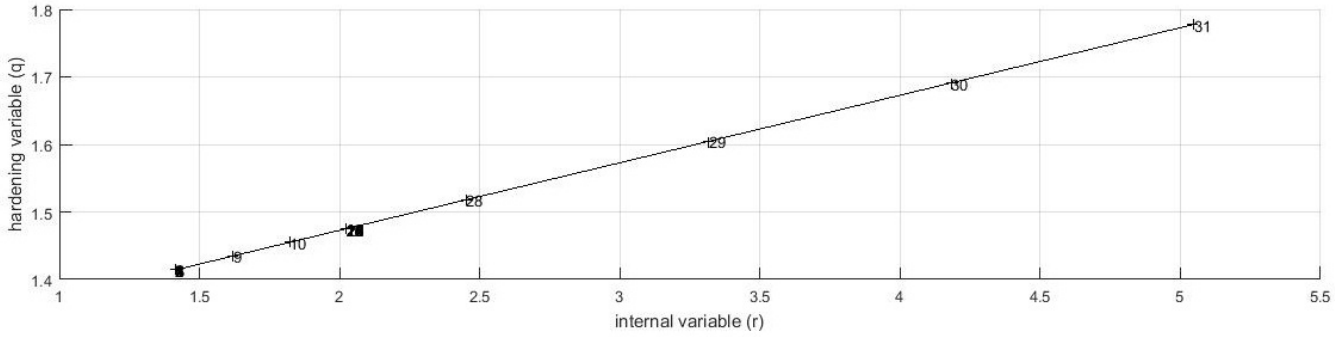
The exponential relation is defined as $q(r) = q_\infty - (q_\infty - r_0)e^{A\left(1-\frac{r}{r_0}\right)}$ where q_∞ is the maximum value which can be reached by q.

Values of q_∞ and A should satisfy to the restriction of Hardening modulus $H \leq \frac{q(r)}{r}$ where $H = A\frac{q_\infty - r_0}{r_0}e^{A\left(1-\frac{r}{r_0}\right)}$.

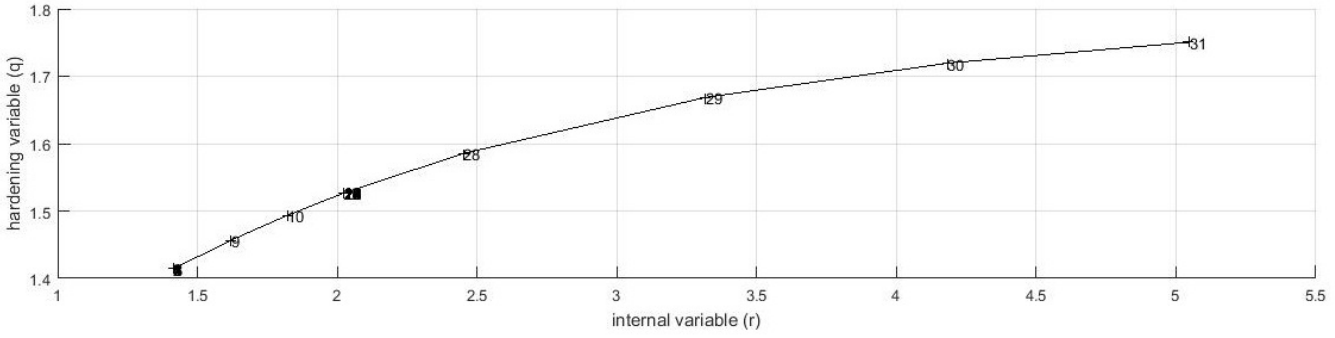
The implementation is represented in rmap_dano.m. If q_∞ and A do not satisfy to restriction, the program throw an error.

Experiments

I order to assess the correctness of the implementation, three different paths at the stress space and stress-strain curve were considered. Values of q_∞ and A were tested and chosen as $q_\infty = 1.8$ and A = 0.8. If $A \geq 1$ the exponential function rises faster than in case when $A < 1$.



a) Linear function



b) Exponential function

Fig. 1: Hardening/softening law.

The figures show relation between hardening and internal variables for both linear and exponential functions. Both graphs behave as expected which means that implementation is correct. In the current work the exponential function is going to be used to confirm that implementation is correct.

Path 1

$$\Delta\bar{\sigma}_1^{(1)} = 300;\ \Delta\bar{\sigma}_2^{(1)} = 0\ (\text{uniaxial tensile loading})$$

$$\Delta\bar{\sigma}_1^{(2)} = -500;\ \Delta\bar{\sigma}_2^{(2)} = 0\ (\text{uniaxial tensile unloading/compressive loading})$$

$$\Delta\bar{\sigma}_1^{(3)} = 700;\ \Delta\bar{\sigma}_2^{(3)} = 0\ (\text{uniaxial compressive unloading/tensile loading})$$

Both tension-only and damage surface for tension-only damage models provide similar results.

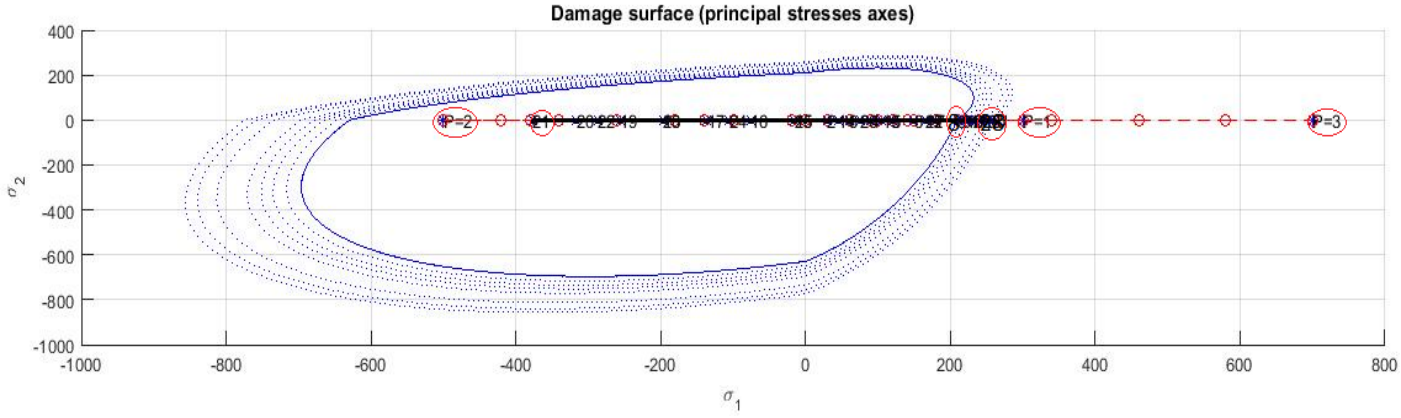


Fig. 2: Non-symmetric compression-tension damage model. The path at the stress space

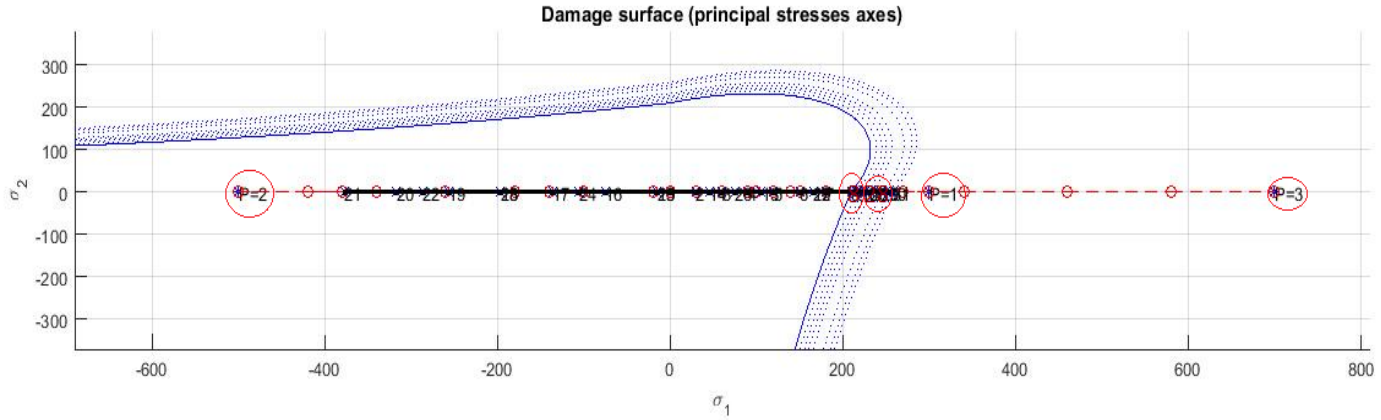


Fig. 3: Tension-only damage model. The stress-strain relation

It can be seen from Fig. 2, which represents the damage surface for the non-symmetric compression-tension model, and Fig.3, which represents the damage surface for tension-only model, that principal stresses satisfy to the damage criterion of inviscid model. As surface expands when stresses are on the boundary, hardening condition can be observed.

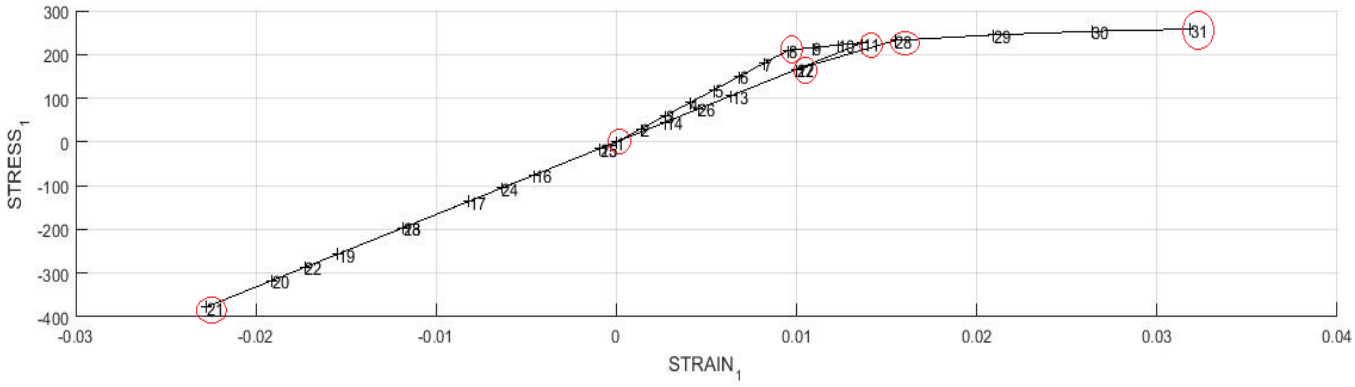


Fig. 4: Non-symmetric compression-tension damage model. The stress-strain relation

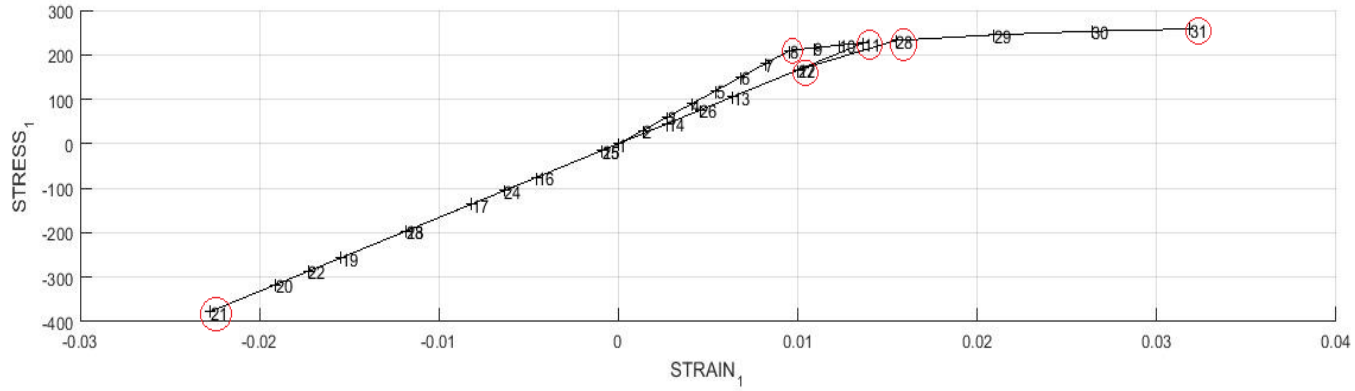


Fig. 5: Tension-only damage model. The stress-strain relation

From Fig.4 and Fig.5 we can see evolution of strain and stress values for non-symmetric compression-tension model and tension-only model respectively. As current loading path is an uniaxial trend, there is a dependence between strain and stress only in direction 1. Starting at point 0 (tensile loading), the graph demonstrates linear behavior up to the path in the elastic domain. At point 8 the slope changes because stresses reach the boundary and stay there until they start to move inside (tensile unloading). From this point stresses stay inside elastic domain that is why the slope does not change. Afterwards, in the third segment (tensile loading) stresses move from the domain to the boundary till the point 27. At the point 28 the slope becomes much slighter because stresses reach boundary.

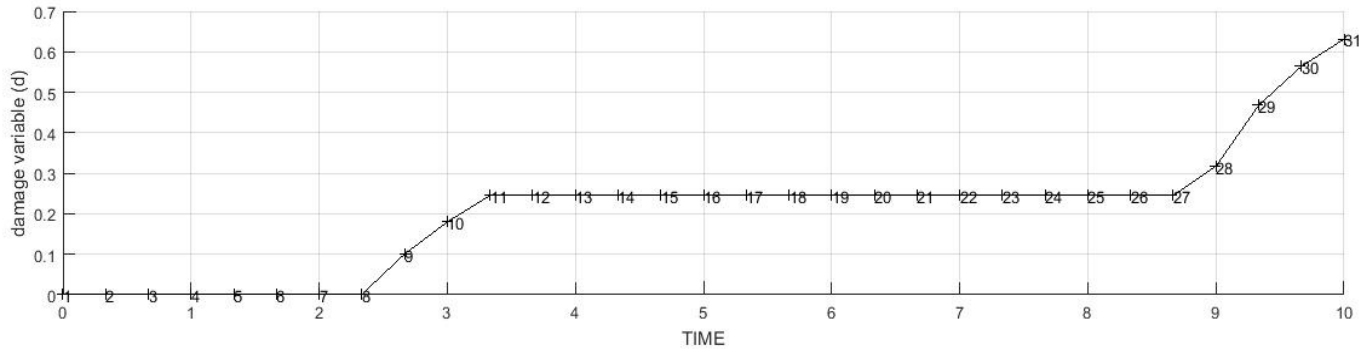


Fig. 6: Non-symmetric compression-tension damage model. Evolution of the damage variable

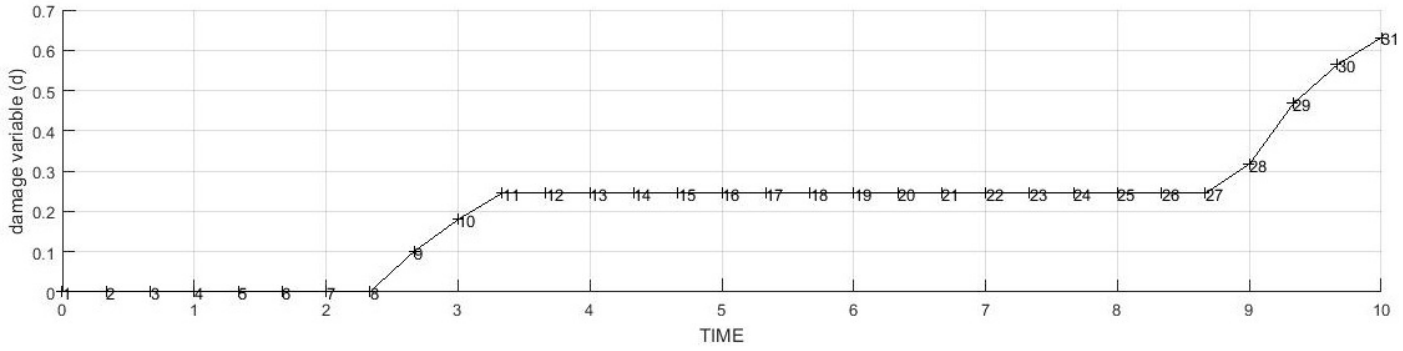


Fig. 7: Tension-only damage model. Evolution of the damage variable

From Fig.6 and Fig.7 it can be observed how damage variable d changes with time values for non-symmetric compression-tension and tension-only damage models accordingly. Till the point of 8 damage variable does not change as stresses are inside of the elastic domain. From the point 8 to 11 slope appears on the graph because the stress moves outside of the boundary. Then till the point 27 damage variable does not change as the stress stay again inside while it starts to move outward.

Path 2

$$\Delta\bar{\sigma}_1^{(1)} = 200;\ \Delta\bar{\sigma}_2^{(1)} = 0 \quad (\text{uniaxial tensile loading})$$

$$\Delta\bar{\sigma}_1^{(2)} = -700;\ \Delta\bar{\sigma}_2^{(2)} = -700 \quad (\text{biaxial tensile unloading/compressive loading})$$

$$\Delta\bar{\sigma}_1^{(3)} = 1000;\ \Delta\bar{\sigma}_2^{(3)} = 1000 \quad (\text{biaxial compressive unloading/tensile loading})$$

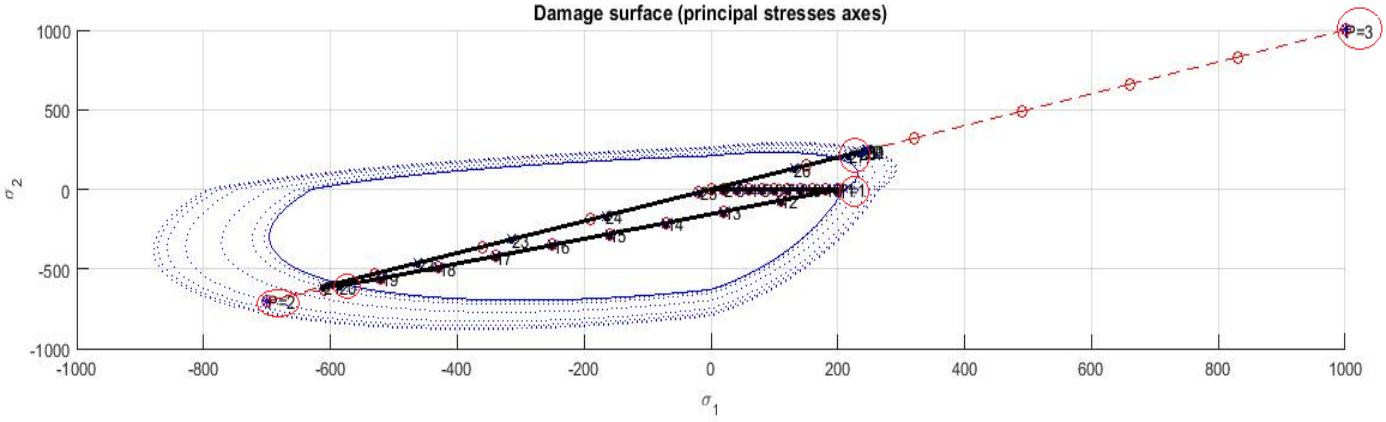


Fig. 8: Non-symmetric compression-tension damage model. The path at the stress space

From Fig.8 it can be seen that the stress remains inside the damage surface in compression in case of tensile uploading (the second segment). However, it can reach

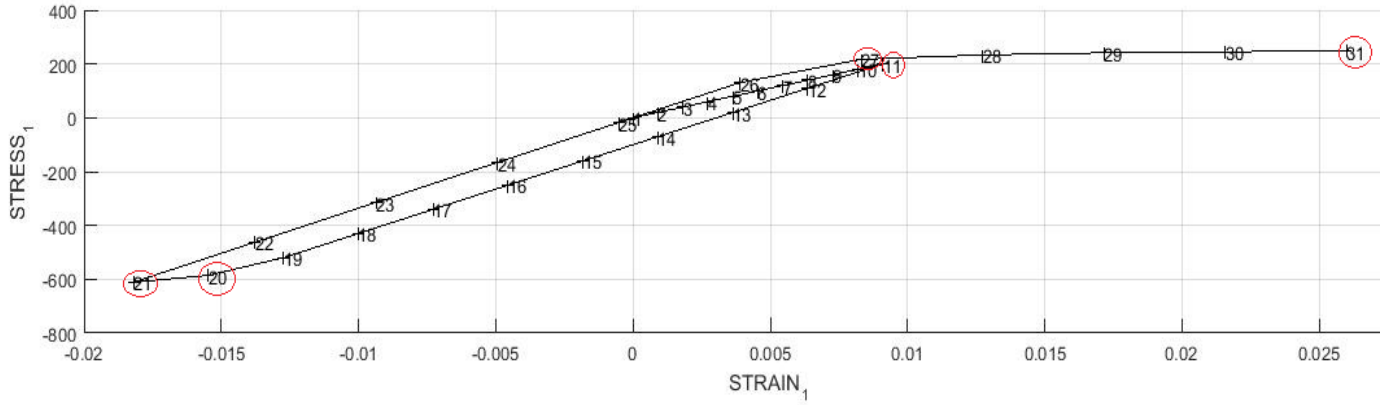


Fig. 9: Non-symmetric compression-tension damage model. The stress-strain relation in direction 1

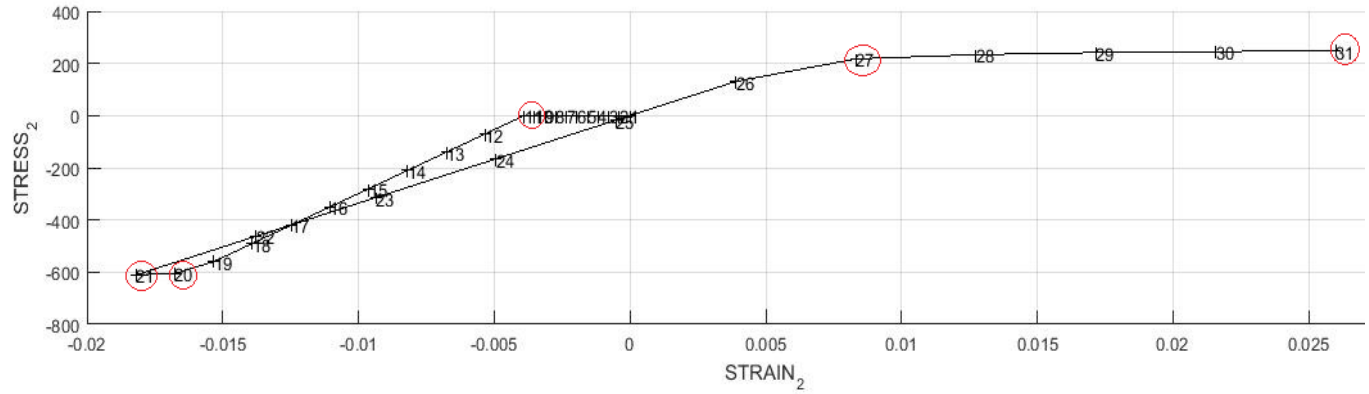


Fig. 10: Non-symmetric compression-tension damage model. The stress-strain relation in direction 2

In Fig.9 and Fig.10 there are represented stress-strain relations in directions 1 and 2 respectively. From Fig. 9 it can be observed that the graph remains straight until the point 8 because the stress stays inside elastic domain. Fig.10 reflects that the stress is equal to 0 until the point 11 while the first segment is loading in the direction 1. As the stresses approach the boundary at the point 20, the slope near this point can be observed in the graphs. Between the points 26 and 27 the stress again reaches the boundary.

Let us consider tensile-only damage model.

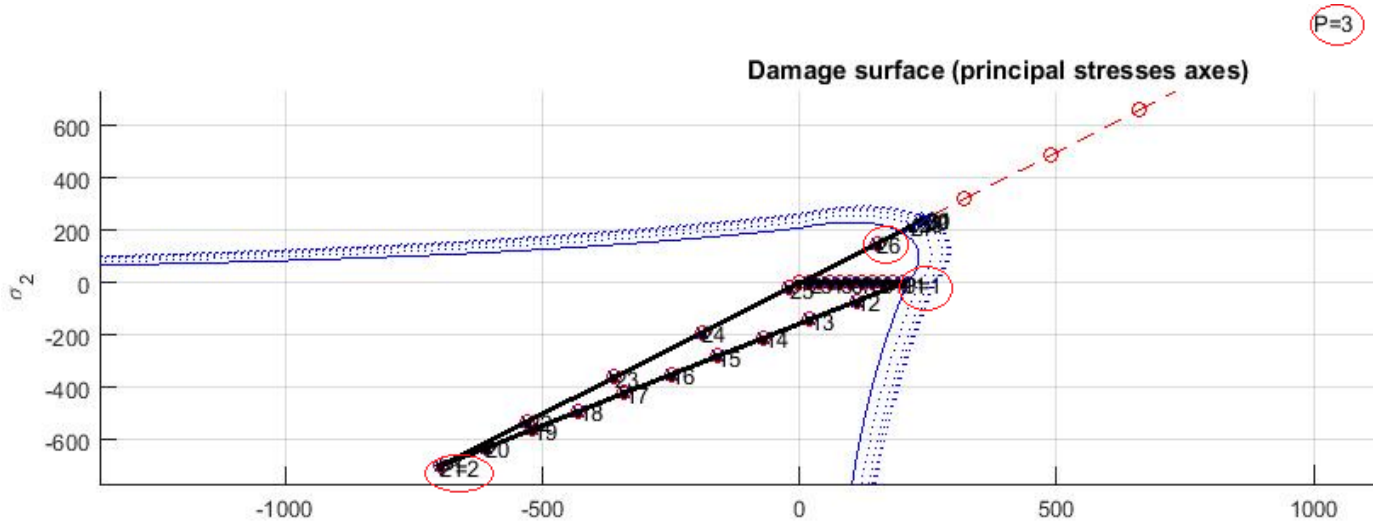


Fig. 11: Tension-only damage model. The stress-strain relation

From Fig. 11 it can be observed that in case of tensile uploading the stresses does not reach the boundary as this model does not consider compression.

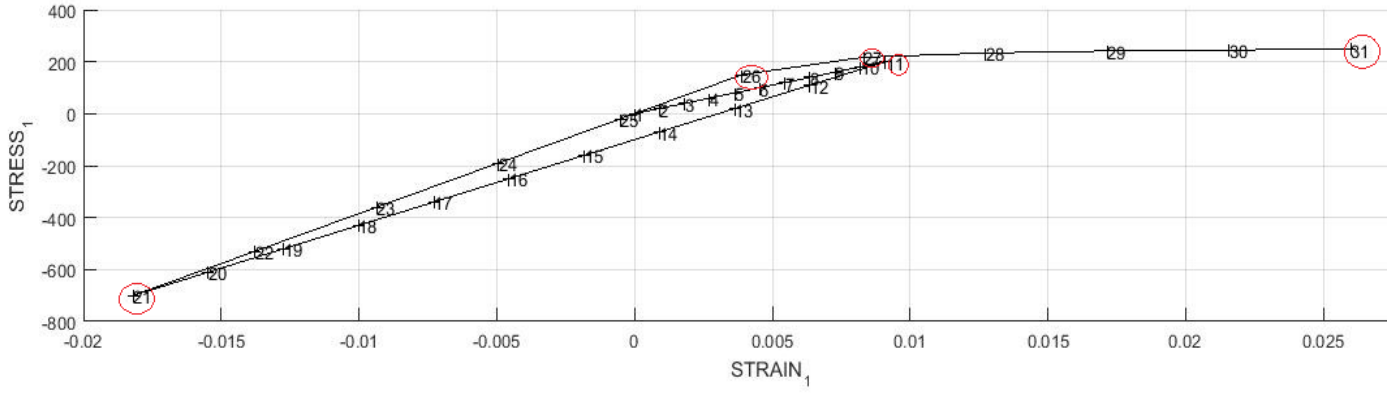


Fig. 12: Tension-only damage model. The stress-strain relation in direction 1

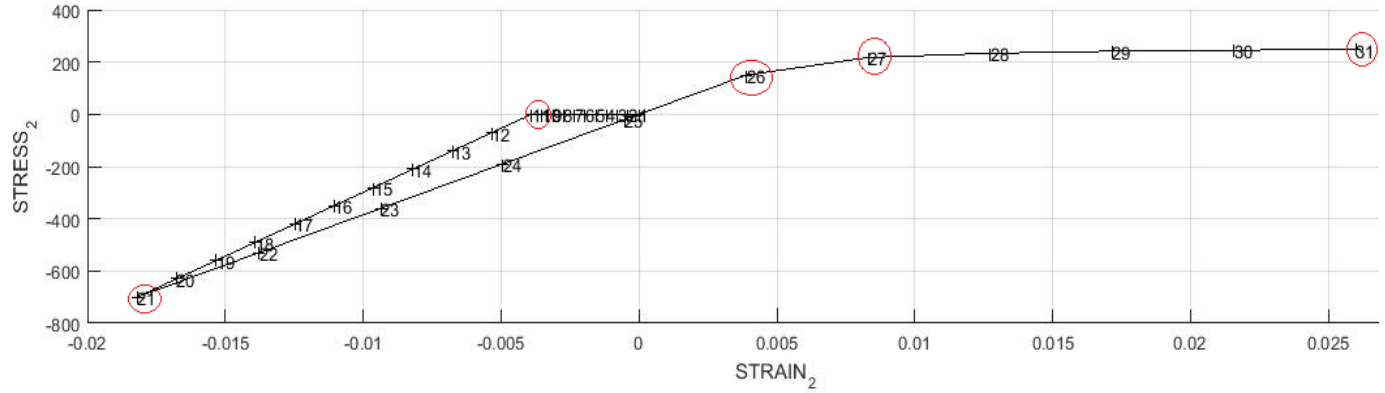


Fig. 13: Tension-only damage model. The stress-strain relation in direction 2

Fig.12 and Fig.13 reflect stress-strain relations in directions 1 and 2 accordingly for tension-only damage model. The behavior is similar to the non-symmetric compression-tension one except there is no slope near the point 20 as the stresses do not reach the boundary.

Path 3

$$\Delta\bar{\sigma}_1^{(1)} = 300;\ \Delta\bar{\sigma}_2^{(1)} = 300 \quad (\text{biaxial tensile loading})$$

$$\Delta\bar{\sigma}_1^{(2)} = -700;\ \Delta\bar{\sigma}_2^{(2)} = -700 \quad (\text{biaxial tensile unloading/compressive loading})$$

$$\Delta\bar{\sigma}_1^{(3)} = 1000;\ \Delta\bar{\sigma}_2^{(3)} = 1000 \quad (\text{biaxial compressive unloading/tensile loading})$$

First, let us consider non-symmetric compression-tension damage model.

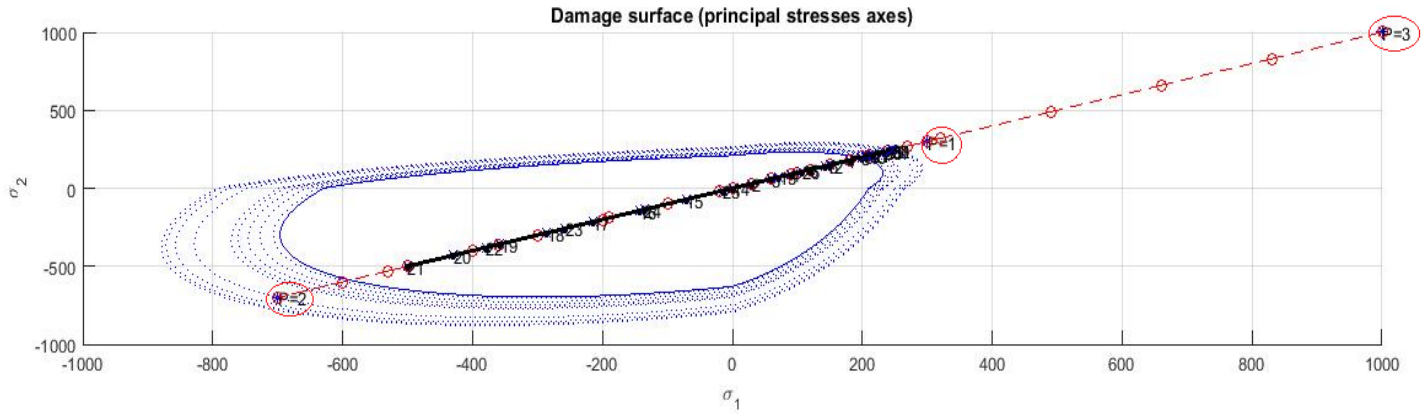


Fig. 14: Non-symmetric compression-tension damage model. The path at the stress space

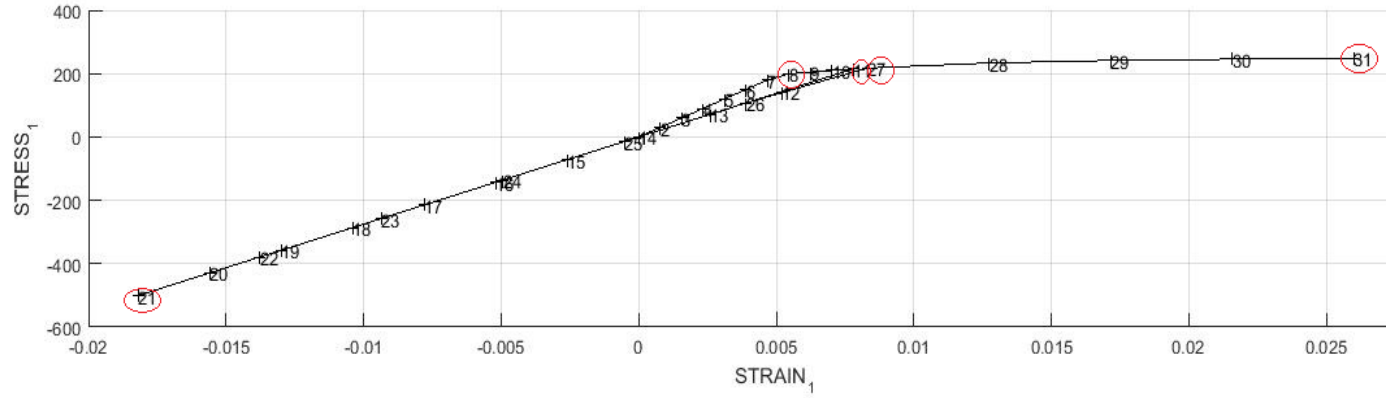


Fig. 15: Non-symmetric compression-tension damage model. The stress-strain relation in direction 1

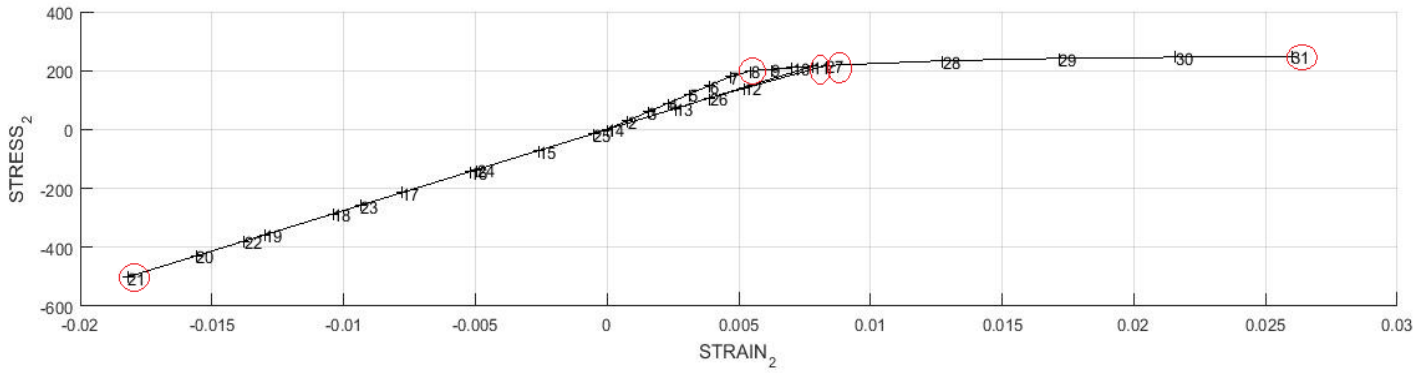


Fig. 16: Non-symmetric compression-tension damage model. The stress-strain relation in direction 2

Fig.15 and Fig.16 represent the stress-strain relations in two directions. These graphs are equal because all segments correspond to biaxial loads. From the figures it can be observed that the graphs remain linear until the point 8 as the stresses take place inside the elastic domain. In case of the third segment the stresses reach the boundary at the point 27.

Let us consider the tension-only damage model.

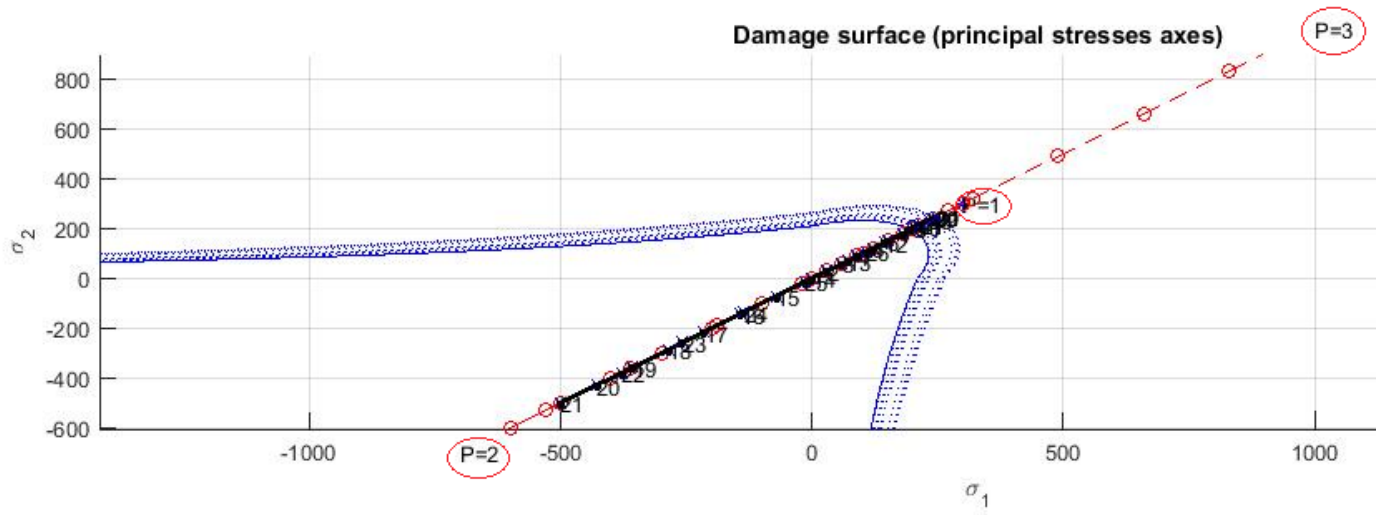


Fig. 17: Tension-only damage model. The stress-strain relation

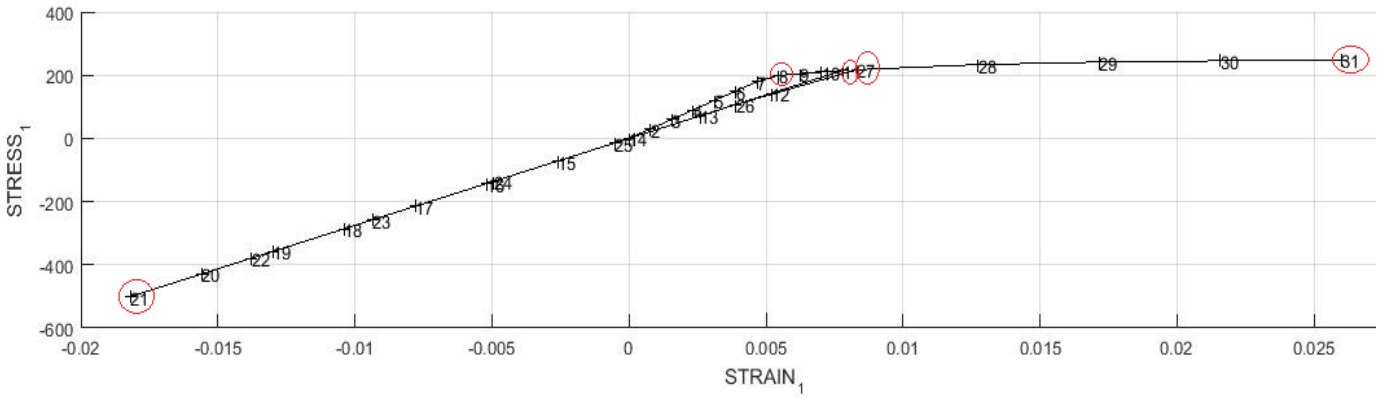


Fig. 18: Tension-only damage model. The stress-strain relation in direction 1

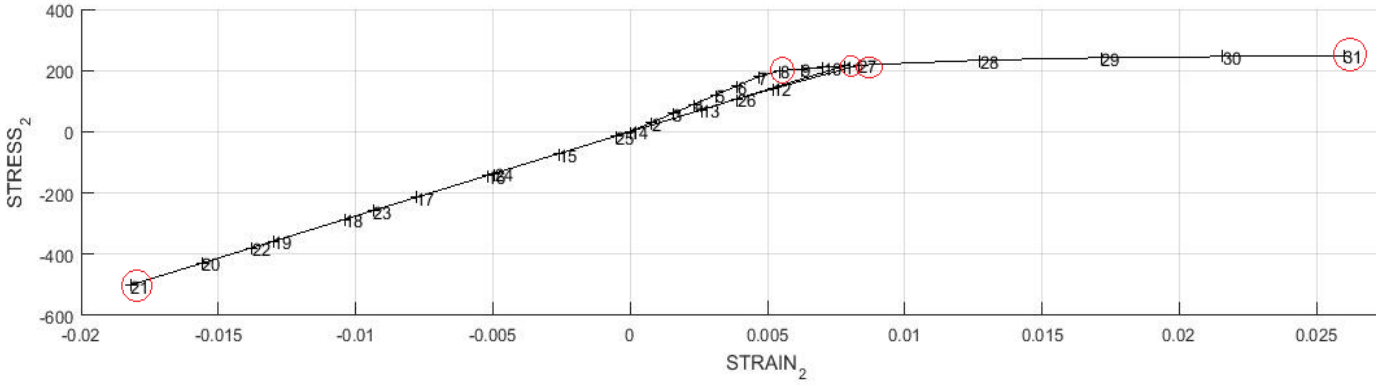


Fig. 19: Tension-only damage model. The stress-strain relation in direction 2

As we can see from the Fig. 18 and Fig. 19, in case of the tension-only model behavior of graphs is similar to the non-symmetric compression-tension model case. It happens because the stresses remain in the elastic domain for the both models.

Part I. Rate Dependent Models

In the current part the integration algorithm for the continuum isotropic visco-damage “symmetric tension-compression” model was implemented.

The viscous case is implemented in the rmap_dano1.m function. In order to compute both norms for the previous and current time steps the additional parameter, which is a previous strain tensor, was added to rmap_dano1.m notation. This strain vector is defined in the function damage_main.m. Also, in the rmap_dano1.m function the equation

$\tau_{\varepsilon_{n+\alpha}} = (1-\alpha)\tau_{\varepsilon_n} + \alpha\tau_{\varepsilon_{n+1}}$ was implemented. Here α is an integration parameter, $\alpha \in [0,1]$, $\tau_{\varepsilon_{n+1}}$ and τ_{ε_n} are strain norms, which are calculated in the Modelos_da_dano1.m function. In order to understand if points are located in unloading or loading path, $\tau_{\varepsilon_{n+\alpha}}$ is compared with r_n, which is the internal variable at the previous step.

To assess the correctness of the implementation, let us consider different cases for a specific given Poisson ratio and linear hardening/softening parameter. For experiments the first path was chosen with following values $\Delta\bar{\sigma}_1^{(1)} = 600;\ \Delta\bar{\sigma}_2^{(1)} = 0;\ \Delta\bar{\sigma}_1^{(2)} = -500;\ \Delta\bar{\sigma}_2^{(2)} = 0; \Delta\bar{\sigma}_1^{(3)} = 100;\ \Delta\bar{\sigma}_2^{(3)} = 0.$

1) **Different viscosity parameter η**; η = 0.3, 0.1, 0.01.

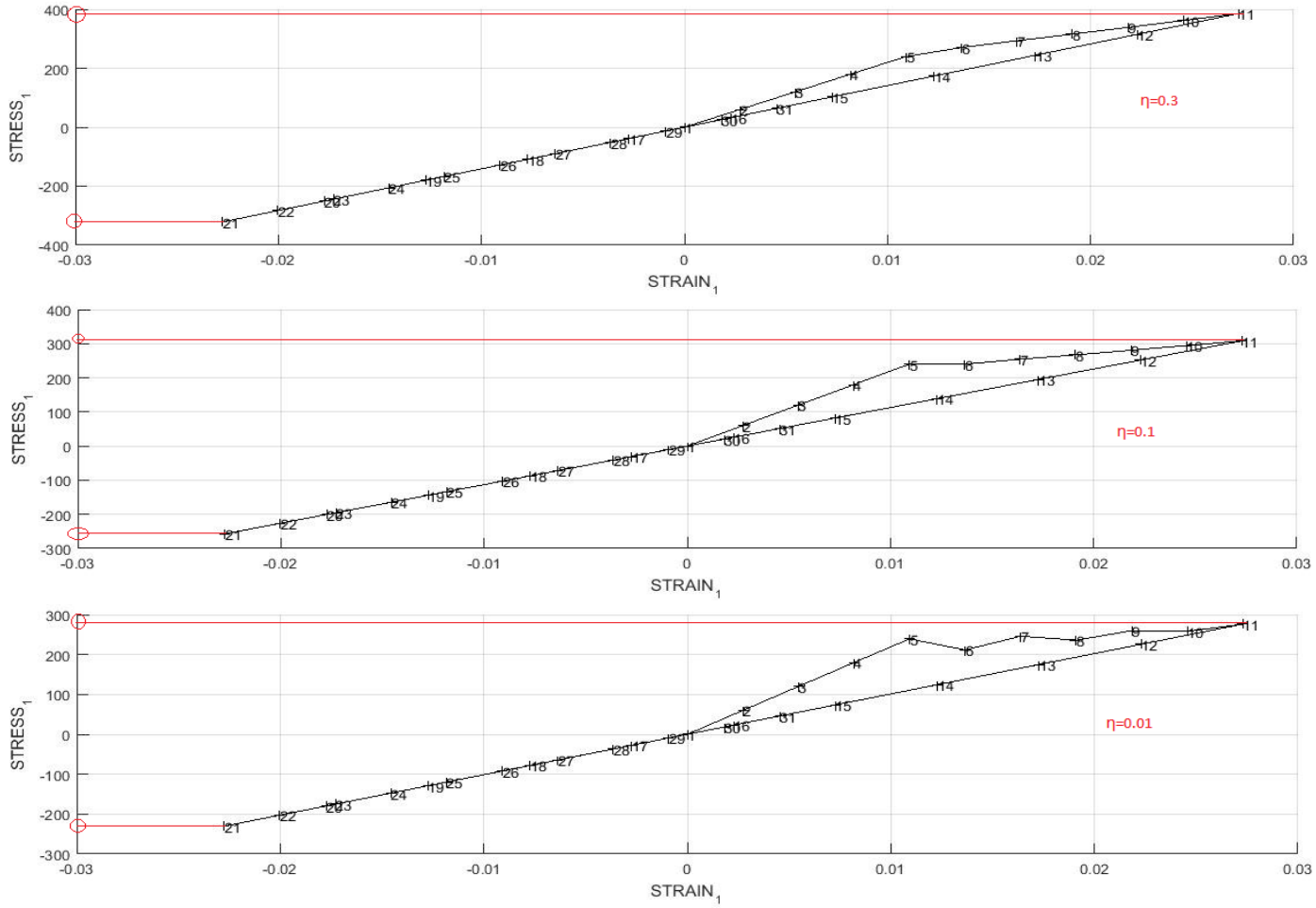


Fig. 20: The stress-strain relations in direction 1 for different η

As we can see from the Fig. 20, the greater value for the η parameter, the higher positive values are reached by the stresses (loading) and the less negative stresses values (unloading).

2) **Different strain rate parameter $\dot{\varepsilon}$**. As the strain rate is the derivative of the strain over time, in order to see its variations, the total time parameter was changed. Total time = 10 s; 1 s; 0.1 s.

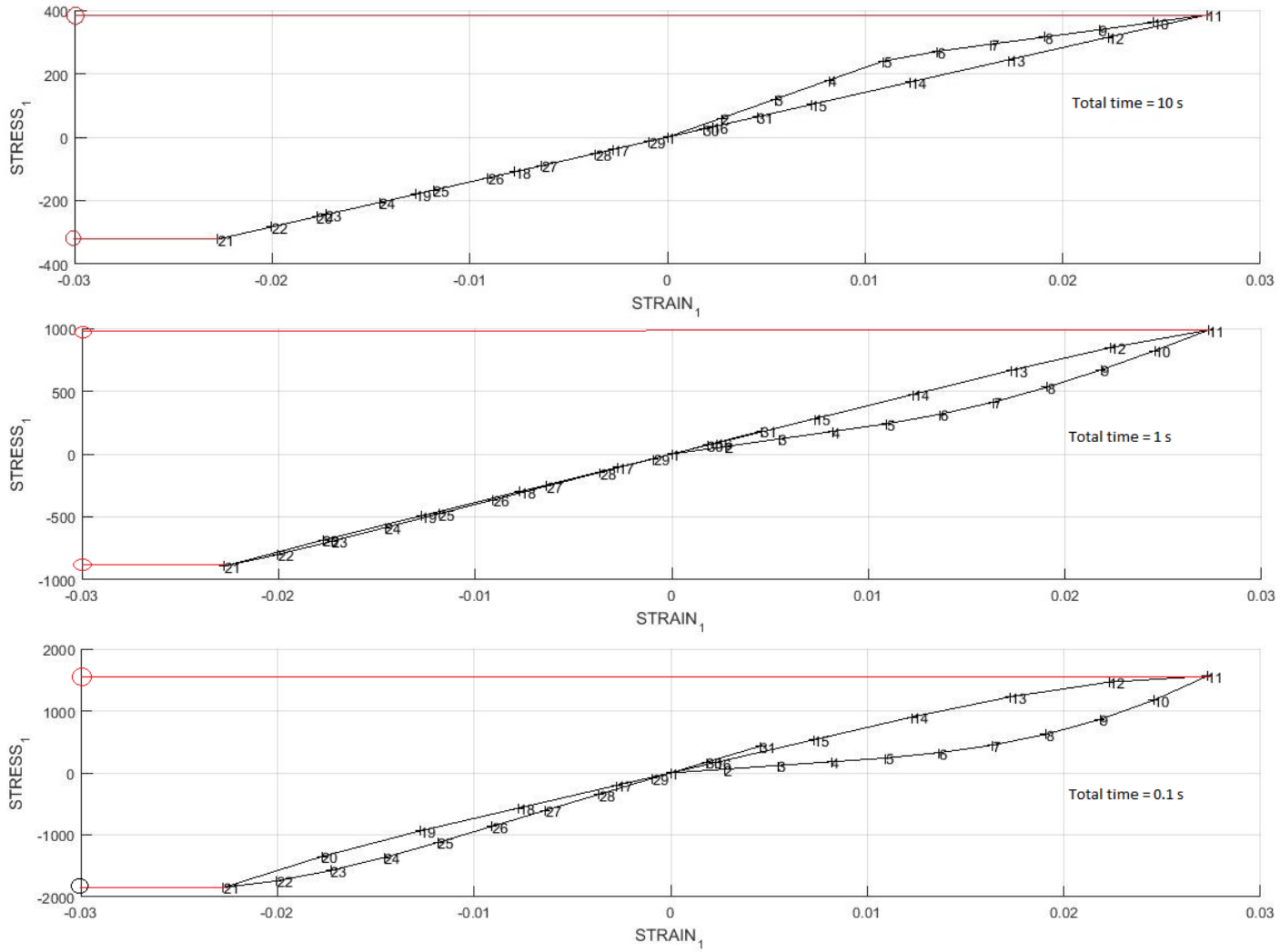
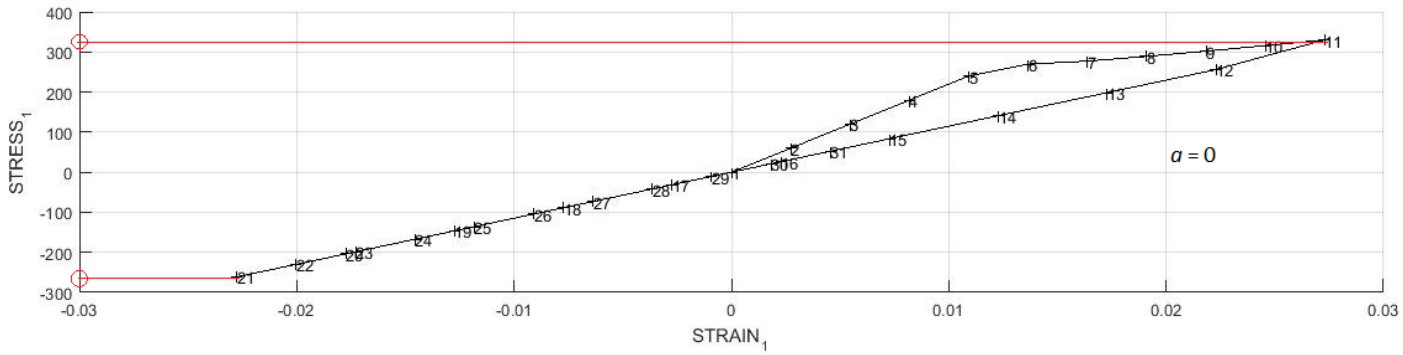


Fig. 21: The stress-strain relations in direction 1 for different total time

It can be seen from Fig. 21 that for negative values (unloading) of the stresses the greater total time value, the greater stresses. Meanwhile for positive values (loading) of the stresses the situation is opposite. The strain rate increases with time increasing.

3) **Different α values**; $\alpha = 0; \frac{1}{4}; \frac{1}{2}; \frac{3}{4}; 1$.



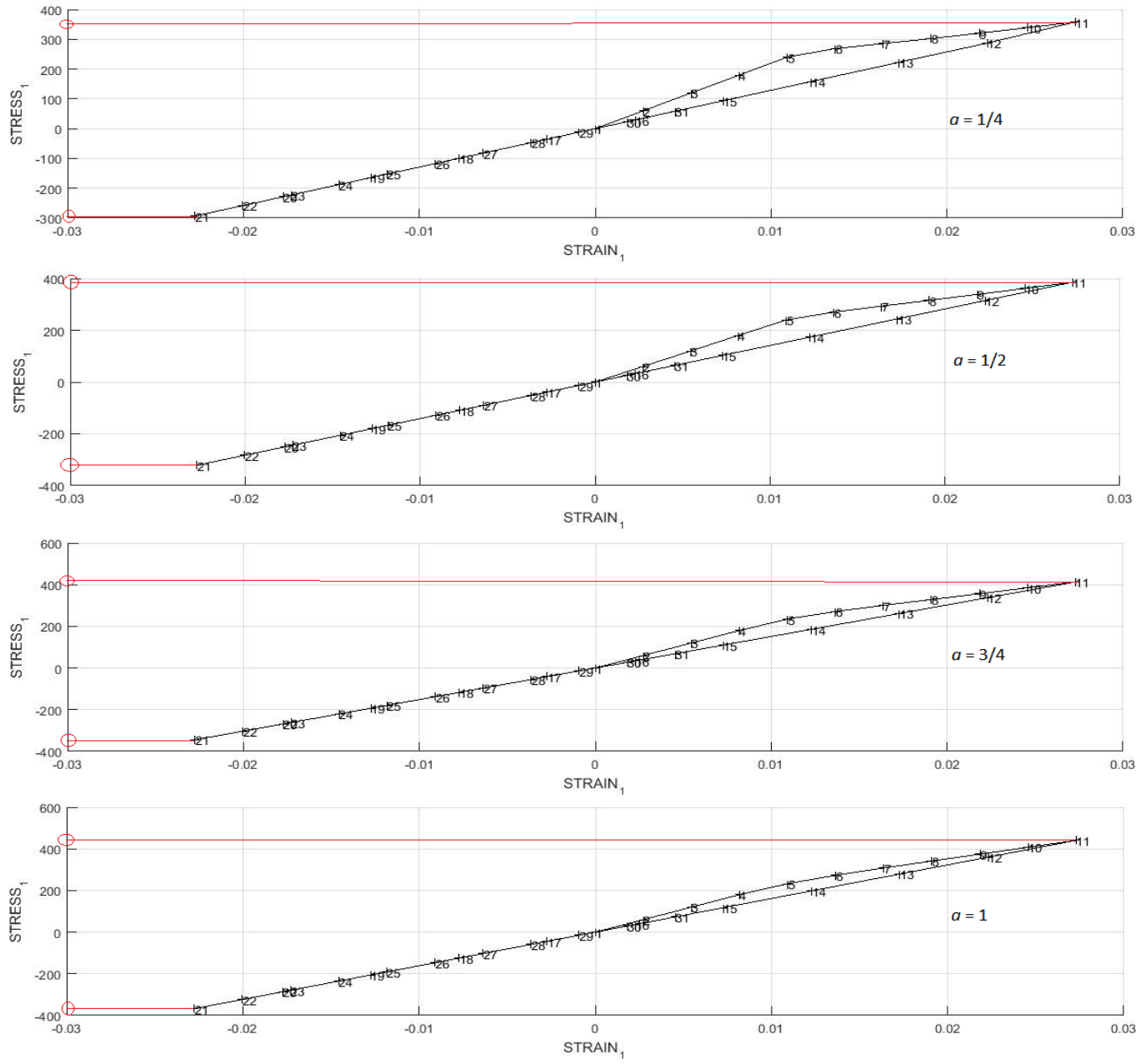


Fig. 22: The stress-strain relations in direction 1 for different α

From Fig. 22 it can be seen that higher values α correspond to higher values of the stresses in positive case and less values of the stresses in the negative case. The similar behavior was observed in experiments with variations of viscous parameter. At the same time the figures show that for values of α is equal to 0 and ¼ the graphs have a little bit different behavior. It happens because stability condition requires α to be in the range [0.5, 1].

In order to assess the correctness of the implementation, the stress-strain graph of the non-symmetric damage model is presented for the loading path 1 and viscous case with viscous parameter η=0 and $\alpha = 1$ which should be the same as for inviscid case. The following figure reflects that even when these parameters are settled, the stress-strain relations are the same as in Fig. 4.

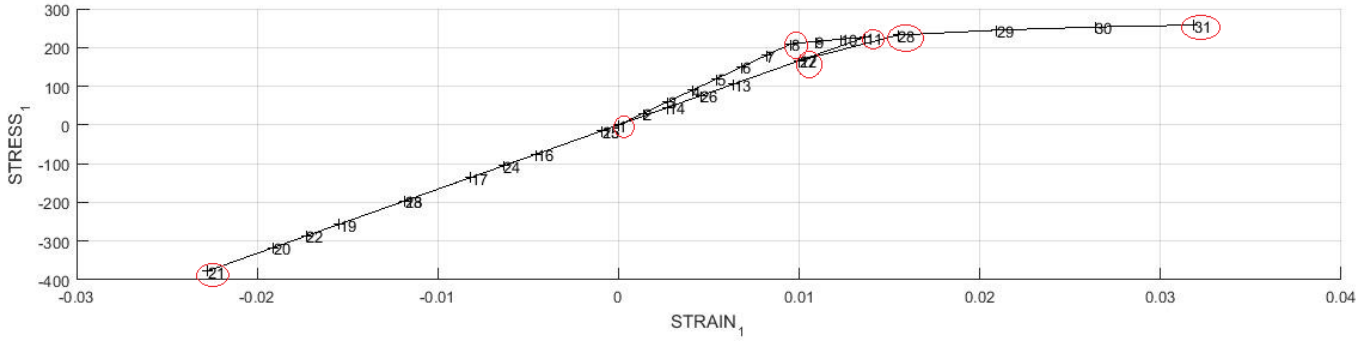


Fig. 23: Non-symmetric compression-tension damage model. The stress-strain relation in direction 1 with settled parameters η=0 and $\alpha = 1$

In order to analyze the effects of the α values on the component of the tangent and algorithmic constitutive operators, using the same previous input data with η=0.3 and total time equals to 0, the code was changed in accordance to following.

In rmap_dano1.m the auxiliary value was calculated as $\frac{q_{n+1} - Hr_{n+1}}{r_{n+1}^3}$. The algorithmic tangent operator was defined as $\mathbb{C}_{alg,n+1}^{\vartheta d} = \begin{cases} (1-d_{n+1})\mathbb{C}\ (unloading) \\ (1-d_{n+1})\mathbb{C} - \frac{\alpha \Delta t}{\eta + \alpha \Delta t} \frac{1}{\tau_{\varepsilon_{n+1}}} d'_{n+1} \bar{\sigma}_{n+1} \otimes \bar{\sigma}_{n+1}\ (loading) \end{cases}$ where $d'_{n+1} = \frac{q(r_{n+1}) - H_{n+1} r_{n+1}}{r_{n+1}^2}$. The analytical tangent operator was obtained with the formula $\mathbb{C}_{tang}^{\vartheta d} = (1-d)\mathbb{C}$.

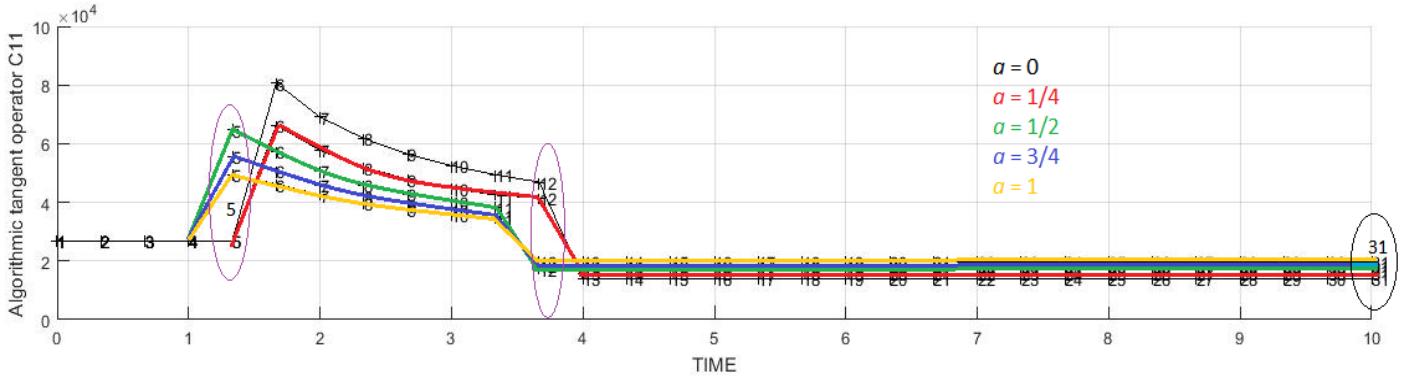


Fig. 24: Algorithmic tangent operator

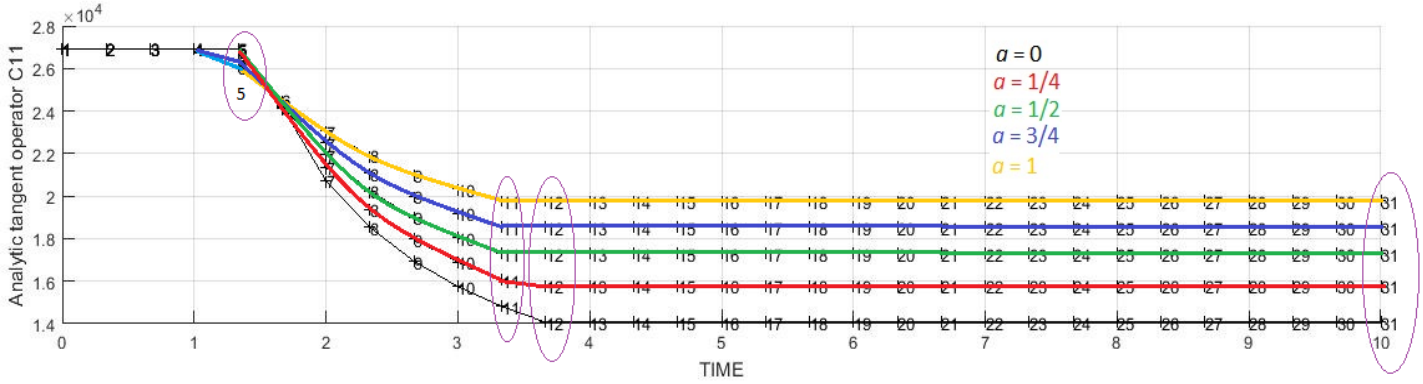


Fig. 25: Analytic tangent operator

From figures 24 and 25 we can see that in the beginning tangent operator does not change until point 5 for $\alpha = 0 \text{ and } 0.25$ and until point 4 for $\alpha = 0.5, 0.75 \text{ and } 1$ as points remain inside of the elastic domain. When points reach the boundaries, the behavior changes in these points. Further it changes again in the points 11 and 12 the corresponding values of α. Graphs slope is decreasing while points remain inside of the boundaries. Also, it is seen that greater values of α correspond to higher values of the tangent operator. The tangent operator never returns to the beginning state because of damage which occurs in the model.

Annex

```
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
    eps_pos = eps_n1;
    eps_pos(find(eps_pos < 0)) = 0;
    rtrial= sqrt(eps_pos*ce*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
    %rtrial= sqrt(eps_n1*ce*eps_n1')
    %if eps_n1(1)<0 && eps_n1(2)<0 && eps_n1(4)<0
    %    rtrial = rtrial*n;
    %end
    eps_pos = eps_n1*ce;
    eps_pos(find(eps_pos < 0)) = 0;

    theta = sum(eps_pos)/sum(abs(eps_n1*ce));
    rtrial= (theta + (1-theta)/n)*sqrt(eps_n1*ce*eps_n1')

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

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
%*************************************************************************
*********
%*                  PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODEL
%*
%*
%*
%*      function [ce] = tensor_elastico (Eprop, ntype)                    %*
%*
%*
%*      INPUTS                                                            %*
%*
%*
%*                     Eprop(4)    vector de propiedades de material
%*
%*                                       Eprop(1)=  E------>modulo de Young
%*
%*                                       Eprop(2)=  nu----->modulo de Poisson
%*
%*                                       Eprop(3)=  H----->modulo de
Softening/hard. %*
%*                                       Eprop(4)=sigma_u----->tensiпiSn
пiSltima        %*
%*                      ntype                                        %*
%*                                  ntype=1  plane stress
%*
%*                                  ntype=2  plane strain
%*
%*                                  ntype=3  3D
%*
%*                      ce(4,4)     Constitutive elastic tensor  (PLANE S.
)    %*
%*                      ce(6,6)                                  ( 3D)
%*
%*************************************************************************
*********


%*************************************************************************
*********
%*         Inverse ce
%*
ce_inv=inv(ce);
c11=ce_inv(1,1);
c22=ce_inv(2,2);
c12=ce_inv(1,2);
c21=c12;
c14=ce_inv(1,4);
c24=ce_inv(2,4);
%*************************************************************************
**********









%*************************************************************************
**********
% POLAR COORDINATES
if MDtype==1
```

```
    tetha=[0:0.01:2*pi];

%*************************************************************************
**********
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*


    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;

    for i=1:Contador
        radio(i)= q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i)
0 ...
            nu*(m1(i)+m2(i))]');

        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);

    end
    hplot =plot(s1,s2,tipo_linea);


elseif MDtype==2
 %   Comment/delete lines below once you have implemented this case
  %  ***************************************************
   tetha=[0:0.01:2*pi];

%*************************************************************************
**********
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*


    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;

    for i=1:Contador
        pos = [m1(i) m2(i) 0 nu*(m1(i)+m2(i))];
        pos(find(pos<0))=0;
        radio(i)= q/sqrt(pos*ce_inv*[m1(i) m2(i) 0 ...
            nu*(m1(i)+m2(i))]');

            s1(i)=radio(i)*m1(i);
            s2(i)=radio(i)*m2(i);

    end
    hplot =plot(s1,s2,tipo_linea);
```

```
elseif MDtype==3
    % Comment/delete lines below once you have implemented this case
    % ************************************************
    %menu({'Damage surface "NON-SYMMETRIC" has not been implemented yet. ';
...
    %    'Modify files "Modelos_de_dano1" and "dibujar_criterio_dano1"' ; ...
    %    'to include this option'},  ...
    %    'STOP');
    %error('OPTION NOT AVAILABLE')

    tetha=[0:0.01:2*pi];

%*************************************************************************
**********
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*


    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;

    theta = 0;

    for i=1:Contador
        radio(i)= q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i)
0 ...
            nu*(m1(i)+m2(i))]');

        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);
        if (s1(i) < 0)
            s1(i) = n*s1(i);
        end
        if (s2(i) < 0)
            s2(i) = n*s2(i);
        end

    end
    hplot =plot(s1,s2,tipo_linea);
end
%*************************************************************************
**********



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

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strain
,MDtype,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   = 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)

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')

    vartoplot{i}(4) = ce(1,1) ;
    vartoplot{i}(5) = ce(1,1);

    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,:) ;
        eps_n = strain(i-1,:);

%*************************************************************************
**********
        %*      DAMAGE MODEL
        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n1,eps_n,hvar_n,Eprop,ce,MDtype,n,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)

        %tangent operators
        if aux_var(1)>0
            vartoplot{i}(4)=(aux_var(2)*ce(1,1)- ...
                -
Eprop(8)*delta_t(iload)*aux_var(3)*sigma_n1(1)^2)/(Eprop(7)+Eprop(8)*delta_t(
iload));
        else
            vartoplot{i}(4)=hvar_n(6)/hvar_n(5)*ce(1,1);
        end
        vartoplot{i}(5)=hvar_n(6)/hvar_n(5)*ce(1,1);
    end
    end

elseif (viscpr == 0)
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,eps_n1,hvar_n,Eprop,ce,MDtype,n,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
end




function [sigma_n1,hvar_n1,aux_var] = rmap_dano1
(eps_n1,eps_n,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
*
%*                     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) ;
viscpr = Eprop(6) ;
alpha=Eprop(8);
eta=Eprop(7);
A = 0.8;
q_inf=1.8;
%*************************************************************************
*********


%*************************************************************************
*********
%*       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
%*
[rtrial1] = 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;

if (viscpr == 0)
if(rtrial1 > r_n)
    %*   Loading

    fload=1;
    delta_r=rtrial1-r_n;
    r_n1= rtrial1  ;
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    elseif hard_type == 1
        H=A*(q_inf-r0)*exp(A*(1-rtrial1/r0))/r0;
            if (H>1)
                error ('Values of q_inf and A do not satisfy to
restriction.');
            else
               q_n1 = q_inf - (q_inf - r0)*exp(A*(1 - rtrial1/r0));
            end
    else
        % Comment/delete lines below once you have implemented this case
```

```
        % ******************************************************
        menu({'Hardening/Softening exponential law has not been implemented
yet. '; ...
            'Modify file "rmap_dano1" ' ; ...
            'to include this option'},  ...
            'STOP');
        error('OPTION NOT AVAILABLE')
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    end


else

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

end
elseif (viscpr == 1)
    [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
    rtrial_a = ((1-alpha)*rtrial)+(alpha*rtrial1);
    if (rtrial_a > r_n)
        fload = 1;
        delta_r = rtrial_a-r_n;
        r_n1 = ((eta-delta_t(1)*(1-alpha))*r_n)/(eta+(alpha*delta_t(1))) +
...
+ (delta_t(1)*rtrial_a)/(eta+alpha*delta_t(1));
display('delta t: '); display(delta_t);
        if (hard_type == 0)
            q_n1 = q_n + H*delta_r;
        elseif (hard_type == 1)
            H=A*(q_inf-r0)/r0*exp(A*(1-rtrial1/r0));
            if (H>1)
                error ('Values of q_inf and A do not satisfy to
restriction.');
            else
               q_n1 = q_inf - (q_inf - r0)*exp(A*(1 - rtrial_a/r0));
            end
        end
        if (q_n1<zero_q)
            q_n1=zero_q;
        end
    else
        fload = 0;
        r_n1 = r_n;
        q_n1 = q_n;

    end
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;
%*************************************************************************
*********
```