Computational Solid Mechanics

Master's Degree in Numerical Methods in Engineering

# Assignment 1: Damage model

*Authors:*
Diego Roldán

*Supervisor:*
Prof. S. Joaquín A. Hernández Ortega

Academic Year 2019-2020

## Contents

# 1 Introduction

The principal goal for this assignment is giving to the student an understanding in the algorithmic structure, especially the numerical integration, of continuum damage constitutive models. The program is focused on the local constitutive response and not on the overall structural response. The user is responsible for giving a prescribed local strain history (at a given point of the continuum). Finally, concepts, for instance, elastic domain, Kuhn-Tucker loading/unloading conditions, damage surface and so on can be assimilated and readily grasped via appropriate graphical representations.

It is provided a program for Matlab software which professor Hernández Ortega has implemented. This is partially completed, therefore, the assignment consist of completing the code and understanding how it works according to the damage model theory in order to obtain its full performance in the case of plane strain case.

To summarize what it is already implemented and not, it is presented the next list:

- Damage models:

Implemented: Symmetric

To be implemented: Tension-only, Non-symmetric

- Softening/hardening law

Implemented: Linear Law

To be implemented: Exponential Law

- Viscous/inviscid case

Implemented: Inviscid

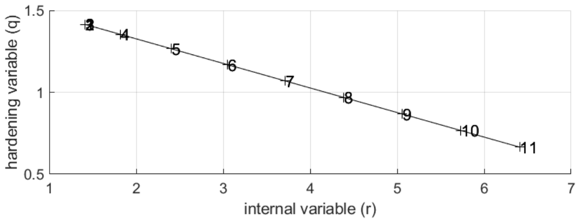
To be implemented: Viscous

At the of the report, it is annexed the code implemented for this assignment with comments in order to facilitate the readability of the code.

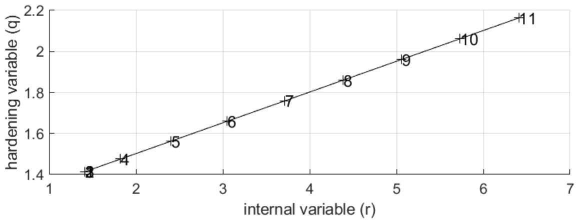
# 2 Rate Independent Models

## 2.1 Linear and exponential hardening/softening

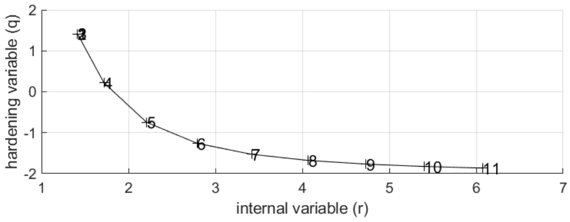
In the first part of the assignment is required to implement the code for linear and exponential hardening (H<0) and softening (H>0). In order to show the correct implementation, it is shown the four cases and its expected behaviour in table 1. To do so, it has been tested the Symmetric case and plotted the variables "hardening variable (q(r))" and 'internal variable (r)'. It has been used only one path [1000 0] with this extensive length to appreciate the exponential behaviour. It is introduced for the exponential case, A=1 and $q_{\infty} = \pm 2$. The other values for the calculation are specified in the next section.



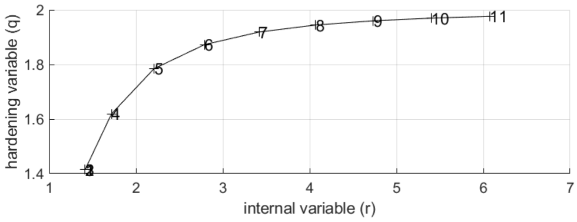

(a) Linear Softening




(b) Linear Hardening




(c) Exponential Softening




(d) Exponential Hardening

Table 1: Linear and exponential hardening/softening

As it is possible to conclude when the softening is produced (H<0), the slope is negative for both linear and exponential. In the same way, when the hardening is produced (H>0) the slope is positive for both linear and exponential. The results coincide with the expected behaviour.

## 2.2 Correctness of the implementation

The damage surface is the elastic material limit. The current stress state has to be inside or on the damage surface in accordance with the damage criteria for inviscid materials. Therefore, there is an elastic behaviour (both elastic loading and unloading) if the stress path is within the damage surface. If the stress state is situated on the surface, there is an inelastic damage state where the internal variable plays a role in its evolution. As it is shown in 12, there is an evolution of the internal variable with pure loading and there is not any increase when unloading.

In order to asses the correctness of the implementation of this program, it is tested different plane strain, rate-independent case models to be studied their behaviour. These tests consist of applying some paths for the stress space and obtaining its respective stress-strain curves for each implemented model. It is also obtained the behaviour of the damage and internal with the time in order to appreciate the correctness of the implementation. This path starts at the point $\sigma_1$ =0 and $\sigma_2$ =0 and follows by three-segments paths given in the strain space below.

1.

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

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

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

2.

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

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

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

3.

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

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

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

Where $\alpha$ is decided to be 350, $\beta$ is -1600 and finally, $\gamma$ is 1700. So, in the given code it is introduced for each one of the cases, the following stresses.

Case 1–> [350 ; 0]–> [-1250 ; 0] –> [450 ; 0]

Case 2–> [350 ; 0]–> [-1250 ; -1600] –> [450 ; 100]

Case 3–> [350 ; 350]–> [-1250 ; -1250] –> [450 ; 450]

These stress paths are chosen in order to show the features coded going thorough the defined limit cases of the corresponding problem.

The properties decided to give to the models are the following ones:

- Young Modulus, E=20000
- Poisson's ratio = 0.3
- Hardening / Softening Modulus, H=0.1
- Ratio compression/tension strength, n=3
- $q_\infty = 2$

For the Tension-only case, it is possible to notice that it behaves as the symmetric case for the positive values of both stresses as it is shown in figures 2, 3 and 6. The principal characteristic is the elasticity and even if it is applied loading or unloading, the model does not suffer any change regarding hardening or softening.

For the Non-Symmetric case, it is is possible to notice that it behaves as the symmetric case and Tension-only for the positives values of both stresses as it is shown in figure

6. In this case, there is an extended area in the pure compression zone of the model related to the ratio of tension/compression strength (n). Therefore, it is produced a softening/hardening in the compressive surface of the stress space affecting the results given by the model.

Next, it is presented the behaviour for the stress space for the specified paths for Tension-only and Non-symmetric models.

### Case 1

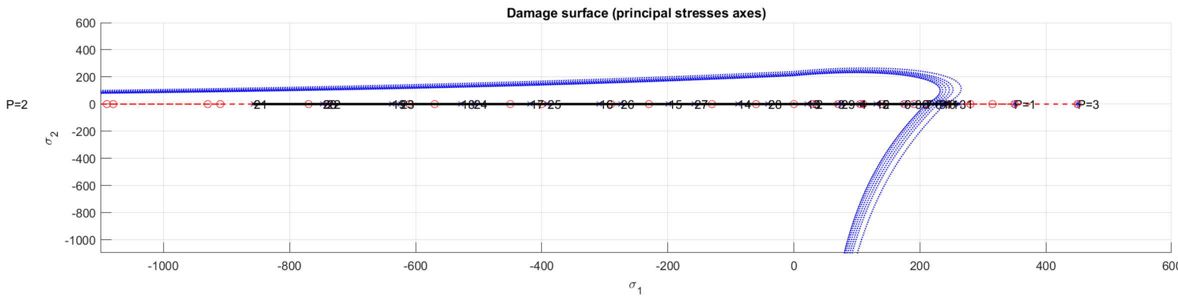



Figure 1: Case 1, Tension-only, stress space path.

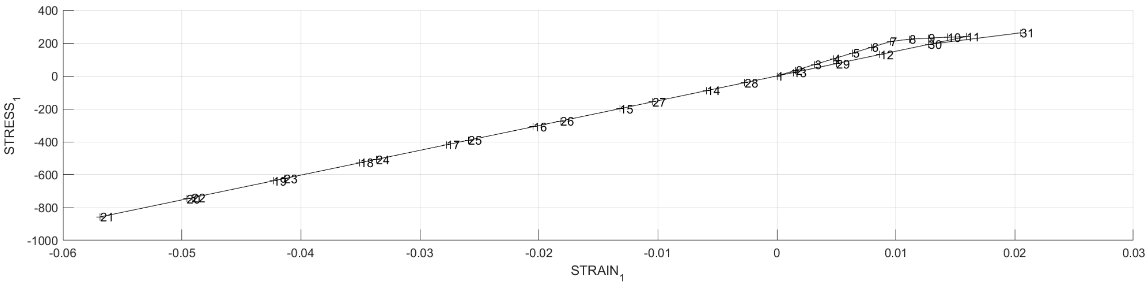



Figure 2: Case 1, Tension-only, Principal stress vs principal strain.

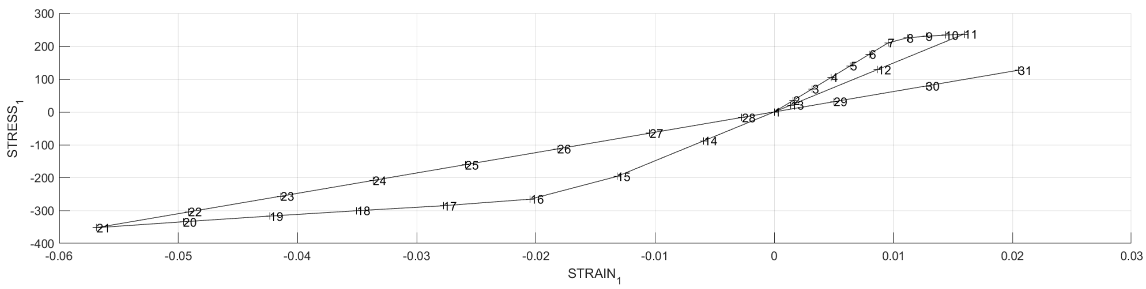



Figure 3: Case 1, Symmetric, Principal stress vs principal strain.

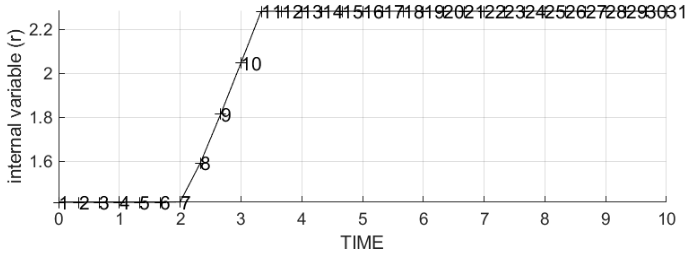



Figure 4: Case 1, Tension-only, Internal variable (r) vs time.

In figure 4, it is possible to see the evolution of the internal variable (r). It increases when the elastic regime is surpassed by the tensile loading and it remains constant due to the fact it has no constraint for compressive loading.

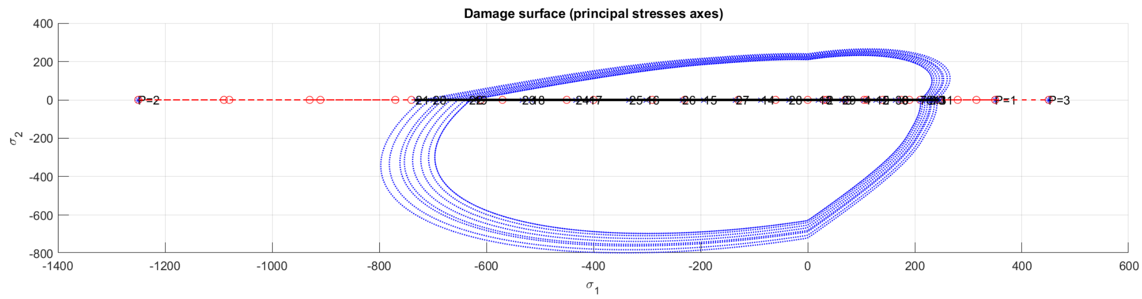



Figure 5: Case 1, Non-symmetric, stress space path.

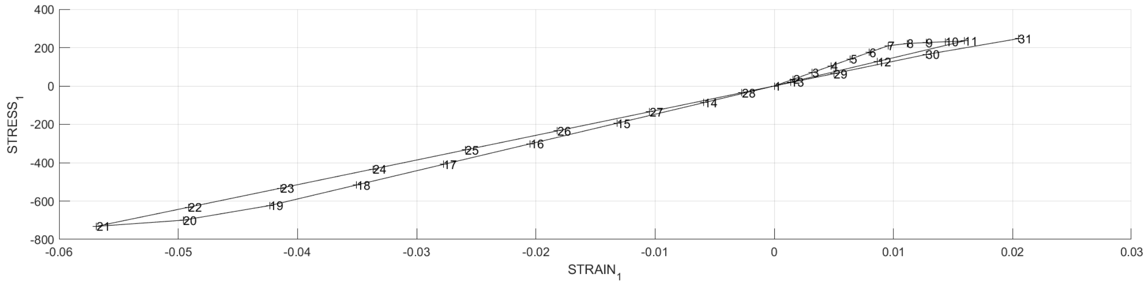



Figure 6: Case 1, Non-symmetric, Principal stress vs principal strain.

## Case 2

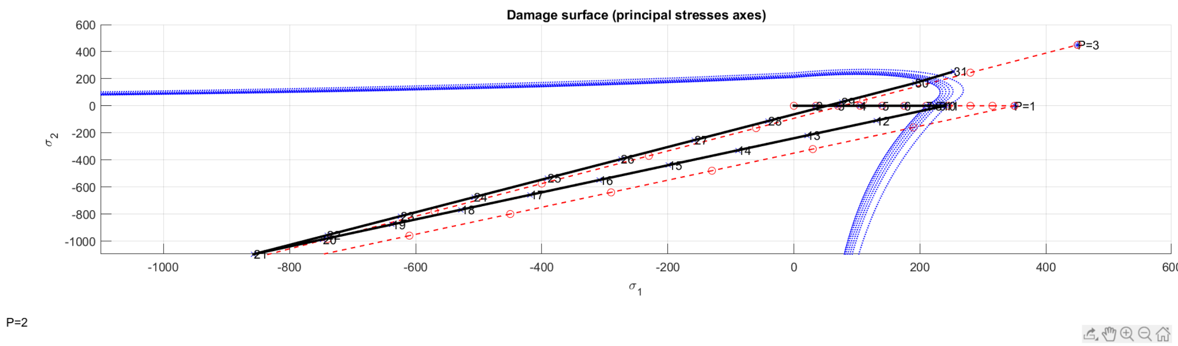



Figure 7: Case 2, Tension-only, stress space path.

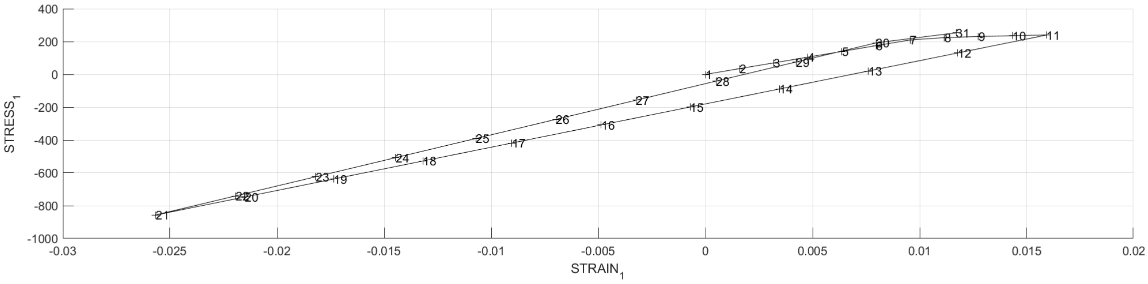


Figure 8: Case 2, Tension-only, Principal stress vs principal strain.

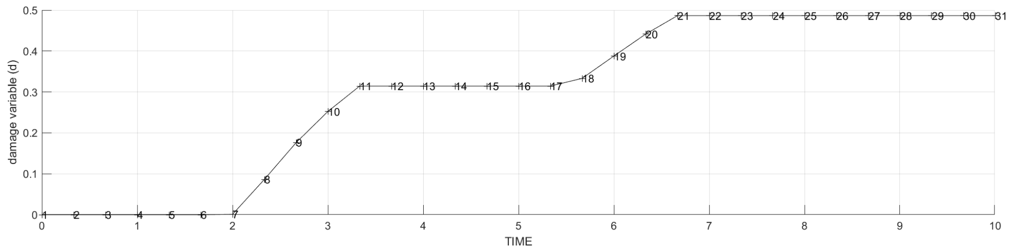


Figure 9: Case 2, Non-symmetric, Damage variable (d) vs time.

**Case 3**

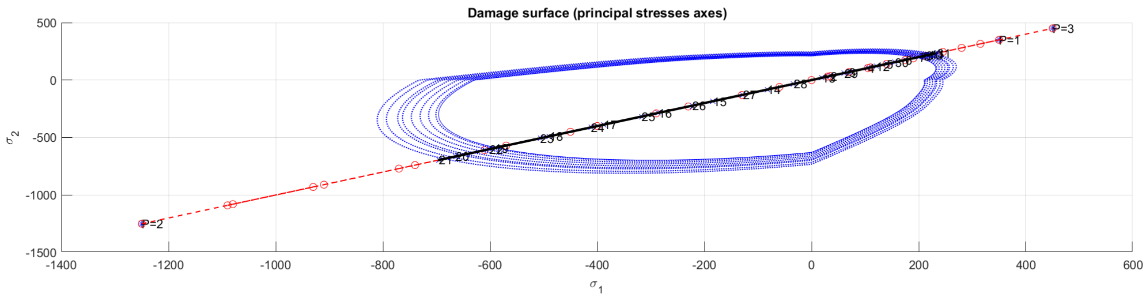



Figure 10: Case 3, Non-symmetric, stress space path.

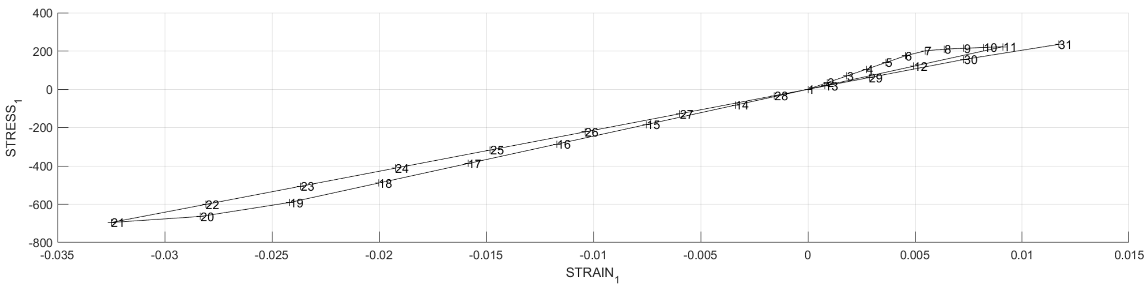


Figure 11: Case 3, Non-symmetric, Principal stress vs principal strain.

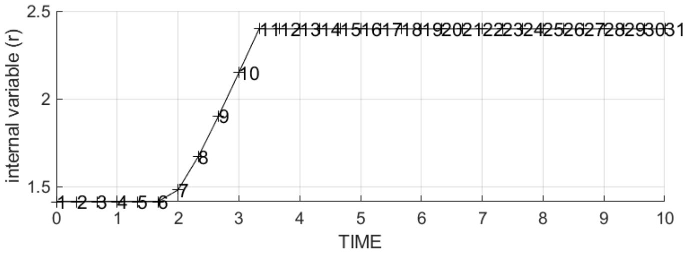



Figure 12: Case 3, Non-symmetric, Internal variable (r) vs time.

Once the graphs are obtained and analysed with the implemented code, it is possible to conclude the correct assessment of the damage model implementation because it behaves as the theory states.

# 3 Rate Dependent Models

In this section, time (t) acts as an independent variable, unlike in rate-independent model that acted as a parameter. Considering the time as an independent variable means that, although the strain is constant throughout time, the stress has not to be constant necessarily. It also provokes that the stress points can be allocated outside the elastic domain.

In order to asses the correctness of the implementation of the code, it is analysed the following cases applied in the visco-damage “symmetric tension-compression” model (for a specific given Poisson ratio and linear hardening/softening parameter given in the previous section). It is created only a one-segmented path starting from the point $\sigma_1 = 0$ and $\sigma_2 = 0$.

## 3.1 Different viscosity parameters $\eta$

It is created only a one-segmented path of [1000 ; 1000] because it is easier to observe the effect of $\eta$. It has been tested for the following values: $\eta = 0$, $\eta = 1$, $\eta = 10$, $\eta = 100$, $\eta = 1000$. All the other parameters have been fixed in order to just study the behaviour of the viscosity; $\alpha = 0.5$ and total time is 10.

It is seen that in the Stress-strain curve in figure 13, when $\eta$ increases, the curve tends to become a unique line because the curve loses the softening/hardening slope and it approaches its limit case. Moreover, it is possible to observe that the higher is the viscosity parameter, the material reaches a higher value of stress.

It is concluded that the implementation is well implemented because it behaves as expected.

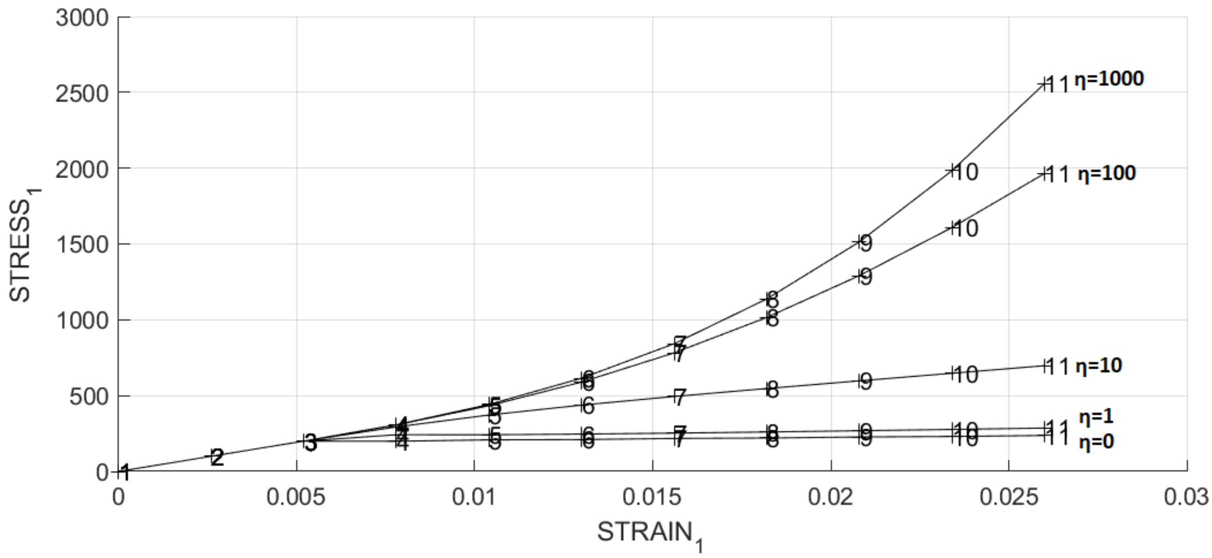



Figure 13: Influence of viscosity ($\eta$) on the stress-strain curve.

## 3.2 Different strain rate, $\dot{\epsilon}$, values

It is created only a one-segmented path of [600; 600] to observe the influence of the strain rate, $\dot{\epsilon}$, in the stress-strain curve. It has been tested a variation of the total time in order to study its behaviour; more time means minor strain rate. The values tested are t=1, t=100, t=1000, t=1000. The viscosity $\eta$ is equal to 1, and $\alpha = 0.5$.

It is observed in figure 14 when the total time increases, the stress decreases. There is a point when the total time is high enough, considered as $\dot{\epsilon} -> 0$, that the curve behaves always as a limit state of values of stress and it acts as the inviscid case.

Therefore, it is concluded that rate dependent models depend on strain rates.

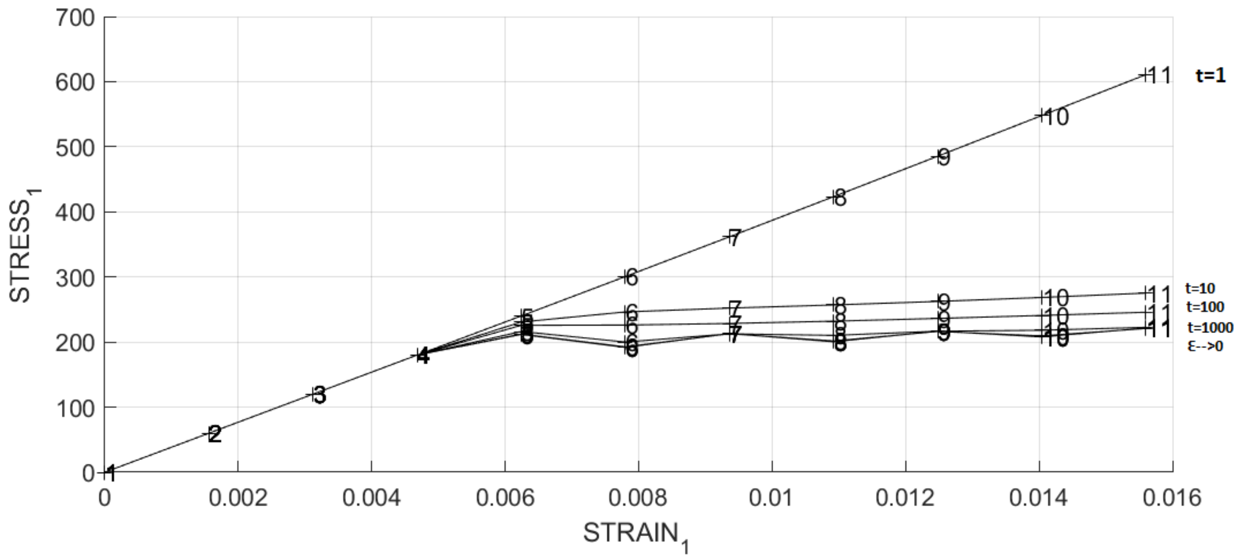



Figure 14: Influence of time ($\dot{\epsilon}$) on the stress-strain curve.

## 3.3 Different $\alpha$ values

Finally, it is tested the influence of $\alpha$ values for the stress-strain curves and created only a one-segmented path of [1000; 1000]. It has been tested different values of $\alpha$ ($\alpha = 0$, $\alpha = 0.25$, $\alpha = 0.5$, $\alpha = 0.75$, $\alpha = 1$. The viscosity $\eta$ is equal to 10, and the total time is 10.

In figure 15, it is seen the influence of alpha on the stress-strain curve. It is observed that when the value $\alpha$ is close 1, the curve is softer, unlike when it approaches 0. It is noticed that until the value of strain 0.005, the values for each one of the $\alpha$ are the same in the curve.

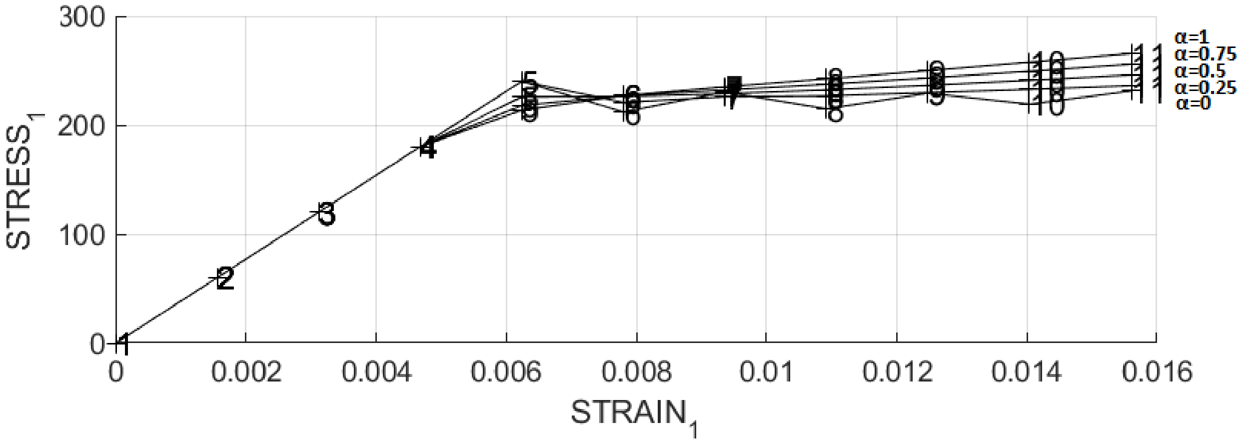
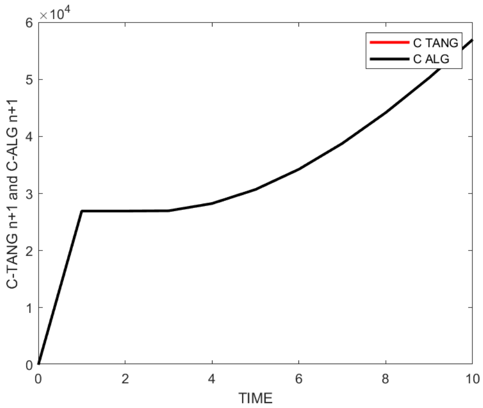


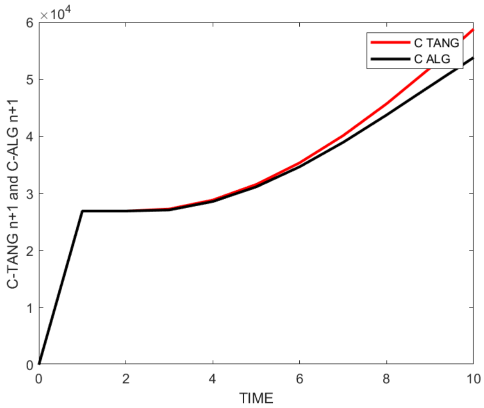

Figure 15: Influence of $\alpha$ on the stress-strain curve.

In table 2, it is presented the effects of $\alpha$ the values, on the evolution along time of the component $C_{11}$ of the tangent and algorithmic constitutive operators.

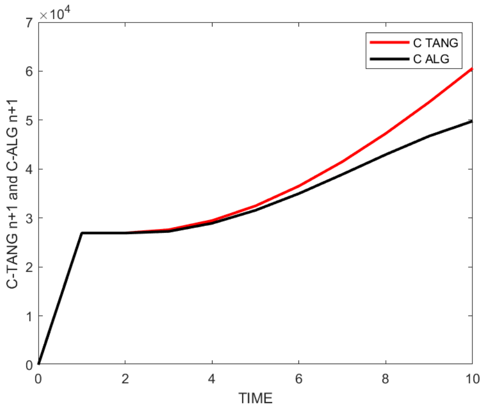
It is possible to observe that for $\alpha = 0$, both constitutive operators behave the same way, but when $\alpha$ increases, $C_{alg}$ modifies its value while $C_{tang}$ maintains almost constant. The reason for this behaviour is found in the expression of the operators, where $C_{alg}$ has an elevated dependency of the term $\alpha$.



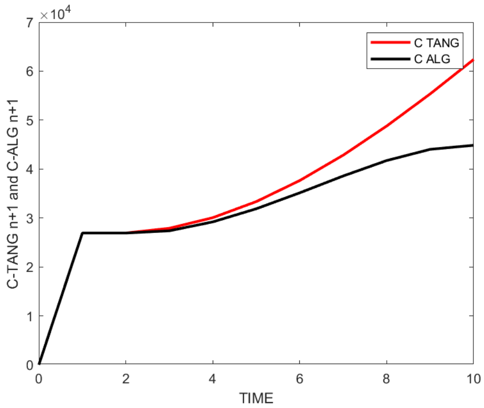

(a) $\alpha = 0$



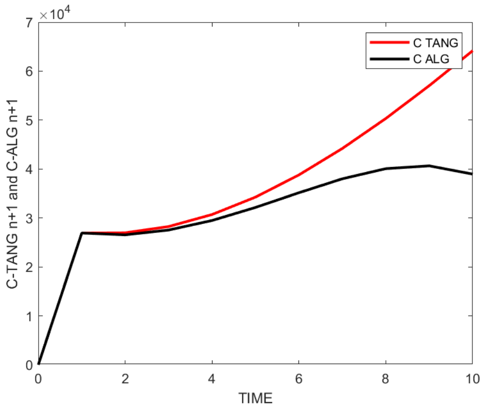

(b) $\alpha = 0.25$




(c) $\alpha = 0.5$




(d) $\alpha = 0.75$




(e) $\alpha = 1$

Table 2: The effects of $\alpha$ the values, on the evolution along time of the component $C_{11}$ of the tangent and algorithmic constitutive operators.

# 4 Conclusions

To conclude this assignment, it is possible to affirm that the codes are well implemented because all the parameters and curves behave as expected and give valid results. This code can compute Damage models as Only-Tension and Non-symmetric for viscous and exponential cases, apart from the ones already implemented.

In order to do further work for this project, it is required to code for the cases of plane stress and 3D theory.

# A Appendix

Next, it is presented the implemented code required for this assignment.

## A.1 Modelos de daño

```
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
s_n1 = ce*eps_n1';
s_n1pos = [max(s_n1(1),0) max(s_n1(2),0) max(s_n1(3),0) max(s_n1(4),0)];
rtrial =sqrt(eps_n1*s_n1pos');


elseif (MDtype==3)  %*Non-symmetric
s_n1 = ce*eps_n1';
s_n1pos = [max(s_n1(1),0) max(s_n1(2),0) max(s_n1(3),0) max(s_n1(4),0)];
abs_s_n1 = abs(s_n1);
teta = (sum(s_n1pos))/(sum(abs_s_n1));
rtrial=  sqrt(eps_n1*ce*eps_n1') * (teta + (1 - teta)/n);

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

## A.2 Dibujar criterio daño

```
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----->tensin ltima        %*
%*        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
    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([max(m1(i),0) max(m2(i),0) 0 ...
        max(nu*(m1(i)+m2(i)),0)]*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);
    axis([-1100 600 -1100 600])

elseif MDtype==3

```

```
    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
      %Compute tetha as function of stresses
      abs_s = sum(abs([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]));
      pos_s = sum([max(m1(i),0) max(m2(i),0) 0 max(nu*(m1(i)+m2(i)),0)]);
      teta = pos_s/abs_s;

        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))]')*(teta+(1-teta)/n));

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

    hplot =plot(s1,s2,tipo_linea);



end
%**************************************



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

## A.3 Damage main

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,Calg,Ctang]=...
damage_main(Eprop,ntype,istep,strain,MDtype,n,TimeTotal,q_inf)
```

```
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)
%
%
%  2) vartoplot{itime}-->
%                                  -----------------------------
%   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)

Ctang=zeros(1,length(istep)*istep(length(istep)));
Calg=zeros(1,length(istep)*istep(length(istep)));

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"
        % ------------------------
        if viscpr==1
            eps_n1=[strain(i-1,:);strain(i,:)];
        else
            eps_n1 = strain(i,:) ;
        end
        %*******************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var,Calg(i),Ctang(i)] =...
        rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,q_inf,delta_t,..
        sigma_v{i-1});

        % 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
```

## A.4 Rmap daño

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

%*********************************
%*
%*           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) ;
%*******************************************
ALPHA_COEFF=Eprop(8);

%Viscosity:
eta=Eprop(7);
viscpr=Eprop(6);

A=1; %exponential
%**********************************
```

```
%*       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
if viscpr == 0
    [rtrial]=Modelos_de_dano1(MDtype,ce,eps_n1,n);
    rtrialn1=rtrial ;

elseif viscpr == 1
    [rtrialn]=Modelos_de_dano1 (MDtype,ce,eps_n1(1,:),n);
    [rtrialn1]= Modelos_de_dano1 (MDtype,ce,eps_n1(2,:),n);
    [rtrial]=ALPHA_COEFF*rtrialn1 + (1-ALPHA_COEFF)*rtrialn ;
    eps_n1=eps_n1(2,:);

end

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


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

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

    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
```

```
        %    Exponential
        q_n1 = q_inf - (q_inf - q_n)*exp(A*(1-r_n1/r_n));
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    end

    dano_n1   = 1.d0-(q_n1/r_n1);
    sigma_n1  =(1.d0-dano_n1).*ce.*eps_n1';
    Ctang=(1-dano_n1)*ce;
    Calg=(1-dano_n1)*ce-(ALPHA_COEFF*delta_t)/(eta+ALPHA_COEFF*delta_t)*...
    (1/rtrialn1)*(q_n1-H*r_n1)/(r_n1^2)*(sigma_n1'*sigma_n1);
    Ctang=Ctang(1,1);
    Calg=Calg(1,1);

else

    %*     Elastic load/unload
    fload=0;

    r_n1= r_n  ;
    q_n1= q_n  ;
    dano_n1   = 1.d0-(q_n1/r_n1);
    Calg=(1-dano_n1)*ce;
    Ctang=Calg;
    Calg=Calg(1,1);
    Ctang=Ctang(1,1);
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;
%******************************************

```