Polytechnic University of Catalonia - CIMNE

Computational Solid Mechanics
Assignment 1

Bruno Aguirre Tessaro

April 8, 2016

1 Rate Independent Models

1.1 Implement in the supplied MATLAB code the integration algorithms (rate independent and plane strain case) for:

1.1.1 The continuum isotropic damage "non-symmetric tension compression damage" model

The implementation of the "non-symmetric tension compression damage" model was made altering the files $Modelos\_de\_dano1.m$ and $dibujar\_criterio\_dano1.m$ to solve and plot the equations

$$\tau_\varepsilon = \left(\theta + \frac{1-\theta}{n}\right)\sqrt{\boldsymbol{\varepsilon} : C : \boldsymbol{\varepsilon}} \qquad \theta = \frac{\sum_1^3 \langle \bar{\sigma}_i \rangle}{\sum_1^3 |\bar{\sigma}_i|},$$

where τ_σ, n, $\boldsymbol{\sigma}$, C and $\bar{\sigma}_i$ are respectively the stress norm, ratio between compression strength and tension strength, the stress tensor, the constitutive operator and the principal stresses. This kind of model is appropriate for materials for materials that behave differently on tension an compression, like concrete.

1.1.2 The "tension-only" damage model.

The implementation of the the "tension-only" follows a similar way as the "non-symmetric" model, with only the files $Modelos\_de\_dano1.m$ and $dibujar\_criterio\_dano1.m$ being altered. The equation to be altered is

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

where $\bar{\boldsymbol{\sigma}}^+$ is the Mcauley bracket of the principal stresses. This kind of model is used when a material ideally can only fail under tension, or has a significant higher yield stress for tension then compression.

1.2 Implement the following cases for each of those models:

1.2.1 Linear and exponential hardening/softening ($H < 0$ and $H > 0$)

The linear law was already implemented on the code and did not required any modification. The exponential law had to be implemented for hardening and softening by modifying the file $rmap\_dano1.m$ to solve the equation

$$q(r) = q_\infty - (q_\infty - r_0)e^{A(1-\frac{r}{r_0})},$$

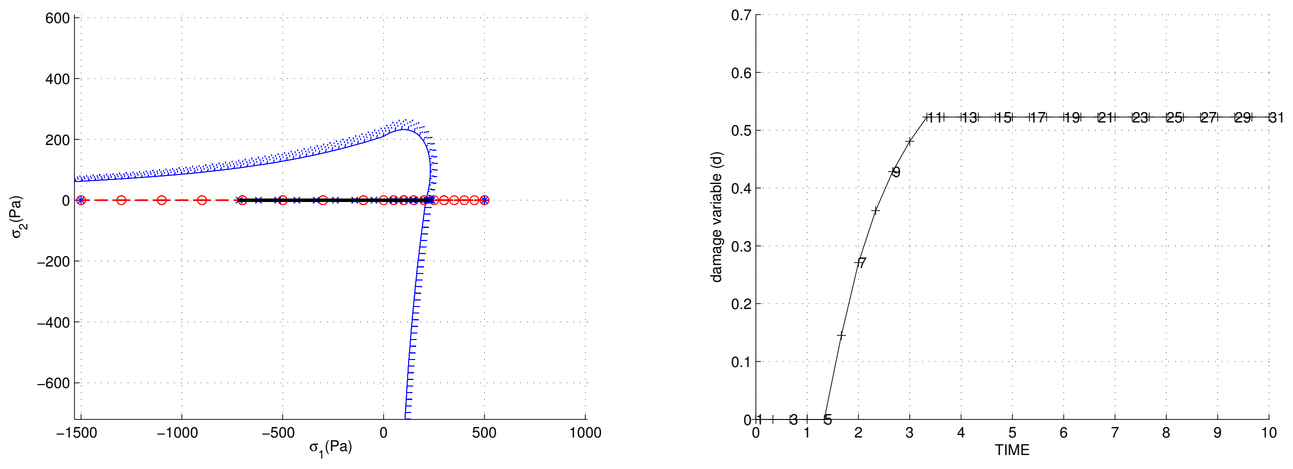
where r, r_0, A and q_∞ are respectively the internal variable, the initial value of the internal variable, a positive only parameter and the asymptote of the limiting value of the hardening parameter. This implementation works for both hardening and softening using a conditional, when $q_\infty > r_0$ its a hardening law, and when $q_\infty < r_0$ its a softening law.

1.3 Assess the correctness of the implementation:

Three test cases will be carried out to check the implementation. All the test cases for rate independent models will be done with the linear hardening law, elastic modulus $E = 20000Pa$, poison's ratio $\nu = 0.3$, hardening modulus $H = 0.1$, the yield stress $\sigma_y = 200Pa$ and $n = 3$. The loading parameters chosen are $\alpha = 500Pa$, $\beta = -2100Pa$ and $\gamma = 2100Pa$.

1.3.1 Test Case I : $\Delta\bar{\sigma}_1^{(1)} = \alpha$; $\Delta\bar{\sigma}_1^{(2)} = -\beta$; $\Delta\bar{\sigma}_1^{(3)} = \gamma$; $\Delta\bar{\sigma}_2^{(1)} = 0$; $\Delta\bar{\sigma}_2^{(2)} = 0$; $\Delta\bar{\sigma}_2^{(3)} = 0$

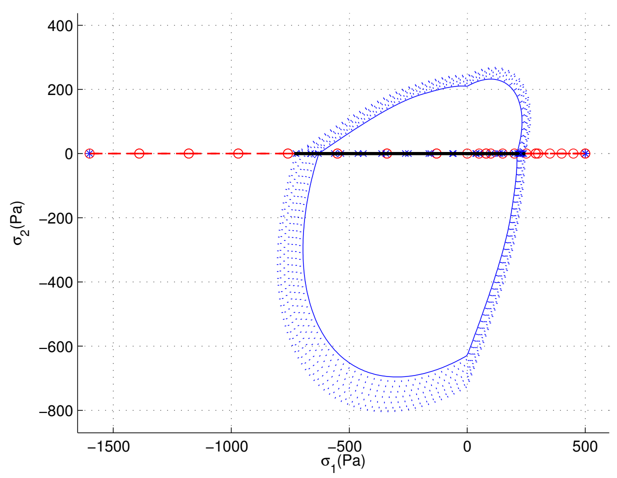
Results obtained for the above load-path using the tension-only model can be seem in Figure 1.1. It is possible to observe that the damage only occurs with the first stress path (uni-axial tensile loading), causing the damage variable to increase. After that no damage occurs so the damage variable remains constant and the damage surface does not increase.



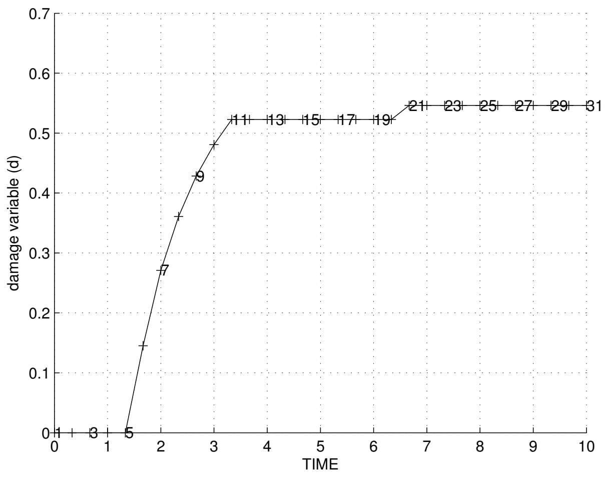
(a) Damage surface (principle stresses) (b) Damage variable evolution

Figure 1.1: Results obtained in Test Case I for the tension-only damage model. In (a) the solid blue line represent initial elastic surface and the doted lines the expansion of the surface due to damage, the red line represents $\bar{\boldsymbol{\sigma}}$ and the black line $\boldsymbol{\sigma}$. In (b) the numbers on the curve represent the time step index of the each point calculated.

The non-symmetric model results can be observed in Figure 1.2. This model admits damage in compression, which is shown in the in Figure 1.2b. It also has damage due to tensile loading in the same way as it occurred in the tension-only model.



(a) Damage surface (principle stresses)



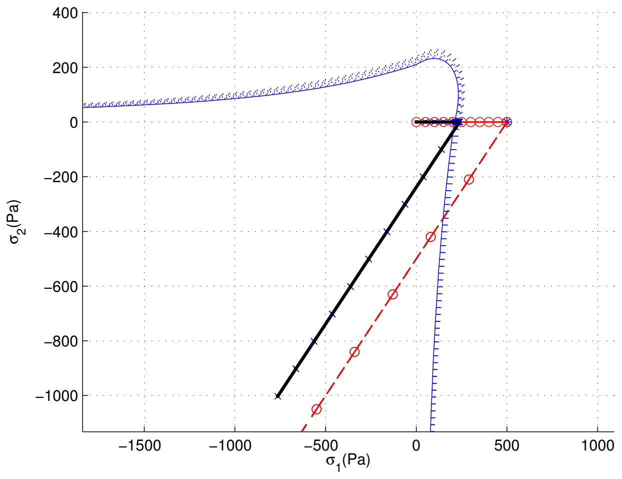
(b) Damage variable evolution

Figure 1.2: Results obtained in the first test for the non-symetric damage model. The plots follows the same configuration as the ones in Figure 1.1

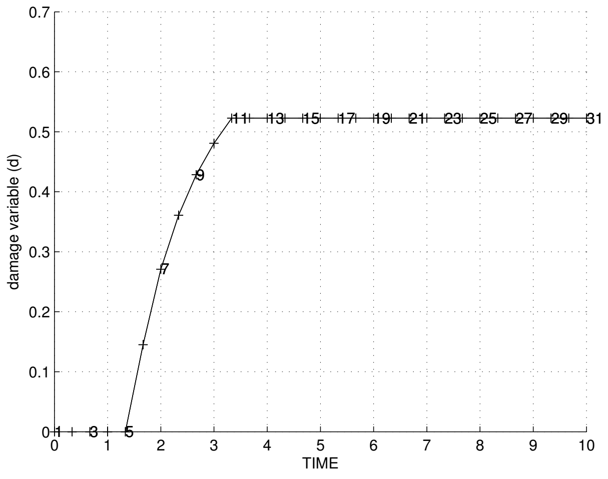
.

1.3.2 Test Case II : $\Delta\bar{\sigma}_1^{(1)} = \alpha$; $\Delta\bar{\sigma}_1^{(2)} = -\beta$; $\Delta\bar{\sigma}_1^{(3)} = \gamma$; $\Delta\bar{\sigma}_2^{(1)} = 0$; $\Delta\bar{\sigma}_2^{(2)} = -\beta$; $\Delta\bar{\sigma}_2^{(3)} = \gamma$

Results obtained for the above load-path using the tension-only model can be seem in Figure 1.3. The behaviour of the damage variable for this load path is the same as seem in Section 1.3.1, because there was no increase in uni-axial tensile loading.



(a) Damage surface (principle stresses)

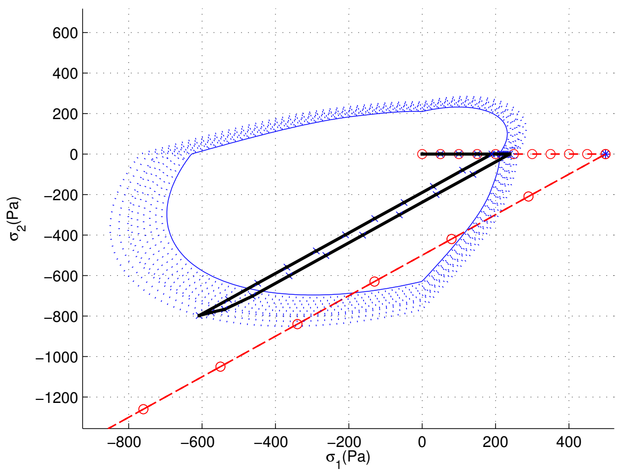


(b) Damage variable evolution

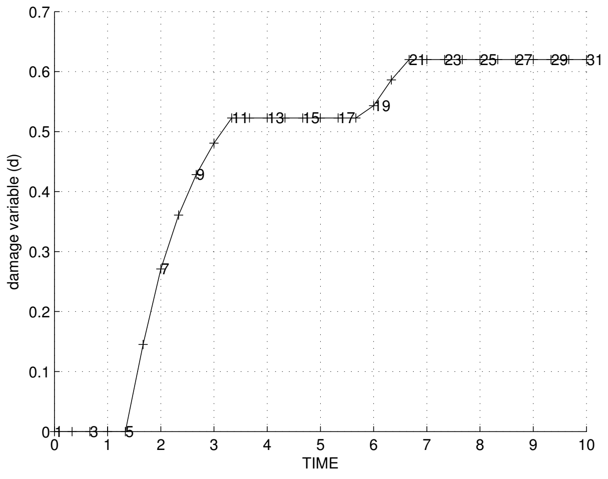
Figure 1.3: Results obtained in Case Test II for the tensile-only damage model. The plots follows the same configuration as the ones in Figure 1.1

.

Results for non-symmetric model can be observed in Figure 1.4. The damage increases in two regions, first one due to the tensile-loading and the second one due to biaxial compressive loading.



(a) Damage Surface (principle stresses)



(b) Damage variable evolution

Figure 1.4: Results obtained in Test Case II for the non-symetric damage model. The plots follows the same configuration as the ones in Figure 1.1

.

1.3.3 Test Case III : $\Delta\bar{\sigma}_1^{(1)} = \alpha$; $\Delta\bar{\sigma}_1^{(2)} = -\beta$; $\Delta\bar{\sigma}_1^{(3)} = \gamma$; $\Delta\bar{\sigma}_2^{(1)} = \alpha$; $\Delta\bar{\sigma}_2^{(2)} = -\beta$; $\Delta\bar{\sigma}_2^{(3)} = \gamma$

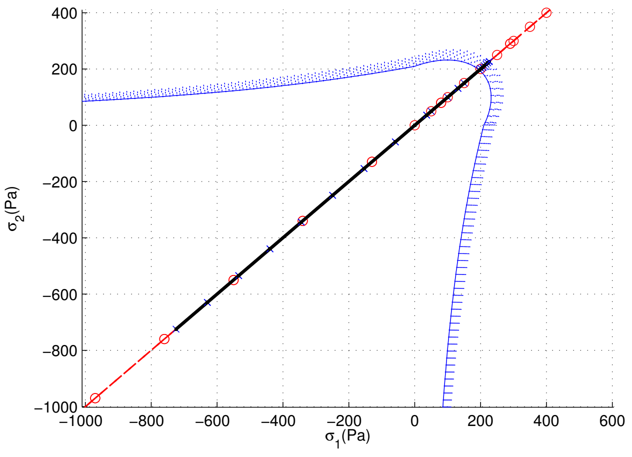
The results obtained for the tension-only damage model using the above load-path can be observed in Figure 1.5. In this case, the damage is caused by biaxial tensile loading, which will reflect a slightly bigger final damage in comparison with the other cases. Once again, there is no damage cause due to compressive loadings.

Results obtained for the tension-only damage model can be observed in Figure 1.6. Again, there is two phases of damage, the first is due to biaxial tensile loading and the second due to biaxial compressive loading.

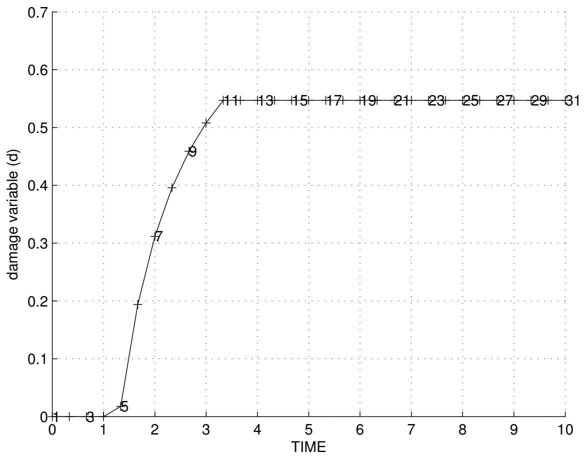
1.3.4 Exponential hardening law assessment

In order to verify the implementation of the exponential hardening/softening law, other test cases will be computed. Figure 1.7 shows the behaviour of the hardening and softening law for the symmetric model. The basic difference between this laws is the shrinking or the expansion of the elastic region in the presence of damage.

To further verify the implementation of this law, a comparison with the linear law will be carried out. Figure 1.8 shows the evolution of the damage variable for each of the three models presented with linear or exponential law. As expected, the laws behave similarly, with a more significant difference in the symmetric case, in which there is a higher increase of the damage surface. The parameter A also plays a big role in the behaviour of the law, and it requires experience to be "tuned" to a optimal value depending on the case needed.



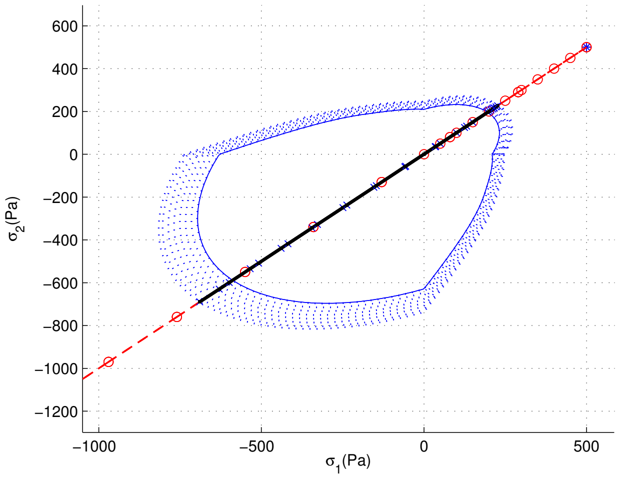
(a) Damage surface (principle stresses)



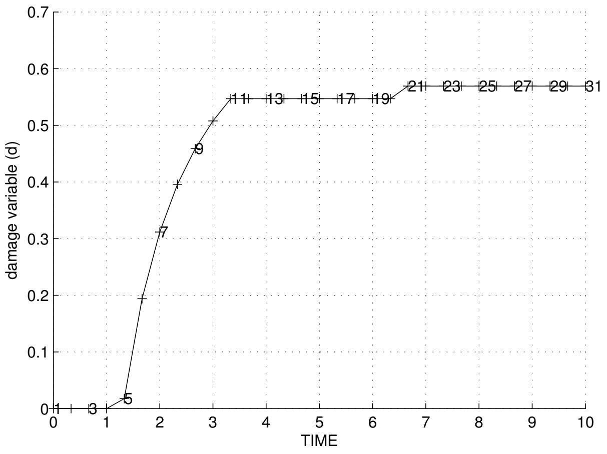
(b) Damage variable evolution

Figure 1.5: Results obtained in Test Case III for the tensile-only damage model. The plots follows the same configuration as the ones in Figure 1.1

.



(a) Damage surface (principle stresses)



(b) Damage variable evolution

Figure 1.6: Results obtained in Test Case III for the non-symetric damage model. The plots follows the same configuration as the ones in Figure 1.1

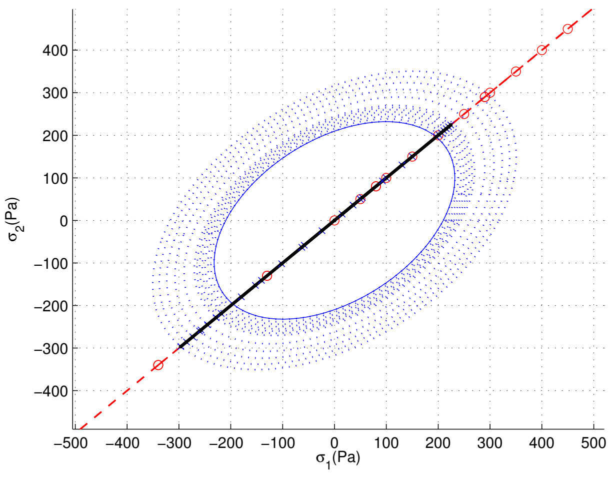
.

2 Rate Independent Models

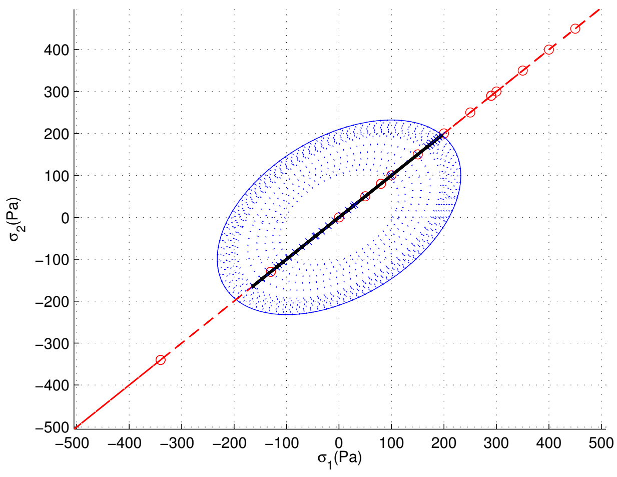
2.1 Implement in the supplied MATLAB code the integration algorithm (plane strain case) for the continuum isotropic visco-damage "symmetric tension-compression" model.

The rate-dependent model required a more involved modification of the MATLAB code provided. First, in the file $rmap\_dano1.m$ the strain norm had to be computed with the equations

$$\tau_{\varepsilon_{(n+\alpha)}} = (1-\alpha)\tau_{\varepsilon_n} + \alpha\tau_{\varepsilon_{(n+1)}},$$



(a) Damage surface - Hardening



(b) Damage surface - Softening

Figure 1.7: Results obtained using the loadpath presented in Section 1.3.3 for hardening and softening using the symetric model

.

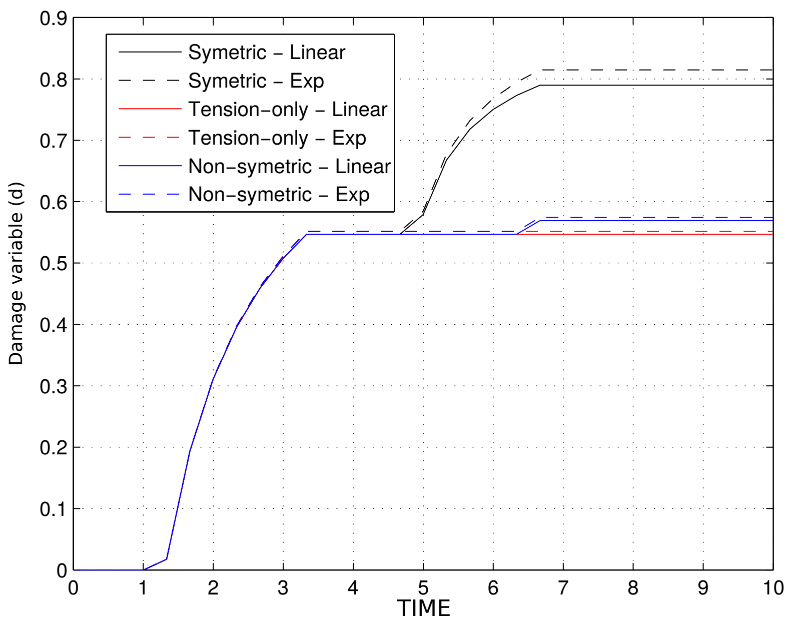


Figure 1.8: Comparison between hardening laws for each of the models presented using loadpath of Section 1.3.3.

where α is the parameter governing the time integration scheme to be used. This strain norm is further used in the computation of the internal variable r, who was modified in the same file to the equation

$$r_{n+1} = \frac{[\eta - \Delta t(1-\alpha)]}{\eta + \alpha \Delta t} r_n + \frac{\Delta t}{\eta + \alpha \Delta t} \tau_{\varepsilon_{(n+\alpha)}}, \tag{2.1}$$

where Δt and η are respectively the time step and viscosity. The algorithmic constitutive tangent operator C^{vd}_{alg} is implemented in the *damage_main.m* file and is calculated as a post process with the equation

$$C^{vd}_{alg,n+1} = (1 - d_{n+1})C + \frac{\alpha\Delta t}{\eta + \alpha\Delta t}\frac{1}{\tau_{\varepsilon_{(n+\alpha)}}}\frac{H_{n+1}r_{n+1-q(r_{n+1})}}{r^2_{n+1}}(\bar{\boldsymbol{\sigma}} \otimes \bar{\boldsymbol{\sigma}}). \tag{2.2}$$

It is important to remark that Equations 2.1 and 2.1 will be only be called when there is evolution of the internal variable. If that is not the case r has the same implementation as the rate independent case and C^{vd}_{alg} assumes the form of

$$C^{vd}_{alg,n+1} = (1 - d_{n+1})C.$$

Differently from the inviscid case, the stresses on rate dependent model have the possibility to leave the elastic domain due to viscous effects. This phenomena can be observed in Figure 2.1, where stresses on the bi-axial tensile loading region leave the damage surface.

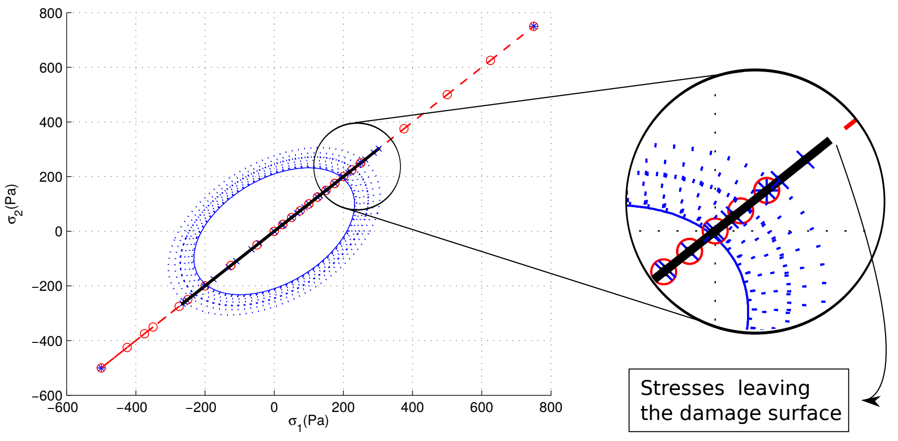


Figure 2.1: Behaviour of rate-dependent cases for the symetric model using a $\alpha = 1/2$ and $\eta = 0.3$ with the magnefied viscous effect.

2.1.1 Effect of viscosity

For this tests, the parameters used will be the same as in Section 1, with the loading path being

$$\Delta\bar{\sigma}^{(1)}_1 = 250Pa, \qquad \Delta\bar{\sigma}^{(2)}_1 = -950Pa, \qquad \Delta\bar{\sigma}^{(3)}_1 = 0Pa,$$
$$\Delta\bar{\sigma}^{(1)}_2 = 250Pa, \qquad \Delta\bar{\sigma}^{(2)}_2 = 450Pa, \qquad \Delta\bar{\sigma}^{(3)}_2 = 1400Pa,$$

and a Crank-Nicholson time integration scheme. The viscosity parameters taking on the test case are $\eta = (0.01 : 0.1 : 1)$. Zero viscosity was not chosen for the tests because they recover the rate-independent case.

Results for this test can be seem in Figures 2.2 and 2.3. It can be observed that viscosity has a big role in the the behaviour of all the parameters tested. With the increase of viscosity damage variable gets smaller, which reflects on the stress-strain curve and the constitutive operators.

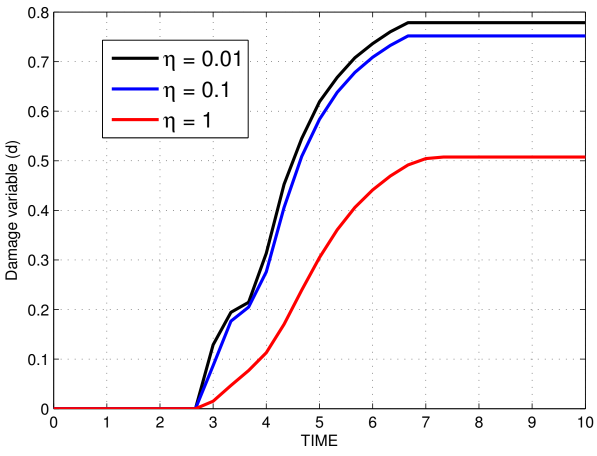
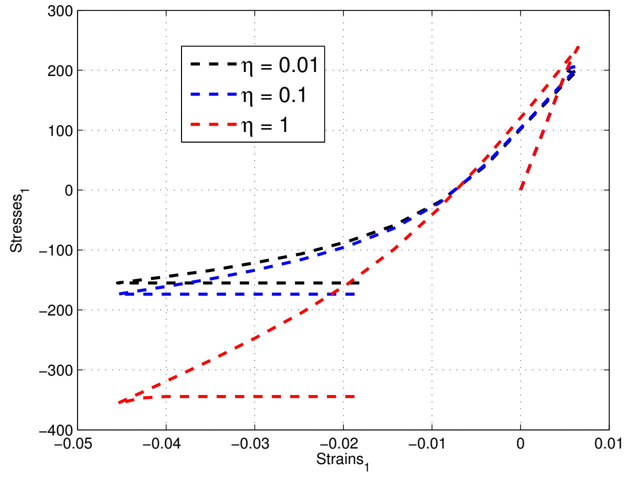
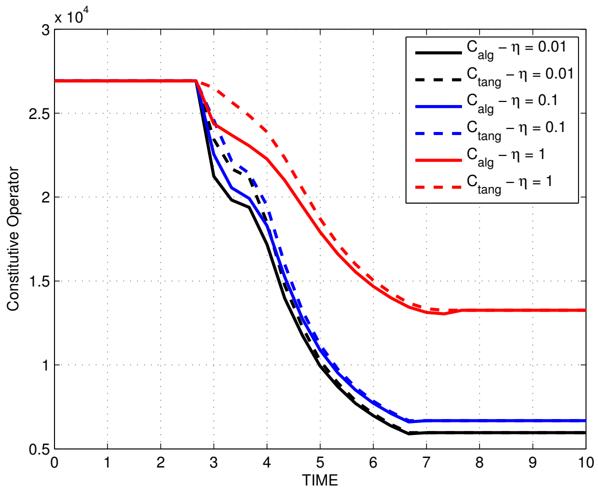


Figure 2.2: Results obtained for the evolution of the damage variable for different values of viscosity using the rate-dependt model

.



(a) Stress-Strain curve



(b) Component C11 of the constitutive operators

Figure 2.3: Results obtained for stress-strain and the Component C11 of the constitutive operators for different values of viscosity using the rate-dependt model

.

2.1.2 Effect of strain-rates

For this tests, the parameters used will be the same as in 2.1.1, including the same load-path for a viscosity of $\eta = 0.1$. Changing the strain rate directly on the code is not possible, so to achieve this a modification on the final time of simulation will be done. Hence, for the same load path, the final simulation time will be $t_{final} = (1s, 5s, 10s)$, indirectly modifying the strain rate.

Results for this tests can be observed in Figures 2.4 and 2.5. The effects of the strain rates over the stress-strain curve, damage variable and the constitutive operators are very similar to the effects of viscosity, as seem in Section 2.1.1. In this case, a high strain rate mimics the effects of a higher viscosity and vice versa.

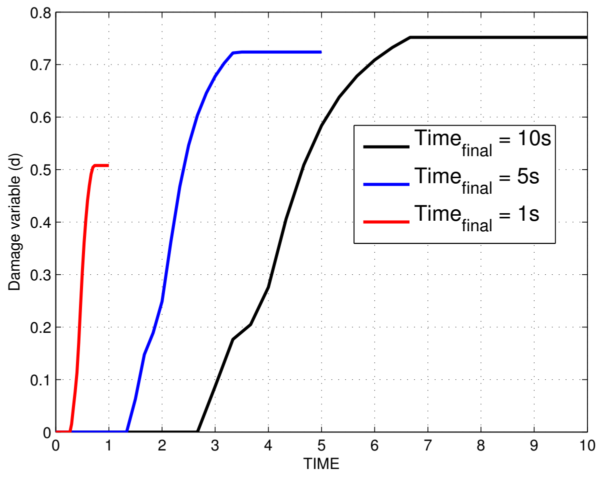
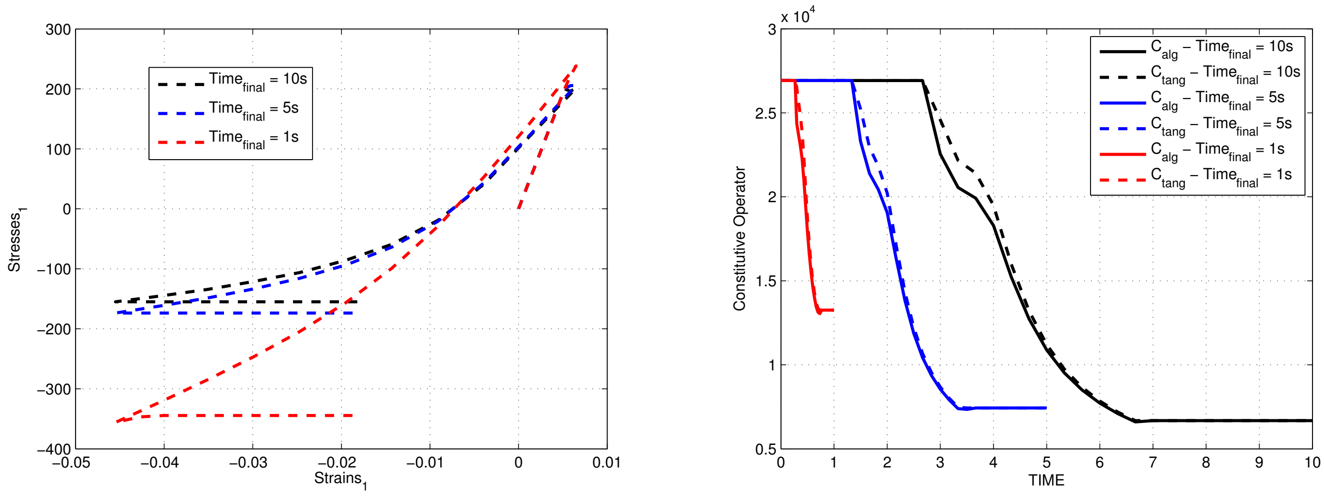


Figure 2.4: Results obtained for the evolution of the damage variable for different final times of simulation using the rate-dependt model

.



(a) Stress-Strain curve

(b) Component C11 of the constitutive operators

Figure 2.5: Results obtained for stress-strain and the Component C11 of the constitutive operators fdifferent final times of simulation using the rate-dependt model

.

2.1.3 Effect of time integration schemes

For this tests, the parameters used will be the same as the ones in Section 2.1.1, with a viscosity of $\eta = 0.1$. The time integration schemes used will be the ones related to $\alpha = (0 : 1/4 : 1/2 : 3/4 : 1)$.

The results obtained can be seem in Figures 2.6 and 2.7. For this case, the time integration scheme has a significant impact on the results of the simulation, Forward Euler has an instability, causing oscillation on all of the presented results. The other time schemes are not presenting oscillations, however, achieve different results when the stresses meet the damage surface. This result show the sensibility of the case towards its numerical method, and the importance to carefully chose the most adequate method for any particular case.

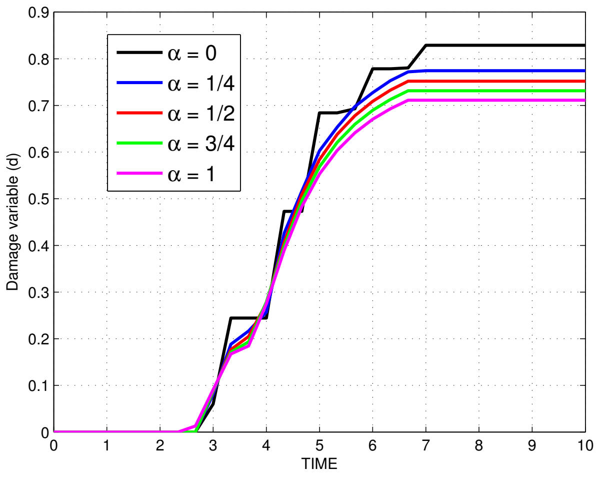
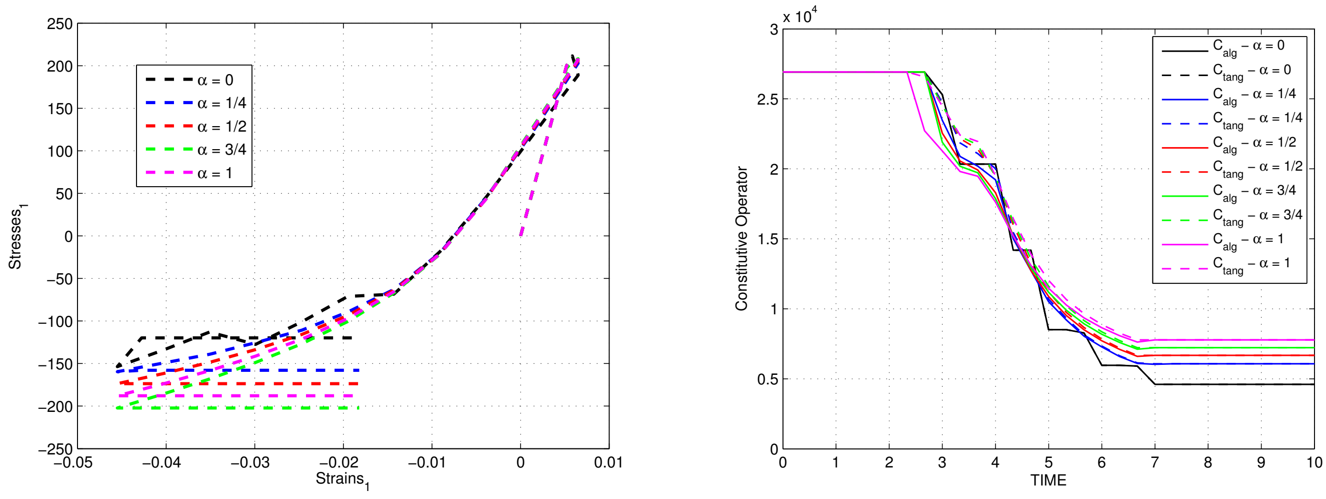


Figure 2.6: Results obtained for the evolution of the damage variable for different time integration schemes using the rate-dependent model

.



(a) Stress-Strain curve

(b) Component C11 of the constitutive operators

Figure 2.7: Results obtained for stress-strain and the Component C11 of the constitutive operators for different time integration schemes using the rate-dependent model

.

3 Concluding Remarks

The MATLAB code provided was modified and tested to meet the required aspects. Two new tension/compression models were implemented, as well as a hardening law for the rate-independent cases. The rate-dependent model was also implemented, with the calculation of the constitutive operator.

The implementation was tested against several cases, and in every case, it met the results expected from the theory. However, the field of Continuum Damage Model is very complex and requires a lot of experience and care from the user when applying it to the real world.

Annex - Modified Routines

File: *damage_main.m*

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,C_tang,C_alg,ce,d]=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    ;
    mhist   = 6    ;
end



totalstep = sum(istep) ;


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


% Elastic constitutive tensor
% ----------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
% Historic variables
% hvar_n(1:4) --> empty
% hvar_n(5) = q --> Hardening variable
% hvar_n(6) = r --> Internal variable
hvar_n  = zeros(mhist,1)  ;

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

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

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
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var,rtrial] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype
            ,n,delta_t(iload),strain(i-1,:)); %----> Included the time step and
            eps_n on the input
        % 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)

        fload(i) = aux_var(1);
    end

end

        %----> Starting the modification
        %----> Calculation of the C algoritmic/tangent operator

        %C_tang = ce;
        C_alg(:,:,1) = ce;
        C_tang(:,:,1) = ce;

        for i=2:length(vartoplot)
            d = vartoplot{i}(3);
            r = vartoplot{i}(2);
            q = vartoplot{i}(1);
            H = Eprop(3);
            alpha = Eprop(8);
            eta = Eprop(7);

            if(vartoplot{i-1}(3)~=vartoplot{i}(3))

                if (viscpr == 0)

                    C_alg(:,:,i) = (1-d)*ce-aux_var(3)*(sigma_n1*sigma_n1')/(1-d)
                        ^2;

                elseif (viscpr == 1)

                    C_alg(:,:,i) = (1-d)*ce+(alpha*delta_t(iload))/(eta+alpha*
                        delta_t(iload))*1/(rtrial)*(H*r-q)/(r^2)*(sigma_n1*sigma_n1
                        ');
                end

            else
                C_alg(:,:,i) = (1-vartoplot{i}(3))*ce;


            end
            C_tang(:,:,i) = (1-vartoplot{i}(3))*ce;
        end

        d(1,1) = 0;
        for i=2:length(vartoplot)
            d(1,i) = vartoplot{i}(3);
        end

        %----> Ending the modification
```

File: $Modelos\_de\_dano1.m$

```
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

    %----> Starting the modification

    sigma_n1 = eps_n1*ce; %----> Getting the stresses

    sigma_n1_positive = zeros(1,length(sigma_n1));  %----> Initializing the vector

    for i=1:length(sigma_n1)
        sigma_n1_positive(1,i) = (sigma_n1(i)>0)*sigma_n1(i); %----> Applying the
            Mcauly Bracket
    end

    rtrial = sqrt(sigma_n1_positive*eps_n1');  %----> Calculating the Norm of
        strains for only tension

    %----> Ending the modification

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

    %----> Starting the modification

    sigma_n1 = eps_n1*ce; %----> Getting the stresses

    theta = (sigma_n1(1)*(sigma_n1(1)>0) + sigma_n1(2)*(sigma_n1(2)>0) + sigma_n1
        (3)*(sigma_n1(3)>0) + sigma_n1(4)*(sigma_n1(4)>0))/...
            (abs(sigma_n1(1)) + abs(sigma_n1(2)) + abs(sigma_n1(3)) + abs(sigma_n1
                (4))); %----> Computing theta

    rtrial = (theta+(1-theta)/n)*sqrt(sigma_n1*inv(ce)*sigma_n1'); %----> Computing
         the norm of stresses for non-symetric model

    %----> Ending the modification

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

File: *rmap_dano1.m*

```
function [sigma_n1,hvar_n1,aux_var,rtrial] = rmap_dano1 (eps_n1,hvar_n,Eprop,ce,
    MDtype,n,delta_t,eps_n)  %----> Included the time step on the input









%*************************************************************
%*                                        *
%*            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) ;

    %----> Starting the modification
    %----> Parameters for the viscous case
    eta =  Eprop(7);
    alpha = Eprop(8);
    viscpr = Eprop(6);

    %----> Ending the modification

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


%*******************************************************
%*       initializing                                                  %*
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
% if(r_n<=0.d0)
%     r_n=r0;
%     q_n=r0;
% end
%*******************************************************


%*******************************************************
%*       Damage surface
                                                                   %*

    %----> Starting the modification
    %----> Including the viscous rtrial

    if viscpr == 0
        [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
    elseif viscpr == 1

        rtrial_n1 = Modelos_de_dano1 (1,ce,eps_n1,n);
        rtrial_n = Modelos_de_dano1 (1,ce,eps_n,n);

        rtrial = (1-alpha)*rtrial_n+alpha*rtrial_n1;
    end

    %----> Ending the modification

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


%******************************************************
%*   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;

    %----> Starting the modification
    %----> Viscous and Non-Viscous cases
    if viscpr == 1
```

```
        r_n1 = (eta−delta_t*(1−alpha))/(eta+alpha*delta_t)*r_n + (delta_t)/(eta+
            alpha*delta_t)*rtrial; %−−−−> Viscous case

    elseif viscpr == 0

        r_n1= rtrial ; %Non−Viscous case, same as before

    end

    %−−−−> Ending the modification

    if hard_type == 0
        %  Linear
        q_n1 = q_n + H*delta_r;
    else
       %−−−−> Starting the modification
       %*  Exponential

       option = 1; %−−−−> Option 1 use the direct form and Option 2 use the
           derivative form
       A = abs(H);  %−−−−> Initializing variable (A > 0)
       if option == 1
           if (H>0)
            q_inf = 2*r0 − zero_q; %−−−−> Defining the Value of q_inf for hardening
           elseif (H<0)
            q_inf = zero_q; %−−−−> Defining the Value of q_inf for softening
           end

       q_n1 = q_inf − (q_inf−r0)*exp(A*(1−r_n1/r0)); %−−−−> Computing q

       elseif option == 2;
           if (H>0)
               q_inf = 2*r0 − zero_q; %−−−−> Defining the Value of q_inf for
                   hardening
           elseif (H<0)
               q_inf = zero_q; %−−−−> Defining the Value of q_inf for softening
           end
           H_deriv = A*(q_inf−r0)/r0*exp(A*(1−r_n1/r0));
           q_n1 = q_n + H_deriv*delta_r; %−−−−> Computing q
       end

       %−−−−> Ending the modification

    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    end


else

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


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

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


%*******************************************************
%* Updating historic variables                                            %*
%  hvar_n1(1:4)  = eps_n1p;
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;
%******************************************************




%******************************************************
%* Auxiliar variables
                                                                 %*
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
%*****************************************************
```

File: *dibujar_criterio_dano1m*

```
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----->tensiï£¡n ï£¡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

    %---> Starting the modification
    %--->  Tensile damage-only model

    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)>0)*m1(i)  (m2(i)>0)*m2(i)  0 nu*((m1(i)>0)*m1(i)+(m2
            (i)>0)*m2(i))]*ce_inv*[m1(i)  m2(i)  0 ...
             nu*(m1(i)+m2(i))]');    %----> Mcauly operator applied on the first
                 stress term only

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

    end

    hplot =plot(s1,s2,tipo_linea);

    %----> Ending the modification

elseif MDtype==3

    %----> Starting the modification
    %---->  Non-Symetric model


    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

       theta = (m1(i)*(m1(i)>0) + m2(i)*(m2(i)>0) + 0 + (nu*(m1(i)+m2(i)))*(((m1(i)
           +m2(i)))>0))/...
            (abs(m1(i)) + abs(m2(i)) + 0 + abs(nu*(m1(i)+m2(i)))); %----> Computing
                 theta

        radio(i)= q/((theta+(1-theta)/n)*(sqrt([m1(i)  m2(i)  0 nu*(m1(i)+m2(i))]*
            ce_inv*[m1(i)  m2(i)  0 ...
            nu*(m1(i)+m2(i))]')));      %----> Computing the radius with the same
                procedure as MDtype=1 but the first term

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

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

    %----> Ending the modification
```

```



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



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