Master's Degree Numerical Methods in Engineering

Universitat Politècnica de Catalunya
BarcelonaTech

Computational Solid Mechanics

Homework 1: Damage Models

Author:
Mariano Tomás Fernandez

Professor:
Xavier Oliver
Alfredo Huespe
Joaquín A. Hernández Ortega

March 30^{th}, 2020
Academic Year 2019-2020

Contents

Abstract

The assignment require the implementation of some material descriptions that, with some previously coded MATLAB functions, will generate a continuum damage constitutive model. These implementations included the exponential hardening/softening implementation which was done correctly, implementation and analysis of the rate independent *tension-only* and *non-symmetric* models, that were implemented and tested finding their similarities and differences, and finally a rate dependent implementation was also performed in which the influence of viscosity, strain rate and α time-integration method were analyzed finding complete accordance with the bibliography.

1 Introduction

Continuum damage mechanics has been used to model materials characterized by a loss in stiffness for increasing stresses. These models have also been used to represent materials that undergo an irreversible degradation. The idea of these models is to describe materials that with a fixed damage initiation threshold (damage process initiation) and growth (propagation) which can represent in real life the apparition of micro-defects, for example micro-pores and micro-cracks (Chaves [1]).

To materialize the continuum damage constitutive models a set of initial MATLAB functions were given by the professor to implement the missing material descriptions. The continuum damage constitutive model used focuses in *local* constitutive response, namely a point instead of a structure [2]. The improvements coded includes the implementation of an *exponential* hardening law used in both *rate dependent* and *rate independent models*, material descriptions such as *tension-only* and *non-symmetric* were implemented. A simplified scheme of these materials descriptions is presented below.

- Rate independent materials (with linear and exponential hardening)
  - Symmetric model: equal tension and compression resistance
  - Tension-Only: compression resistance equal zero
  - Non-symmetric model: tension/compression resistance is a factor different than one
- Rate dependent materials (with linear and exponential hardening)
  - Symmetric model: equal tension and compression resistance
  - Tension-Only: compression resistance equal zero
  - Non-symmetric model: tension/compression resistance is a factor different than one

In the case of hardening materials what happens is that the elastic region changes once the stress state makes the damage parameter r surpass the initial damage parameter r_0. This elastic region can expand, in the case of hardening materials, or shrink for softening materials. To implement this in a continuum model, a parameter H is used an represents the slope of the *hardening/softening* evolution. This would mean that after the damage parameter r is bigger than an initial value r_0, the elastic region can be bigger or smaller, and when unloading and reload again, the damage can appear before or after the initial value r_0. In the case of the *stress-strain constitutive modeling* implementations, some typical examples of their importance can be for *tension-only* model the case of a hanger steel bar for bridges which only has relevant stiffness when subjected to tension, and the *non-symmetric* that can represent the typical case of a concrete structure, where the tension stress is about seven to ten times smaller than the compression strength.
Finally, the *rate dependency* can be interpreted as the viscous effects that some materials may experiment or not, in the case of the concrete settlements, reaction forces and structural characteristic reactions change over time due to the viscous effect of this material. This material description may reproduce the concrete effects mentioned previously when relevant (for example to allow an engineer having a detailed settlement estimation to ensure no settlement will interfere with the normal use of a high speed trains).

2 Hardening description

In Figure 2.1 the implementation of the exponential hardening law is shown and compared to the linear hardening. As shown in the figure, both descriptions starts with the same slope H and then the exponential description varies smoothly from q_{r0} to q_{∞}. The linear hardening develops higher variations of the hardening variable Δq for the same variation of the internal variable Δr.
It is important to point out that the implemented *exponential* hardening law is bounded both up and down with q_{∞}, and this is not the case of the linear description which is monotonically increasing. The decreasing case, both q, linear or exponential, are bound with q_0 that is defined as 10^{-6} r_0. In Appendix A.1 the code used for implementing this behaviour is shown.

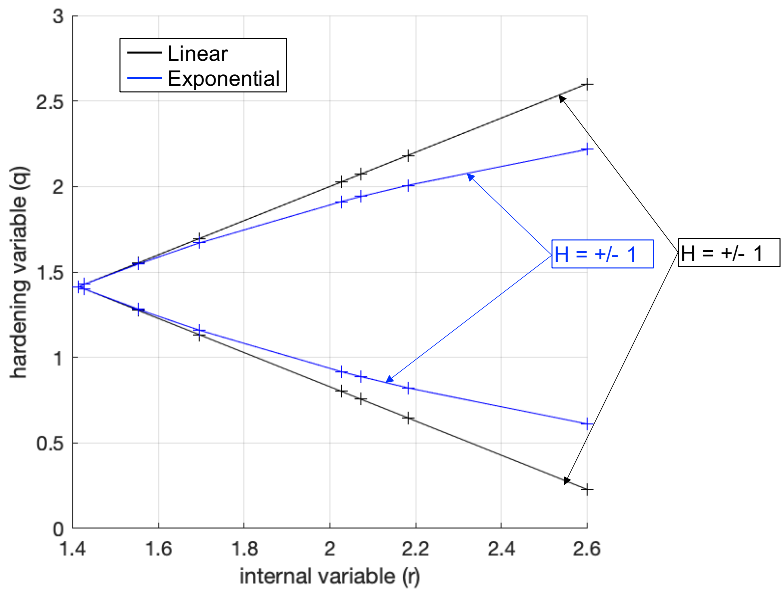


Figure 2.1: Hardening variable q variation in terms of internal variable r for values of slope $H = \pm 1$ using linear and exponential descriptions.

3 Rate independent models

The idea in this section is to assess the correctness of the implementations coded (*tension-only* and *non-symmetric* models) by analyzing different stress paths as *uniaxial tensile loading/unloading/compression* and *biaxial tensile loading/unloading/compression*. The material parameters adopted are shown in Table 3.1, where the hardening law as adopted using exponential description. The coded functions using MATLAB are shown in Appendix **??**

Property, symbol	Units	Value
Young's Modulus, E	Pa	20000
Poisson ratio, ν	-	0.3
Yield stress, σ_Y	Pa	200
Hardening parameter, H	-	-0.5
Compression/tensile ratio, n	-	2

Table 3.1: Material properties.

3.1 First loading trial

In Table 3.2 the first loading path trial is described with the initially unloaded material ($\sigma_{01} = \sigma_{02} = 0$) applying uniaxial loading both in tension and compression.

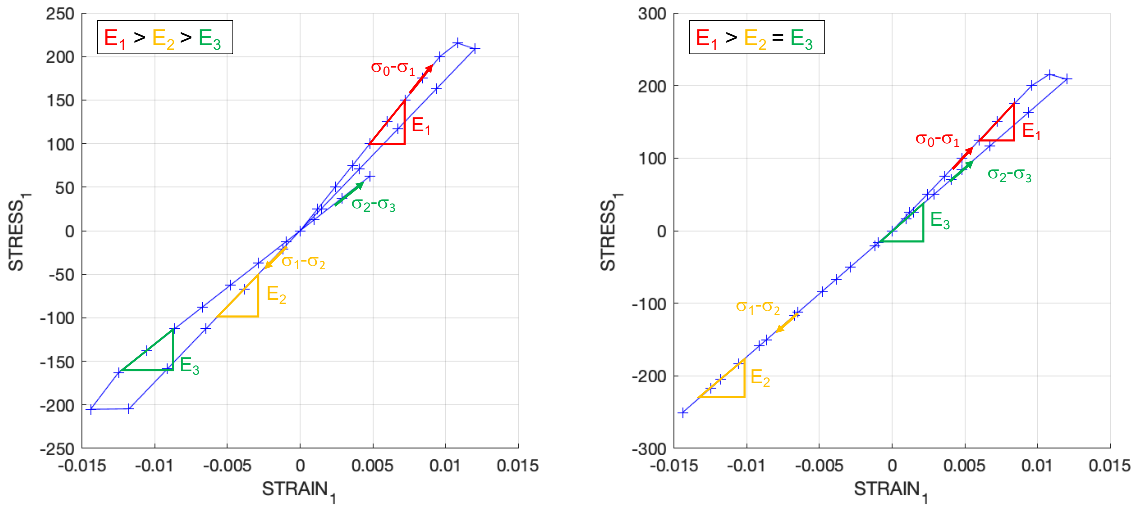
- Symmetric model: The first load ($\Delta\sigma_{1_1} = \alpha$) is bigger than the yield stress of the material in the tension direction, therefore the material registers an elastic loading (E_1) from the unloaded position to the Yield stress and then damage occurs until reaching the first loading stress state σ_{1_1}, the point $P1$ is outside the elastic domain in Figure 3.3a. The second load ($\sum\sigma_{1_2}$) is bigger than the yield stress in the compression side (see $P2$ in Figure 3.3a), therefore after unloading from σ_1 with the degraded Young's modulus resulting from the first loading step (E_2) the material experiments damage once surpassing the Yield stress in the compression side, and the Young's modulus degrades again up to E_3. Finally, the third load applied ($\sum\sigma_{1_3}$) is below the yield stress σ_Y, and the material moves from σ_2 to σ_3 with the last degraded Young's Modulus (E_3). These effects are observed and confirmed in Figure 3.1a and Figure 3.3a.

- Tension-only model: The first load ($\Delta\sigma_{1_1} = \alpha$) goes from the unloaded material to the yield stress (σ_{1_1}) with the undamaged Young's Modulus (E_1), when the stress is bigger damage is produced (see $P1$ outside elastic domain Figure 3.3b). The second load ($\sum\sigma_{1_2}$) behaves elastic as no negative stress is preserved with the degraded Young's modulus (E_2). Finally, the third loading step applied ($\sum\sigma_{1_3}$) is reached with the same Young's Modulus as the material did not registered any further degradation in the compression side. These loading paths in terms of stress-strain curves are presented in Figure 3.1b and Figure 3.3b.

- Non-symmetric model: This material description under the conditions being tested will have a very similar *stress-strain* description to the *tension-only* model. This is because as the compression part has $n = 2$ times the tension yield stress, the only load generating damage would be in the first step after surpassing the tension yield stress. The loading paths in terms of stress-strain curves are presented in Figure 3.2 and Figure 3.3c.

Stress applied		Value (σ_1)	Value (σ_2)	$\sum\sigma_{1_i}$	$\sum\sigma_{2_i}$
$\Delta\sigma_{1_i}$	$\Delta\sigma_{2_i}$	[Pa]	[Pa]	[Pa]	[Pa]
σ_{i1}	σ_{i2}	0	0	0	0
α	0	250	0	250	0
β	0	-550	0	-300	0
γ	0	400	0	100	0

Table 3.2: First loading trial path.



(a) Stress-strain loading path using symmetric model (b) Stress-strain loading path using tension-only model

Figure 3.1: First loading-unloading uniaxial path using symmetric and tension only models.

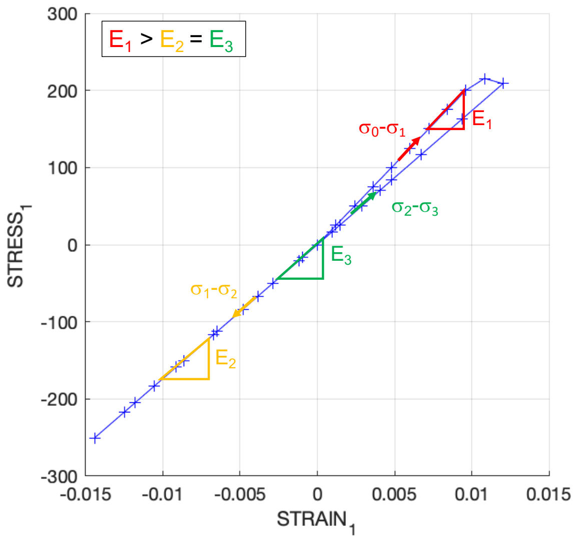
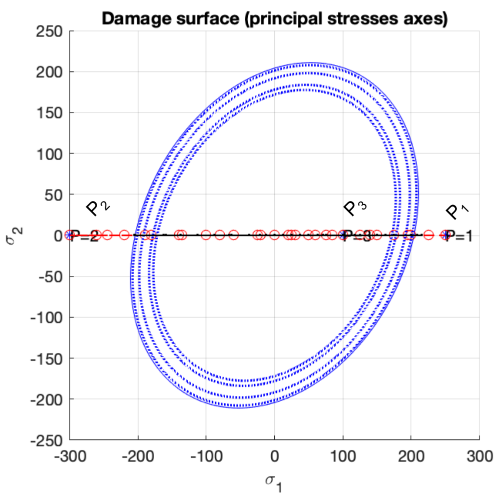
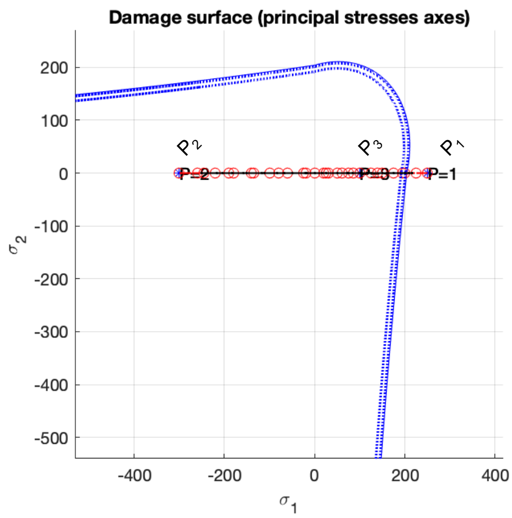


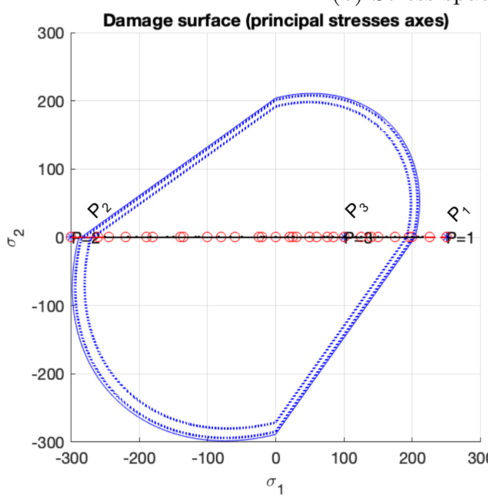
Figure 3.2: First loading-unloading uniaxial path using non-symmetric model. Stress-strain loading path.



(a) Stress space using symmetric model



(b) Stress space using tension-only model



(c) Stress space using non-symmetric model

Figure 3.3: First loading-unloading uniaxial path using symmetric, tension only and non-symmetric models.

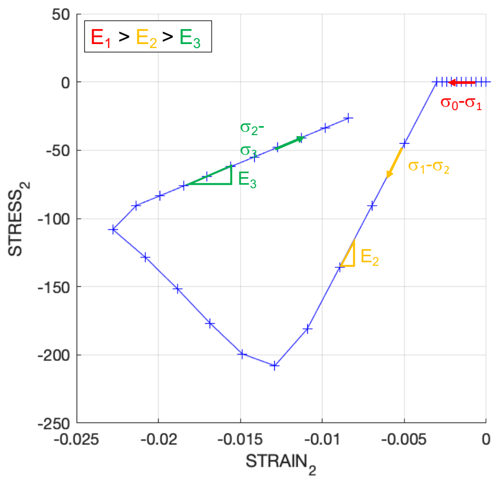
3.2 Second loading trial

In Table 3.3 the second loading path trial is described with the initially unloaded material ($\sigma_{01} = \sigma_{02} = 0$) applying uniaxial first and then biaxial stress loading states in tension and compression. In this case, to cover a wider range of stress-strain behaviours, in Figures 3.4a, 3.4b and 3.5 the plot relating stress and strains is shown in terms of σ_2 and ε_2.

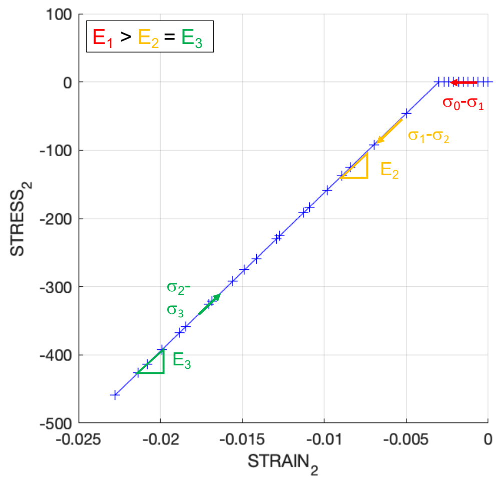
- Symmetric model: The first load ($\Delta\sigma_{1_1} = \alpha$) is bigger than the yield stress of the material in the tension direction (in terms of σ_1, that is the reason why this is not represented in Figure 3.4a) and the material registers damage until reaching the first loading stress state σ_1 (out of the elastic domain in Figure 3.6a). The second load ($\sum\sigma_{2_2}$) is bigger than the yield stress in the compression side, in terms of σ_{2_2}, therefore after unloading from $\sigma_{2_1} = 0$ with the degraded Young's modulus resulting from the first loading step (E_2) the material experiments damage once surpassing the Yield stress in the compression side, and the Young's modulus degrades again up to E_3, also the effect of damage combined with *softening* is plays an important roll in the stress decay at increasing strains before loading with E_3 slope. Finally, the third load applied ($\sum\sigma_{2_3}$) is below the yield stress σ_Y, and the material moves from σ_2 to σ_3 with the last degraded Young's Modulus (E_3). These effects are observed and confirmed in Figure 3.4a and Figure 3.6a.
- Tension-only model: The first load ($\Delta\sigma_{1_1} = \alpha$) goes from the unloaded material to the yield stress (σ_1) with the undamaged Young's Modulus (E_1), when the stress is bigger damage is produced. The second load ($\sum\sigma_{2_2}$) behaves elastic as no negative stress is preserved with the degraded Young's modulus (E_2). Finally, the third loading step applied ($\sum\sigma_{1_3}$) is reached with the same Young's Modulus as the material did not registered any further degradation in the compression side. These loading paths in terms of stress-strain curves are presented in Figure 3.4b and the stress space is shown in Figure 3.6b.
- Non-symmetric model: This material description under the conditions being tested registers damage in when moving from stress state *zero* to *one* (bigger than σ_Y in tension) and when reaching stress state *two* as this stress state in terms of σ_{2_2} is higher than $n \cdot \sigma_Y = 2\sigma_Y$. Therefore, three Young's modulus are registered in this case with E_1 higher than E_2 higher than E_3. The loading paths in terms of stress-strain curves are presented in Figure 3.5 and the stress space is shown in 3.6c.

Stress applied		Value (σ_1)	Value (σ_2)	$\sum\sigma_{1_i}$	$\sum\sigma_{2_i}$
$\Delta\sigma_i$	$\Delta\sigma_i$	[Pa]	[Pa]	[Pa]	[Pa]
σ_{i1}	σ_{i2}	0	0	0	0
α	0	250	0	250	0
$-\beta$	$-\beta$	-550	-550	-300	-550
γ	γ	400	400	100	-150

Table 3.3: Second loading trial path.



(a) Stress-strain loading path using symmetric model



(b) Stress-strain loading path using tension-only model

Figure 3.4: Second loading-unloading biaxial path using symmetric and tension only models.

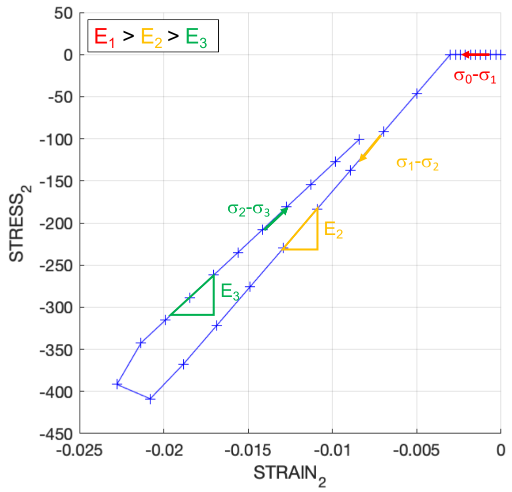
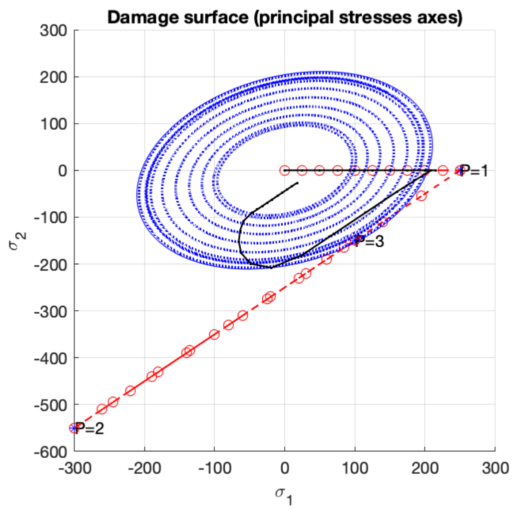
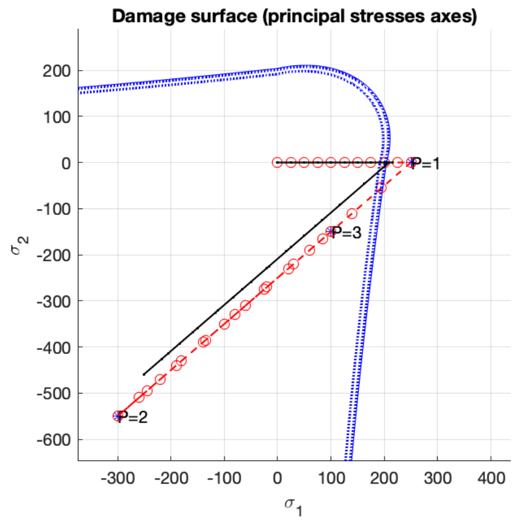


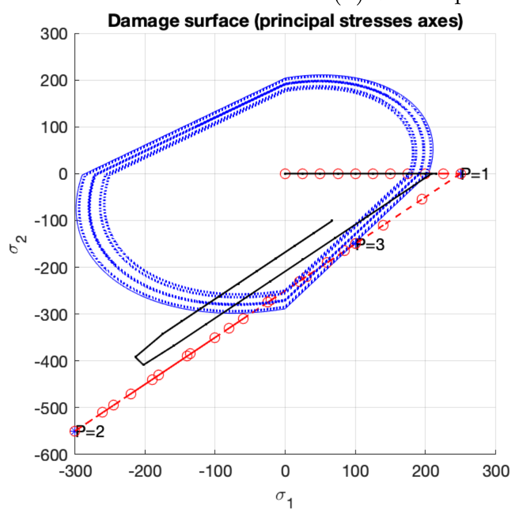
Figure 3.5: Second loading-unloading biaxial path using non-symmetric model. Stress-strain loading path.



(a) Stress space using symmetric model



(b) Stress space using tension-only model



(c) Stress space using non-symmetric model

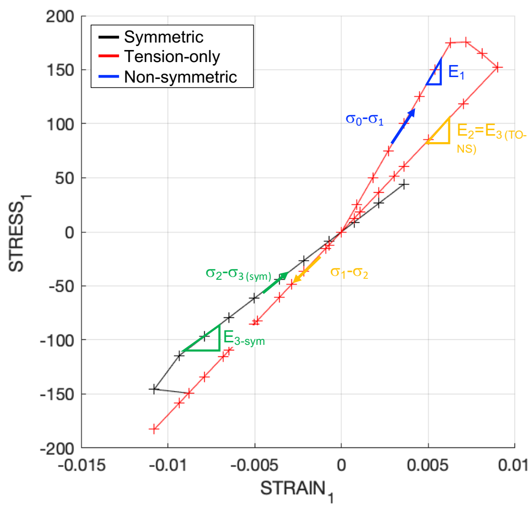
Figure 3.6: Second loading-unloading biaxial path using symmetric, tension only and non-symmetric models.

3.3 Third loading trial

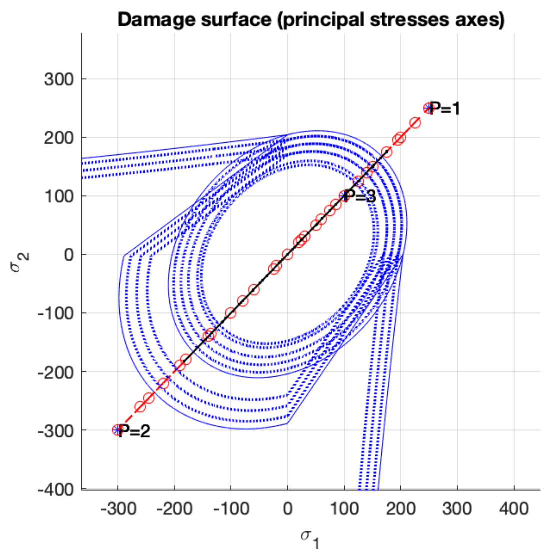
In Table 3.4 the third loading path trial is described with the initially unloaded material ($\sigma_{01} = \sigma_{02} = 0$) applying biaxial stress loading states in tension and compression. This loading case is similar to the tested in the previous section 3.2, therefore the stress-strain plots will be in only one figure, and another variables will be shown and discuss.
From Figure 3.7a it is clear that the only difference between the models is registered in the symmetric model in compression, as it registers damage. Then, tension-only and non-symmetric models follow equal stress-strain paths, that is why the non-symmetric curve cannot be seen in the plot. Therefore, the Young's modulus in the models are $E_1 > E_2 > E_{3_{sym}}$ and $E_1 > E_2 = E_{3_{OT-NS}}$ (where OT is only-tension and NS is non-symmetric).
In Figure 3.7b the plot shows the stress space $\sigma_1 - \sigma_2$ and the loading path. The loading path represents an hidrostatic loading $\sigma_i = \sigma_j$, and in all the cases the stress in three is inside the elastic domain. The trends denoted for each of the material models can be confirmed by looking at Figure **??** where again the tension-only and non-symmetric models coincide and the symmetric model registers damage in two stages: first in tension then in compression.

Stress applied		Value (σ_1)	Value (σ_2)	$\sum \sigma_{1_i}$	$\sum \sigma_{2_i}$
$\Delta\sigma_i$	$\Delta\sigma_i$	[Pa]	[Pa]	[Pa]	[Pa]
σ_{i1}	σ_{i2}	0	0	0	0
α	α	250	250	250	250
$-\beta$	$-\beta$	-550	-550	-300	-300
γ	γ	400	400	100	100

Table 3.4: Third loading trial path.



(a) Stress-strain loading path



(b) Stress path σ_1-σ_2

Figure 3.7: Third loading-unloading biaxial path using symmetric, tension-only and non-symmetric model.

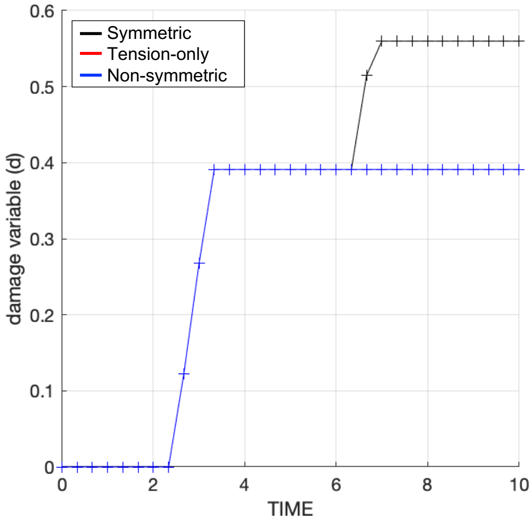


Figure 3.8: Damage variable against time evolution using symmetric, tension-only and non-symmetric model.

4 Rate dependent models

The idea in this section is to analyze the performance of the rate dependent continuum damage model, considering the symmetric tension-compression model, using a constant Poisson ratio and a linear hardening/softening material. The rate dependent model, introduce what is known as viscous damage, which leads to the introduction of damage not only with stress but also with time. To assess these effects a some parameters will be tested:

- Different viscosity values, ν;
- Different strain ratios, $\dot{\varepsilon}$;
- Different α values: $\alpha = 0$, $\alpha = 1/4$, $\alpha = 1/2$, $\alpha = 3/4$ and $\alpha = 1$ (for the α time-integration method).

By analyzing these parameters obtain results showing:

1. The effects of the previous values on the obtained stress-strain curves in appropriate loading paths.
2. The effects of the α values, on the evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators.

The material properties used to evaluate these effects are shown in Table 4.1, with linear softening law. The implemented codes can be found in Appendix A.3.

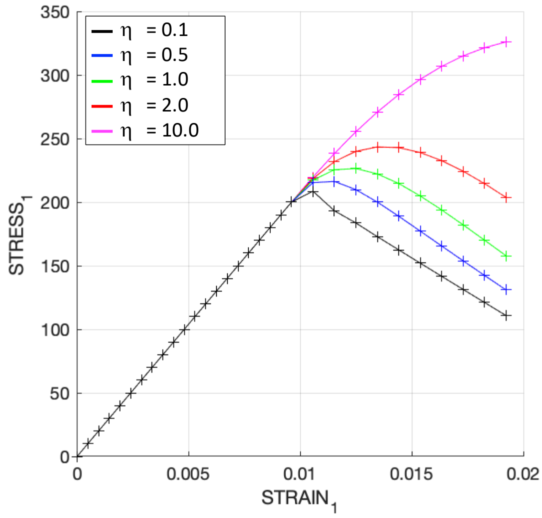
Property, symbol	Units	Value
Young's Modulus, E	Pa	20000
Poisson ratio, ν	-	0.3
Yield stress, σ_Y	Pa	200
Hardening parameter, H	-	-0.5

Table 4.1: Material properties.

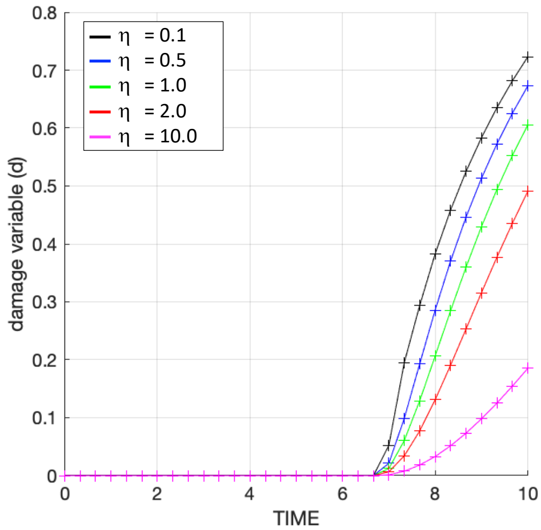
4.1 Viscous coefficient effect

To assess the viscous effect over the development of stresses-strains in the model, a set of five different values for ν are used, $\nu = 0.1,\ 0.5,\ 1.0,\ 2.0,\ 10.0$. The total time for the evaluation of the problem is $t = 10$, α for the time integration-method is set equal to $\alpha = 0.5$ (Crank-Nicholson method) and loading path is $\underline{\sigma}^{(1)} = [100,\ 0]$, $\underline{\sigma}^{(2)} = [200,\ 0]$ and $\underline{\sigma}^{(1)} = [400,\ 0]$.
Results show that for higher viscous values, the damage develops in a slower manner. That is to say, that for the same total time (in this case $t = 10$) lower damage is developed or a larger strain is required to register the same

strain. The obtained results are shown in Figure 4.1a and Figure 4.1b where the stress and damage described are observed.



(a) Stress-strain relationship



(b) Damage variables

Figure 4.1: Response of the material using different viscous parameters (η).

4.2 Strain rate effect

To analyze the effect of the strain rate in the development of stress-strain curves a set of five different values of $\dot{\varepsilon}$ are used, $\dot{\varepsilon} = 5 \cdot 10^{-3},\ 1 \cdot 10^{-2},\ 1,\ 10,\ 100$. The viscous factor for the evaluation of the problem is $\nu = 1.0$, α for the time integration-method is set equal to $\alpha = 0.5$ (Crank-Nicholson method) and loading path is $\underline{\sigma}^{(1)} = [100,\ 0]$, $\underline{\sigma}^{(2)} = [200,\ 0]$ and $\underline{\sigma}^{(1)} = [400,\ 0]$.
Results show as expected [3] that higher values of strain rate generate a stiffer less damaged model as can be observed in Figure 4.2.

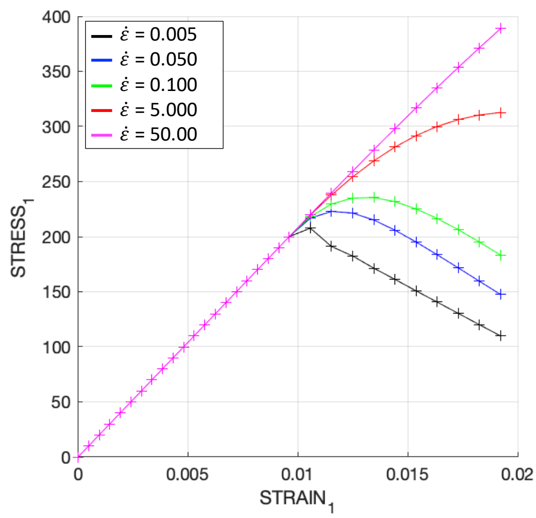
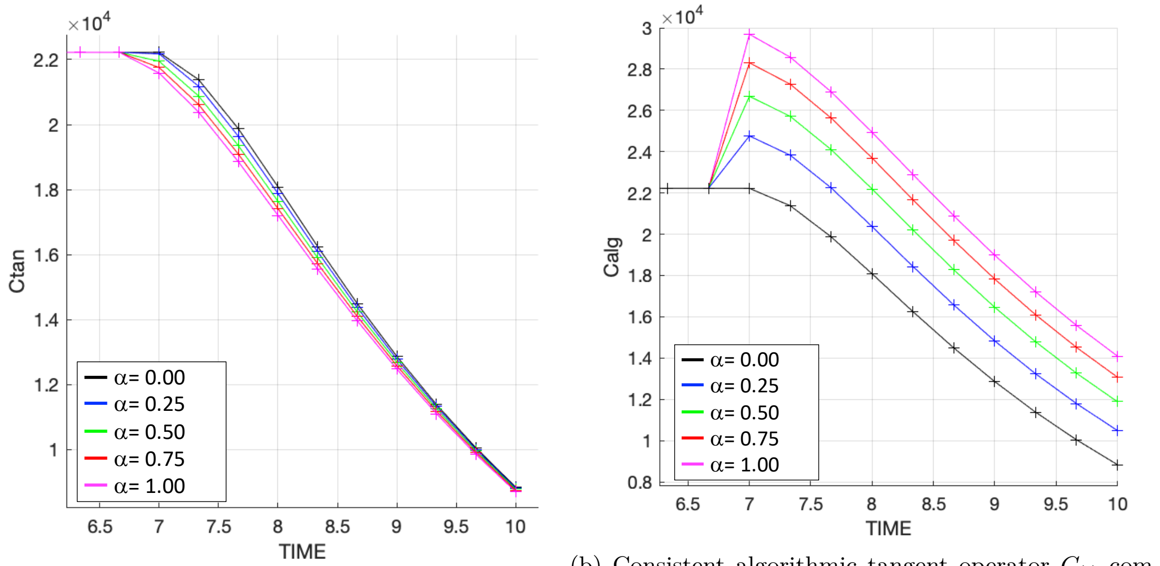


Figure 4.2: Assess of the influence of strain rate parameter ($\dot{\varepsilon}$) varying from 0.005 to 50.

4.3 Alpha time-integration method effect

To assess the importance of the time-integration method five different α values were analyzed, $\alpha = 0,\ 0.25,\ 0.5,\ 0.75,\ 1$. The viscous factor for the evaluation of the problem is $\nu = 1.0$, the total time is set to $t = 10$ ($\dot{\varepsilon}$ approximately 0.05) and loading path is again $\underline{\sigma}^{(1)} = [100,\ 0]$, $\underline{\sigma}^{(2)} = [200,\ 0]$ and $\underline{\sigma}^{(1)} = [400,\ 0]$.

This parameter is of keen importance to ensure the stability of the time integration scheme. When $\alpha = 0.5$ or $\alpha = 1.0$ the method is unconditionally stable (this method is known as Crank-Nicholson). On the other hand when $\alpha = 0.0$ the method is conditionally stable, when this stability conditions are reached the convergence will depend on the relationship between the strain rate $\dot{\varepsilon}$ and the α coefficient.
The results analyzing the C_{11} component of the tangent operator and consistent tangent operator shows that a non stable scheme can introduce a variability in the stiffness of approximately 40% for the consistent algorithmic tangent operator (and it is importance to consider that this number depends on the α and the $\dot{\varepsilon}$ used for this check). Therefore, it is very important to choose an unconditionally stable scheme as $\alpha = 1,\ \alpha = 0.5$.



(a) Tangent operator C_{11} component

(b) Consistent algorithmic tangent operator C_{11} component

Figure 4.3: Variation of a given component of the tangent operator

5 Conclusions

In the development of the continuum damage mechanics model some of the main findings of this area were analyzed and proved with the implemented code. Some of the most important implementations were:

- exponential hardening law was implemented which was proved to be a smoother transition for hardening/-softening materials bounded up and down by $\pm q_\infty$;
- for rate independent models three materials descriptions were used for three different loading cases proving their features, checking the initiation of damage, the evolution of stress-strain when damage occurs, also their stress paths were analyzed to see were these loading cases were held (inside or outside the elastic domains), and in all of the cases the code behaved as expected;
- for the rate dependent models, the symmetry model was used as an example and these viscous models were analyzed by changing the viscosity parameter, were the model shown a *stiffer* behaviour for higher viscosity parameters. The strain rate application was also verified and it was found that higher strain rates represents lower damages on the material. Finally, the time integration method was studied and it was checked that for the strain rate and viscosity parameter chosen, the material response can be importantly modified with differences in the consistent algorithmic tangent operator up to 40%.

References

[1] E. W. V. Chaves, *Notes on Continuum Mechanics*. Springer, 2013.

[2] J. H. Ortega, "Numerical integration of constitutive damage models using matlab," *Computational Solid Mechanics - Universitat Politecnica de Catalunya*, 2011.

[3] D. O. Eduardo De Souza Neto, Djordje Peric, *Computational Methods for Plasticity. Theory and Applications.* Wiley, 2008.

A Implementation codes

Even though some other changes were also made, as for example in the function *plotcurvesNEW.m* its inclusion does not alter the essence of the code and would just add irrelevant information.

A.1 rmap_dano1.m function

```
function [sigma_n1,hvar_n1,aux_var,C_tan,C_alg] = rmap_dano1 (eps_n,eps_n1,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) ;
visc = Eprop(6);

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




%*****************************************************************************
%*       Damage surface                                                    %*
alpha = Eprop(8);
eta = Eprop(7);
[tau_n] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
[tau_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);

[rtrial] = (1-alpha)*tau_n + alpha*tau_n1;
delta_tt = delta_t(1);

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


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

fload=0;


if(rtrial > r_n)          % verifies if the load is bigger than the damage
    fload=1;              % previous value of r
```

```
    r_n1 = ((eta - delta_tt*(1-alpha))*r_n + delta_tt*rtrial)...
        *(1/(eta+delta_tt*alpha));
    delta_r=r_n1-r_n;




    %*       initializing
     r0 = sigma_u/sqrt(E);
     zero_q=(1e-6)*r0;
     A_1 = abs(H);             % this slope accomplishes approx 98% q_inf
                               % in r = 2*r0


 % definition of q_inf for exponential law definition
    if H > 0
        q_inf = 2*r0;         % the q_inf variable is developed in 2*r0
    else
        q_inf = zero_q;       % the material degrades until the minimum val
    end                       % value of q



    if hard_type == 0           % if the hardening type is linear executes
        q_n1= q_n+ H*delta_r;   % this structure, if not the only other
                                % option is exponential
    else
        % Exponential softening/hardening law
        q_n1 = q_inf - (q_inf - r0)*exp(A_1*(1-(r_n1/r0))); %Lecture 4
                                                             %Slide 13
    end

    if(q_n1<zero_q)             % if the hardening variable is less than
        q_n1=zero_q;            % the minimum, is the minimum
    end

else

    %*     Elastic load/unload
    fload=0;
    r_n1= r_n  ;
    q_n1= q_n  ;
    %q_vec = [q_vec;q_n1];
end

% Damage variable
% ---------------
dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
sigma_n = ce*eps_n1';

if visc==1
    if rtrial>r_n
        C_tan = (1-dano_n1)*ce;
        C_alg = C_tan + ((alpha*delta_tt)/(eta+alpha*delta_tt))*(inv(tau_n1))*...
            ((q_n1-H*r_n1)/(r_n1)^2)*(sigma_n*(sigma_n'));
    else
        C_tan = (1-dano_n1)*ce;
        C_alg = C_tan;
    end
else
    if rtrial > r_n
        C_tan = (1-dano_n1)*ce;
        C_alg = C_tan-((q_n1-H*r_n1)/(r_n1)^3)*(sigma_n*(sigma_n'));
    else
        C_tan = (1-dano_n1)*ce;
        C_alg = C_tan;
    end
end
```

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

A.2 Rate independent implementation

A.2.1 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
    sigmaOT = ce*eps_n1';
    for i=[1:4]
        if sigmaOT(i)<0
            sigmaOT(i)=0;
        else
            sigmaOT(i) = sigmaOT(i);
        end
    end
    ceI=inv(ce);
    eps_n1n = ceI*sigmaOT;
    rtrial=sqrt(eps_n1n'*ce*eps_n1n);

elseif (MDtype==3)  %*Non-symmetric
    denom_NS = 0;
    numer_NS = 0;
    sigmaNS1 = ce*eps_n1';
    sigmaNS = [sigmaNS1(1) sigmaNS1(2) sigmaNS1(4)];
    for i = 1:3
        denom_NS = denom_NS + abs(sigmaNS(i));
        if sigmaNS(i)<0
```

```
            numer_NS= numer_NS;
        else
            numer_NS = numer_NS + sigmaNS(i);
        end
    end
    theta_NS = numer_NS/denom_NS;
    ceI=inv(ce);
    %eps_n1NS = ceI*sigmaNS1;
    %rtrial=(theta_NS + ((1-theta_NS)/n))*sqrt(eps_n1NS'*ce*eps_n1NS);
    rtrial=(theta_NS + ((1-theta_NS)/n))*sqrt(sigmaNS1'*ceI*sigmaNS1);
end

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

A.2.2 dibujar_criterio_dano1

```
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----->    %*
%*                      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
    delta_th = 0.04; %this parameter helps the surface to avoid going to
                     %infinite when only tension model is used
    tetha = [-(pi/2)+delta_th:0.01:pi-delta_th]; % domain where values of there;
% Radius                                 % are values of s1 > 0 or s2 > 0
    D=size(tetha);
    m1=cos(tetha);
    m2=sin(tetha);
    Contador=D(1,2);

    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;
    mp_1  = m1;
    mp_2  = m2;

    for i = 1:Contador
        if mp_1(i)<0
            mp_1(i) = 0;
        else
            mp_1(i) = mp_1(i);
        end
        if mp_2(i)<0
            mp_2(i) = 0;
        else
            mp_2(i) = mp_2(i);
        end

        radio(i)= q/sqrt([mp_1(i) mp_2(i) 0 nu*(mp_1(i)+mp_2(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==3
    theta_1 = [0:0.01:pi/2];                    % tension domain
    theta_2 = [pi+0.01:0.01:3*pi/2];            % compression domain
    theta = [theta_1 theta_2 theta_1(1)];       % sum of the domains

    D=size(theta);
    m1=cos(theta);
    m2=sin(theta);
    Contador=D(1,2);

    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;
    mp_1  = m1;
    mp_2  = m2;

    for i = 1:Contador
        if (mp_1(i)<=0) & (mp_2(i)<=0)
            mp_1(i) = (1/n)*mp_1(i);
            mp_2(i) = (1/n)*mp_2(i);
        else
            mp_1(i) = mp_1(i);
```

```
            mp_2(i) = mp_2(i);
        end

         radio(i)= q/sqrt([mp_1(i) mp_2(i) 0 nu*(mp_1(i)+mp_2(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);
end
return
```

A.3 Rate dependent implementation

A.3.1 damage_main

```
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    ; %this maybe related to the number of strain components
    mhist   = 6    ; %no idea what this is
end

if viscpr == 1
    mstrain = 4    ;
    mhist   = 6 ;
else
end


totalstep = sum(istep) ;


% INITIALIZING GLOBAL CELL ARRAYS
% ------------------------------
sigma_v = cell(totalstep+1,1) ;
TIMEVECTOR = zeros(totalstep+1,1) ;
delta_t = TimeTotal./istep/length(istep) ;


% Elastic constitutive tensor
% ----------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
% Historic variables
% hvar_n(1:4) --> empty
```

```
% hvar_n(5) = q --> Hardening variable
% hvar_n(6) = r --> Internal variable
hvar_n  = zeros(mhist,1)  ;

% INITIALIZING  (i = 1) !!!!
% ***********i*
i = 1 ;
r0 = sigma_u/sqrt(E);
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_n
eps_n1 = strain(i,:) ;
sigma_n1 =ce*eps_n1'; % Elastic
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0  sigma_n1(4)];

nplot = 3 ;
vartoplot = cell(1,totalstep+1) ;
vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5)  ; %  Damage variable (d)

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;
        % Total strain at step "i"
        % ------------------------
        eps_n = strain(i-1,:) ;
        eps_n1 = strain(i,:) ;
        %*************************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var,C_tan, C_alg] = rmap_dano1(eps_n,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

        Ctan = C_tan(1,1);
        Calg = C_alg(1,1);
        vartoplot{i}(4) = Ctan; % C11 tangent constitutive tensor
        vartoplot{i}(5) = Calg; % C11 algorithmic constitutive tensor

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