Computational Solid Mechanics
Assignment 1 - Damage Models

Federico Valencia Otálvaro

Master's in Numerical Methods in Engineering
Universitat Politècnica de Catalunya

March 30th, 2020

Contents

List of Figures

1 Part I: Rate Independent Models

1.1 Damage Models Implementation

In order to implement the tension only and non-symmetric damage models in the provided Matlab program, the `Modelos_de_dano1` (see appendix A.1, lines 24 to 36) and `dibujar_criterio_dano1` (see appendix A.2, lines 64 to 158) routines had to be modified. The following graphics display the damage surfaces corresponding to the tension only and non-symmetric models respectively, according to slides 16-19 of lecture 4 of the course.

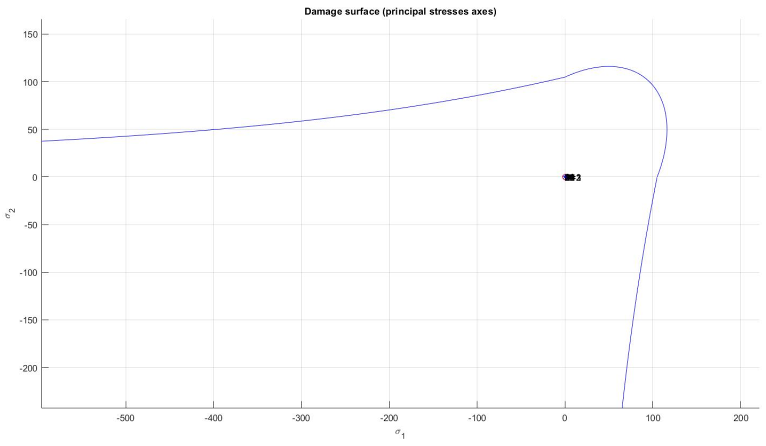

Figure 1: Tension Only Damage Surface

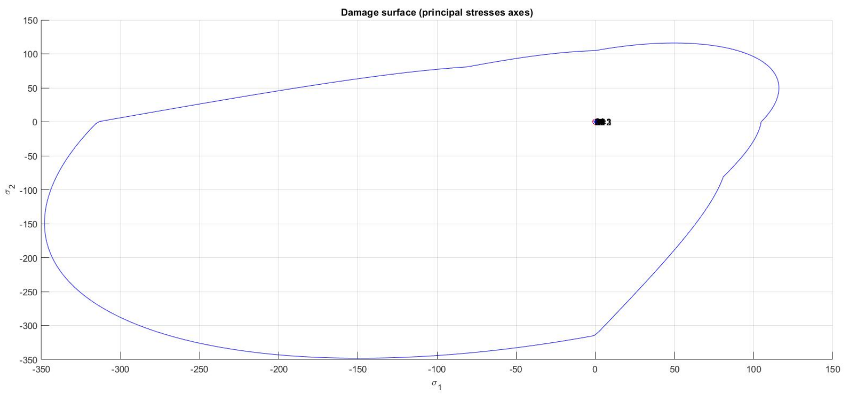

Figure 2: Non-Symmetric Damage Surface

1.2 Exponential Hardening Law

For the implementation of the exponential hardening law, the `rmap_dano1` routine (see Appendix A.3, lines 50-56 and 108) was modified. In order to obtain the value of the material parameter A, the expression for H(r) presented at slide 13 of lecture 4 was evaluated for$r = r_0$. Isolating A yields:

$$A = \frac{H(r)r_0}{q_\infty - r_0} \qquad A > 0 \tag{1}$$

Where q_∞ was arbitrarily defined as:

$$q_\infty = r_0 \times 10^{-6} \quad (Lower\ bound) \tag{2}$$

$$q_\infty = r_0(2 - 10^{-6}) \quad (Upper\ bound) \tag{3}$$

Once the value of A is known, q(r) for the exponential hardening law may be computed using the expression shown in slide 13 of lecture 4.

1.3 Implementation Assessment

1.3.1 Test 1: Uniaxial Loading

The first test consists of **uniaxial** loading and unloading of a material which hardens exponentially when subjected to damage. The three segments of the loading path describe an initial tensile loading phase, followed by a an unloading/compressing loading phase, and a final tensile loading 50% greater than the initial tensile load. Table 1 contains the final values of each one of the segments of the loading path.

	$\Delta\overline{\sigma}_1$	$\Delta\overline{\sigma}_2$
1	300	0
2	-200	0
3	450	0

Table 1: Loading Path, Test 1 (Part I)

The following material properties were used for the assessment of both the tension only and non-symmetric damage models.

E	ν	H	f_y	Hard. Type
20000	0.3	0.1	100	Exp.

Table 2: Material Properties, Test 1 (Part I)

Where E is the Young's modulus, ν is the Poisson ratio, H is the hardening modulus, and fy is the yield stress.

- Tension Only Damage Model

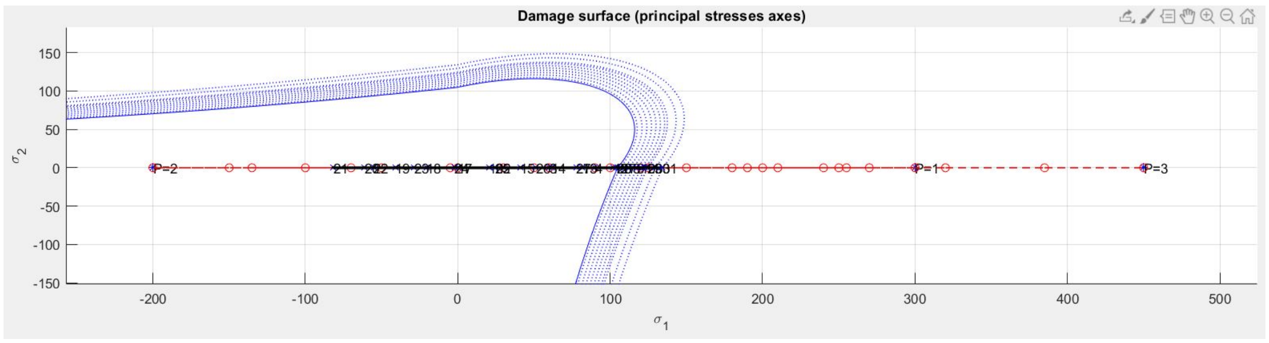

Figure 3: Stress Path, Tension Only Model, Test 1 (Part I)

For the tension only model, there should be no damage during the unloading/compressive loading phase. However, for the first and third segments of the loading path, the tensile stresses exceed the yield stress of the material, and therefore, it should present a certain amount of damage. It may also be observed that the stress path remains within the damage surface, which is an indicator of a correct implementation.

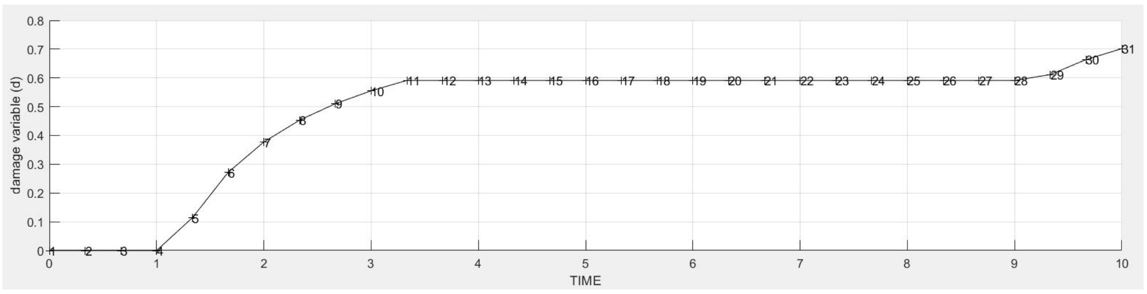

Figure 4: Damage Variable vs. Time, Tension Only Model, Test 1 (Part I)

As expected, damage increment is present only during the tensile loading phases, while the damage remains constant during the compressive loading stage. As a consequence, the stress vs. strain curve should display loss of strength with a certain degree of material hardening ($H > 0$) in intervals where damage grows, due to the damage-induced alteration of the constitutive tensor.

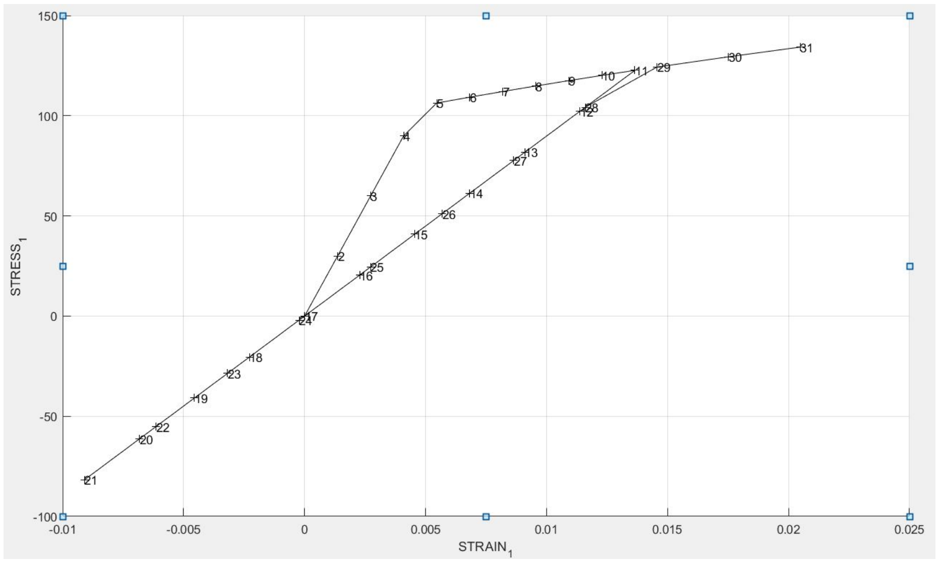

Figure 5: Stress I vs. Strain I, Tension Only Model, Test 1 (Part I)

The stress vs. strain curve correctly illustrates the expected behavior of the material, since there is evidence of hardening during tensile loading and perfectly elastic behavior during compressive loading.

- Non-Symmetric Damage Model

For the non-symmetric model, the compression/tension ratio of the material was chosen as n=3, while all material properties and loads were kept identical to those used for the tension only model.

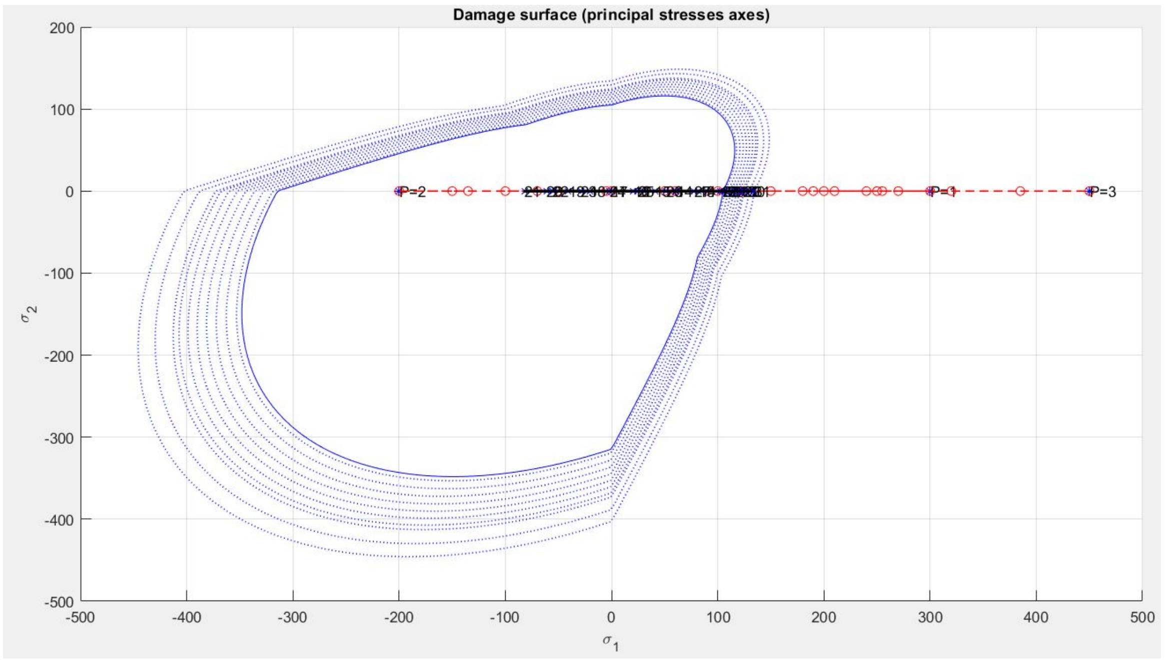

Figure 6: Stress Path, Non-Symmetric Model, Test 1 (Part I)

We may observe that the load capacity of the material is not exceeded during the compressive loading phase and therefore, damage should only exist during tensile loading. Since both models were assigned identical material properties and loads, in this particular case, both of them should display the same behavior.

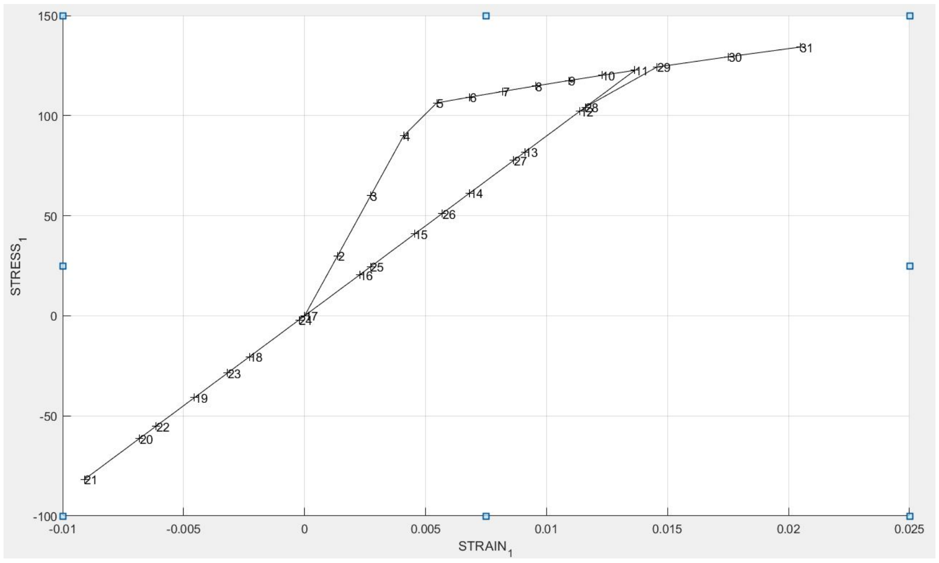

Figure 7: Stress I vs. Strain I, Non-Symmetric Model, Test 1 (Part I)

By comparing Figure 5 and Figure 7 it is evident that both models exhibit the same behavior under this particular set of conditions, as it was expected.

1.3.2 Test 2: Biaxial Loading

Test 2 consists of an initial uniaxial tensile loading phase, followed by a biaxial unloading and compressive loading phase, and a final biaxial tensile loading. The coordinates of the final point of each segment of the loading path are displayed in Table 3.

	$\Delta\overline{\sigma}_1$	$\Delta\overline{\sigma}_2$
1	80	0
2	-500	-500
3	700	700

Table 3: Loading Path, Test 2 (Part I)

The following material properties were used for the assessment of both damage models.

E	ν	H	f_y	Hard. Type
40000	0.2	0.1	100	Exp.

Table 4: Material Properties, Test 2 (Part I)

It is worth noting that the maximum stress applied during the uniaxial tensile loading phase is smaller than the yield stress of the material and as a consequence, the behavior of both models should be elastic during this stage.

- Tension Only Damage Model

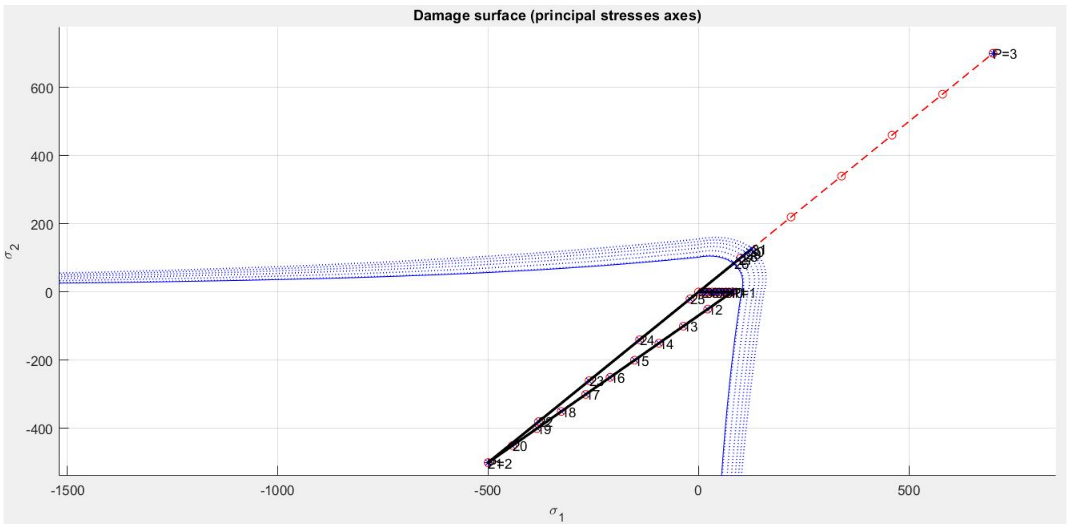

Figure 8: Stress Path, Tension Only Model, Test 2 (Part I)

Segments 1 and 2 of the stream path remain inside the elastic range of the material. Therefore, the damage variable should not increase during these two stages and the constitutive behavior should be completely elastic.

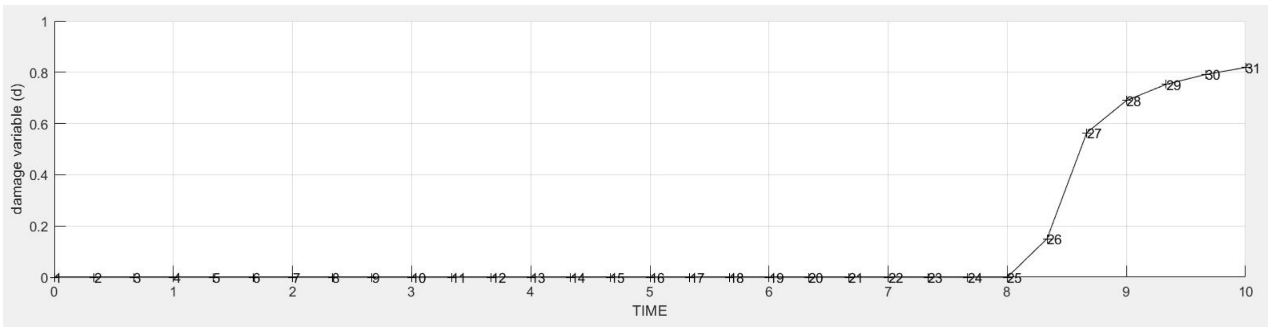

Figure 9: Damage Variable vs. Time, Tension Only Model, Test 2 (Part I)

Figure 9 shows damage starting to show during the 2 final seconds of the simulation, which correspond to the final tensile loading phase. From Figure 8 we may clearly see that the period of time when damage appears corresponds to the period when the stress path goes beyond the damage surface of the material.

- Non-Symmetric Damage Model

For Test 2, the compression/tension ratio of the material was kept unchanged with respect to the one used for Test 1 (n = 3). However, since the compressive loads in this case are much larger, they should exceed the capacity of the material and the damage variable should increase during the compressive loading phase, unlike Test 1.

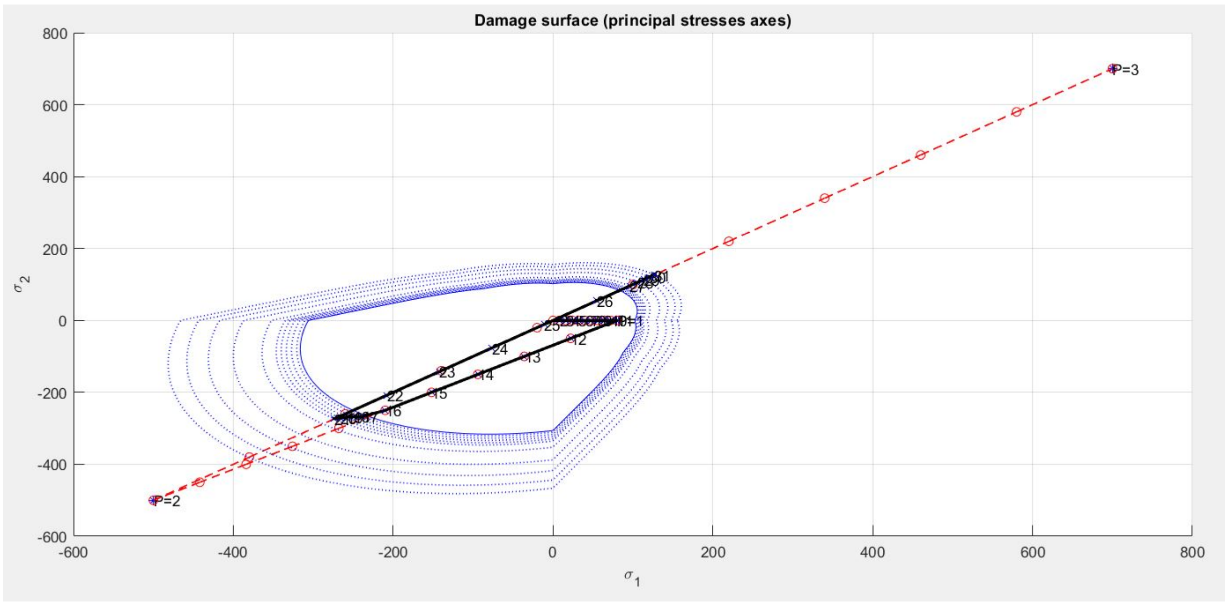

Figure 10: Stress Path, Non-Symmetric Model, Test 2 (Part I)

The stress path goes beyond the material's capacity during the compressive loading phase and the second tensile loading phase.

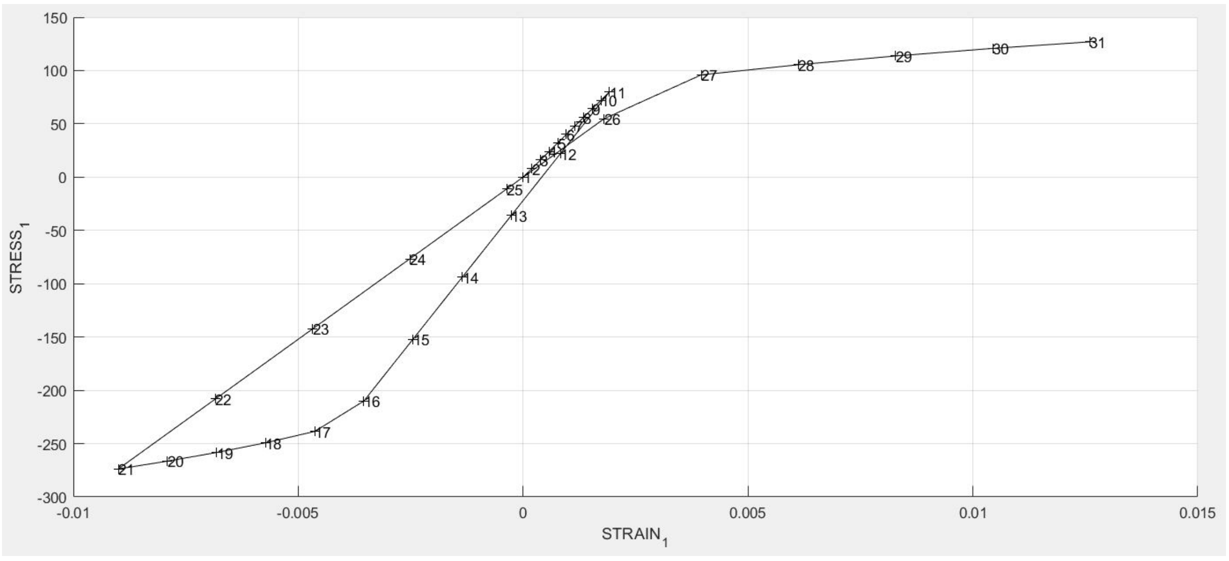

Figure 11: Stress I vs. Strain I, Non-Symmetric Model, Test 2 (Part I)

As expected, there is a loss of material strength along with hardening during the second (steps 16 to 21) and third (steps 26 to 31) loading stages, while elastic behavior holds during the first tensile loading stage.

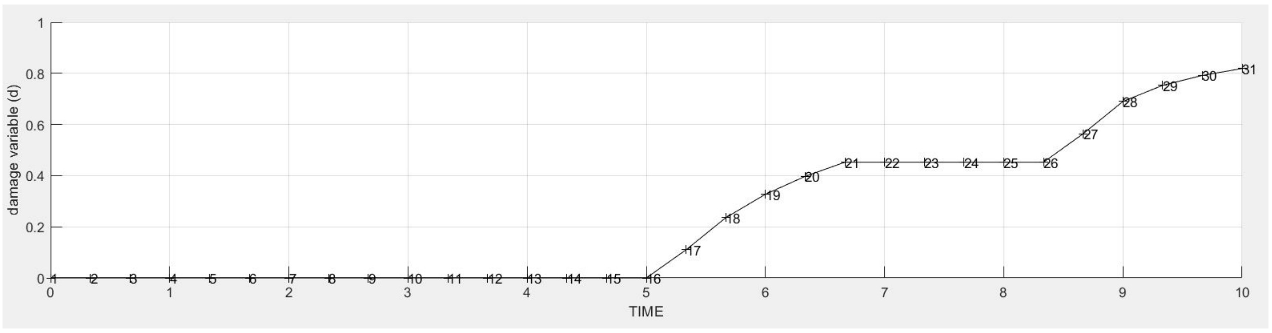

Figure 12: Damage Variable vs. Time, Non-Symmetric Model, Test 2 (Part I)

The damage variable in Figure 12 increases accordingly to what is evidenced in the stress vs. strain plot (Figure 11). In general, the implemented solution for the non-symmetric model shows satisfactory results.

1.3.3 Test 3: Biaxial Loading

Test 3 consists of an initial biaxial tensile loading phase, followed by biaxial compressive loading with a magnitude twice as large than the one corresponding to the same phase in Test 2, and a final biaxial tensile loading phase with stress 50% larger than the one applied in phase 1.

	$\Delta\overline{\sigma}_1$	$\Delta\overline{\sigma}_2$
1	300	300
2	-1000	-1000
3	450	450

Table 5: Loading Path, Test 3 (Part I)

The following material properties were used for both the tension only and non-symmetric models.

E	ν	H	f_y	Hard. Type
20000	0.3	-0.2	100	Lin.

Table 6: Material Properties, Test 3 (Part I)

Since the hardening modulus is now negative and the hardening type was changed to linear, the material should exhibit linear softening when subjected to damage.

- Tension Only Damage Model

  Linear softening of the material is evidenced during both tensile loading phases, while elastic behavior is maintained during the compressive loading phase. This is also reflected on the damage variable plot, where the largest increment occurs on the first third of the simulation time lapse, corresponding to the first tensile loading phase. On the second third, damage remains constant while the compressive loading is being applied and finally, there is a slight increase during the final time step corresponding to the final tensile loading.

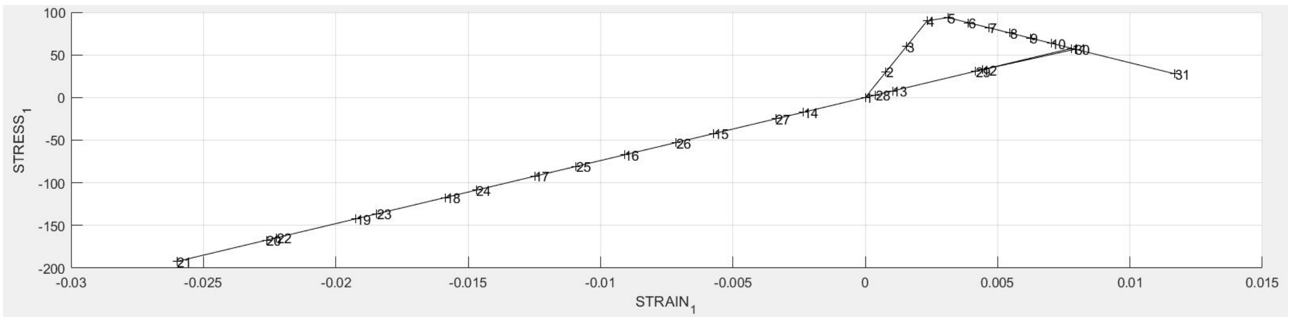

Figure 13: Stress I vs. Strain I, Tension Only Model, Test 3 (Part I)

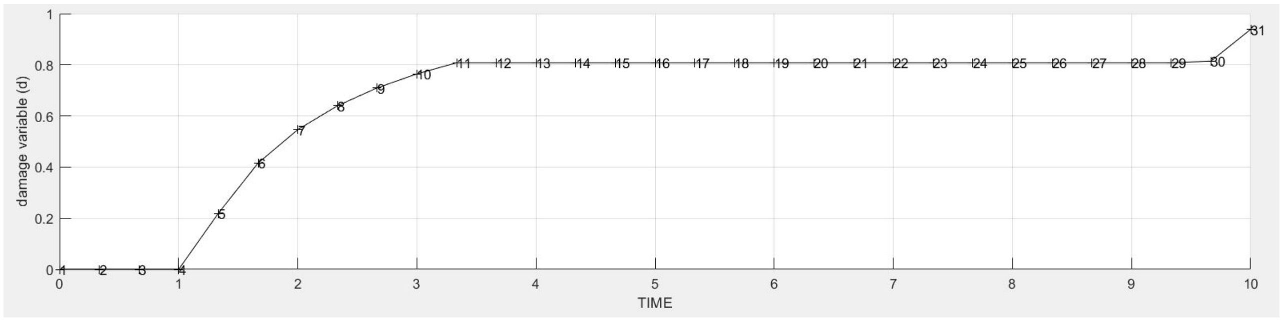

Figure 14: Damage Variable vs. Time, Tension Only Model, Test 3 (Part I)

- Non-Symmetric Damage Model

  The non-symmetric model exhibits linear softening during the first and second loading stages. However, unlike the tension only model, it remains elastic during the third loading stage. The final values for strain and the damage variable are identical for both models but the trajectories these variables followed varied accordingly to the characteristic behavior of each model. Their behavior is identical from time steps 0 to 18, point at which the non-symmetric model begins to suffer compression damage while the tension only model remains elastic.

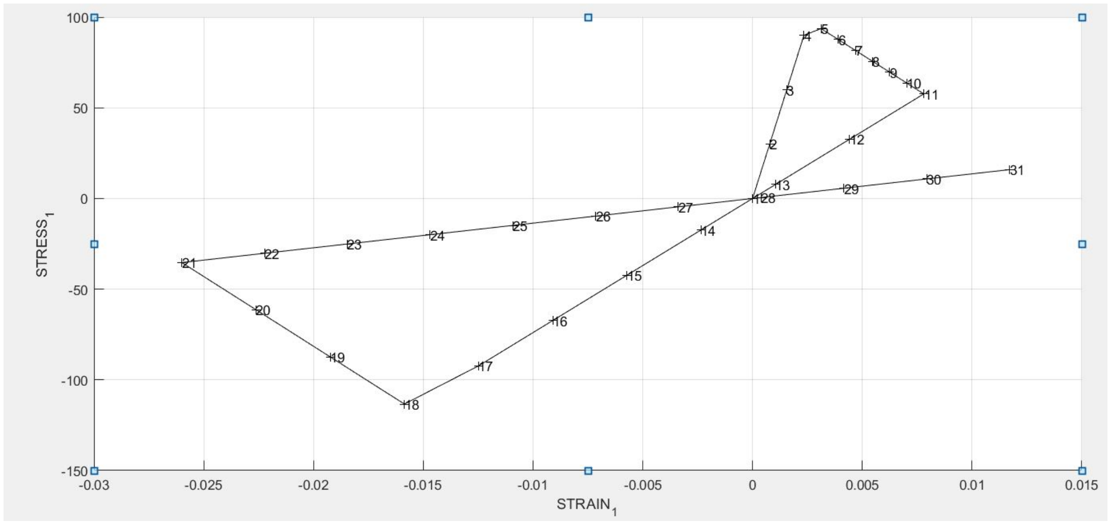

Figure 15: Stress I vs. Strain I, Non-Symmetric Model, Test 3 (Part I)

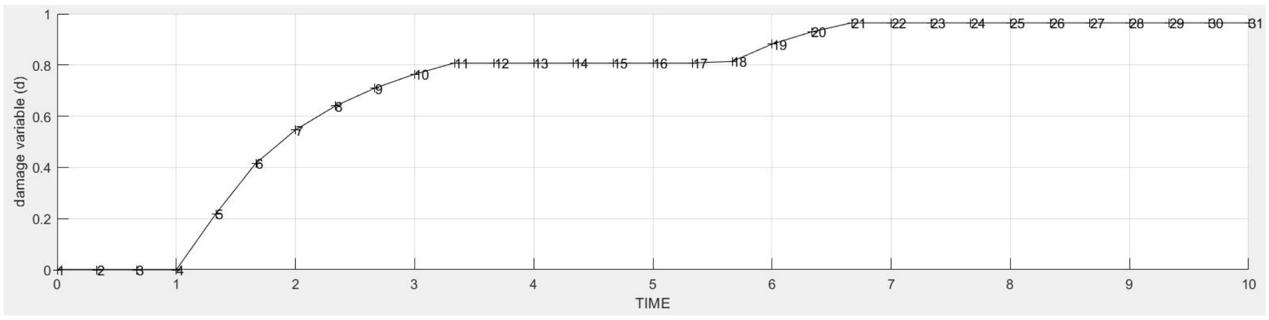

Figure 16: Damage Variable vs. Time, Non-Symmetric Model, Test 3 (Part I)

2 Part II: Rate Dependent Models

2.1 Visco-Damage Symmetric Tension-Compression Model Implementation

For the implementation of the rate dependent symmetric damage model, the Matlab routines `rmap_dano1` (see appendix A.3) and `damage_main` (see appendix A.4) were modified, as well as their input and output parameters.

In order to compute r_{n+1} during a loading stage, it is necessary to first calculate $\tau_{\varepsilon_{n+\alpha}}$ and therefore, ε_n had to be included as an input parameter for `rmap_dano1` in addition to ε_{n+1} (see slide 15, lecture 5). Furthermore, since we are dealing with a numerical time integration scheme, Δt was introduced as an input parameter for this function as well.

In addition to all the necessary modifications for the rate dependent model implementation, the tangent constitutive tensor and the consistent (algorithmic) tangent constitutive tensor were implemented as outputs of the `rmap_dano1` function and are later used in `damage_main` to build vectors containing the position C_{11} of each tensor for every time step.

2.2 Implementation Assessment

The correctness of the implemented solution was assessed with a contiously increasing biaxial loading problem with a loading path of three segments described by the following coordinates in the principal stresses plane:

	$\Delta\overline{\sigma}_1$	$\Delta\overline{\sigma}_2$
1	300	300
2	500	500
3	700	700

Table 7: Loading Path, Part II

Three different tests were performed, modifying a different parameter in each one of them and analyzing the change in the stress vs. strain curve for different values of said parameter. The viscous coefficient η, the time integration parameter α and the strain rate are the variables modified in each one of the performed analysis. Table 8 contains the material properties which were used for all three tests.

E	ν	H	f_y	Hard. Type
20000	0.3	0.1	100	Lin.

Table 8: Material Properties, Part II

2.2.1 Test 1: Variation of η

- $\eta = [0.05, 0.1, 0.2, 0.5, 1]$
- $\alpha = 0$ (Explicit time integration scheme)
- Time interval = 10 seconds. 30 time steps

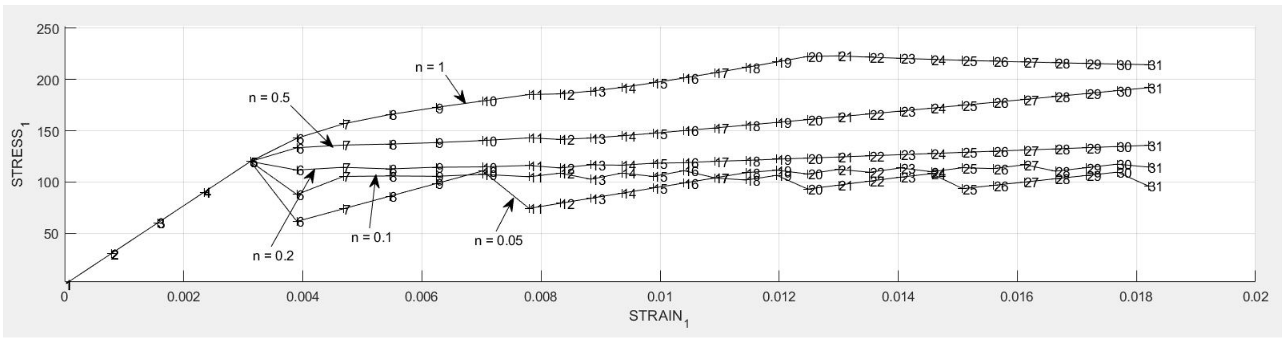

Figure 17: Stress I vs. Strain I for Different Values of η (Part II)

We may observe that there is an oscillation in stress values for the two first curves ($\eta = 0.05$ and $\eta = 0.1$). This is due to instability in the integration scheme, which is a possibility since explicit integration is being performed ($\alpha = 0$). The problem ceases to exist for higher values of η and there is a clear tendency of larger stresses (hardening) for larger viscous parameters.

2.2.2 Test 2: Variation of Strain Rate

- $\eta = 1$
- $\alpha = 0$ (Explicit time integration scheme)
- Time interval = 10 seconds
- Time steps = $[6, 9, 12, 18, 30]$

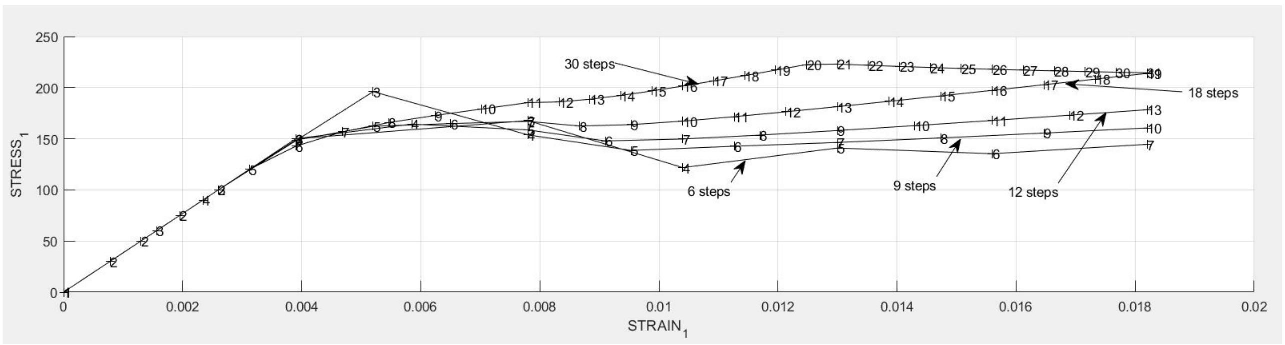

Figure 18: Stress I vs. Strain I for Different Strain Rates (Part II)

For Test 2, a larger amount of time steps in the same time interval should provide more accurate results, since means increased precision of the time integration scheme. The curve generated by the scheme with 6 time steps ($\Delta t = 3.33s$) clearly displays instability in the stress vs. strain curve, while curves generated for a smaller Δt are smooth. There is once again a tendency of larger stresses for a more refined time discretization.

2.2.3 Test 3: Variation of α

- $\eta = 0.05$
- $\alpha = [0, 0.25, 0.5, 0.75, 1]$
- Time interval = 10 seconds. 30 time steps

The value of $\eta = 0.05$ was purposefully chosen since it was the one which presented the largest instability for Test 1, in order to evaluate the change in this phenomenon by switching to an implicit time integration scheme, which is unconditionally stable.

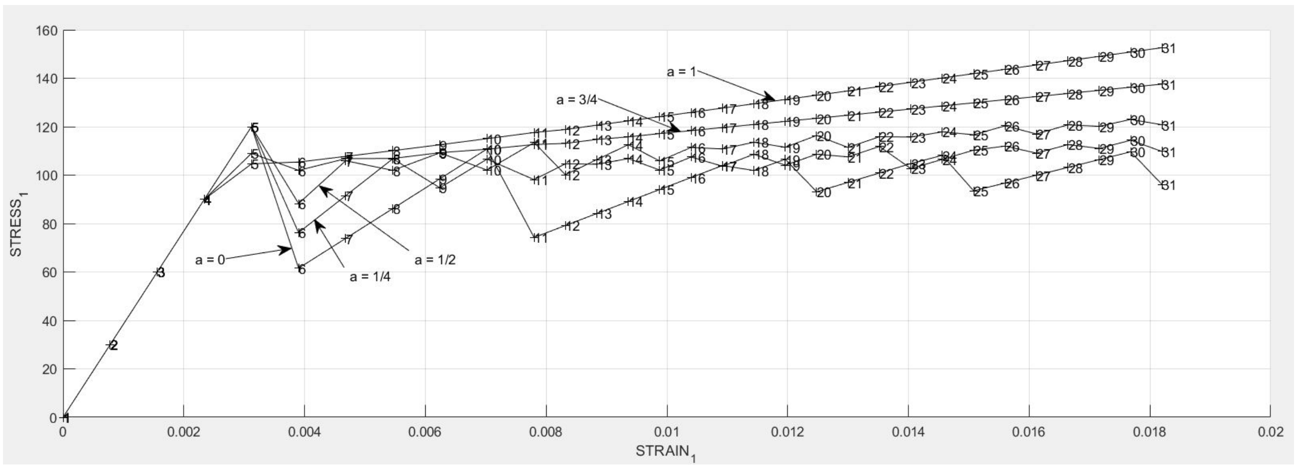

Figure 19: Stress I vs. Strain I for Different Values of α (Part II)

As expected, the instability problem disappears for implicit time integration. However, the peak in the curve at time step 6 persists for all integration schemes, which allows us to conclude that it is not a consequence of instability but of the way the system behaves under the selected viscous coefficient when damage initially appears.

The instability of the explicit schemes also reflects on both the tangent and algorithmic constitutive tensors as they present oscillations which are directly related to the oscillations in the damage variable. It may also be observed that these two tensors are not equal but as time passes (and damage increases) they tend to converge towards the same value, independently of the value of α.

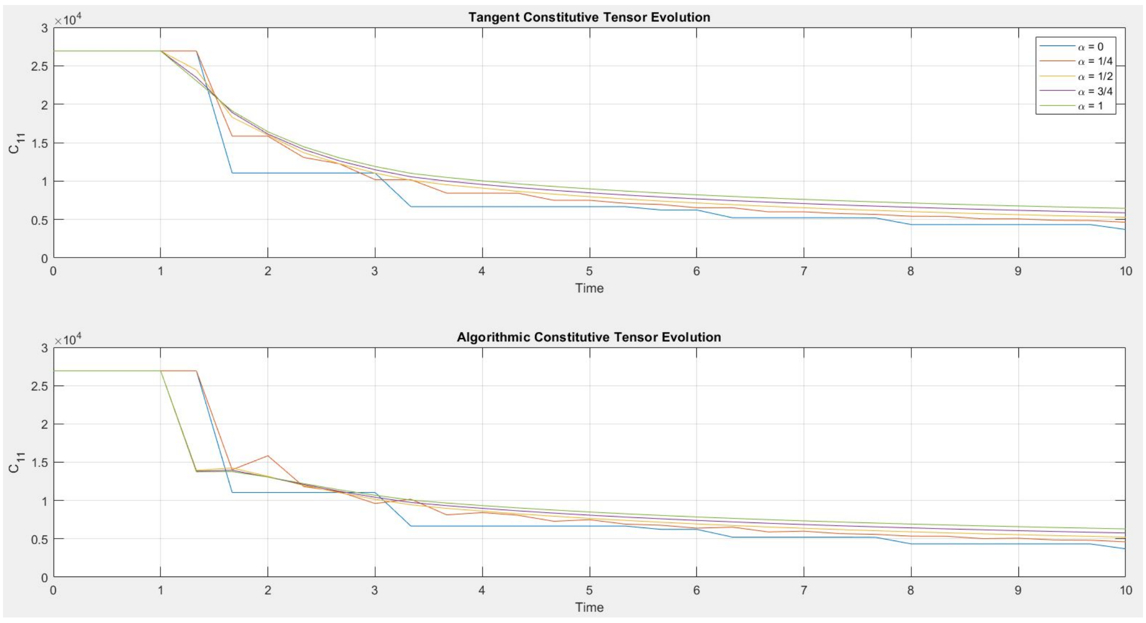

Figure 20: Tangent and Algorithmic Constitutive Tensors Evolution for Different Values of α (Part II)

A Appendices: Modified Matlab Routines

A.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');
```

```
% IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif (MDtype==2)  %* Only tension
    sigma = ce*eps_n1';
    pos = sigma > 0;    % boolean vector with positive values of sigma
    sigma_p = sigma.*pos;   %sigma+
    rtrial = sqrt(sigma_p'*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
    sigma = ce*eps_n1';
    pos = sigma > 0;    % boolean vector with positive values of sigma
    sigma_p = sigma.*pos;   %sigma+
    o = sum(sigma_p) / sum(abs(sigma));  % theta in the slides
    rtrial = (o + (1-o)/n) * sqrt(eps_n1*ce*eps_n1');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

A.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----->t e n s i n   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

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

    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    m1_ = [];     %cos(tetha) for sigma+
    m2_ = [];     %sin(tetha) for sigma+

    for i=1:length(tetha)
        if tetha(i) >= 0 && tetha(i) <= pi/2
            m1_(i) = cos(tetha(i));
            m2_(i) = sin(tetha(i));

        elseif tetha(i) > pi/2 && tetha(i) <= pi
            m1_(i)= 0;
            m2_(i)= sin(tetha(i));

        elseif tetha(i) > pi && tetha(i) <= 3*pi/2
            m1_(i)= 0;
            m2_(i)= 0;

        else
            m1_(i) = cos(tetha(i));
            m2_(i) = 0;

        end
    end
    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==3
% IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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

    %* RADIUS
    D=size(tetha);                      %*  Range
    m1=cos(tetha);                      %*
    m2=sin(tetha);                      %*

    o = [];          % tetha in the slides

    for i=1:length(tetha)
        if tetha(i) >= 0 && tetha(i) <= pi/2
            o(i) = 1;

        elseif tetha(i) > pi/2 && tetha(i) <= pi
           o(i) = m2(i)*(1+nu)/(abs(m1(i)) + abs(m2(i)) + ...
               abs(nu*(m1(i)+m2(i))));

        elseif tetha(i) > pi && tetha(i) <= 3*pi/2
            o(i) = 0;

        else
            o(i) = m1(i)*(1+nu)/(abs(m1(i)) + abs(m2(i)) + ...
                abs(nu*(m1(i)+m2(i))));

        end
    end

    Contador=D(1,2);                    %*


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


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

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

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




return
```

A.3 rmap dano1

```
function [sigma_n1,hvar_n1,aux_var,c_tan,c_alg] = rmap_dano1 (eps_n,eps_n1,hvar_n,Eprop,ce,M

%*************************************************************************************
%*                                          *
%*           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);    % initial value, equal to r0
q_n      = hvar_n(6);    % initial value, equal to r0
E        = Eprop(1);
nu       = Eprop(2);
H        = Eprop(3);
sigma_u  = Eprop(4);
hard_type = Eprop(5) ;

% IMPLEMENTED %%%%%%
viscpr = Eprop(6);
eta = Eprop(7);
alpha = Eprop(8);
%%%%%%%%%%%%%%%%%%%%%

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


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

```
%*       initializing                                            %*
 r0 = sigma_u/sqrt(E);

 zero_q = 1.d-6*r0;      % lower bound

 % IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 inf_q = 2*r0 - zero_q; % upper bound

 if H < 0    % find the value of A when r = r0
    A = H * r0 / (zero_q - r0);
 else
    A = H * r0 / (inf_q - r0);
 end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% if(r_n<=0.d0)
%     r_n=r0;
%     q_n=r0;
% end
%*************************************************************************


%*************************************************************************
%*       Damage surface
%*
[rtrial] = Modelos_de_dano1(MDtype,ce,eps_n1,n);
%*************************************************************************

% IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MODIFY rtrial FOR VISCOUS CASE
if viscpr == 1       % ONLY IMPLEMENTED FOR SYMMETRIC MODEL
    Tao_e = sqrt(eps_n*ce*eps_n');
    Tao_e1 = rtrial;
    rtrial = (1-alpha)*Tao_e + alpha*Tao_e1;
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;

    % IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if viscpr == 0       % INVISCID CASE
        r_n1 = rtrial;
    else                 % VISCOUS CASE
        r_n1 = (eta-delta_t*(1-alpha))/(eta+alpha*delta_t)*r_n + ...
            delta_t/(eta+alpha*delta_t)*rtrial;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % HARDENING LAW
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;

        if q_n1 < zero_q
        q_n1 = zero_q;

        elseif q_n1 > inf_q
            q_n1 = inf_q;

        end

    else

        % IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Exponential Hardening
        q_n1 = inf_q - (inf_q - r0)*exp(A*(1-r_n1/r0));
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    end


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

% Damage variable
% ---------------
dano_n1   = 1.d0-(q_n1/r_n1);

%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';

% IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c_tan = (1.d0-dano_n1)*ce;      % tangent tensor

if rtrial > r_n     % Loading
    c_alg = c_tan + (alpha*delta_t*(H*r_n1-q_n1))/((eta + alpha*delta_t ...
        *Tao_e1)*r_n1^2)*(sigma_n1*sigma_n1');
else
    c_alg = c_tan;
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;

end
```

A.4 damage main

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,c_tan_11,c_alg_11]=damage_main(Eprop,ntype,
global hplotSURF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONTINUUM DAMAGE MODEL
% ----------------------
% Given the almansi strain evolution ("strain(totalstep,mstrain)") and a set of
% parameters and properties, it returns the evolution of the cauchy stress and other
variables
% that are listed below.
%
% INPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% ---------------------------------------------------------------------
% Eprop(1) = Young's modulus  (E)
% Eprop(2) = Poisson's coefficient (nu)
% Eprop(3) = Hardening(+)/Softening(-) modulus (H)
% Eprop(4) = Yield stress (sigma_y)
% Eprop(5) = Type of Hardening/Softening law  (hard_type)
%            0 --> LINEAR
%            1 --> Exponential
% Eprop(6) = Rate behavior (viscpr)
%            0 --> Rate-independent (inviscid)
%            1 --> Rate-dependent   (viscous)
%
% Eprop(7) = Viscosity coefficient (eta)  (dummy if inviscid)
% Eprop(8) = ALPHA coefficient (for time integration), (ALPHA)
%             0<=ALPHA<=1 , ALPHA = 1.0 --> Implicit
%                           ALPHA = 0.0 --> Explicit
%            (dummy if inviscid)
%
% ntype    = PROBLEM TYPE
%            1 : plane stress
%            2 : plane strain
%            3 : 3D
%
% istep = steps for each load state (istep1,istep2,istep3)
%
% strain(i,j) = j-th component of the linearized strain vector at the i-th
%               step, i = 1:totalstep+1
%
% MDtype      = Damage surface criterion %
%            1 : SYMMETRIC
%            2 : ONLY-TENSION
%            3 : NON-SYMMETRIC
%
%
% n          = Ratio compression/tension strength (dummy if MDtype is different from 3)
%
% TimeTotal  = Interval length
%
%  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%  --------------------------------------------------------------------
%  1) sigma_v{itime}(icomp,jcomp)  --> Component (icomp,jcomp) of the cauchy
%                                      stress tensor at step "itime"
%                                      REMARK: sigma_v is a type of
%                                      variable called "cell array".
%
```

```
%
%  2) vartoplot{itime}              −−> Cell array containing variables one wishes to plot
%                                     ────────────────────────────────────────
%   vartoplot{itime}(1) =   Hardening variable (q)
%   vartoplot{itime}(2) =   Internal variable (r)%

%
%  3) LABELPLOT{ivar}              −−> Cell array with the label string for
%                                    variables of "varplot"
%
%          LABELPLOT{1} => 'hardening variable (q)'
%          LABELPLOT{2} => 'internal variable'
%
%
%  4) TIME VECTOR  − >
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SET LABEL OF "vartoplot" variables  (it may be defined also outside this function)
% ────────────────────────────────────
 LABELPLOT = {'hardening variable (q)','internal variable'};

E      = Eprop(1) ; nu = Eprop(2) ;
viscpr = Eprop(6) ;
sigma_u = Eprop(4);


if ntype == 1
    menu('PLANE STRESS has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3−DIMENSIONAL PROBLEM has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
else
    mstrain = 4    ;
    mhist   = 6    ;
end

% IMPLEMENTED %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if viscpr == 1
        eta = Eprop(7);
        alpha = Eprop(8);

        if MDtype ~= 1
        menu({'Viscous model is implemented for Symmetric Damage Model only.'},  ...
            'STOP');
        error('OPTION NOT AVAILABLE FOR THIS MDtype')
        end
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) --> empty1
% 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)

% IMPLEMENTED %%%%%%%%
c_tan_11 = [];
c_alg_11 = [];

c_tan_11(i) = ce(1,1);
c_alg_11(i) = ce(1,1);
%%%%%%%%%%%%%%%%%%%%%%

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;
        % Total strain at step "i"
        % ------------------------

        % IMPLEMENTED %%%%%%%%
        eps_n = eps_n1;
        %%%%%%%%%%%%%%%%%%%%%%

        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(iload));

        % IMPLEMENTED %%%%%%%%%%%
        c_tan_11(i) = c_tan(1,1);
        c_alg_11(i) = c_alg(1,1);
        %%%%%%%%%%%%%%%%%%%%%%%%%

        % 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
```