Computational Solid Mechanics

Assignment 1

Numerical integration of constitutive damage models using MATLAB

Oscar Salvador
[Date]

SUMMARY

Background

There are many engineering problems in which the emergence of micro-cracks and its propagation cause large differences between the actual and the "elastic" behaviour. For that reason, the classical linear elastic models are not appropriate to represent the physics of this kind of problems, and some deeper analysis is required. Along this report, some continuum damage models are computed in order to predict this damage propagation, and in this way, a complex and computationally-expensive microscopic description of the problem is avoided. The analysis is focused on the damage evolution, therefore, the strain history of the problem is prescribed as an input of the model.

Aim

The objective of the survey is to compute an inviscid damage model, as well as a viscous one capable of predict the propagation of the damage under several loading paths. In order to assess the correctness of the models, the evolution of the damage surface on the stress-space as well as the stress-strain curves are analysed to compare the results with the theory.

Rate independent models

Regarding this type of non-viscous models, three of them are implemented in the code.

- Symmetric (tension/compression) model
- Tensile-damage model
- Non-symmetric (tension/compression) model

The symmetric model has already been previously implemented, therefore, the following assessment is focused on the correctness of the tensile model and the non-symmetric one, as well as on the modifications of the routines to compute them.

Tensile-damage model

This model characterizes materials which only fail under tension, and does not take into account compression failure. To represent this behaviour, the elastic domain is open in the compression space and the elastic limit cannot be reached when all the principal stress are negative. In order to compute this special case of elastic domain the following expression is implemented on the code.

$$\tau_\varepsilon = \sqrt{\bar{\sigma}^+ : \varepsilon}$$

Where $\bar{\sigma} = \mathbb{C} : \varepsilon$ and $\bar{\sigma}^+$ makes zero the negative eigenvalues of $\bar{\sigma}$.

Non-symmetric tension-compression model

This model is intended to capture the behaviour of materials whose tensile strength differs significantly from its compression strength. For that reason the elastic domain is not symmetric and is enlarged in the compression space. Some new parameters are added to "modelos_de_dano1" and "dibujar_criterio_dano" routines. The implemented expression is described below.

$$\tau_\varepsilon = \left[\theta + \frac{1-\theta}{n}\right]\sqrt{\varepsilon : \mathbb{C} : \varepsilon} \tag{1}$$

$$\theta = \frac{\sum_1^2 \langle \sigma_i \rangle}{\sum_1^2 |\sigma_i|} = \frac{\sum_1^2 \langle \bar{\sigma}_i \rangle}{\sum_1^2 |\bar{\sigma}_i|} \tag{2}$$

Being theta and n parameters which control the shape of the compression part.

Exponential hardening/softening law

An exponential hardening/softening law is implemented by means of an appropriate selection of different parameters which are presented below

$$H(r) = \frac{dq(r)}{dr} = A\frac{(q_\infty - r_0)}{r_0} e^{A\left(1-\frac{r}{r_0}\right)} \quad being\ A > 0 \tag{3}$$

If the parameter "delta_t" is small enough the previous expression can be linearized and q_{n+1} can be computed as follows

$$q_{n+1} = q_n + A\frac{q_\infty - r_0}{r_0} e^{A\left(1-\frac{r_{n+1}}{r_0}\right)}(r_{n+1} - r_n) \tag{4}$$

where A is a prescribed hardening/softening constant.

After applying the appropriate modifications to the code, some results of the q-r curve are plotted to ensure the proper implementation of the exponential law.

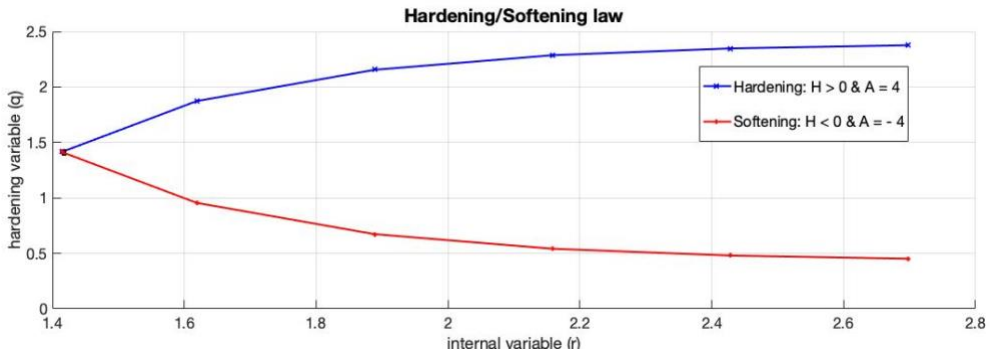

Figure 1: Exponential Hardening/Softening law

The value of q_∞ has been parametrized as follows

$$q_\infty = r_0 + (r_0 - zero\_q) \tag{5}$$

being zero_q a bound for $q(r)$ to avoid negative values.

Verification

With the aim of verifying the code some loading paths are applied to check the response of the non-symmetric and tension-only damage models. The loading paths are composed of three-segment paths in the strain space, which are defined in terms of their corresponding effective stresses.

The different loading paths are shown in the following table

Table 1: Loading paths applied to verify the models

Model	**Path**	**Seg.1 ($\Delta\bar{\sigma}_1, \Delta\bar{\sigma}_2$)**	**Seg. 2 ($\Delta\bar{\sigma}_1, \Delta\bar{\sigma}_2$)**	**Seg. 3 ($\Delta\bar{\sigma}_1, \Delta\bar{\sigma}_2$)**
Tension-only	#1	(400,0)	(−900,0)	(500,0)
	#2	(400,0)	(−700, −700)	(800,800)
	#3	(400,400)	(−600, −600)	(800,800)
Non-symm.	#1	(500,0)	(−2500,0)	(2000,0)
	#2	(400,0)	(−1200, −1200)	(1000,1000)
	#3	(400,400)	(−1900, −1900)	(2500,2500)

The total simulation time is set to 10, and the number of time steps is 10 per segment.

Tensile-model

Path 1

The evolution of the damage surface as well as the loading path are shown below

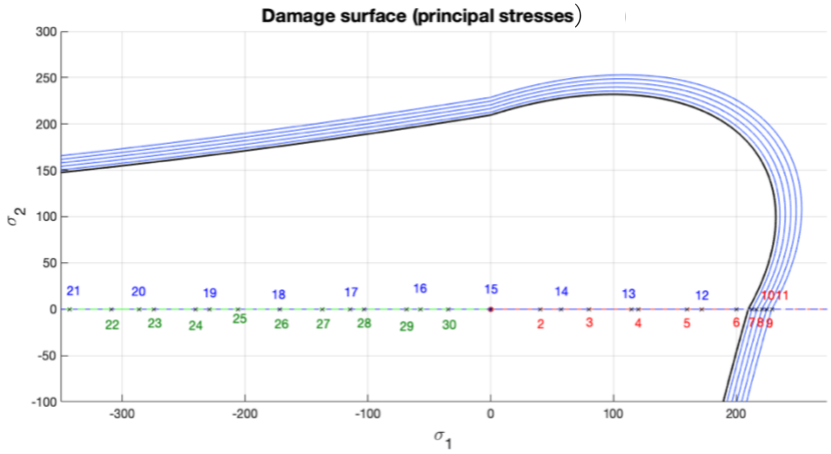

Figure 2: Damage surface of the path 1 (only-tension model)

As can be noted from the above plot, this loading path only works along the axis of the principal stress σ_1. At time step n=7 the load exceeds the elastic domain and the damage surface responds accordingly growing. Then, it keep expanding up to the end of the first path segment (n=11). This behaviour corresponds to the expected one, as the theory suggests.

The stress-strain curve of the loading path is presented below

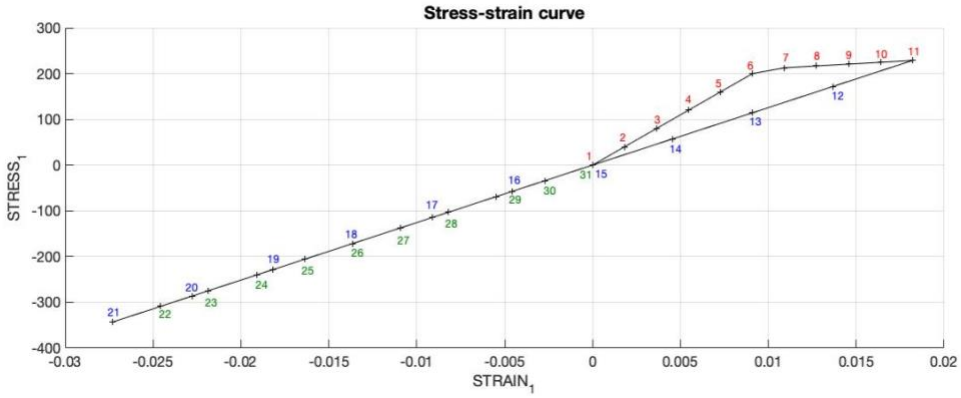

Figure 3: Stress-strain curve of the path 1 (only-tension model)

It is worth noting that the slope of the curve along the second and the third segment (blue and green respectively) remains constant and differs from the one of the first segment (red). As the first load goes beyond the elastic limit, the damage (d) starts to grow. Then, when the second path starts to discharge on the opposite direction, the damage remains constant and the slope is reduced by a factor (1-d). Finally, as the theory sustains, the compressive load don't reach the elastic limit and the third segment (tension) keep the same slope.

Path 2

At this point, the model is tested by applying a loading path that combines stresses and deformation of both principal stresses axis.

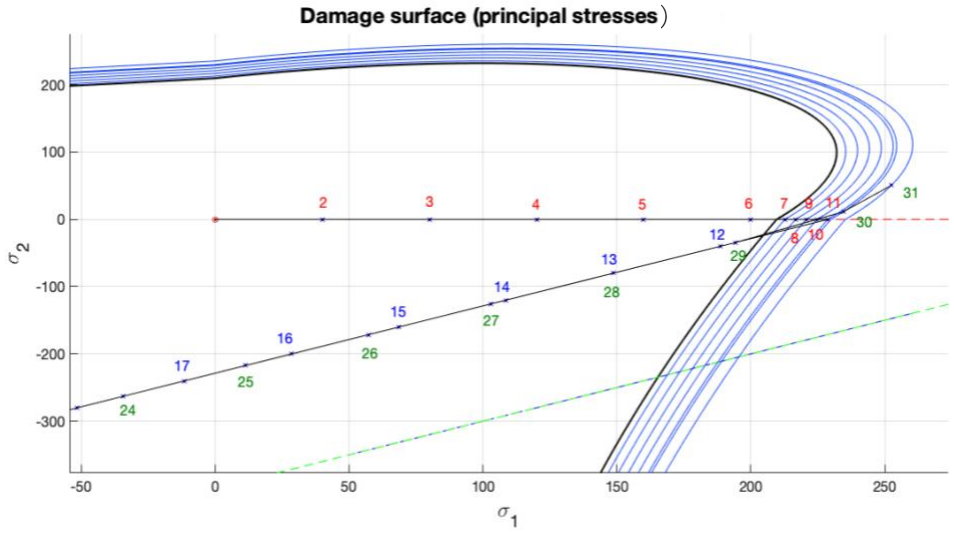

Figure 4: Damage surface of the path 2 (only-tension model)

Here, the plot does not clearly show the loading path, but it is focused on the damage surface evolution (black solid line). During the first segment the damage surface grows. After that, there is a discharge during the second segment, and finally, the third segment ends out of the previous damage surface forcing it to expand.

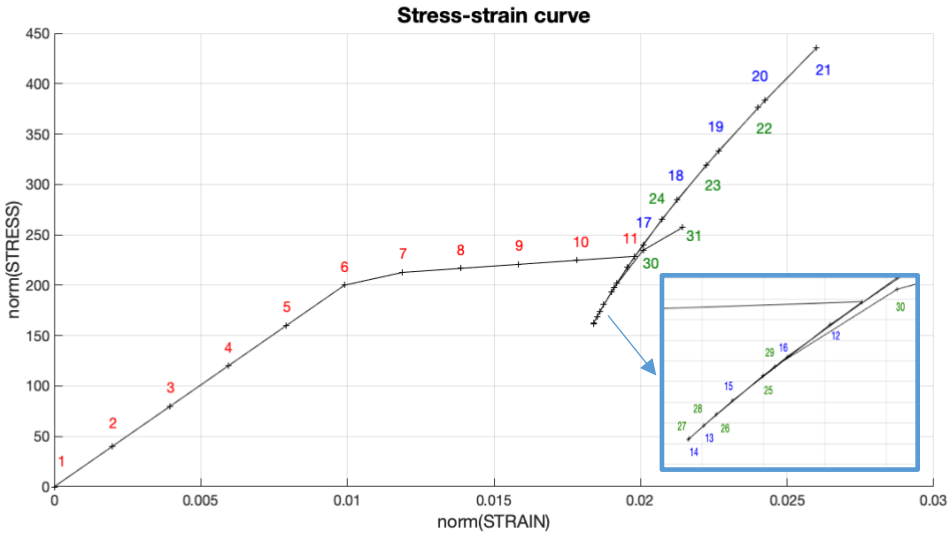

Figure 5: Stress-strain curve of the path 2 (only-tension model)

In this plot the first loading segment exceeds the elastic domain and damage starts to propagate by means of a linear hardening law. After that, the second segment returns to the (0,0) point and after the time step n=17 the compression load begins to grow linearly as it was expected.

Path 3

The last loading path to asses the implementation of the model is a symmetric path respect to both principal stresses axis.

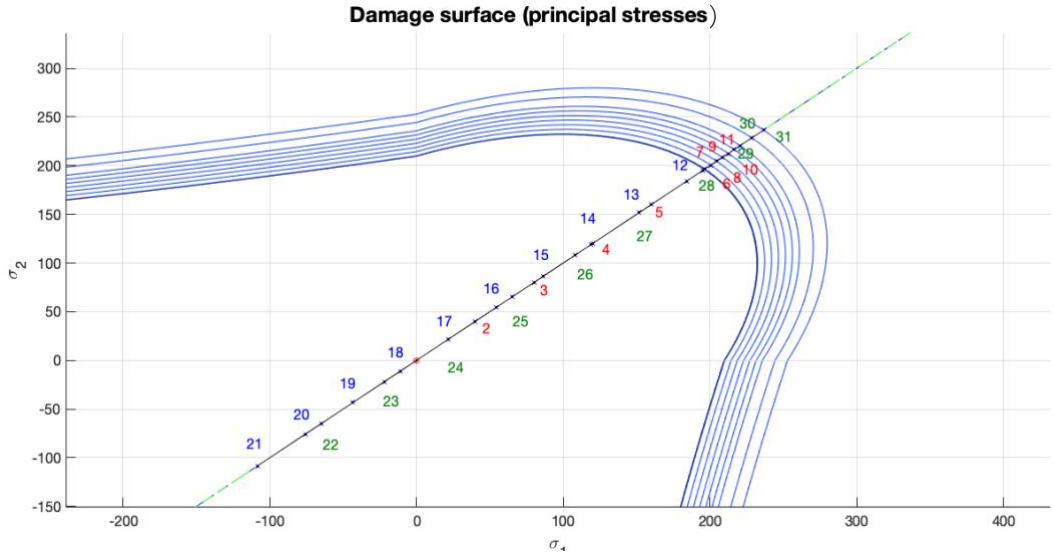

Figure 6: Damage surface of the path 3 (only-tension model)

The evolution of the damage surface is symmetric respect to the principal axis. Firstly, the damage surface grows up to n=11 and after that, as the previous loading paths have shown, the compression segment (blue) is entirely elastic. Finally, the third segment increases the size of the damage model in the same way.

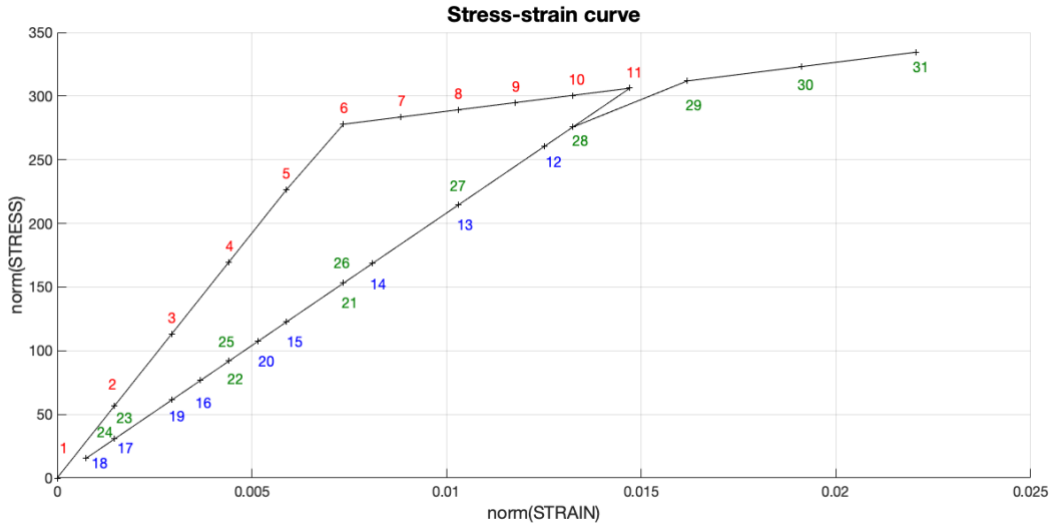

Figure 7: Stress-strain curve of the path 3 (only-tension model)

The first segment shows a linear hardening law (from n=6 to n=11). After that, the second segment goes to the neutral point elastically and grows in their last three points. Finally, the third segment

increases linearly and after n=28 the damage propagates. This behaviour is the same that the theory sustains.

Non-symmetric model

Path 1

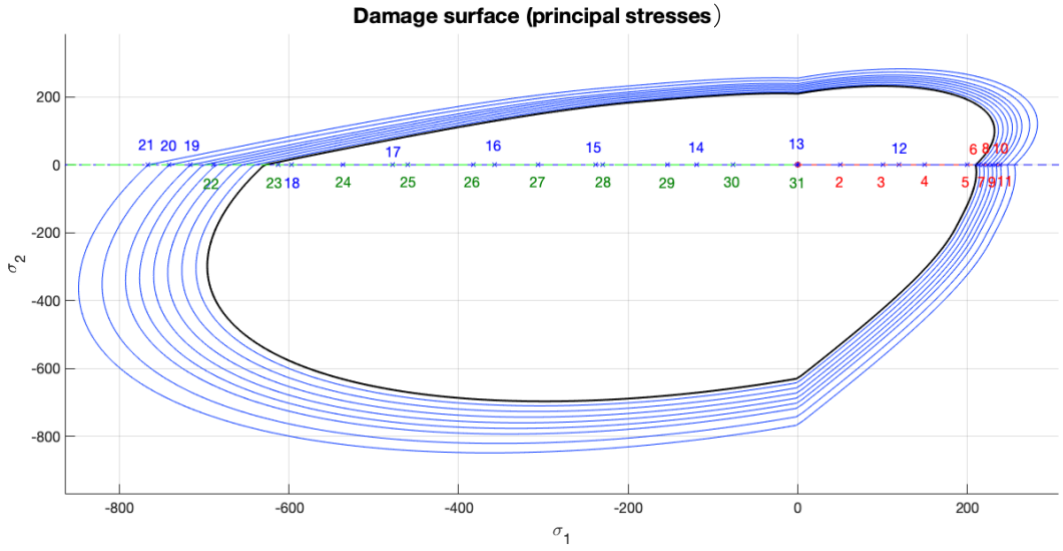

Figure 8: Damage surface of the path 1 (Non-symmetric model)

It is worth noting that the damage surface behaves growing its size during the first and the second segment (red & blue), as it was expected. The particular shape of this non-symmetric damage surface involves an elastic domain more extended along the compression part. This non-symmetry can be controlled trough the factor n.

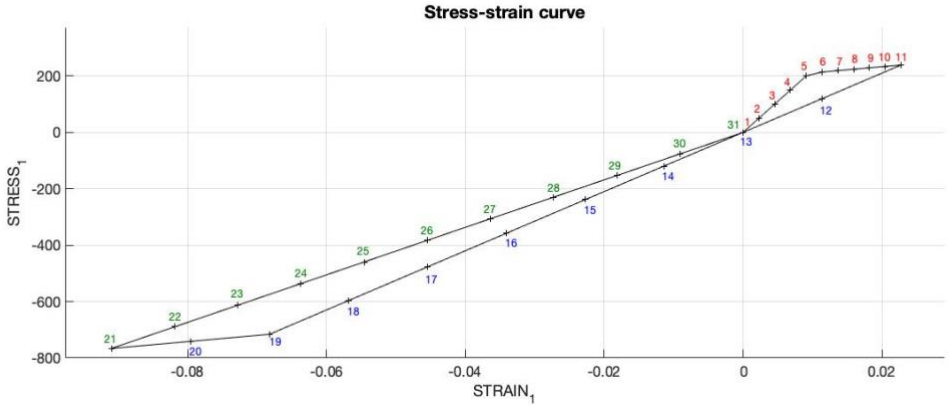

Figure 9: Stress-strain curve of the path 1 (non-symmetric model)

As can be seen, the first segment (red) arrives quickly to the non-elastic domain (n=5), meanwhile the second segment (blue) needs about four times more compression loading to reach the boundary of the elastic domain. This behaviour is due to the non-symmetry nature of the model, and is consistent with the theory presented.

Path 2

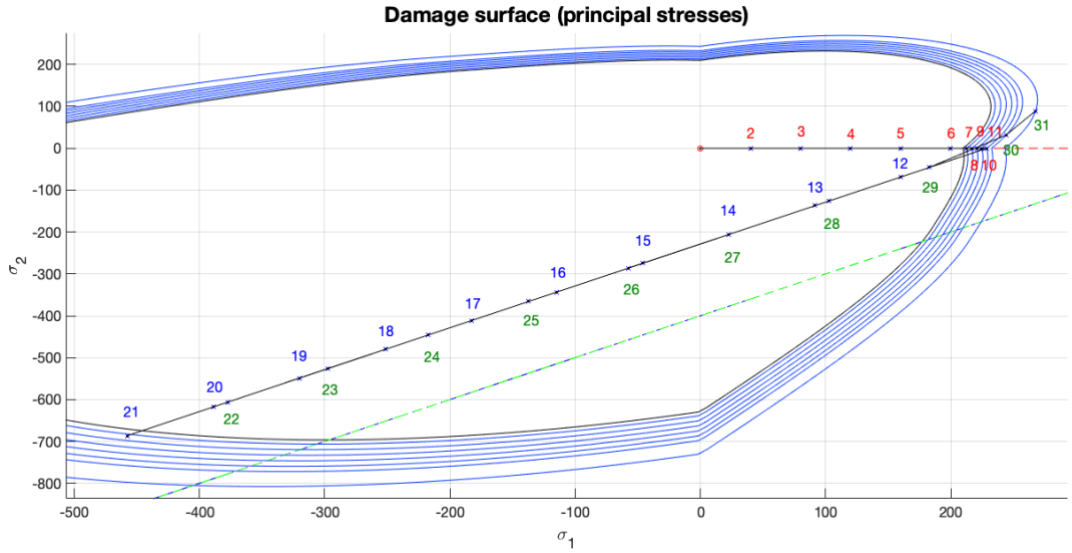

Figure 10: Damage surface of the path 2 (non-symmetric model)

This plot shows how a high compression loading remains on the elastic domain (blue), and furthermore, how two lower tension loadings (red & green) exceed the elastic domain due to the non-symmetry. Therefore, regarding the consistency with the theory, this loading path does not add anything in this sense

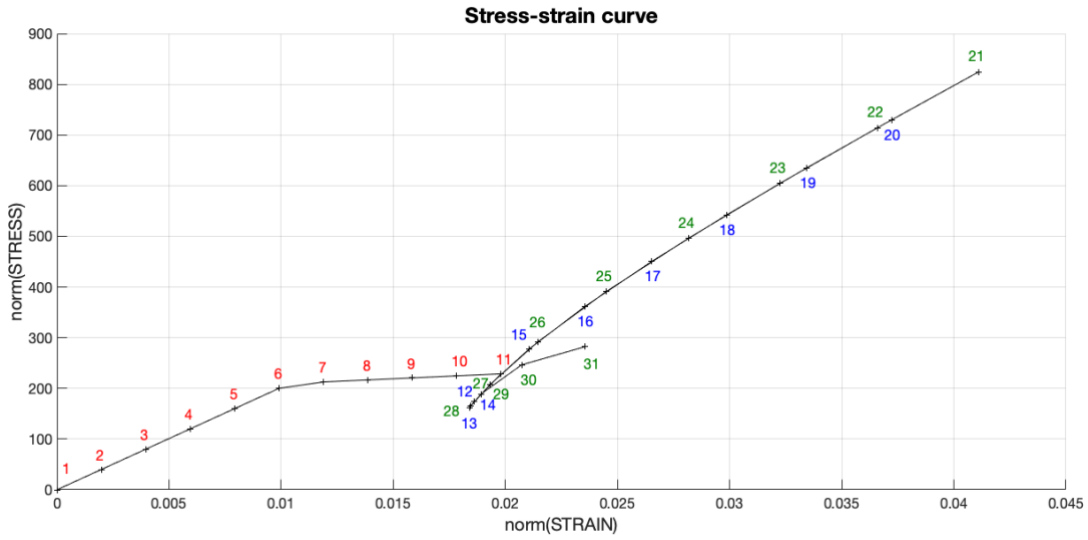

Figure 11: Stress-strain curve of the path 2 (non-symmetric model)

More conclusions can be extracted of the stress-strain curve. As can be seen, the damage starts to grow from n=6, and the slope is affected by this issue during the following segments. As it was expected due to the non-symmetry, the second segment (blue), which is a compression loading, does not produce more damage and the discharge (green) is carried out keeping the same slope.

Path 3

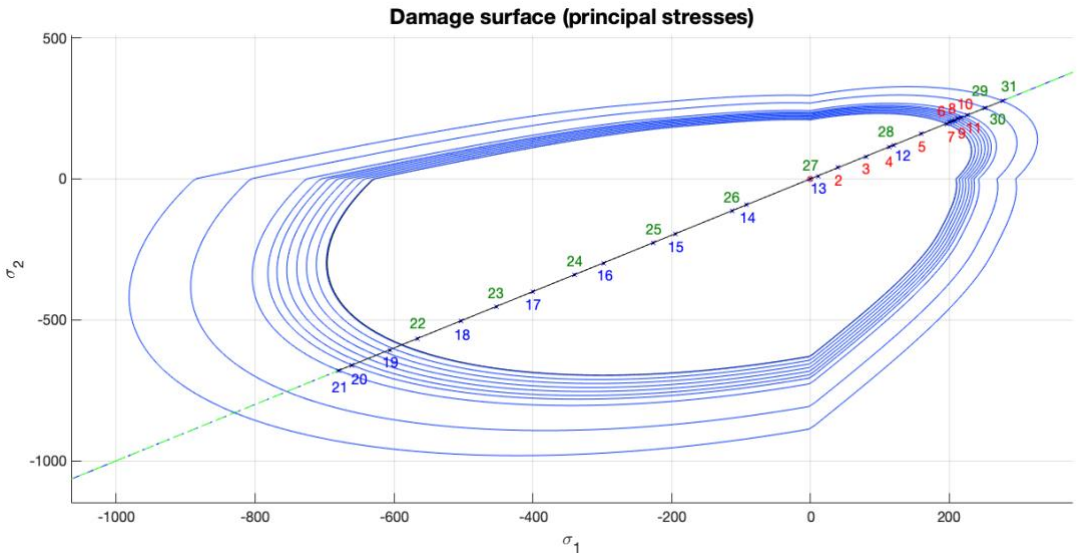

Figure 12: Damage surface of the path 3 (non-symmetric model)

This loading path makes the damage surface grow symmetrically as shown with a similar path in the tensile-damage model. All the three segments exceed the elastic domain, and in consequence, the damage surface grows accordingly.

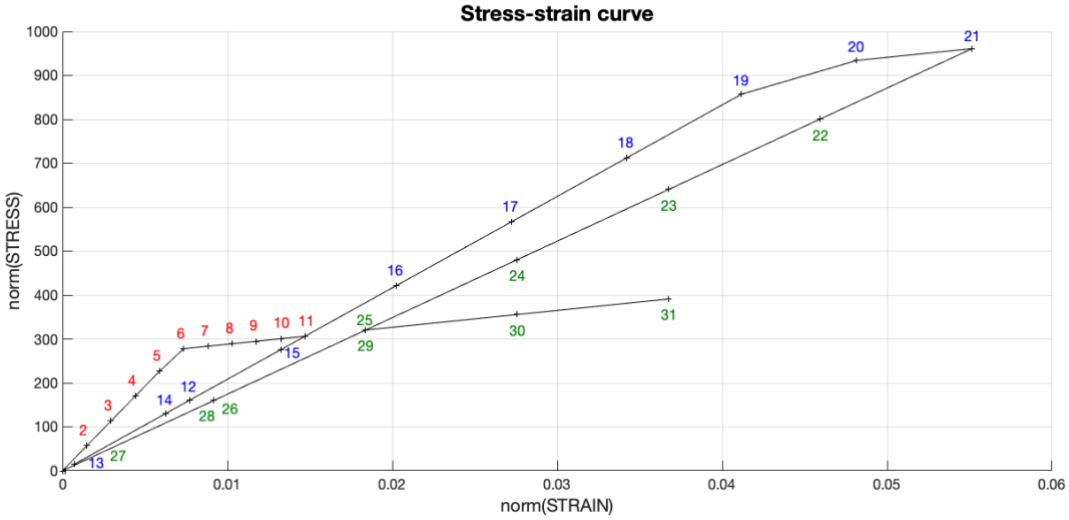

Figure 13: Stress-strain curve of the path 2 (non-symmetric model)

The stress-strain curve shows clearly the three periods when the elastic domain is exceeded and the damage parameter grows. The slopes are affected by the factor (1-d), and it is worth noting that the after every damage period, the following slope decreases. To sum up, the results of the assessment of the non-symmetric damage model correspond to the ones that the theory states.

Rate dependent model

After the implementation of the rate independent model, a further step is taken by computing a visco-damage model. In order to check the correctness of this implementation, some parameters are fixed to evaluate the evolution of different variables such as the viscosity parameter η, the strain rate $\dot{\varepsilon}$, and the α value.

Visco-damage "symmetric tension-compression model

First of all, it is worth checking how the damage surface evolves in order to ensure a proper behaviour. As the theory sustains, the stress/strain state can lay outside the elastic domain as can be seen in the following figure.

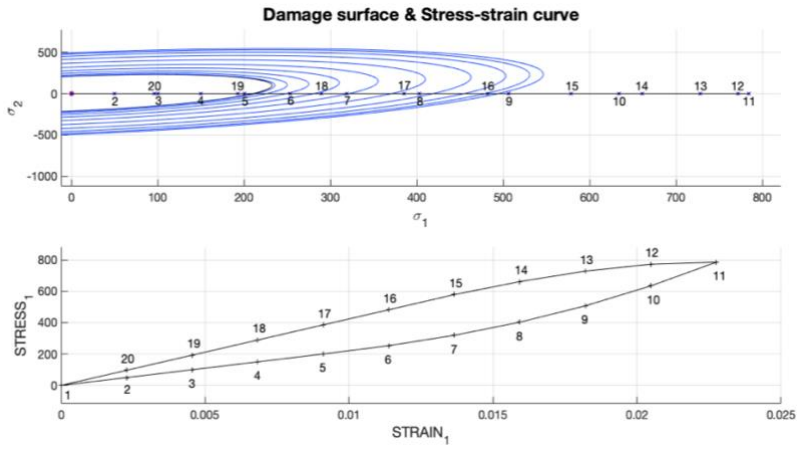

Figure 14: Damage surface & stress-strain curve of the viscous model

Another prove of this behaviour is the discharging segment (n=12 to n=20) of the stress-strain curve. Here it can be proven that this segment is not linear up to n=16 when the stress-strain state returns to the elastic domain.

In order to verify the rate dependency, the time evolution of the stress is compared with the one of the rate independent model

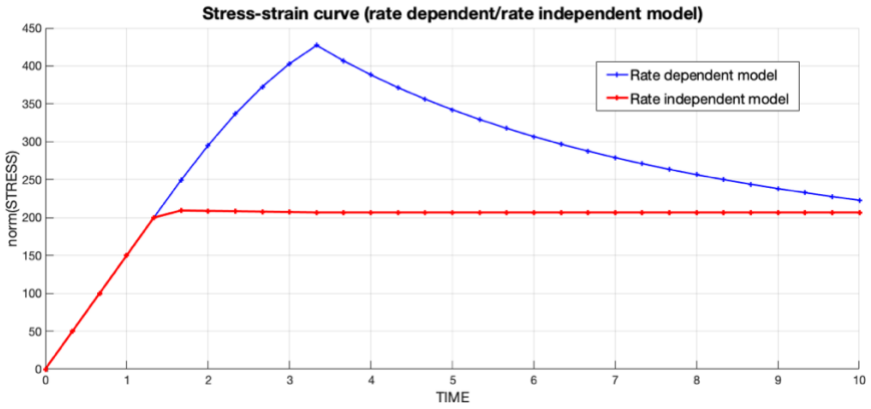

Figure 15: Stress evolution of rate dependent/independent models

The figure clearly shows that in the rate dependent model the stress tensor can change even if the strain remains constant.

Verification

Effect of different viscosity parameters η

Fixed parameters:

$\nu = 0.3$	$H = -0.02$	$\alpha = 1$	Loading path: (150,0);(300,0);(600,0)

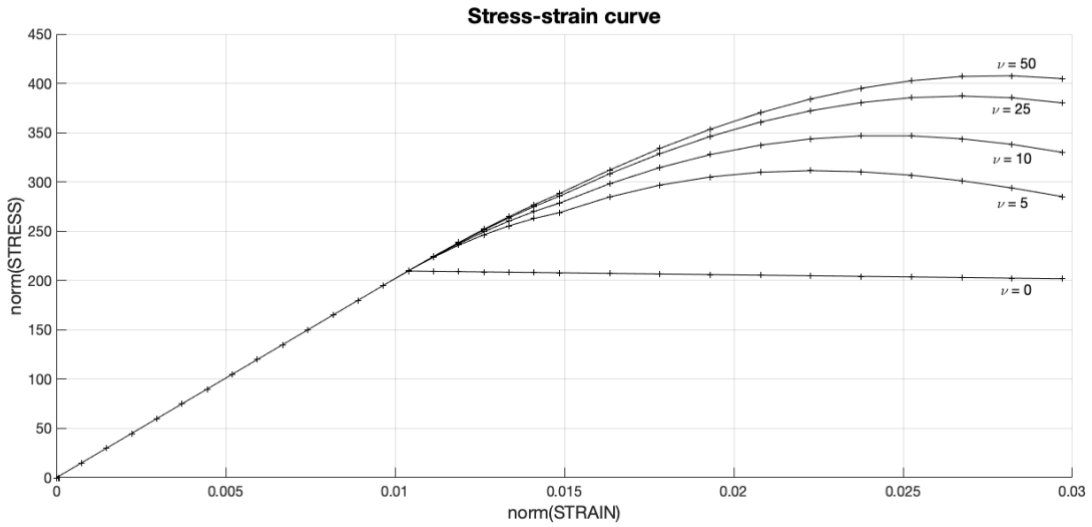

Figure 16: Stress-strain curve of the viscosity study

It is worth noting that the higher the viscosity value the smoother is the softening segment. In other words, the increasing of this parameter delays the softening and therefore, higher values of stress appear for the same values of strain. Furthermore, the results for null viscosity match with the ones obtained with the rate independent model, proving that when this parameter tends to zero the previous model is recovered.

Effect of different strain rates

Fixed parameters:

$\nu = 0.3$	$H = -0.02$	$\alpha = 1$	$\eta = 5$	Loading path1: (330,0);(660,0);(1000,0) Loading path2: (0,0);(500,0);(1000,0) Loading path3: (0,0);(0,0);(1000,0)

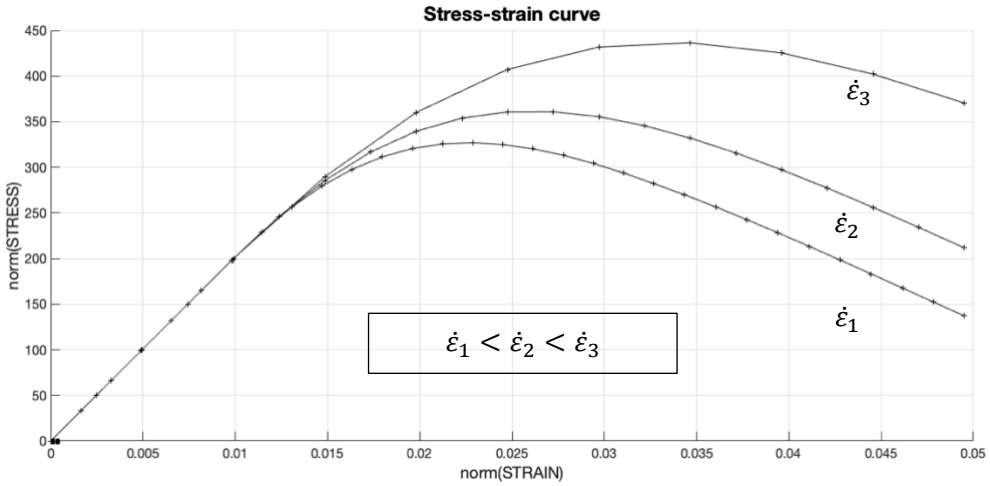

Figure 17: Stress-strain curve of the strain rate study

As can be seen, the increasing of the strain rate retards the softening of the material and provokes the appearance of higher stress values for the same strain. If the analysis is divided into the elastic and the damage region, the first part does not depend on the strain rate since there is no evolution of the internal variables. However, along the damage part, the influence of the strain rate generates larges differences on the corresponding stresses. If the figure is compared with the previous one, it could be noted a similitude between the viscosity and the strain rate influence on the stresses.

Effect of different alpha values

The following analysis tries to develop the influence of the numerical scheme on the results of the model. The purpose is to show what the theory states about the stability of the different schemes of the alpha method.

$\nu = 0.3$	$H = 0$	$\alpha = \{0, 0.25, 0.5, 0.75, 1\}$	$\eta = 0.1$	Loading path: (200,0);(400,0);(600,0)

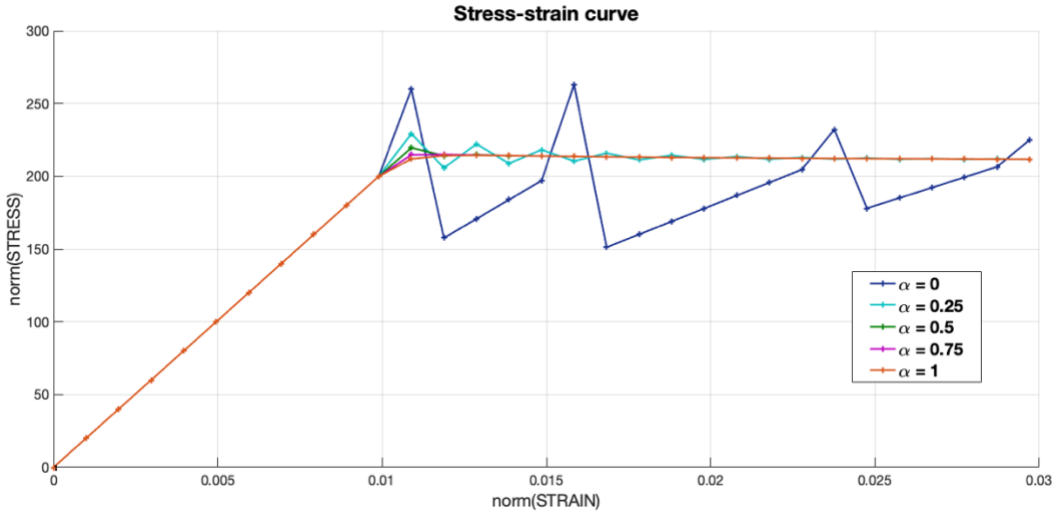

Figure 18: Stress-strain curve of the alpha method analysis

The results with the explicit method ($\alpha = 0$) and $\alpha = 0.25$ show instabilities. By making further experiments with several time steps it can be state that these methods can reach stable results with an appropriate refinement of the time discretization. From $\alpha = 0.5$ to 1 the presented results are consistent and stable, which coincides with the theory. It is worth noting that the higher the viscosity value the more stable the model turns to be, which results logical when the internal variable expression is analysed.

Effect of alpha on the evolution of the tangent and algorithmic constitutive operators

The following pictures show the time evolution of the C11 component of the tangent and algorithmic constitutive operators for different alpha values

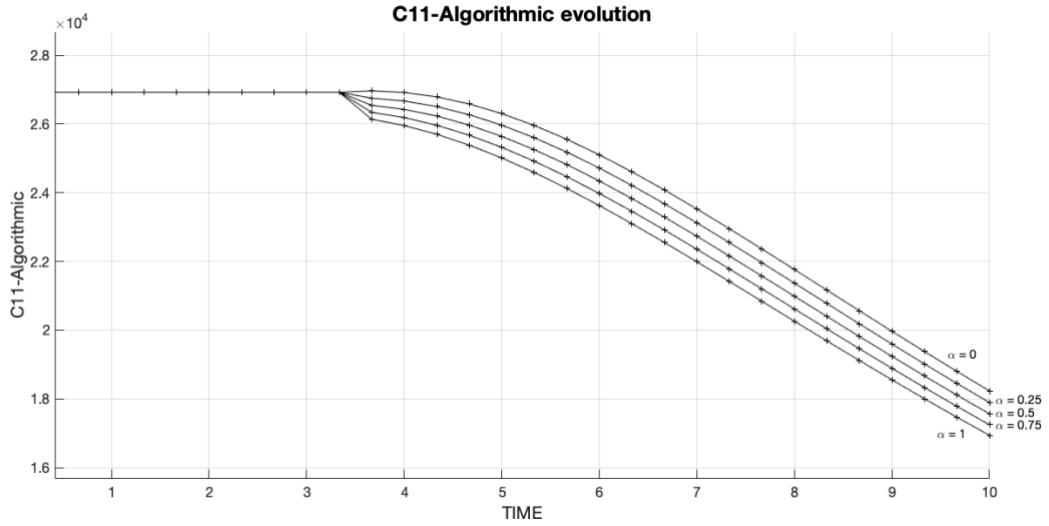

Figure 19: Time evolution of component (1,1) of the algorithmic operator

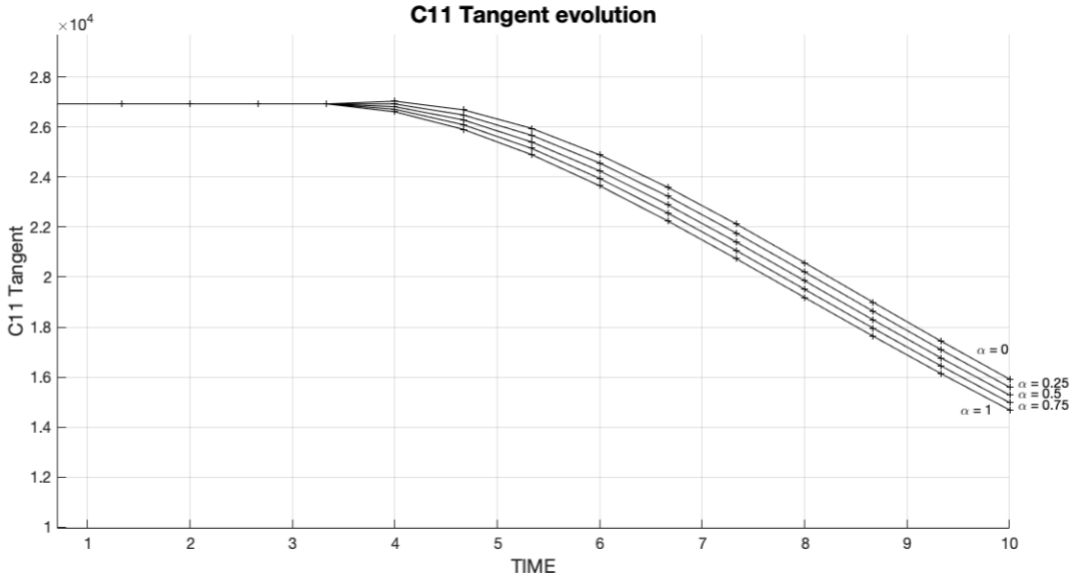

Figure 20: Time evolution of component (1,1) of the tangent operator

As can be seen, when $\alpha = 0$ the tangent and the algorithmic operators match. However, when alpha increases, the operators give values lower than the ones of the explicit case.

In order to go deeper in the analysis the time step is refined to check the response of the algorithmic operator.

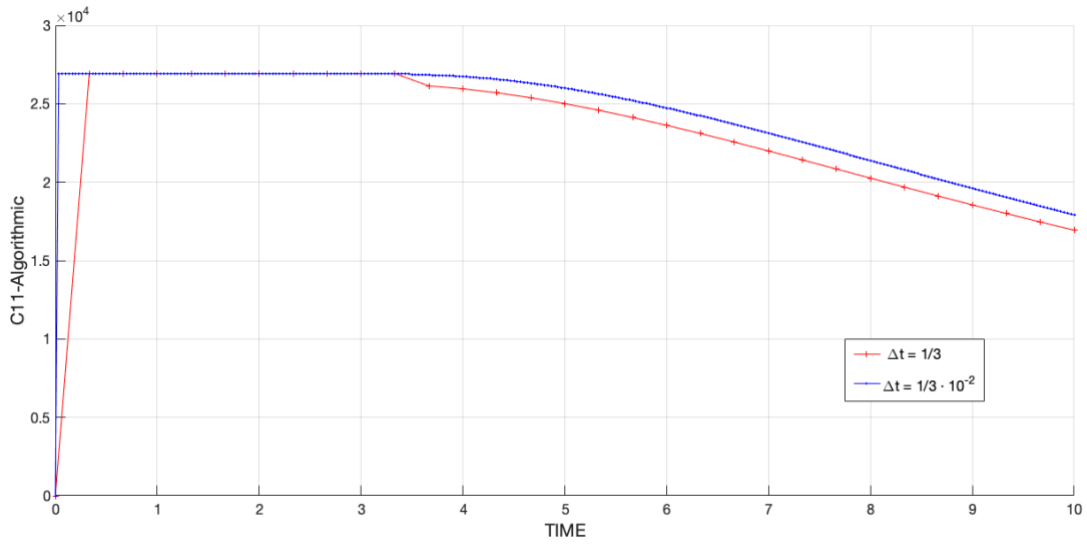

Figure 21: Time evolution of algorithmic operator for different time steps

The results clearly show that a refinement of the time discretization makes the algorithmic operator match with the analytic tangent one. In this way the consistency is checked.

Also, it is worth noting that if the viscosity parameter tends to zero the algorithmic operator ends up matching the one of the non-viscous model.

The parameters used to calculate the previous results are

$\nu = 0.3$	$H = -0.02$	$\alpha = \{0, 0.25, 0.5, 0.75, 1\}$	$\eta = 10$	Loading path: (200,0);(400,0);(600,0)

Annex

Only the modified parts of the code are reported along this part.

Damage_main.m

```
% 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)
CTANG=zeros ( totalstep +1 ,1);  %  Tangent operator
CALG=zeros ( totalstep +1 ,1);  %  Algorithmic operator

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)

        eps_n0=strain(i,:);
        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,Calg,Ctang] =
rmap_dano1(eps_n0,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),
'-',MDtype,n );
            set(hplotSURF(i),'Color',[0 0.2 1],'LineWidth',0.1)
;
        end

        CALG(i)=Calg(1,1);
        CTANG(i)=Ctang(1,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
```

```
%*****************************************************************
*
        % 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)=CTANG(i); % C11 Tangent
        vartoplot{i}(5)=CALG(i); % C11 Algorithmic
        vartoplot{i}(6)=hvar_n(6)/hvar_n(5); %  q/r
    end
end
```

rmap_dano1.m

```
function [sigma_n1,hvar_n1,aux_var,Calg,Ctang] = rmap_dano1
(eps_n0,eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t)

% Viscous parameters

viscpr  = Eprop(6);
eta     = Eprop(7);
ALPHA_COEFF = 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_n] = Modelos_de_dano1 (MDtype,ce,eps_n0,n); % Viscous model
parameter
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
[rtrial_nalpha]=(1-ALPHA_COEFF)*rtrial_n+ALPHA_COEFF*rtrial ; %
Viscous model parameter

%*****************************************************************
***************
```

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

if viscpr == 1
    if(rtrial> r_n)
        % Loading
        fload=1;
        r_n1=((eta-delta_t*(1-
ALPHA_COEFF))/(eta+ALPHA_COEFF*delta_t))*r_n+...
        (delta_t*rtrial_nalpha)/(eta+ALPHA_COEFF*delta_t); % r(n+1)
        delta_r=r_n1-r_n;

    if hard_type==0
        %Linear
        H_n1=H;
        q_n1= q_n+ H*delta_r;
    else
        %Exponential
        q_inf=r0+(r0-zero_q);
        if H>0
            H_n1 = H*((q_inf-r0)/r0)*exp(H*(1-rn1/r0));
            q_n1=q_n+((H*(q_inf-r0)/r0)*exp(H*(1- rn1/r0)))*delta_r;
        else
            H_n1 = H*((q_inf-r0)/r0)*(1/exp(H*(1-rn1/r0)));
            q_n1=q_n+((H*(q_inf-r0)/r0)*(1/exp(H*(1-
rn1/r0))))*delta_r;
        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

else
    % No viscous model

    if(rtrial > r_n)
        %*   Loading

    fload=1;
    delta_r=rtrial-r_n;
    r_n1= rtrial  ;
    if hard_type == 0
        %  Linear
        H_n1=H;
        q_n1= q_n+ H*delta_r;
```

```
    else
        % Exponential
        q_inf = r0+(r0-zero_q);

         if H>0
            H_n1 = H*((q_inf-r0)/r0)*exp(H*(1-rtrial/r0));
            q_n1= q_n+ ((H*(q_inf-r0)/r0)*exp(H*(1-
rtrial/r0)))*delta_r;
         elseif H<0
            H_n1 = H*((q_inf-r0)/r0)*(1/exp(H*(1-rtrial/r0)));
            q_n1 = q_n+ ((H*(q_inf-r0)/r0)*(1/exp(H*(1-
rtrial/r0))))*delta_r;
         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

end


% Damage variable
% ---------------
dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
sigmab_n1=ce*eps_n1';
%hold on
%plot(sigma_n1(1),sigma_n1(2),'bx')

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

% Analytic/Algorithmic tangent operator

if(rtrial>r_n)
    Ctang=(1-dano_n1)*ce;

Calg=Ctang+((ALPHA_COEFF*delta_t)/(eta+ALPHA_COEFF*delta_t))*(1/rtrial
)*...
    ((H_n1*r_n1-q_n1)/(r_n1^2))*(sigmab_n1*sigmab_n1');
else
    Calg=(1-dano_n1)*ce;
    Ctang=Calg ;
end
```

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

Dibujar_criterio_dano1

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)

%*********************************************************************
****************
% 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
    % ****************************************************
%     menu({'Damage surface "ONLY-TENSION" 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) ;

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

        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) ;

    for i=1:Contador
        sigma= [m1(i) m2(i) 0 nu*(m1(i)+m2(i))];
        sigma_pos=sigma.*(sigma>0);
        Th=sum(sigma_pos)/sum(abs(sigma));

        radio(i)= (q/sqrt(sigma*ce_inv*sigma'))/(Th+(1-Th)/n);

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

    hplot=plot(s1,s2,tipo_linea);


end
return
```

modelos_de_dano1

```
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=eps_n1*ce;
sigma_pos=sigma.*(sigma>0);

rtrial=sqrt(sigma_pos*eps_n1');

elseif (MDtype==3)  %*Non-symmetric

    sigma=eps_n1*ce;
    sigma_pos=sigma.*(sigma>0);
    sigma_abs=abs(sigma);
    theta=sum(sigma_pos)/sum(sigma_abs);

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

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