# Computational Solid Mechanics - Assignment 1

Ana Salas Ordóñez

8th April 2016

## 1. Introduction.

After the implementation on Matlab of the continuum isotropic damage ”non-symmetric tension-compression damage” model, the ”tension-only” damage model, the exponential hardening/softening law and the continuum isotropic visco-damage ”symmetric tension-compression” model (see the modified routines in Annex), the results of each implementation will be shown and explained in this report. As constant input parameters for each plot we have: Young modulus (E) = 20000 MPa, Poisson's coefficient ($\nu$) = 0.3 and Yield stress = 200 MPa.

## 2. Results.

### 2.1. Part I - Rate independent models.

In order to check the correctness of the implementation of the different models, we are going to represent different loading paths in the stress space and the stress-strain curve for each one of them. Furthermore, for each model the linear hardening/softening law is going to be used. Once the correctness of the different models is checked, we will proceed to show the proper functioning of the implementation of the exponential hardening/softening law, but only for one model and for different values of the parameters $q_\infty$ and $A$.

The three paths that we used, defined in terms of their corresponding effective stress (MPa), are the following,

$$\begin{array}{ccc}
(1) & (2) & (3) \\
\bar{\sigma}_1^{(1)} = 500; \quad \bar{\sigma}_2^{(1)} = 0 & \bar{\sigma}_1^{(1)} = 500; \quad \bar{\sigma}_2^{(1)} = 0 & \bar{\sigma}_1^{(1)} = 500; \quad \bar{\sigma}_2^{(1)} = 500 \\
\bar{\sigma}_1^{(2)} = 300; \quad \bar{\sigma}_2^{(2)} = 0 & \bar{\sigma}_1^{(2)} = 500; \quad \bar{\sigma}_2^{(2)} = -200 & \bar{\sigma}_1^{(2)} = 300; \quad \bar{\sigma}_2^{(2)} = 300 \\
\bar{\sigma}_1^{(3)} = 700; \quad \bar{\sigma}_2^{(3)} = 0 & \bar{\sigma}_1^{(3)} = 700; \quad \bar{\sigma}_2^{(3)} = 500 & \bar{\sigma}_1^{(3)} = 700; \quad \bar{\sigma}_2^{(3)} = 700
\end{array}$$

### 2.1.1. Tension-only damage model.

For the tension-only damage model the three previous paths are used (Path (1) Figure 1, Path (2) Figure 2, Path (3) Figure 3). In Figure 1, it can be easily seen that the material deforms elastically with an expansion (hardening) of the elastic domain, besides looking at the location of the stress state in relation with the prevailing damage surface, one can ascertain at each step whether the material deforms elastically with or without damage. Thus, we can see how this model is implemented correctly since none of the stress state is outside the elastic domain, which it can occur for viscous cases (Part II) but not for the inviscid ones. For different paths, when the slope is constant on the stress strain curve the material deforms elastically without damage, as soon as the slope decrease the material deforms elastically with damage.

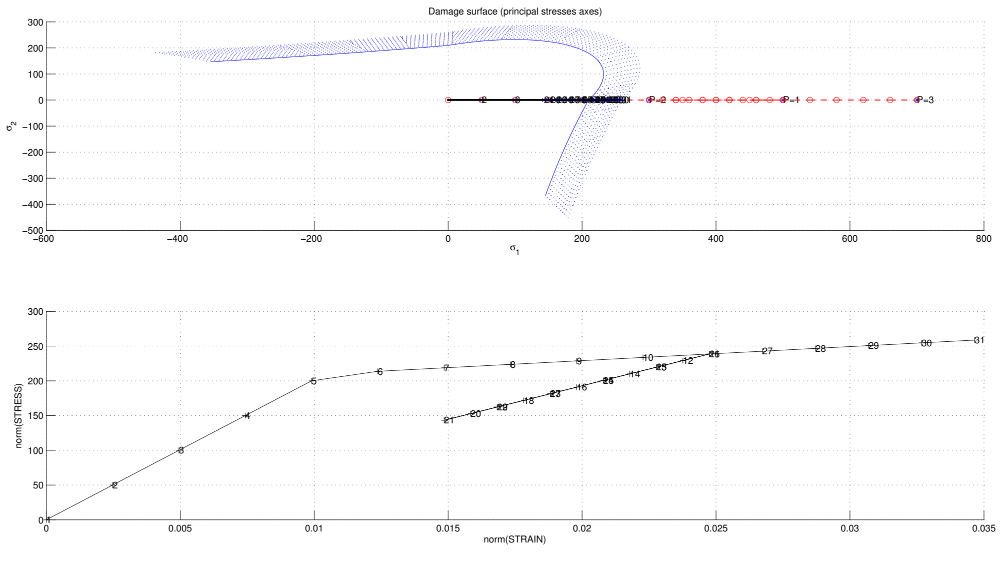

Figure 1: Loading path (1) in the stress space and stress-strain curve for tension-only damage model.

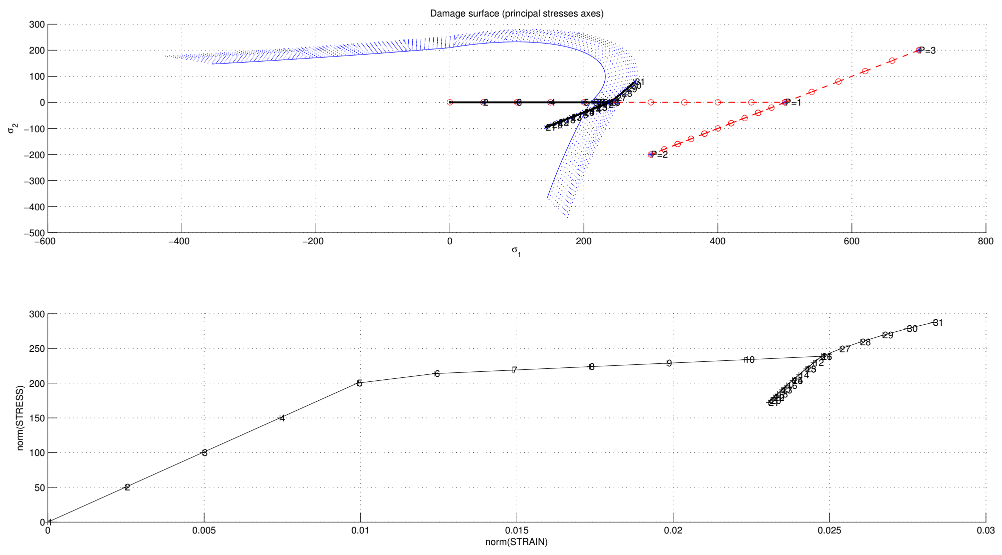

Figure 2: Loading path (2) in the stress space and stress-strain curve for tension-only damage model.

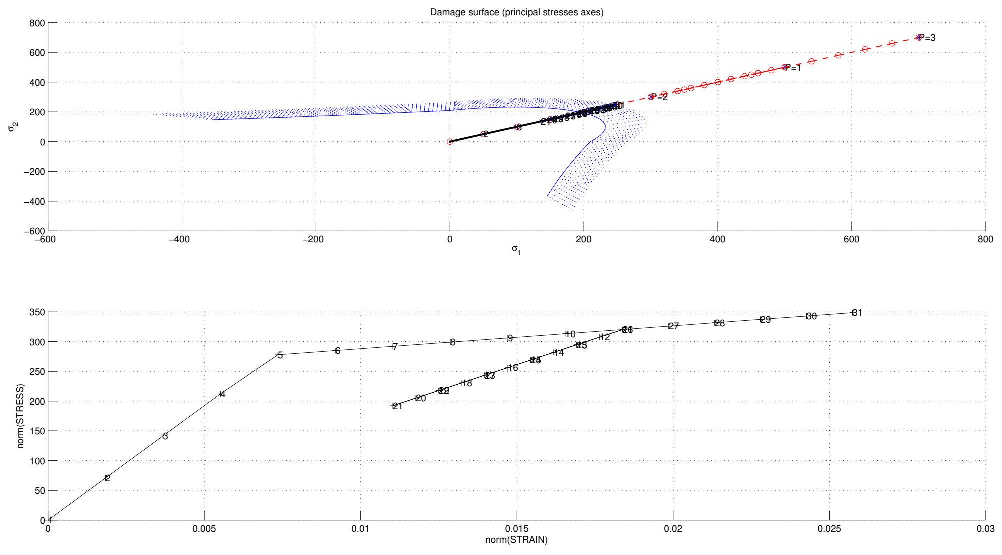

Figure 3: Loading path (3) in the stress space and stress-strain curve for tension-only damage model.

### 2.1.2. Non-symmetric tension-compression damage model.

For the non-symmetric tension-compression damage model the three previous path are used (Path (1) Figure 4, Path (2) Figure 5, Path (3) Figure 6). As it has been seen for the tension-only damage model, the curve stress strain for each path are exactly the same for both models, and indeed the material deforms elastically with an expansion (hardening) of the elastic domain. As we saw in the tension-only model, for different paths, the slope is constant as long as there is no damage until a certain point it begins to curve, as soon as the slope decrease the material deforms elastically with damage.

None of the stress state lay outside the elastic domain, which is consistent with our model and inviscid case, thus, this model is implemented correctly.

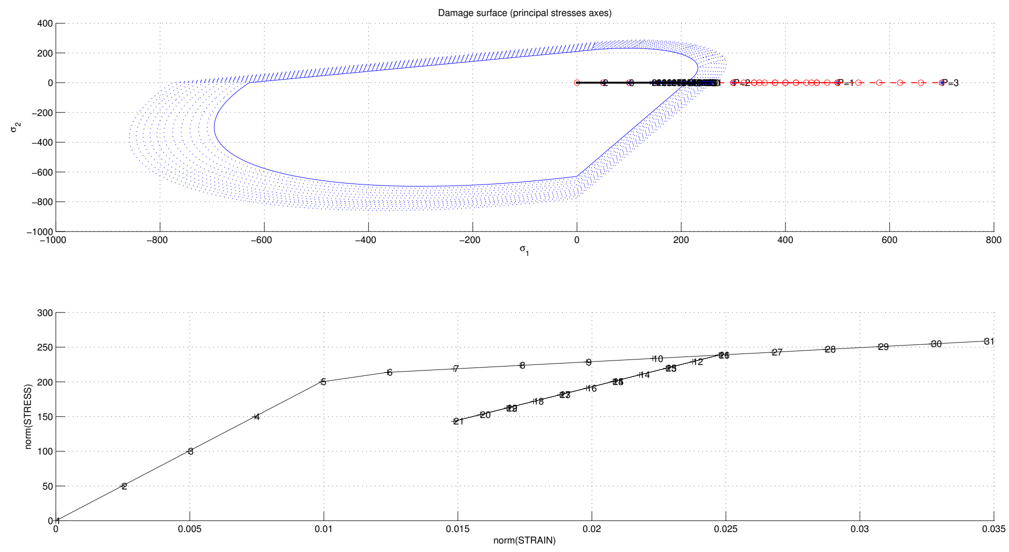

Figure 4: Loading path (1) in the stress space and stress-strain curve for non-symmetric tension-compression damage model.

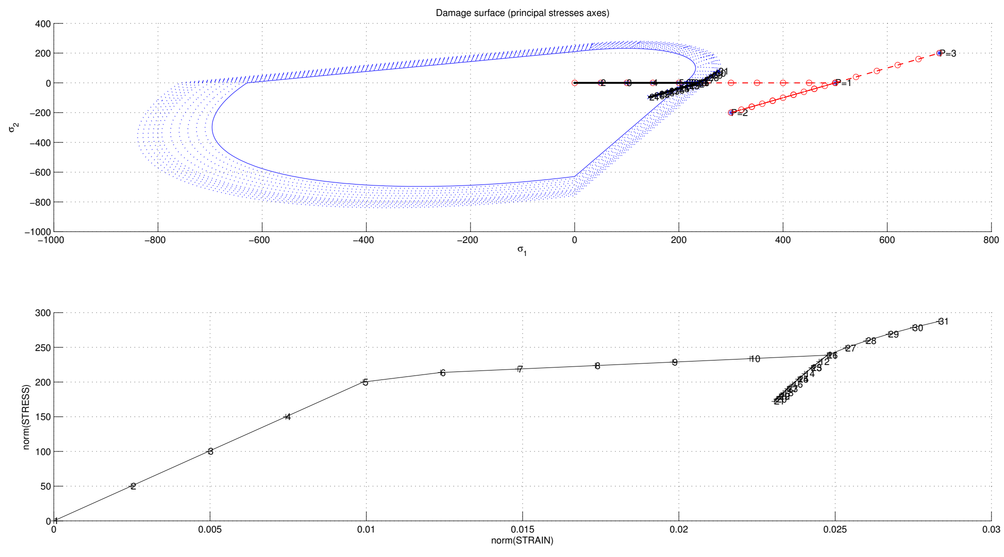

Figure 5: Loading path (2) in the stress space and stress-strain curve for non-symmetric tension-compression damage model.

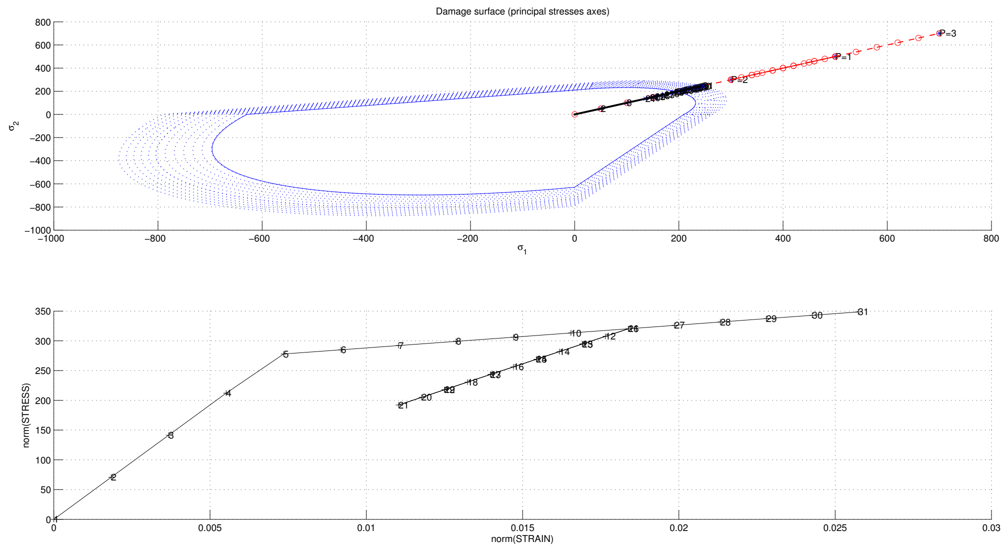

Figure 6: Loading path (3) in the stress space and stress-strain curve for non-symmetric tension-compression damage model.

### 2.1.3. Exponential hardening/softening law.

In this section, the correctness of the exponential hardening/softening law will be tested. To do so, a non-symmetric tension-compression damage model and the following loading path are used,

$$\begin{aligned} \bar{\sigma}_1^{(1)} &= 300; \quad \bar{\sigma}_2^{(1)} = 400 \\ \bar{\sigma}_1^{(2)} &= 500; \quad \bar{\sigma}_2^{(2)} = 400 \\ \bar{\sigma}_1^{(3)} &= 500; \quad \bar{\sigma}_2^{(3)} = 0 \end{aligned}$$

For the exponential hardening/softening law it is allowed to change the value of the parameters $q_\infty$ and $A$, being $A$ always bigger than zero. In Figure 7 it is represented the stress strain curve for different values of $q_\infty$. It can be seen that for $H > 0$ ($q_\infty >$r$_0 \rightarrow$ Damage with hardening (green line)), the elastic zone increases, for $H < 0$ ($q_\infty <$r$_0 \rightarrow$ Damage with softening (blue line)), the elastic zone decreases, and for $H = 0$ ($q_\infty =$r$_0 \rightarrow$ Perfect damage (red line)), the elastic zone remains the same.

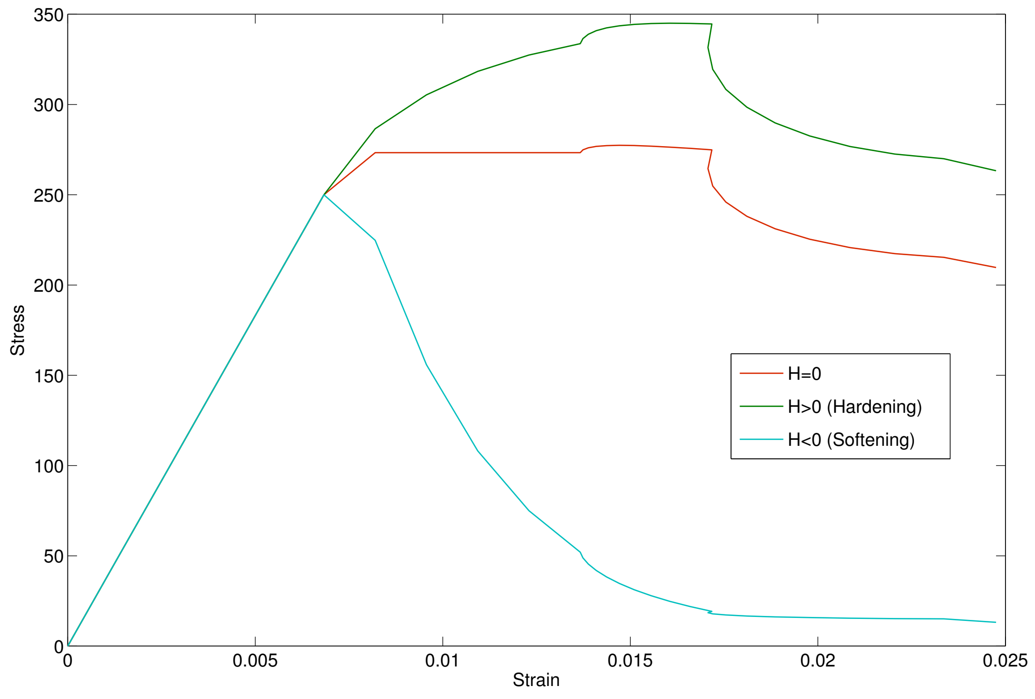

Figure 7: Curve stress strain for different values of $q_\infty$.

Moreover, if the value of $q_\infty$ remains constant and the value of $A$ is changed Figure 8 is obtained ($q_\infty > r_0 \rightarrow$ Hardening). While the value of $q_\infty$ it is what makes the elastic zone to increase or decrease, increasing the value of $A$ makes, if the damage is with hardening, that the elastic zone increases more, if the damage is with softening, that the elastic zone decreases more.

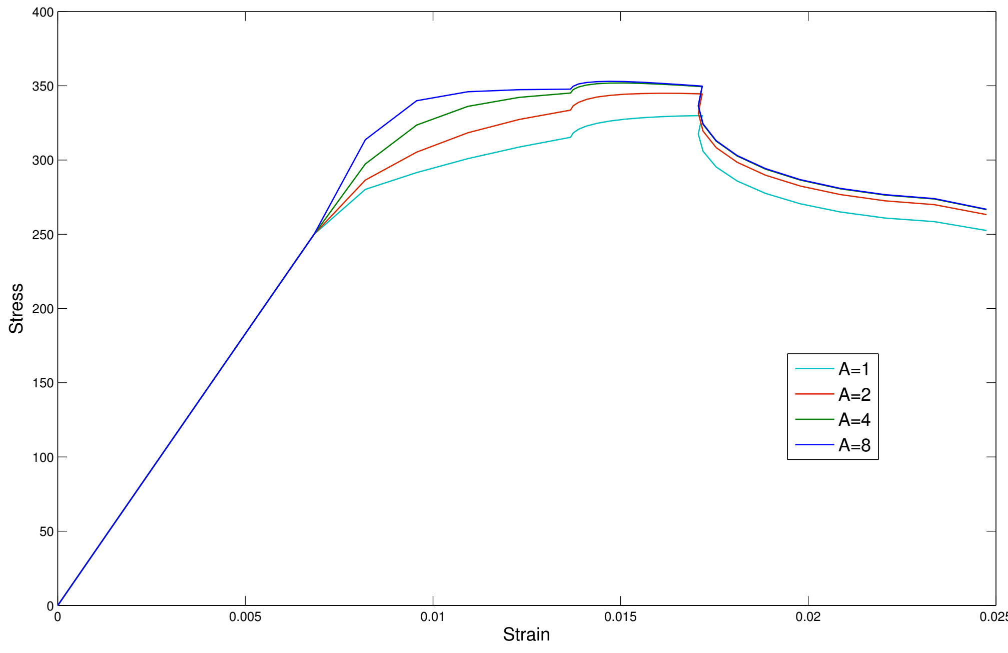

Figure 8: Curve stress strain for different values of $A$.

## 2.2. Part II - Rate dependent models.

In order to check the correctness of the implementation for the continuum isotropic visco-damage "symmetric tension- compression" model, for a specific given Poisson ratio (0.3) and linear hardening/softening parameter we are going to represent the stress space and the stress strain curve for the same loading path used in section 2.1.3. for different values values of viscosity parameter ($\eta$), strain rate ($\dot{\varepsilon}$) and $\alpha$. Once the correctness of this model is checked, we will show the effects of the $\alpha$ values, on the evolution along time of the $C_{11}$ component of the tangent and algorithmic constitutive operators.

### 2.2.1. Effects of the different values of viscosity parameter, strain rate and $\alpha$ on the obtained stress-strain curves in appropriate loading paths.

In Figure 9 it is shown how increasing the value of the viscosity parameter the values of the stress increase as well, which is consistent with the theory given in class. Being this the viscous case, the stress states can lay outside the elastic domain, as it can be seen in the stress space.

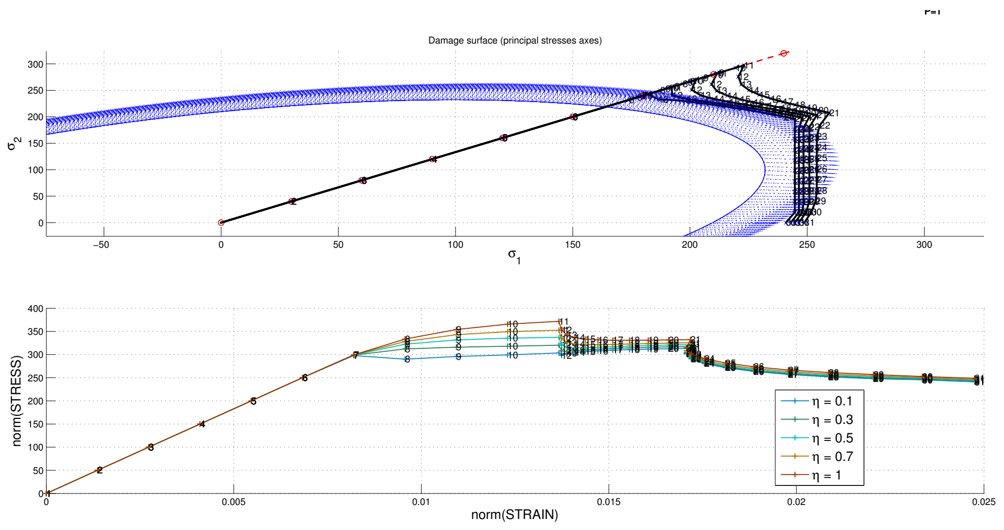

Figure 9: Stress space and stress-strain curve for the continuum isotropic visco-damage "symmetric tension- compression" model for different values of $\eta$.

Comparing Figures 9 and 10 can be confirmed that the strain rate and the viscosity have similar effects on the strain-stress relation, when both the strain rate and the viscosity parameter increase the strain-stress relation increase as well.

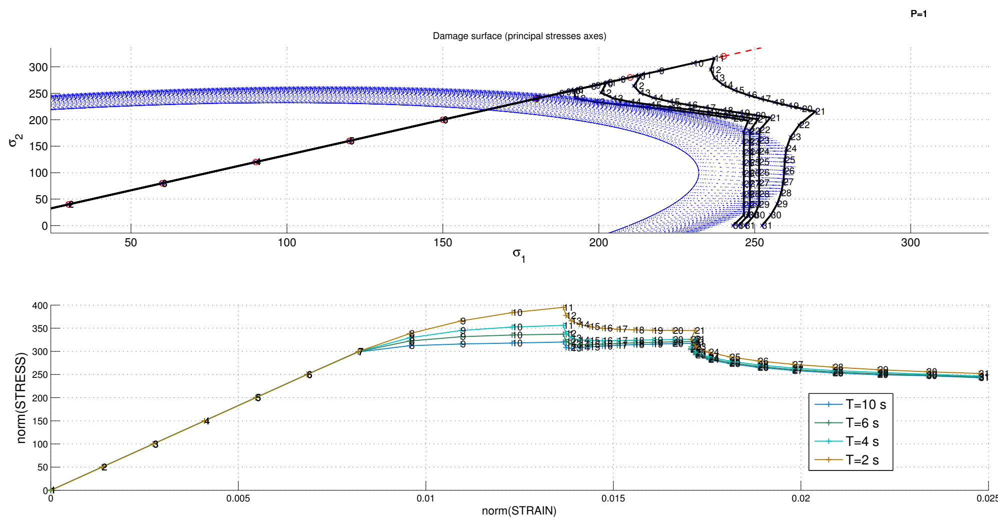

Figure 10: Stress space and stress-strain curve for the continuum isotropic visco-damage "symmetric tension- compression" model for different values of strain rate.

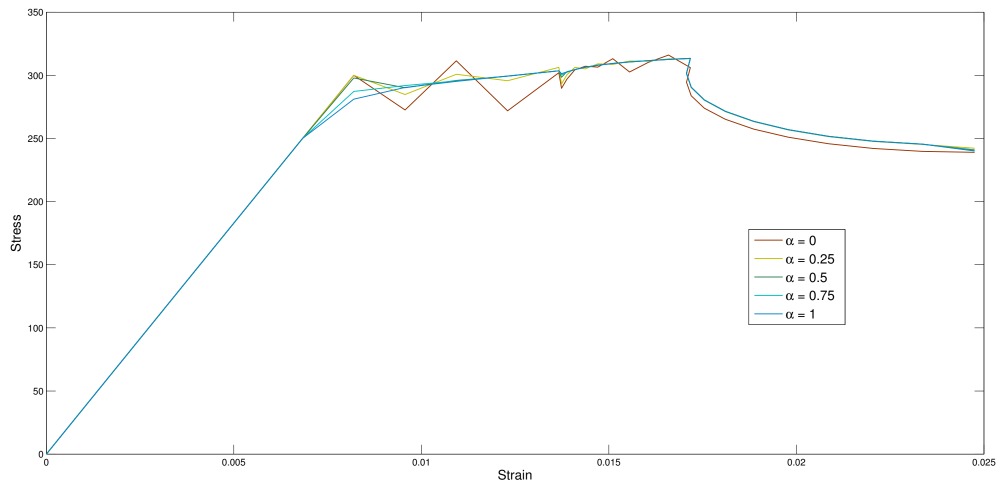

Figure 11: Curve stress strain for different values of $\alpha$ for $\eta = 0.1$.

Plotting the curves stress strain for different values of $\alpha$, it is shown that for values of $\eta$ close to 0, as it can be $\eta = 0.1$, the method is unstable for values of $\alpha$ between 0 and 0.5 since non-physical oscillations appear (Figure 11), which it is consistent with the analytical stability analysis of this method. Furthermore, when the value of $\eta$ is increased ($\eta = 0.5$) the method is stable for all values of $\alpha$, as we can see in Figure 12, but only for $\alpha = 0.5$ second order accuracy can be obtained.

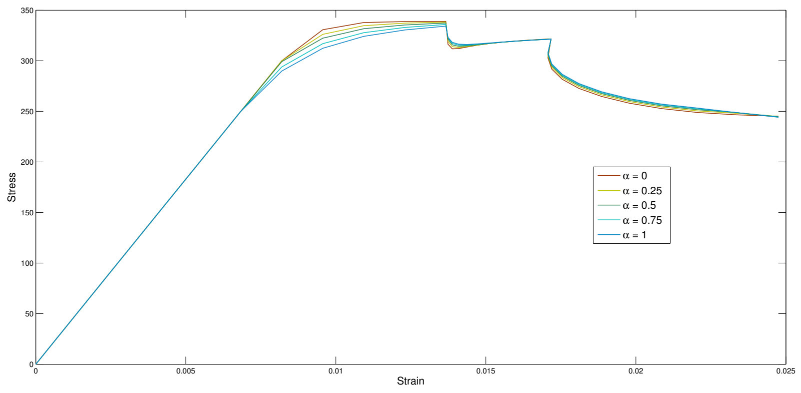

Figure 12: Curve stress strain for different values of $\alpha$ for $\eta = 0.5$.

### 2.2.2. Effects of the $\alpha$ values on the evolution along time of the $C_{11}$ component of the tangent and algorithmic constitutive operators.

As we can see in Figure 13 and 14, the initial value for both constitutive operators is the same due to there is not damage yet, as soon as the damage of the elastic domain begin the value of the $C_{11}$ component of the tangent constitutive operator is bigger than the value of the algorithmic one, which is consistent with the formula. Moreover, when the value of $\alpha$ increases we get lower values of the $C_{11}$ component of the algorithmic constitutive operator.

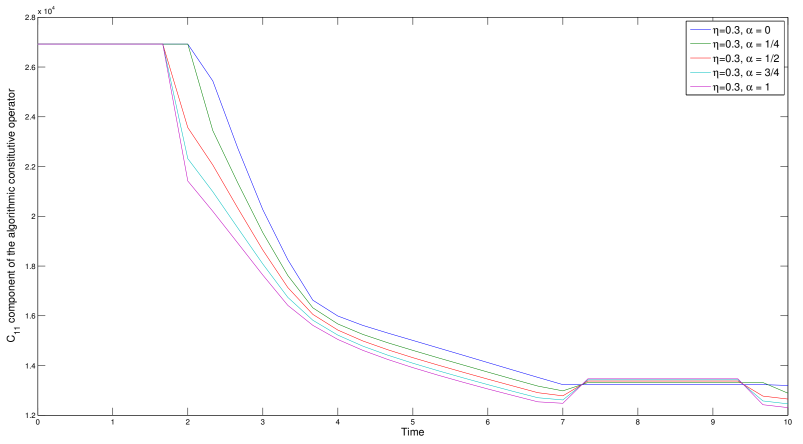

Figure 13: C11 component of the algorithmic constitutive operators for different values of $\alpha$.

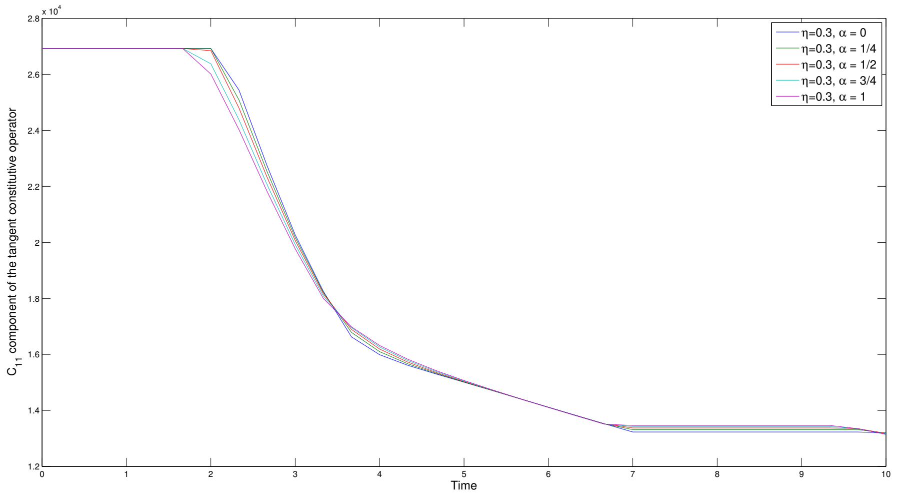

Figure 14: C11 component of the tangent constitutive operators for different values of $\alpha$.

# 3. Annex

## 3.1. Tension-only and Non-symmetric tension-compression damage models

### 3.1.1. dibujar_criterio_dano1.m

```
elseif MDtype==2 %Tension-only

    tetha=[-0.38*pi:0.001:0]; %Fourth quadrant
    m1=cos(tetha);
    m2=sin(tetha);
    D=size(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) 0 0
            ...
            nu*(m1(i)+0)]');

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


    tetha1=[0:0.001:pi/2]; %First quadrant
    m1=cos(tetha1);
    m2=sin(tetha1);
    D=size(tetha1);
    Contador1=D(1,2);

    for i=1:Contador1
        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+Contador)=radio(i)*m1(i);
        s2(i+Contador)=radio(i)*m2(i);

    end

    tetha2=[pi/2+0.01:0.001:7*pi/8]; %Second quadrant
    m1=cos(tetha2);
    m2=sin(tetha2);
    D=size(tetha2);
    Contador2=D(1,2);
    for  i=1:Contador2
         radio(i)=q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[0 m2(i) 0
             ...
             nu*(0+m2(i))]');

         s1(i+(Contador1+Contador))=radio(i)*m1(i);
         s2(i+(Contador1+Contador))=radio(i)*m2(i);

    end

    hplot =plot(s1,s2,tipo_linea);




elseif MDtype==3 %Non-symmetric

    tetha1=[0+0.001:0.001:pi/2];  %First quadrant
    m1=cos(tetha1);
    m2=sin(tetha1);
    D=size(tetha1);
    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

    tetha2=[pi/2+0.01:0.01:pi]; %Second quadrant
    m1=cos(tetha2);
    m2=sin(tetha2);
    D=size(tetha2);
    Contador1=D(1,2);
    s_u=s2(Contador);

    for  i=1:Contador1
           s2(i+Contador)=s_u/(1+1/(-n*tan(tetha2(i))));
           s1(i+Contador)=s2(i+(Contador))/tan(tetha2(i));
    end

    tetha3=[pi+0.01:0.01:(3*pi/2)-0.01]; %Third quadrant
    m1=cos(tetha3);
    m2=sin(tetha3);
    D=size(tetha3);
    Contador2=D(1,2);

    for i=1:Contador2
        radio(i)=(q*n)/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+(Contador1+Contador))=radio(i)*m1(i);
        s2(i+(Contador1+Contador))=radio(i)*m2(i);
    end

    tetha=[-pi/2:0.01:0]; %Fourth quadrant
    m1=cos(tetha);
    m2=sin(tetha);
    D=size(tetha);
    Contador3=D(1,2);

    for i=1:Contador3
          s2(i+(Contador1+Contador+Contador2))=-n*s_u/(1-n/(tan(tetha(i))))
              ;
          s1(i+(Contador1+Contador+Contador2))=s2(i+(Contador1+Contador+
              Contador2))/tan(tetha(i));
    end

    hplot =plot(s1,s2,tipo_linea);

end
```

### 3.1.2. Modelo_de_dano1.m

```
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_p=sigma_v;

  for i=1:4
```

```
        if sigma_p(i)<0
           sigma_p(i)=0;
        end
     end

     tetha=(sigma_p(1)+sigma_p(2)+sigma_p(4))/(abs(sigma_v(1))+abs(sigma_v(2))
          +abs(sigma_v(4)));
     rtrial=(tetha+(1-tetha)/n)*sqrt(sigma_v*eps_n1');

end
```

## 3.2. Exponential law and continuum isotropic visco-damage ”symmetric tension- compression” model

### 3.2.1. Damage_main.m

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,C_tang,C_alg]=damage_main(
    Eprop,ntype,istep,strain,MDtype,n,TimeTotal,A)
global hplotSURF

 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

if viscpr == 0
     Eprop(7)=0;
     Eprop(8)=1;
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)
rtrial=0;
C_tang=ones*ce; %C_tangent tensor
C_alg=ones*ce; %C_algorithmic tensor

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        deltat=delta_t(iload);
        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,rtrial] = rmap_dano1(eps_n1,hvar_n,Eprop,
            ce,MDtype,n,A,deltat,i,rtrial);
        % 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


        %C_tangent and C_algorithmic tensors

        if(aux_var(1)>0)
            d=1-aux_var(2);
            C_alg(:,:,i)=(1-d)*ce+aux_var(3)*(sigma_n1*sigma_n1')/((1-d)^2)
                ;
            C_tang(:,:,i)=(1-d)*ce;
        else
            d=1-aux_var(2);
            C_alg(:,:,i)=(1-d)*ce;
            C_tang(:,:,i)=(1-d)*ce;
        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
```

### 3.2.2. rmap_dano1.m

```
function [sigma_n1 ,hvar_n1 ,aux_var ,rtrial] = rmap_dano1 (eps_n1 ,hvar_n ,
    Eprop ,ce ,MDtype,n ,A, deltat ,i ,rtrial)

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) ;
eta      = Eprop(7);
alpha    = Eprop(8);
viscpr   = Eprop(6);
%*****************************************************************
%***************************************************************
%*        initializing                                              %*
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
% if(r_n<=0.d0)
%      r_n=r0;
%      q_n=r0;
% end
%**************************************************************
rtrial_n=rtrial;
%**************************************************************
%*        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;

tau_nalpha=(1-alpha)*rtrial_n+alpha*rtrial;

if(tau_nalpha > r_n)
    %*    Loading

     fload=1;
    r_n1=((eta-deltat*(1-alpha))/(eta+alpha*deltat))*r_n+(deltat/(eta+alpha
        *deltat))*tau_nalpha;
    delta_r=r_n1-r_n;

    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        % Exponential
        if H>0 %Hardening
        q_inf=1.8; %Introducir valor
        q_n1=q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
        elseif H<0 %Softening
        q_inf=zero_q; %Introducir valor
        q_n1=q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
        elseif H==0
        q_n1=q_n;
        end
    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
% 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;
if viscpr==1
    aux_var(3) = alpha*deltat/(eta+alpha*deltat)*(1/rtrial)*(H*r_n1-q_n1)/(
        r_n1^2);
else
    aux_var(3) = -(q_n1-H*r_n1)/(r_n1^3);
end
%*********************************************************
```