UPC - BARCELONA TECH
Master on Numerical Methods in Engineering
Spring 2018

Computational Solid Mechanics

Assignment # 1: Continuum Damage Models

Prasad Adhav

Due date: 23 March 2018

Contents

List of Figures

1 Rate independent model

For the first part of the assignment, we are asked to obtain the path at the stress space and the stress-strain curve for different loading/unloading cases. The results obtained through the simulation are presented in the subsequent subsections down below. In order to obtain the requested plots, we have provided the code with the following basic material properties,

$$E = 20000\ MPa \quad ; \quad \sigma_Y = 200\ MPa \quad ; \quad \nu = 0.3 \quad ; \quad H = 0.25$$

where E is the Young modulus, σ_Y is the so-called yield stress, ν is the Poisson ratio and H stands for the hardening/softening modulus. Also, a ratio of compression strength to tension strength $n = 2$, and a total time of $t = 10\ s$ have been considered.

1.1 Case 1: complete uniaxial path

The first case of analysis consists in a loading path comprising an uniaxial tensile loading, an uniaxial tensile unloading/compressive loading and an uniaxial compressive unloading/ tensile loading. Mathematically,

$$\Delta\bar{\sigma}_1^{(1)} = \alpha,\ \Delta\bar{\sigma}_2^{(1)} = 0 \quad ; \quad \Delta\bar{\sigma}_1^{(2)} = -\beta,\ \Delta\bar{\sigma}_2^{(2)} = 0 \quad ; \quad \Delta\bar{\sigma}_1^{(3)} = \gamma,\ \Delta\bar{\sigma}_2^{(3)} = 0 \tag{1}$$

being α, β and γ arbitrary parameters chosen by the student. For this case, we have considered $\alpha = 150\ MPa$, $\beta = -700\ MPa$ and $850\ MPa$. Figure 1.1 down below, shows a graphical representation of the obtained path in the stress space for the tension-only damage model. It also includes the points in the time discretization.

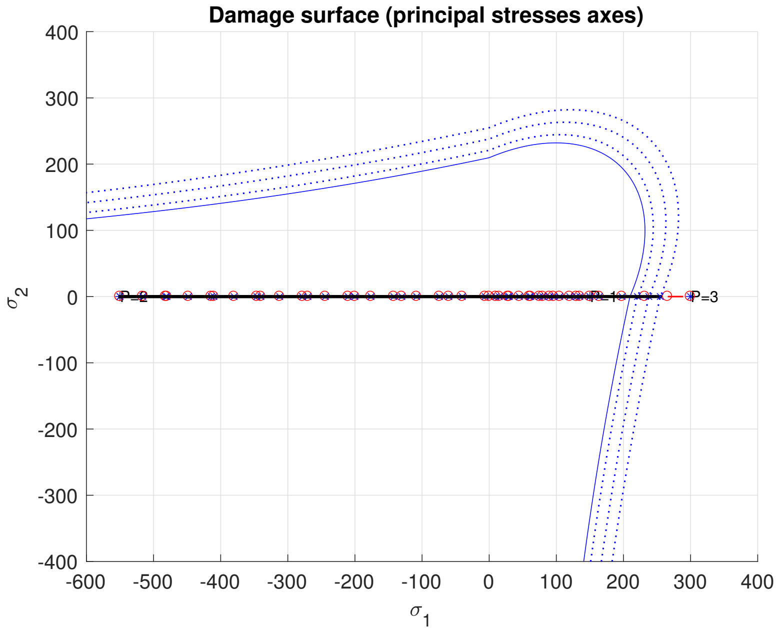


Figure 1.1: Graphical representation of the loading/unloading path in the stress space considered for the first case of analysis for the tension-only damage model. This path is completely uniaxial.

In the chosen path, we have first done a load stage without going beyond the limit value σ_Y, then we have compressed the material and finally we have loaded again until we cross the boundary of the initial domain, so to speak. The dotted blue lines represent the evolution of the stress space, which has expanded as a result of the hardening in order to keep the points on the boundary.

Figure 1.2, shows the stress-strain curve obtained for the already described loading path. It is important to note that the black-crossed line represents the first load stage, the blue-circled line shows the points corresponding to the second stage and the green line contains the third load stage.

As we have not gone beyond the yields stress on the first load stage, all the points lie in a straight line of slope E. In the compression stage, this situation is replicated but in the opposite direction. In fact, we could have given a higher increment on the load in the second stage and the points would still lie on the same straight line as there is no border to cross in compression. This is in fact the main characteristic of the tension-only model, i.e. a material can only fail by tension and not in compression.

The interesting part of the plot comes once we overpass the original elastic surface. From here, the material no longer shows an elastic behavior, it goes through a hardening process (recall that we considered a positive H) and the evolution of the internal variable stops being zero.

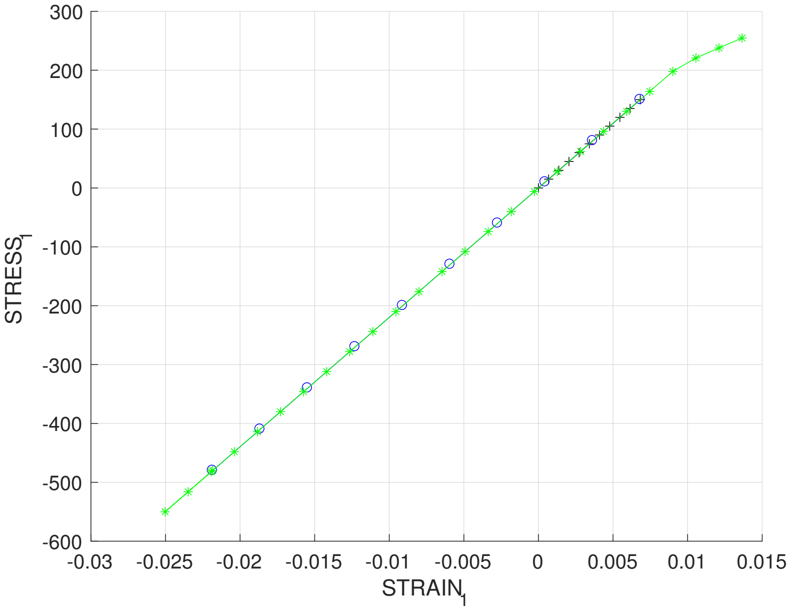


Figure 1.2: Stress-strain curve for the first case of analysis considering the tension-only damage model. The black-crossed line represents the first stage, the blue-circled line shows the second stage and the green-starry one represents the evolution on the third load stage.

Let us now introduce the results obtained for this simulation in the case of the non-symmetrical damage model. This model is useful to simulate materials, such as concrete, whose tension domain really differs from the compression one. In Figure 1.3, the path in the stress space considered in this uniaxial case is shown. It is important now to note the huge difference between this plot and the one from Figure 1.1. Now, a material is allowed to fail in compression or, in other words, we can now leave the elastic domain by means of a compressive load. This is in fact what happened for the value of β chosen for the computations. As we have exceeded the domain in the compression side, we expect the material to undergo some kind of damage and the surface to evolve so as to keep points on the boundary (points outside the interior of the domain or the boundary are not feasible).

In order to examine the considered path using the non-symmetrical model, Figure 1.4 offers the graphical representation of the stress-strain curve for this case.

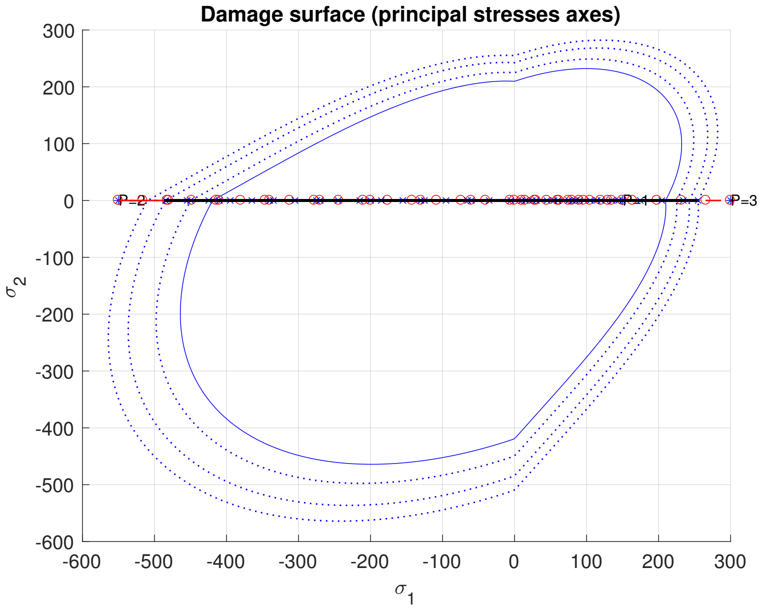


Figure 1.3: Plot of the load/unload path in the stress space considered in the uniaxial case of analysis for the non-symmetrical damage model. The path is completely uniaxial.

As before, the first state remains on the elastic domain, therefore its representation is just a straight line of slope E. As commented before, for the second stage we introduced a value so that we exit the elastic domain. Therefore the internal variable starts to evolve and this fact is translated into a curve in the lower part of the plot. This basically means that the material undergoes a hardening process (in the compression part). Finally, as the material is loaded again, the point moves towards the interior of the domain and it crosses it from the compression part to the tension part. Thus, this is represented as a straight line in the stress-strain curve, being now the slope the so-called secant modulus E_s. This is again fulfill until we overpass the damage threshold and the material goes through hardening in the tensile part.

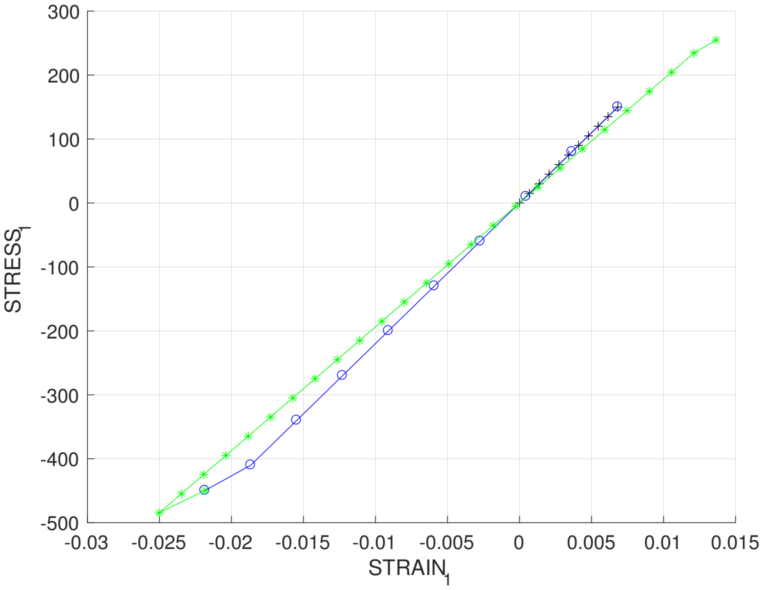


Figure 1.4: Stress-strain curve for the case of the non-symmetrical damage model and the uniaxial path. As explained the material now also leaves the elastic domain on the second compression stage.

1.2 Case 2: uniaxial/biaxial path

In this section, plots for the results obtained for the uniaxial/biaxial case are presented for both the only-tension and non-symmetrical damage models. The second case proposed to analyze considers the following stress increments,

$$\Delta\bar{\sigma}_1^{(1)} = \alpha,\ \Delta\bar{\sigma}_2^{(1)} = 0 \quad ; \quad \Delta\bar{\sigma}_1^{(2)} = -\beta,\ \Delta\bar{\sigma}_2^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_1^{(3)} = \gamma,\ \Delta\bar{\sigma}_2^{(3)} = \gamma \tag{2}$$

where now it is considered, $\alpha = 150\ MPa$, $\beta = -700\ MPa$ and $\gamma = 1000\ MPa$.

Figure 1.5 is a representation of the load path for the tension-only damage model. The first stage is an uniaxial elastic load, the second one presents a biaxial compressive loading. Here it is important to note that, due to the characteristics of this only-tension model, this step is always going to be within the elastic regime. Therefore, the values of β is somewhat irrelevant if it is negative, in the way that the material response will be elastic. In the final load step, as the point moves outwards the interior by means of a biaxial tensile loading the elastic domain expands as the Karush/Kuhn/Tucker mathematically state.

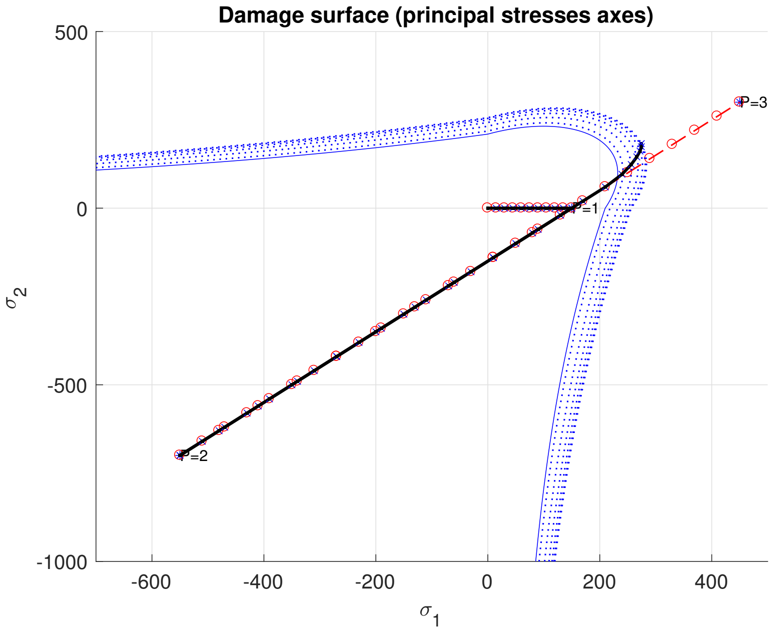


Figure 1.5: Plot showing the path in the stress space for the second case of analysis for the tension-only dame model. Blue dotted lines, represent the evolution of the elastic surface.

In Figure 1.6, a representation of the obtained stress-strain curve is shown. As stated before, the first stage is an elastic load, thus this reads as a straight line with slope E in the graph (black line). The second stage is a pure compressive state which is know to be always elastic for this damage model. Again, this gives an straight line in the stress-strain plot but we need to take into account that the slope is not same as before as the relationship between stresses and strains has now changed. Now the important part appears when in the third load stage the point moves all the way outwards the elastic domain. Then, it undergoes some hardening (in this case an exponential law has been chosen). When the material point tries to go outwards the domain as a result of this loading, the internal variable evolves and this is shown in Figure 1.7. Note that the points in the stress-strain curve which represent the hardening are exactly those which correspond to the change of the internal variable. Previous to that change, the internal variable has the initial value $r_0 = \sigma_Y/\sqrt{E}$, which in fact can be seen as a mechanical property of the material also obtained in the laboratory.

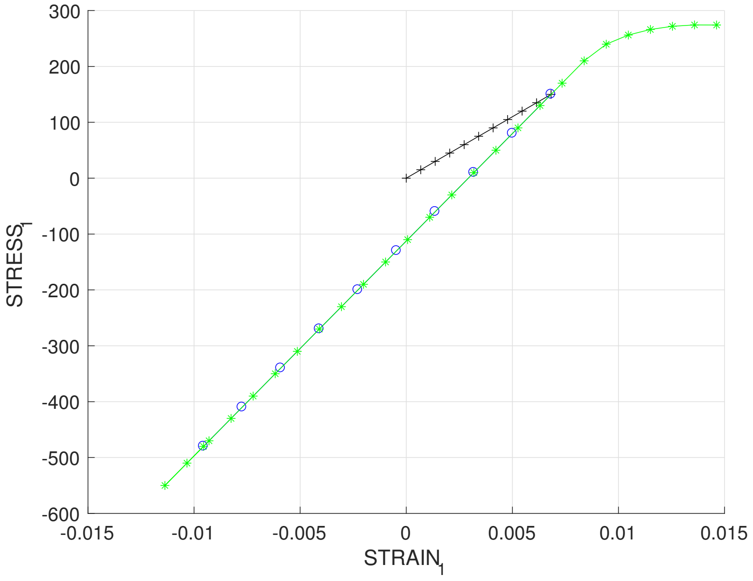


Figure 1.6: Stress-strain curve obtained for the tension-only damage model when the material undergoes a uniaxial/biaxial path. It is important to note that the compressive part is purely elastic.

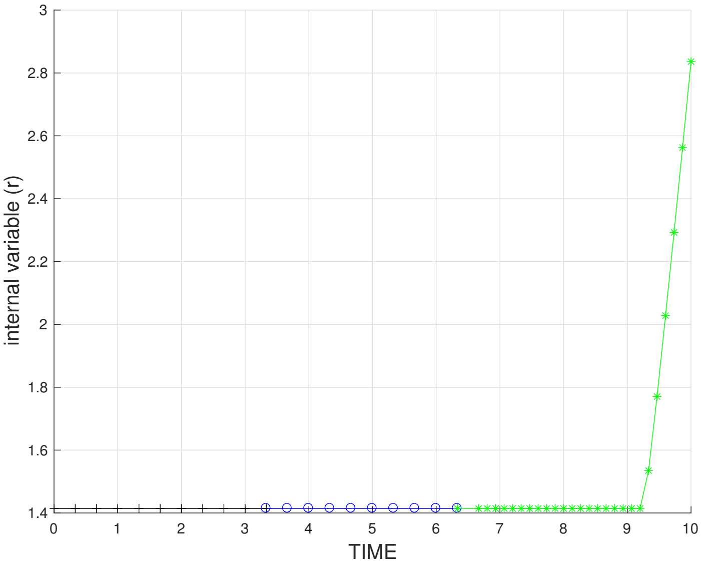


Figure 1.7: Plot of the evolution of the internal variable r. As expected, changes is the value of r appear as the material undergoes the third loading stage which results in a hardening process.

It is also important to note that, had we chosen a first load step so that the point exceeds the limit σ_Y, the material would suffer hardening and even though the internal variable would have already evolved in this first load step, it would have remained constant through the compressive state and finally change again for the final state as the point tries to go out of the domain.

Let's now analyze the behaviour of this path when selecting the non-symmetrical damage model. A representation of the path in the stress space with the corresponding elastic surface is represented in Figure 1.8. As always the blue dotted lines account for the evolution of the elastic surface in order to fulfill the consistency conditions.

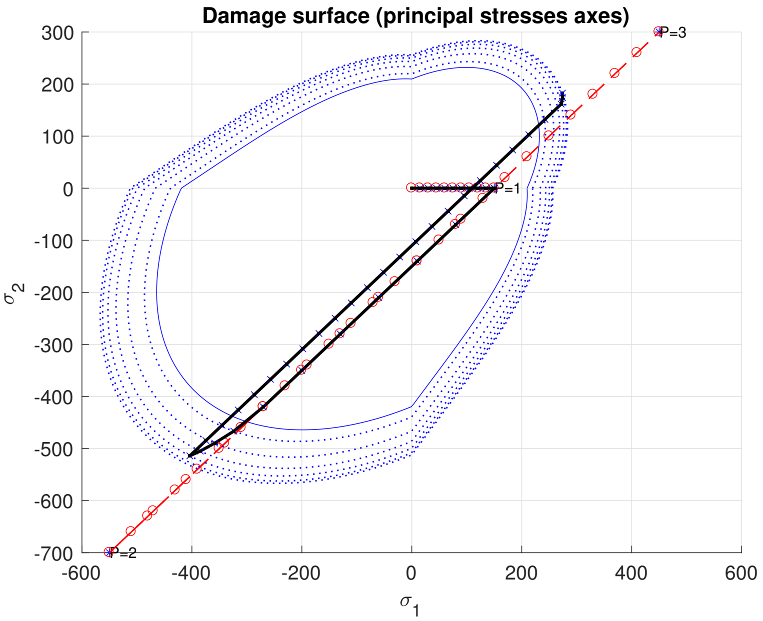


Figure 1.8: Graphical representation of the load path in the stress space for the uniaxial/biaxial case considering the non-symmetrical damage model.

In the next Figure 1.9, we show the results for the stress-strain curve. It is important to note that now, the compressive stage appears with a different slope than that in Figure 1.4 in which both straight lines corresponding to the first and second load/unload stages were coincident. Again, the material suffers damage in the compressive loading as we have introduced a value for β which lies outside the elastic domain. Thus the material undergoes a hardening process. Then, the domain evolves so as to keep the point on the boundary and the third load stage is elastic until the point reaches the limit value in tension and the internal variable would show an evolution again.

In order to show some insight in this fact, Figure 1.10 offers a representation of the evolution of the hardening (stress like) variable, which in fact controls the size of the elastic surface in the stress space. As shown in the plot, variations of the hardening variable q appear when there is an evolution of the domain, i.e. an evolution of the internal variable.

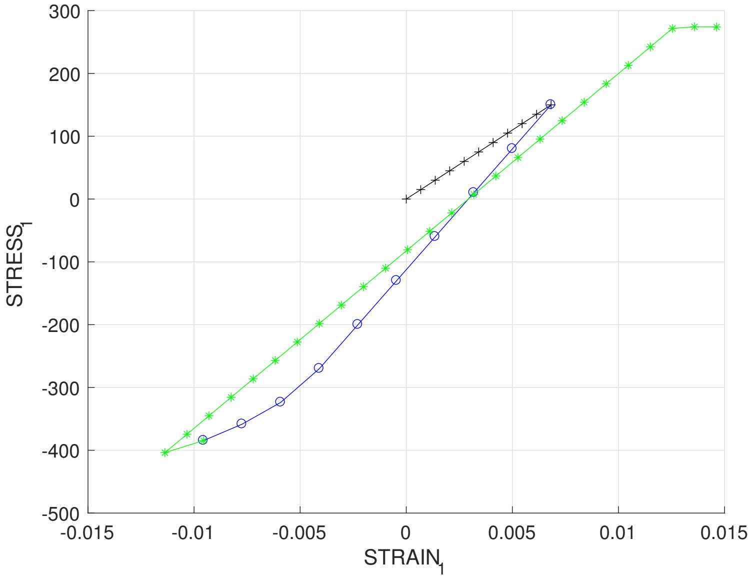


Figure 1.9: Stress-strain curve for the uniaxial/biaxial case of analysis when considering the non-symmetrical damage model.

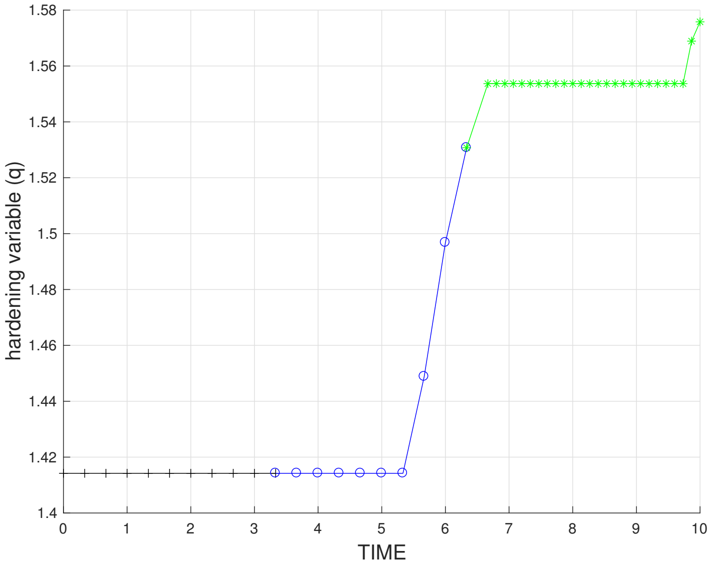


Figure 1.10: Evolution of the hardening (stress-like) variable for the non-symmetrical damage model and the second case of analysis.

1.3 Case 3: complete biaxial path

The last case of analysis consists in a complete biaxial loading/unloading path of the form,

$$\Delta\bar{\sigma}_1^{(1)} = \alpha,\ \Delta\bar{\sigma}_2^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_1^{(2)} = -\beta,\ \Delta\bar{\sigma}_2^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_1^{(3)} = \gamma,\ \Delta\bar{\sigma}_2^{(3)} = \gamma \tag{3}$$

where we are going to use the same values of α, β and γ as for the previous case. The difference of this case, apart from the fact that is completely biaxial, is that we are now going to consider a negative value of the hardening modulus, $H = -0.25$. Thus, we will simulate know a process involving a softening (linear).

Let us examine first the results for the tension-only damage model. Figure 1.11 displays the loading path. As expected, the dotted blue lines are now in the interior of the initial domain, since as we have considered a negative value of the hardening modulus H, now the domain undergoes a contraction so as to keep a point on the border.

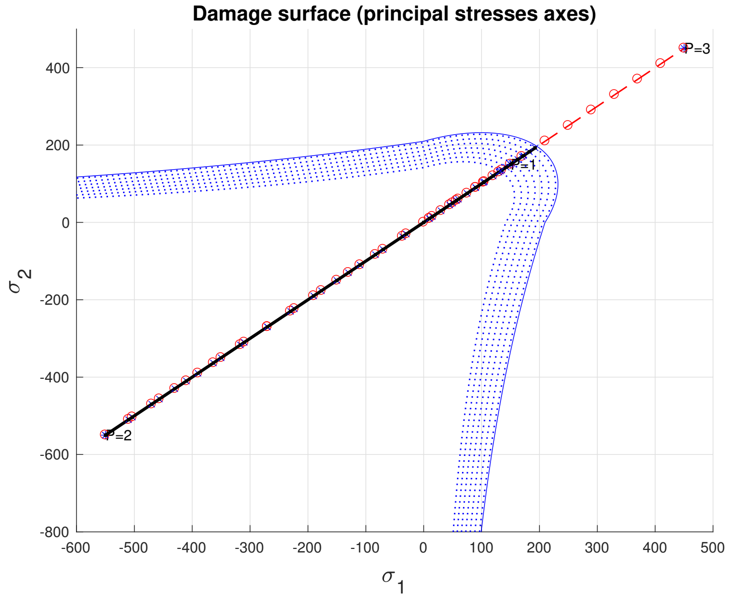


Figure 1.11: Representation of the loading path for the complete biaxial case considering the tension-only damage model. As softening has been considered for the computations, the elastic domain reduces its size in order to keep a point on the border.

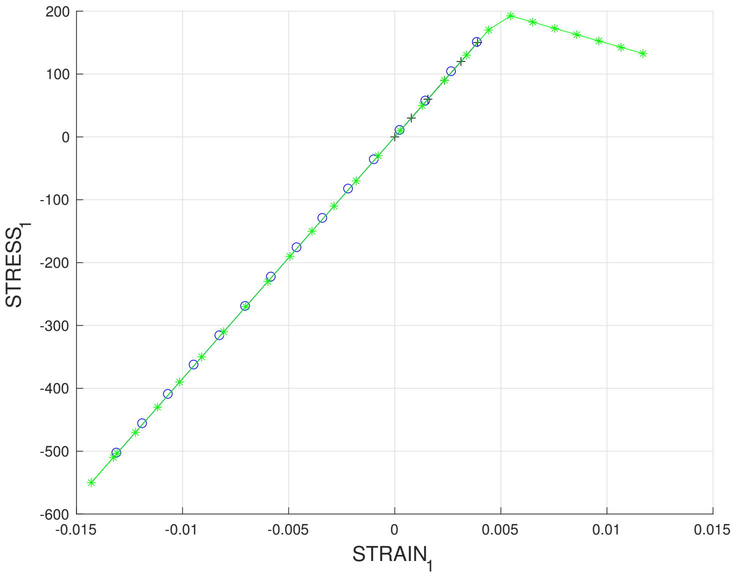


Figure 1.12: Stress-strain curve for the complete biaxial loading path when considering the tension-only damage model. Once the material leaves the elastic behavior, a decrease in the stresses appear as a consequence of the softening process.

Figure 1.12 is a representation of the stress-strain curve for this model. As we can see, this plot does not differ in shape much with the one obtained for the complete uniaxial case, (see Figure 1.2). This is because the load path also follows an straight line and the compressive part is purely elastic, as already discussed for this tension-only model. The difference appear since now we have decided to give the material a negative hardening modulus. Therefore, it goes through softening and the stress is reduced once the material behaves non-elastically.

Finally, Figures 1.13 and 1.14 represent the loading path and the stress-strain curve for the non-symmetrical damage model, respectively.

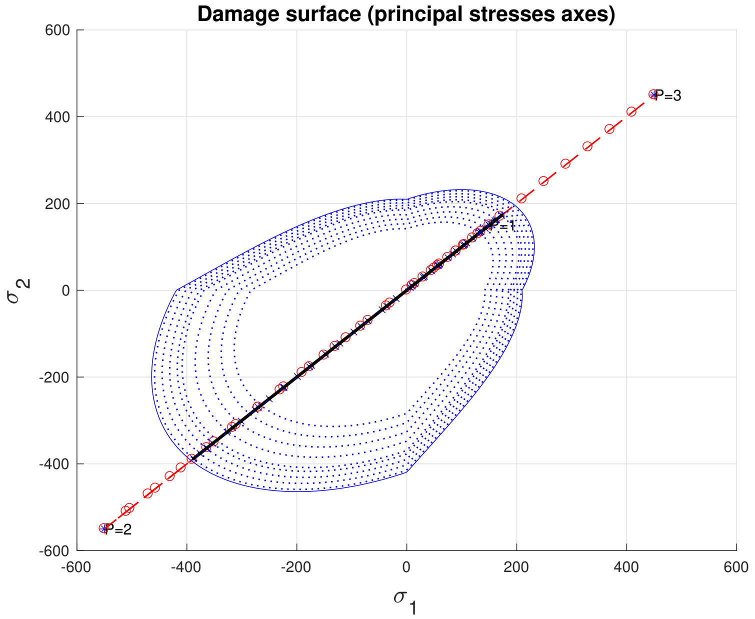


Figure 1.13: Plot showing the loading path for the third case of analysis considering the non-symmetrical damage model.

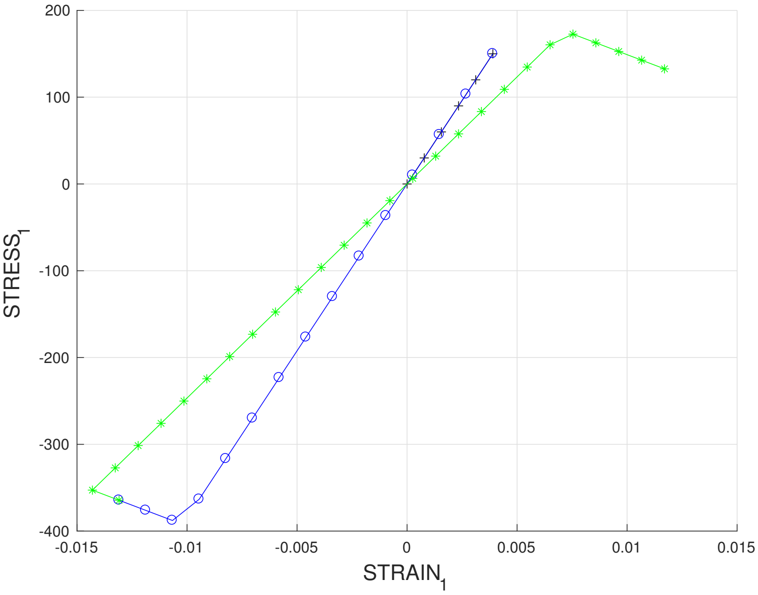


Figure 1.14: Stress-strain curve for the complete biaxial loading path when considering the non-symmetrical damage model.

The loading path is completely represented on a straight line, consisting on a elastic tensile load, a compressive load with and increment which lies outside the reference surface and a compressive unloading with the same characteristic. The subsequent dotted blue lines, represent how the elastic domain contracts. It is important to realize now how these facts are shown in the stress-strain curve. The first and second stages are in a straight line of the same slope, until we overpass the limit of the elastic surface. Then the material suffers damage, and since softening is considered this time, the stress become less negative and the given material would fail easily in compression. Now for the third step of the load path, the points would cross the elastic domain with an slope lower than previously since the material has already been damaged.

2 Rate dependent model

In this second section we are going to asses the correctness of the implementation of the rate dependent damage model. One of the key properties of the viscodamage model is that now, even though the strain tensor $\boldsymbol{\varepsilon}$ remains constant, the stress tensor $\boldsymbol{\sigma}$ can still change, i.e. time is now an independent variable of the problem. Geometrically speaking, now stress/strain states which lie outside the elastic domain are feasible.

In the subsequent subsections, we are going to analyze the effects of the different new parameters that are to be taken into account for the implementation. We will present first the influence of the viscosity parameter, then the dependency of the strain rate and the α parameter (for the α integration method) are studied. Finally, a discussion on the evolution of the algorithmic tangent operator is provided.

For all the plots presented from now on, the symmetric tension-compression model is considered.

2.1 Effect of viscosity on stress-strain curve

In Figure 2.1 a graphical representation of the influence of the viscosity parameter on the point response on the stress-strain curve is shown. A uniaxial load path has been considered for simplicity, similar as in Section 1.1. As we can see, the part corresponding to the elastic behaviour of the material remains unchanged. In the inelastic region, a higher value of viscosity implies a higher value of the stresses as discussed in the lecture. This fact is completely consistent with the damper model, which required a higher force for a higher material viscosity, η.

2.2 Effect of strain rate on stress-strain curve

Figure 2.2 shows a plot containing, for fixed values of viscosity and integration parameters, several stress-strain curves with different total times of simulation for a same loading path. This is in fact the way of simulate a change in the strain rate. As it was regarded for the viscosity dependency, the elastic part remains equal for all the cases. When we go beyond the yield stress σ_Y, i.e. when damage starts triggering, a directly proportional relationship between stress and the total time of simulation is observed. Higher values of strain rate (loading path applied in less time) derive into higher values of stress. This fact clearly explains the dependency of stress not only on the strains but also on the strain rates, which is the key aspect of the viscodamage model.

Note: in order to completely asses the correctness of the implementation, we have checked that, provided a material with null viscosity, $\eta = 0$, all the stress-strain curves for different times of simulation are exactly equal, in other words, the results of the rate independent model are recovered.

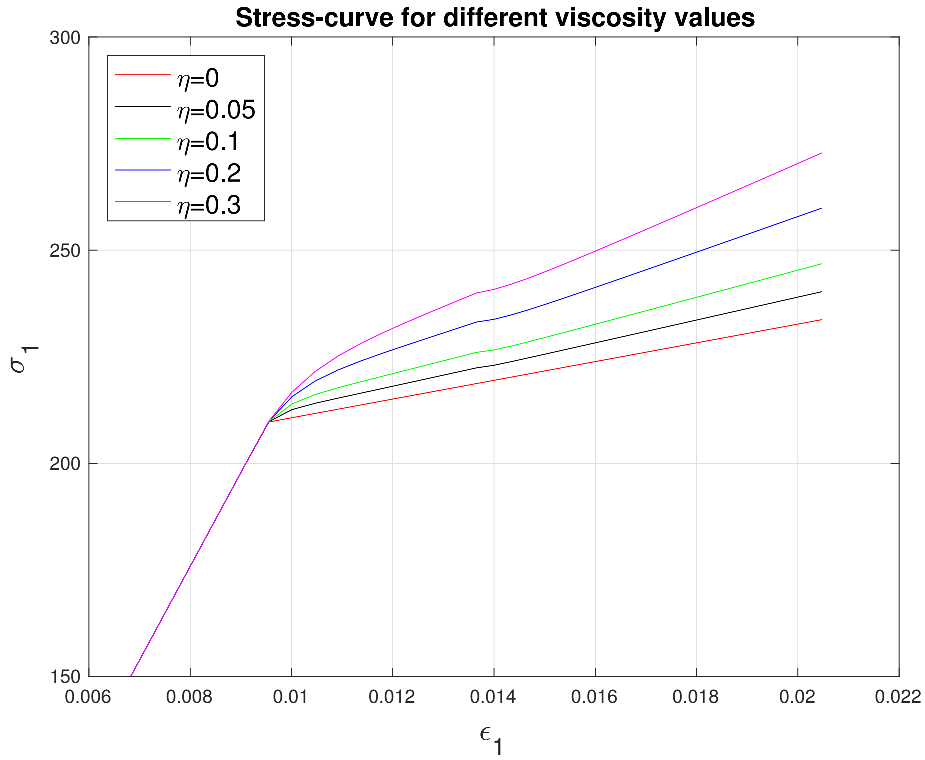


Figure 2.1: Influence of the material viscosity on the stress-strain curve for a uniaxial load path. The elastic part remains unchanged for all the cases.

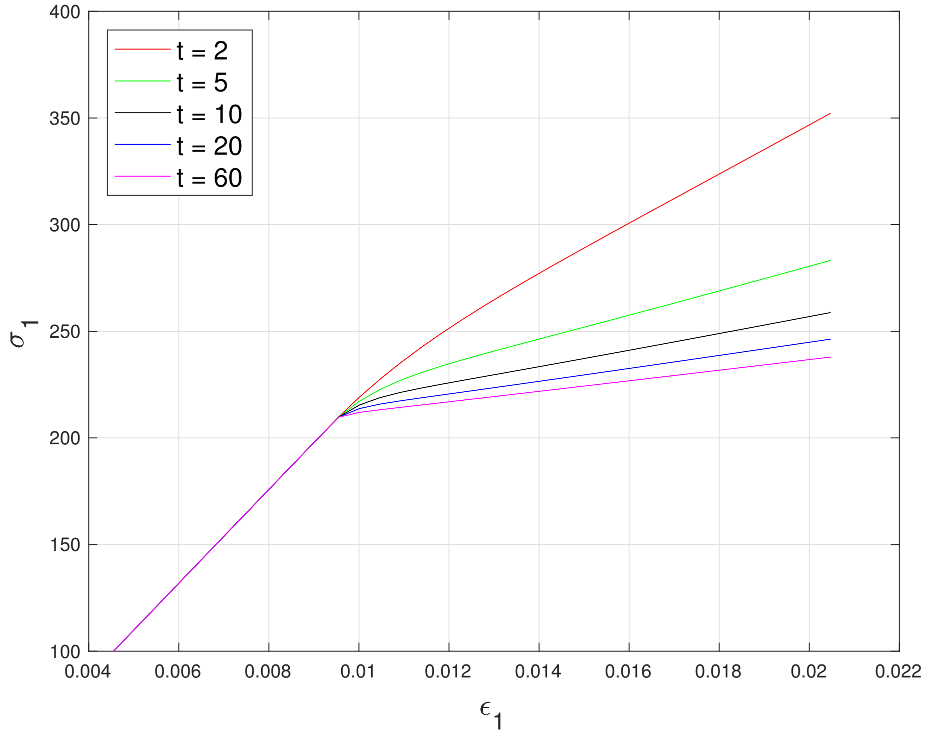
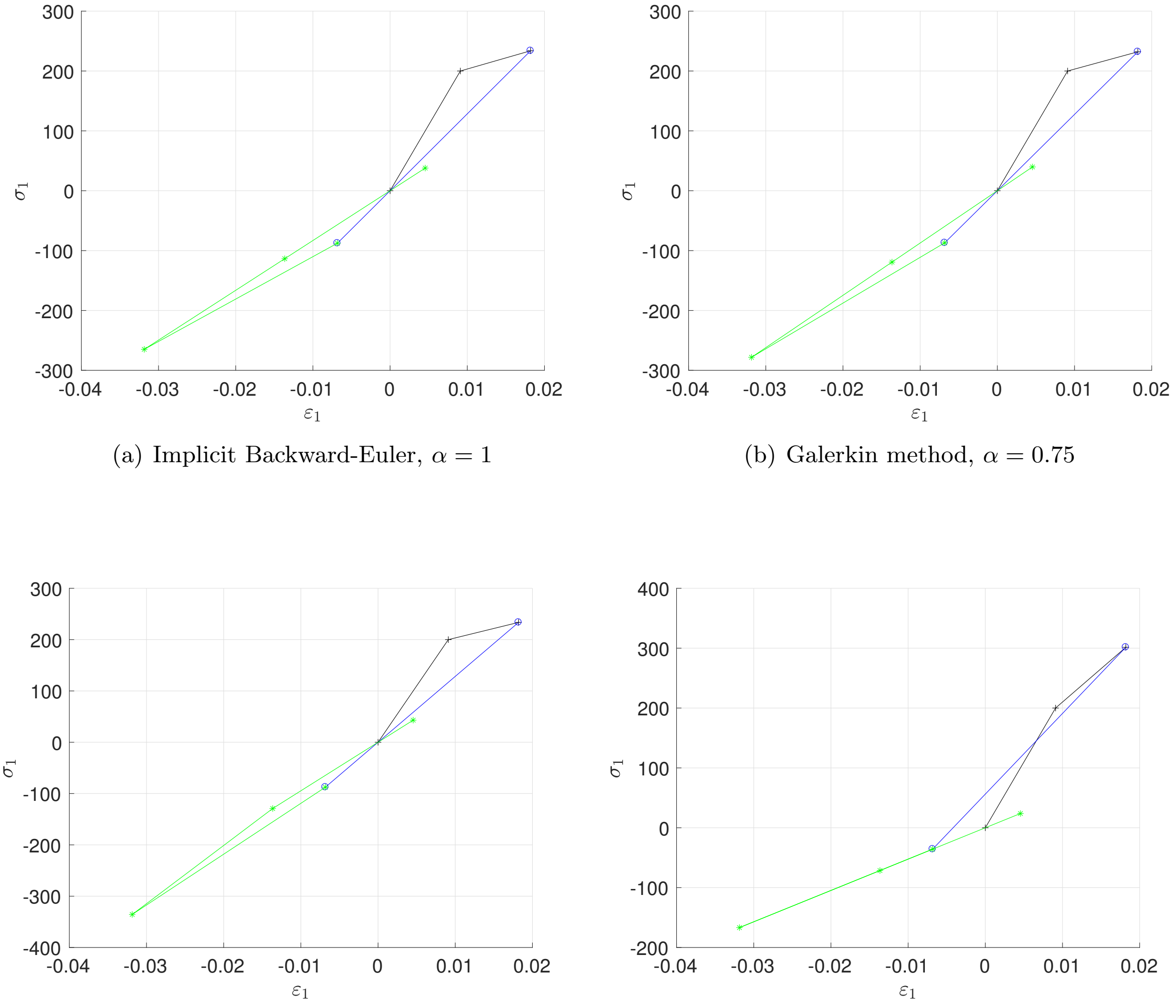


Figure 2.2: Representation of the effect of the strain rate on the stress-strain curve for an uniaxial load path. Higher strain rates on the loading turn into a higher stress in the inelastic part of the curve.

2.3 Effect of α integration parameter on the stress-strain curve

Let us now analyze the dependency of the solution on different integration schemes. Several values of the integration parameter α were considered, $\alpha = 0$ (explicit Forward-Euler scheme), $\alpha = 0.25$, $\alpha = 0.5$ (implicit Crank-Nicholson scheme), $\alpha = 0.75$ (Galerkin scheme), $\alpha = 1$ (implicit Backward-Euler scheme). All these schemes are first order accurate $\mathcal{O}(\Delta t)$, while the Crank-Nicholson provides second order accuracy $\mathcal{O}(\Delta t^2)$. As we now, values of α lower than 0.5 can induce spurious solutions and stability problems arise when using inappropriate time steps. For this analysis, we have considered an uniaxial loading path for simplicity, linear hardening, a fixed viscosity value $\eta = 0.2$, a total time for the simulation of $t = 25$ s and 2 steps per stage for the integration ($istep = 2$).

Figure 2.3 contains the stress-strain curves obtained for different values of α. For values higher or equal than 0.5, the solution seems to have physical meaning. The loading starts from point $(0, 0)$ and as we go beyond the yield stress σ_Y, the material undergoes a hardening. Then the material is elastically unloaded as the point crosses the elastic surface. Again, as we introduced an increment for the stress exceeding the limit, the material suffers damage but now in the compressive part. Finally, the material is loaded again. On the other hand, when the problem is to be solved using an explicit scheme, we observe the presence of non meaningful solutions specially in the compressive part where now points on the third load stage appear in the same line which is something totally unrepresentative. Therefore, even though explicit schemes have lower computational cost, they should be in general avoided as they turn into stability problems and spurious solutions.

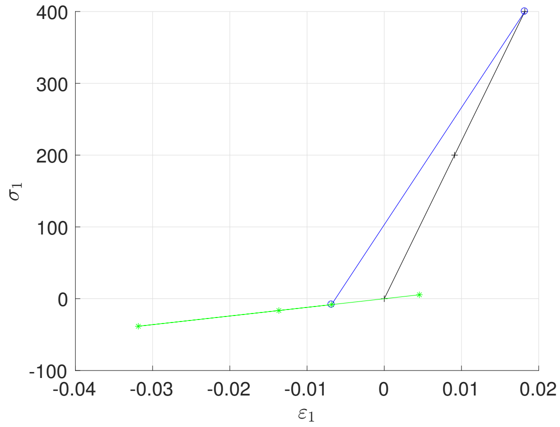


(a) Implicit Backward-Euler, $\alpha = 1$

(b) Galerkin method, $\alpha = 0.75$

(c) Crank-Nicholson, $\alpha = 0.5$

(d) Explicit scheme, $\alpha = 0.25$



(e) Explicit Forward-Euler, $\alpha = 0$

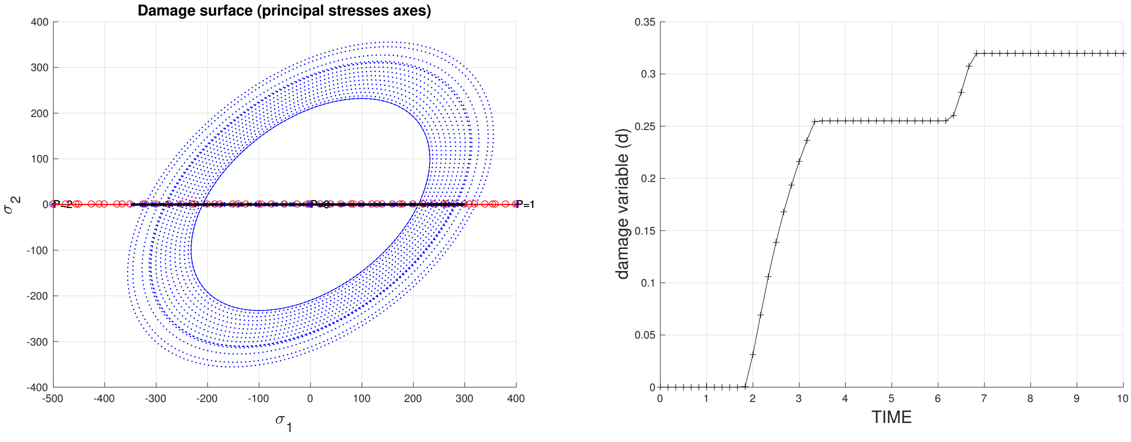
Figure 2.3: Stress-strain curve obtained for different integration schemes. (a) Backward-Euler, (b) Galerkin method, (c) Crank-Nicholson, (d) $\alpha = 0.25$, (e) Forward-Euler. As we can see, schemes with $\alpha < 0.5$ produce spurious solutions. Total time of integration $t = 25s$, $istep = 2$.

2.4 Tangent and algorithmic constitutive operators

Finally, we are going to discuss some facts regarding the algorithmic and analytic tangent constitutive operators. Generally speaking, the tangent constitutive operator expresses a relationship between the derivative of the stresses with respect to the derivative of the strains. Let us first recall the importance of the algorithmic tangent constitutive operator on a complete FEM code.

At the end of the day, what it is pretended to be solved, is a problem with a non-linear constitutive equation, being the solution of the problem (after a finite element discretization) the displacements of the nodes of a certain solid. Generally speaking, the problem of the non-linearity of the equation is overcome through a process of linearization using the Newton method, which will return the so called tangent stiffness matrix, which varies along the deformation process of the solid. This stiffness matrix needs to be computed in terms of the algorithmic tangent constitutive operator. In the case of computing this matrix by using the analytical operator, the convergence of the method would not be optimal.

Figure 2.4 (a) shows the loading path chosen for this analysis, where the dotted blue lines represent the evolution of the elastic surface. In Figure 2.4 (b) the evolution of the damage variable is presented. As we can see, it remains constant while elastic processes take place and evolves once damage appears in the material as the point tries to escape the elastic domain. Also, as expected, this figure shows the monotonically increasing character of d.

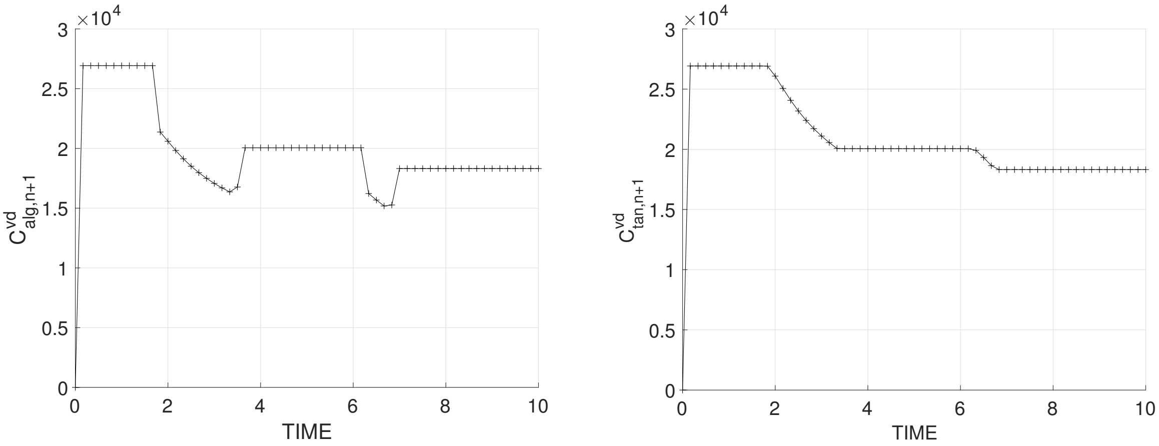


(a) (b)

Figure 2.4: (a) Uniaxial loading path chosen to analyze the evolution of the tangent constitutive operators and (b) evolution of the damage variable in the process.

Figure 2.5 displays the evolution of both analytical and algorithmic tangent constitutive operators when the problem is solved using the so called Crank-Nicholson scheme. Recall that the algorithmic operator is the one relating the derivatives of the current values of the stresses with respect to the current values of the strains. The values of both operators start being equal, as they coincide with those of the tensor of elastic constants $\mathbb{C}$. Then, once damage appears the values start to change. Note how the values of the algorithmic operator become lower as it considers also a negative term with the effect of the viscosity, the α integration parameter and the time interval. Once the point goes back through the elastic domain, the values match again.

Figure 2.6 represents also the evolution but when the problem is solved using the implicit Backward-Euler scheme. Similar results are displayed but now, as α is also higher, the negative part on the equation to compute the algorithmic operator becomes more important are therefore the values of the graph are lower.



(a) (b)

Figure 2.5: Evolution of the algorithmic (a) and analytical (b) tangent constitutive operators for $\alpha = 0.5$.

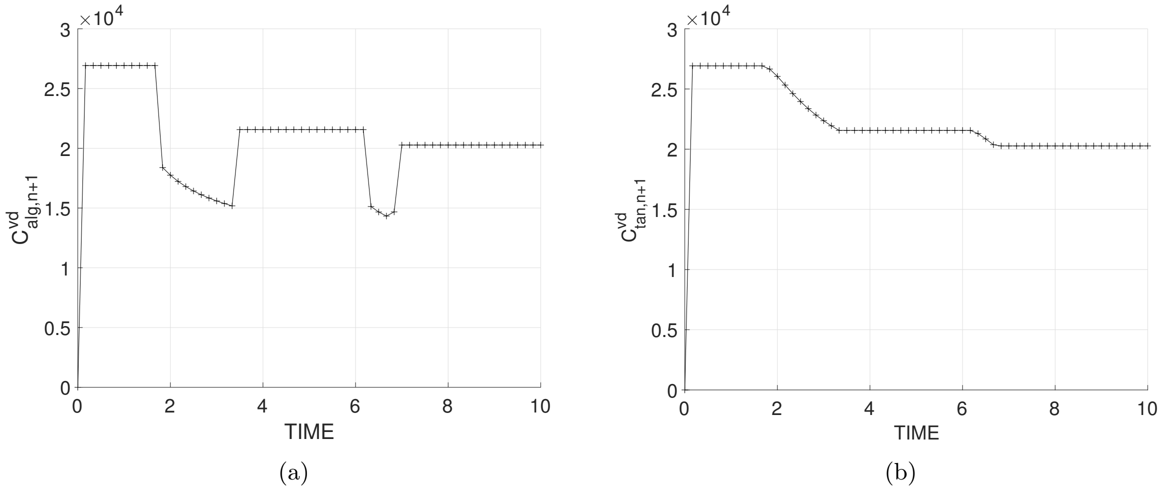


Figure 2.6: Evolution of the algorithmic (a) and analytical (b) tangent constitutive operators for $\alpha = 1$.

Finally, the key aspect that needs to be checked in order to prove the correctness of the implementation is that when we consider the explicit Forward-Euler scheme, i.e. $\alpha = 0$, the values of both constitute operators have to match. This fact is precisely collected in Figure 2.7. It is also important to note that, mathematically speaking, both operators tend to match as we consider lower and lower interval of integration. In the limit, $\Delta t = 0$, they completely match.

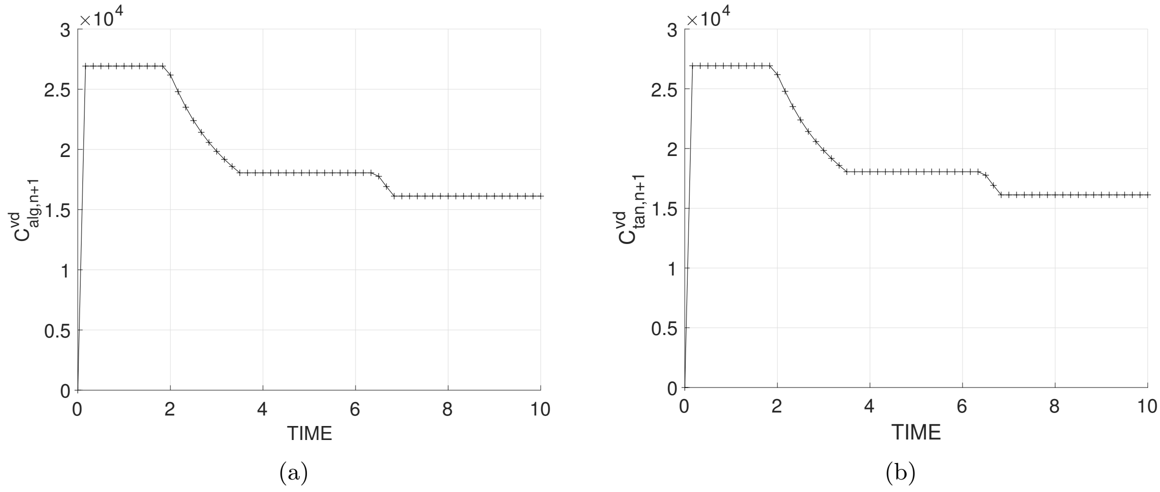


Figure 2.7: Evolution of the algorithmic (a) and analytical (b) tangent constitutive operators for the implicit scheme, $\alpha = 0$. As expected, the values for both operators coincide.

The Newton Raphson method needs to obtain the tangent stiffness matrix which varies along the deformation process of the solid, computed in terms of the algorithmic tangent constitutive operator (if we introduce the analytical one the convergence is not optimal in such case, the Newton Raphson algorithm does not convergence at the maximum rate that it should be, i.e. quadratic.)

A Modified routines

Finally in this appendix, we include all the modified routines used to succesfully complete this assignment. Apart from the given scripts and functions, a new function has been programmed. This function, referred here as *rmap_ dano_ visc.m* was developed to update the values of the Cauchy stress, internal variable, hardening variable and compute the algorithmic constitutive operator, which would be needed in a complete FEM code.

The modified routines, appearing in order are: *damage_ main.m*, *rmap_ dano1.m*, *rmap_ dano_ visc.m*, *dibujar_ criterio_ dano1.m* and *Modelos_ de_ dano1.m*.

- *damage_ main.m*

```
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','C_alg','C_tan'};

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 = 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)
%vartoplot{i}(4) = C_alg(1,1);
```

```
%vartoplot{i}(5) = C_tan(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"
        % ------------------------
        eps_n1 = strain(i,:) ;
        %       DAMAGE MODEL
        if viscpr ==1
            eps_n = strain(i-1,:);
            [sigma_n1,hvar_n,aux_var,C_alg, C_tan] = rmap_dano_visc(eps_n,eps_n1,delta_t,hvar_n,Eprop,ce);
        else
        [sigma_n1,hvar_n,aux_var] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n);
        end
        % 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)
        vartoplot{i}(4) = C_alg(1,1);
        vartoplot{i}(5) = C_tan(1,1);
    end
end
```

- *rmap_dano1.m*

```
function [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n1,hvar_n,
   Eprop,ce,MDtype,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) ;

% Iinitializing
r0 = sigma_u/sqrt(E);
zero_q=1.d-6*r0;
inf_q = 2*r0 - zero_q; % Define q_inf

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

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

if(rtrial > r_n)
    %   Loading
    fload=1;
    delta_r=rtrial-r_n;
    r_n1= rtrial   ;
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        % Exponential
         if H>0 %hardening
          dqdr = H*((inf_q - r_n)/r_n)*exp(H*(1-(r_n1/r_n)));
          q_n1 = q_n + dqdr*delta_r;
        elseif H<0 %softening
           dqdr = H*((r_n-inf_q)/r_n)*exp(H*(1-(r_n1/r_n)));
           q_n1 = q_n - dqdr*delta_r;
        end
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1>inf_q) %Acotar r por arriba
        q_n1 = inf_q;
    end
else
    % Elastic load/unload
    fload=0;
    r_n1= r_n   ;
    q_n1= q_n   ;
end

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

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

%Updating historic variables
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;

% Auxiliar variables
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
```

- *rmap_dano_visc.m*

```
function [sigma_n1,hvar_n1,aux_var,C_alg,C_tan] = rmap_dano_visc(eps_n,eps_n1,delta_t,hvar_n,Eprop,ce)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                              %
%           RATE DEPENDENT-VISCO DAMAGE MODEL                  %
%                                                              %
%                                                              %
% [sigma_n1,hvar_n1,aux_var, C_tan] = rmap_dano1_visc          %
%                                     (eps_n1,hvar_n,Eprop,ce) %
%                                                              %
%                                                              %
%    INPUTS:       eps_n    strain (almansi)    step n         %
%                  eps_n1   strain (almansi)    step n+1       %
%                               vector R4     (exx eyy exy ezz)%
%                  delta_t  time interval for integration      %
%                  hvar_n   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_n1(6)  Internal variable , step n+1    %
%                  hvar_n1(5)  Hardening variable , step n+1   %
%                  aux_var     Auxiliar variables for plot     %
%                  C_alg       Algorithmic tangent operator    %
%                  C_tan       Analytical tangent operator     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Variables
hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5)  ;
eta = Eprop(7);
alpha = Eprop(8);

% Initializing
r0 = sigma_u/sqrt(E);
zero_q=1.d-6*r0;
q_inf=2*r0-zero_q; %Define q_infinite

% Define damage surface
% Symmetric tension-compresion model
```

```
rtrial_n1 = sqrt(eps_n1*ce*eps_n1');
rtrial_n = sqrt(eps_n*ce*eps_n');

% Integration
rtrial_alpha = (1-alpha)*rtrial_n+alpha*rtrial_n1;

if(rtrial_alpha > r_n)
    fload=1;
    %Loading
    delta_r=rtrial_alpha-r_n;
    r_n1= (eta-delta_t*(1-alpha))/(eta+alpha*delta_t)*r_n + ...
          (delta_t/(eta+alpha*delta_t))*rtrial_alpha  ;
    if hard_type == 0
        %  Linear hardening
        q_n1= q_n+ H*delta_r;
    else
        % Exponential hardening
        dqdr = H*(q_inf - r_n)/r_n*exp(H*(1-r_n1/r_n));
        q_n1 = q_n + dqdr*delta_r;
    end
    % Restrict value to q_infinite if needed
    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1 > q_inf)
        q_n1=q_inf;
    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';

% Algorithmic tangent operator
if (rtrial_alpha > r_n)
sigma_barra = ce*eps_n1'; %effective stress
C_alg = (1-dano_n1)*ce + ...
    (alpha*delta_t)/((eta+alpha*delta_t)*rtrial_n1)*...
    ((H*r_n1-q_n1)/r_n1^2)*(sigma_barra'*sigma_barra);
    %Only linear case is considered
else
    C_alg = (1-dano_n1)*ce ;
end

% Analytic tangent operator
C_tan = (1-dano_n1)*ce ;
```

```

% Updating historic variables
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;

% Auxiliar variables
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
```

- *dibujar_ criterio_ dano1.m*

```matlab
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----->tension ultima             %
%            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
    limitINF = -pi/2*0.99;
    limitSUP = pi*0.99;
    tetha=[limitINF:0.01:limitSUP];
    % 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)*(m1(i)>0) m2(i)*(m2(i)>0) 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

    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
        tetha_aux = (m1(i)*(m1(i)>0) + m2(i)*(m2(i)>0))/(abs(m1(i)) + abs(m2(i))) ;
        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))]')/(tetha_aux + ((1 - tetha_aux)/n)
```

```
               );

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

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

end
return
```

- *Modelos_ de_ dano1.m*

```
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
rtrial = sqrt(eps_n1.*(eps_n1>0)*ce*eps_n1');

elseif (MDtype==3)  % Non-symmetric
s_n1 = ce*eps_n1';
s1=s_n1(1); s2=s_n1(2);
tetha_aux = (s1*(s1>0) + s2*(s2>0))/(abs(s1)+abs(s2));
rtrial = (tetha_aux +(1 - tetha_aux)/n)* sqrt(eps_n1*ce*eps_n1')
   ;
end
return
```