Computational Solid Mechanics

Master de Mètodes Numèrics en Enginyeria

Juan Diego Iberico Leonardo

March 23, 2018

ASSIGNMENT 1

Part 1: Rate Independent Models

Part 2: Rate Dependent Models

Contents

1 Rate Independent Models

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

1.1.1 The continuum isotropic damage “non-symmetric tension-compression damage” model and the “tension-only” damage model.

The first and necessary part is implement the program for the remaining damage models. That implementation is carried on the functions *Modelos de daño 1* and *Dibujar criterios de daño 1.*

Tensile damage model: the definition of the angle for this model is restricted to positive values of the stress. The intention is to represent the infinite elastic range for compression and a closed damaged surface for tension, leading to use the McAuley brackets.

The implementation of *rtrial* is based on $\sqrt{\bar{\sigma}^{+}\,\varepsilon_{t+\Delta t}}$. The evaluation at each step is done by comparing the *rtrial* with r_t and deciding if we are under loading condition (hardening or softening) for $rtrial > r_t$ or inside the elastic domain for $rtrial \leq r_t$.

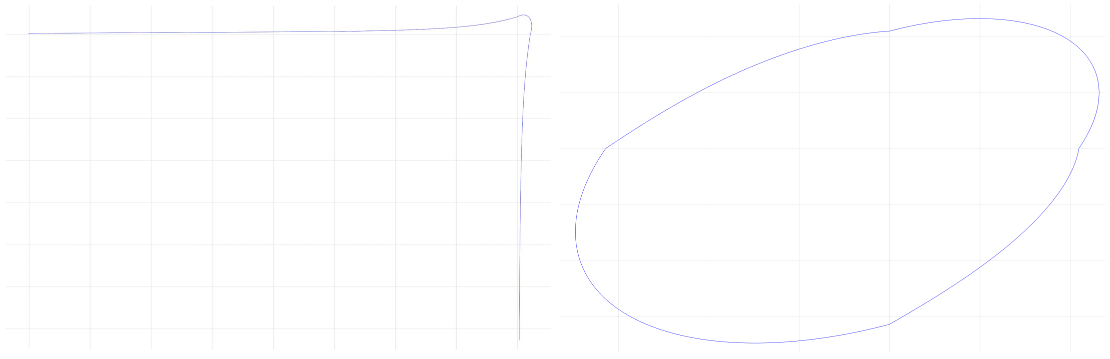


Figure 2: Only tension and Non-symmetric tension-compression models

Non-symmetric tension-compression model: for this model, the implementation of the code is based on the different damage surfaces for compression and tensile, characterized with n. In this case *rtrial* is considering both regions (1st and 3rd quadrant), leaving the remaining regions as transitions plotted as straight lines.

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 exponential implementation on the code for all the damage models is coded on the function *rmap daño 1.* That implementation require change the H constant by $H(r)$ on the equation $q_{t+\Delta t} = q_t + H(r)\,\Delta r$

1.3 Assess the correctness of the implementation: obtain the path at the stress space and the stress-strain curve

From the previous point, the two models of damage implemented are tested under different kind of path loads as well as material parameters.

Here we present the results and conclusions for the three specific cases, keeping constant the following datum:

PARAMETERS	ONLY TENSION	NON - SYMMETRIC
Young modulus	20000	20000
Poisson	0.1	0.1
Yield Stress	100	100
Hard/Soft Modulus	0.5	0.5
Hard/Soft Ev.	Exponential	Exponential
Ratio comp/tension	-	1.5
Increments (istep)	20	20
$\alpha \quad N/mm^2$	200	200
$\beta \quad N/mm^2$	300	600
$\gamma \quad N/mm^2$	400	400

Table 1: Set up of values and parameters

1.3.1 Case 1: Uniaxial loading with α, $-\beta$, γ

On the following lines are presented the tests for Only Tension and Non Symmetric damage models.

Only Tension: As it can be shown on the figure 3, the plot of stress-strain have three main regions related to tensile loading (200 N/mm^2), tensile unloading/compression loading (-300 N/mm^2) and tensile loading/compression unloading (400 N/mm^2).

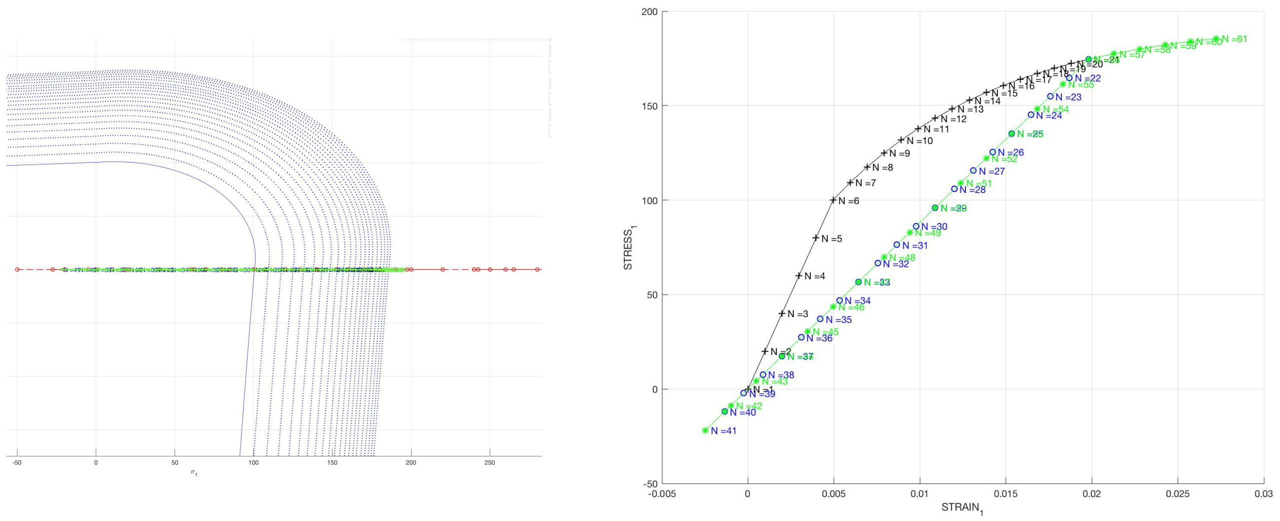


Figure 3: Uniaxial loading path and Stress-Strain plot (only tension model)

The fist part of the plot, corresponding to the tensile loading, represent an elastic behavior just until σ_y, with slope E (young modulus). Increasing the load, and been on the damage surface, the material start to experiment damage, evolution of the damage parameter (d). Considering the hardening modulus 0.5, bringing the capacity to the material to support more stress but with different strain response (not linear). As a remark, consider the material doesn't reach the value of

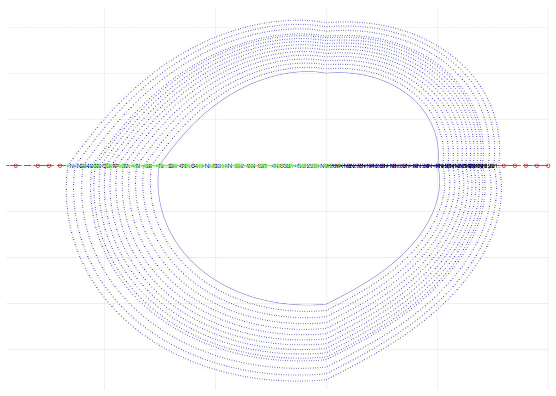
200 N/mm^2, due to the evolution of the damage parameter, arising $\sigma = (1-d)\,200N/mm^2$.

Once the tensile loading is reached, the second load path is applied defining the tensile unloading and 100 N/mm^2 of compressive loading. This load path, on the strain-stress plot is defined as an straight line with a slope $E_d = (1-d)E$ corresponding to inelastic stress.

The last step, is related to compression unloading/ tensile loading (with the same slope E_d) until reach the damage surface (not in σ_y). From that point, the evolution of the damage parameter and consequently the damage of the material increase, leading a non linear behavior.

In that damage model (only tension) the material damage it only can happen under tension loading, never under compression due to the fact we consider a material like concrete (next damage model will reproduce better that behavior).

Non symmetric: In order to reproduce the damage under compressive load, a new definition of the path load it is necessary (for better comprehension of the model).



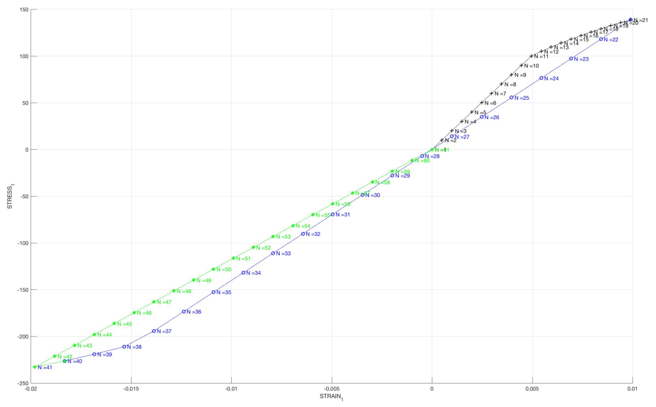


Figure 4: Uniaxial loading path and Stress-Strain plot (non symmetric model)

The uniaxial loading/unloading it is projected on figure 4, where it can be appreciated the closed damaged surface with straight lines on the second and fourth quadrant as transition between tensile and compression.

The loading path stars with the tensile loading of value 200 N/mm^2. Before σ_y the response is elastic characterized with a straight line with slope E. Once the load is over the yield stress, the damage internal variable starts to grow,giving as a result a non-linear response on the strain-stress space.

The second path apply 600 tensile unloading/ compressive loading. this loading path on the strain-stress space it is reflected as a straight line going through the point (0,0) due to the uniaxial load. The slope correspond to E_d, less steep respect to E.

The damage variable it remains constant until achieve the updated damage surface corresponding to $140*n = 210N/mm^2$ for n compressive surface damage. Further this point, the re-

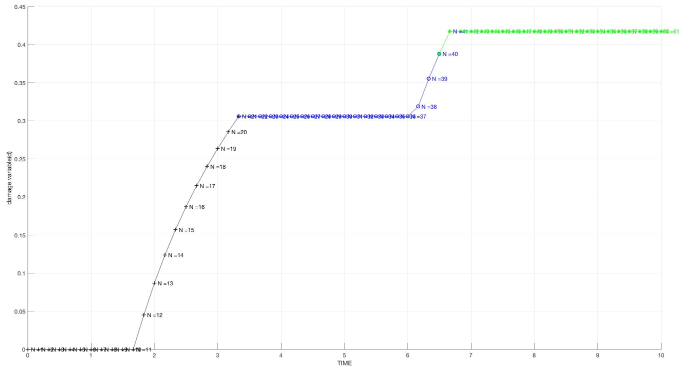


Figure 5: Evolution of the damage variable d respect to time.

sponse it correspond to increase
the damage internal variable (non-linear) until close the cycle (compressive unloading). That unloading is a straight line, even lees steep than the first unloading.

1.3.2 Case 2: Uniaxial loading α and biaxial unloading/loading $-\beta$, γ.

The evaluation of the correctness of both damage models in that case, are introducing biaxial stress.

Only Tension: In this stage, the loading path starts with uniaxial tensile loading of value 200 N/mm^2, followed by -300,-300 N/mm^2 tensile unloading and compressive loading. and the las one with 400,400 N/mm^2.

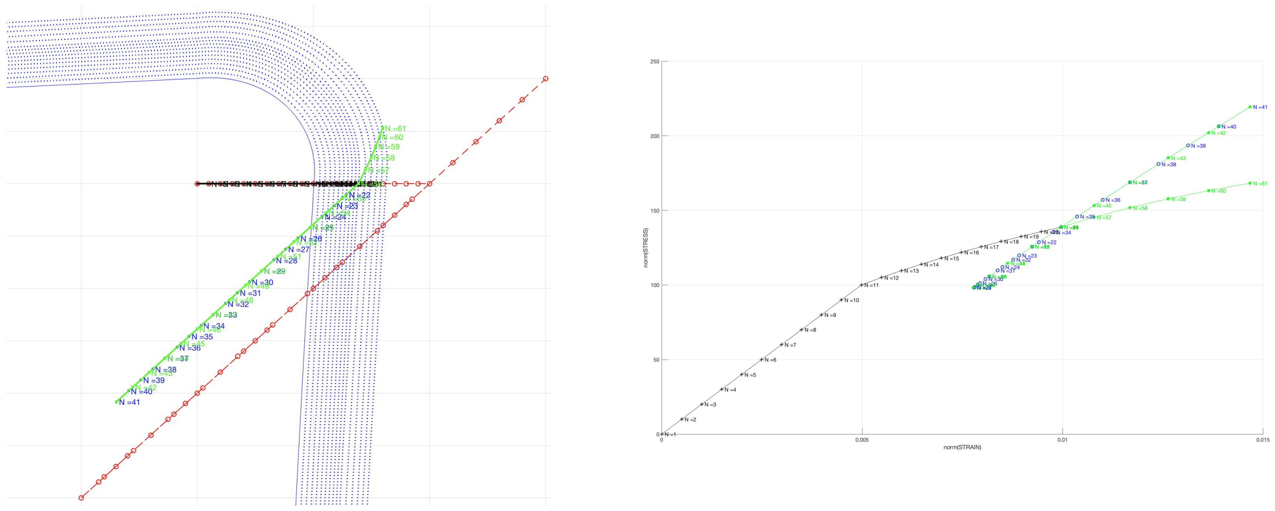


Figure 6: Uniaxial-biaxial loading path and Norm Stress-Strain plot (only tension model)

As it is shown on the figure 6, the load path reflects the uniaxial and biaxial loads/unloads. The remarkable point of this plot, rises from the permanent strain once the damage surface it is reached. This permanent strain deformation is the inelastic strain

Non- symmetric: The same load path uniaxial-biaxial is applied but with different values as it is shown in the table 1.

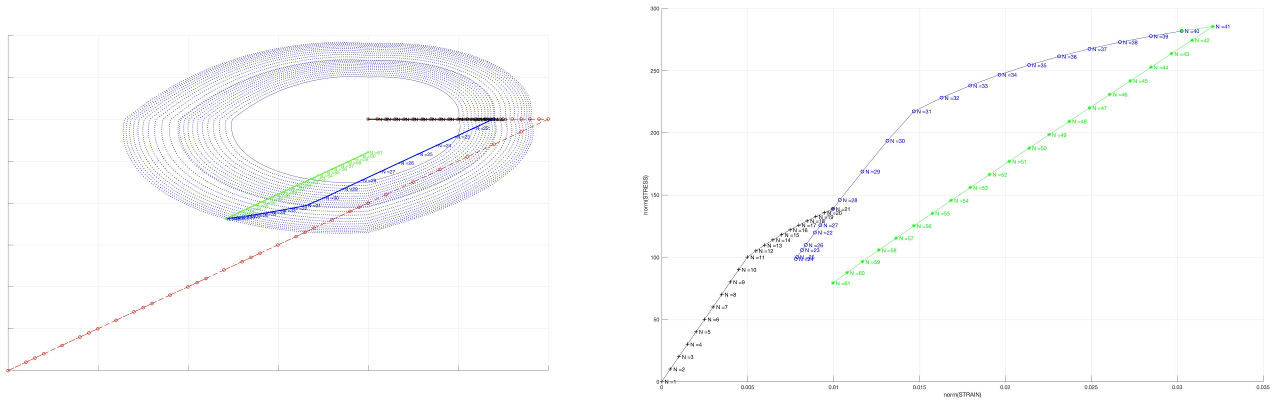


Figure 7: Uniaxial-biaxial loading path and Norm Stress-Strain plot (non symmetric)

A similar behavior as the only tension model it can been appreciate from the figure 7. The remarkable difference rise from the inelastic strain due to the biaxial load, having a permanently positive strain value under non applied load. From the other side, also the damage surface for the compressive load is n times equivalent.

1.3.3 Case 3: Biaxial loading/unloading α, $-\beta$, γ.

On this last part of the rate independent model, the path load are under biaxial tensile and compressive loading/unloading.

<u>**Only Tension:**</u> The first approach is 200,200 N/mm^2 tensile loading. The second path is -600,-600 N/mm^2 tensile unloading/compression loading and the last 400,400 N/mm^2 unloading.

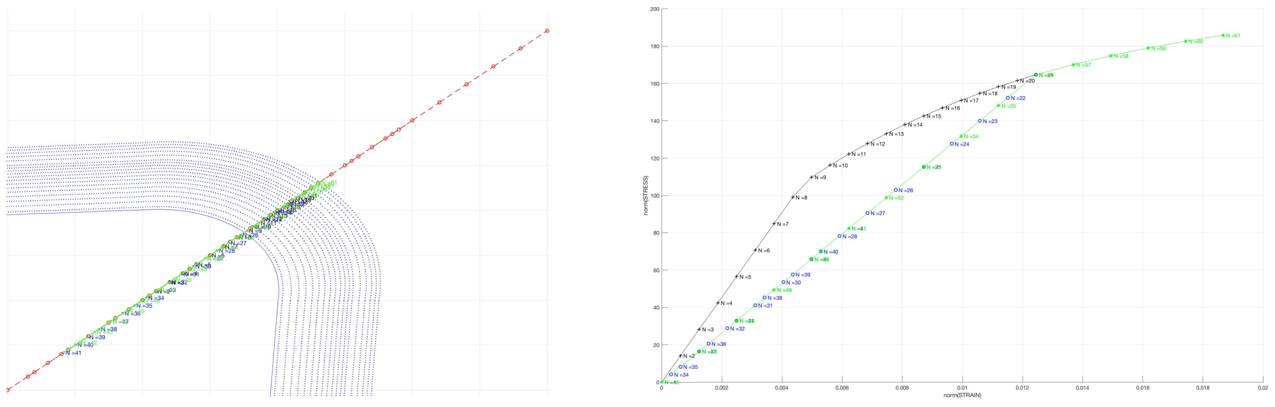


Figure 8: Biaxial loading path and Norm Stress-Strain plot (only tension)

Due to the biaxial load applied, and taking account the plot is in the norm strain-stress space, there is a similar behavior of the material respect to the pure uniaxial load

<u>**Non- symmetric:**</u> On the other side, this particular damage model, shows perfectly how the steep of the straight lines are decreasing, leading to $E > E_d(1) > E_d(2)$

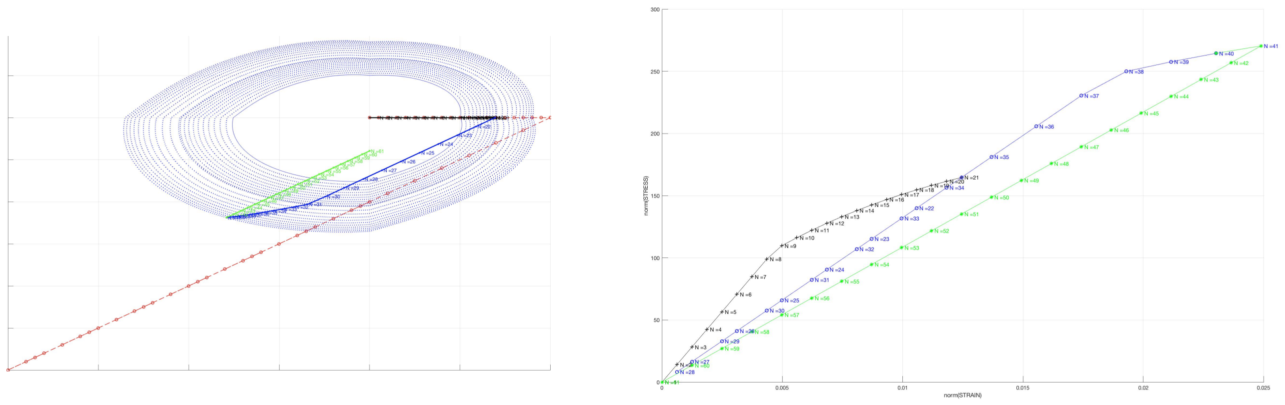


Figure 9: Biaxial loading path and Norm Stress-Strain plot (non symmetric)

1.3.4 Additional tests.

In this section, it will be explain the different behavior of the damage models from the point of view of the input data different from the load.

- Linear/exponential: All the figures presented are plotted with exponential hardening/softening law. This choose is motivated for the better accuracy/approximation presented on the exponential implementation with few point of discretization (load path). Furthermore, the difference of the response between linear and exponential is more considerable as the hardening/softening modulus increase as it is shown on the figure 10.

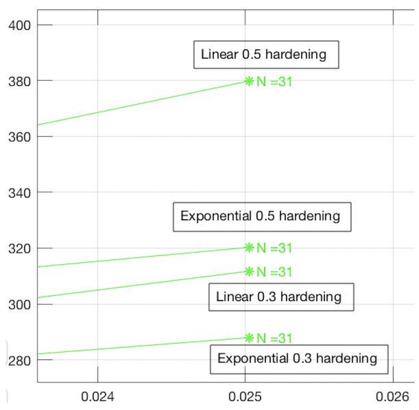


Figure 10: Linear and exponential for different values of hardening/softening values

- Hardening/softening modulus: The variance of this parameter allows to change the slope of the non-linear response (at strain-stress space) during the evolution of the damage variable.

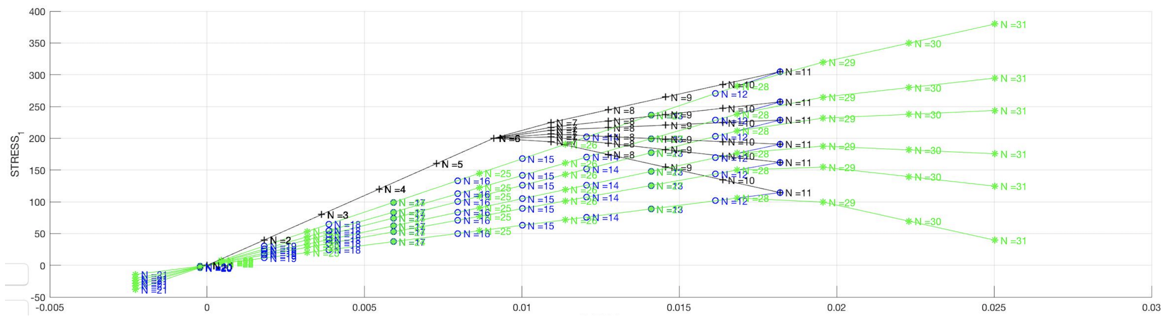


Figure 11: Different values of Hardening and Softening modulus

- Poisson value: The influence of the poisson material parameter has a direct impact on the eleastic range, varying the slope of that region. Furthermore, for hight values of poisson, appears a zone of transition form the elastic range until the hardening damage surfaces.

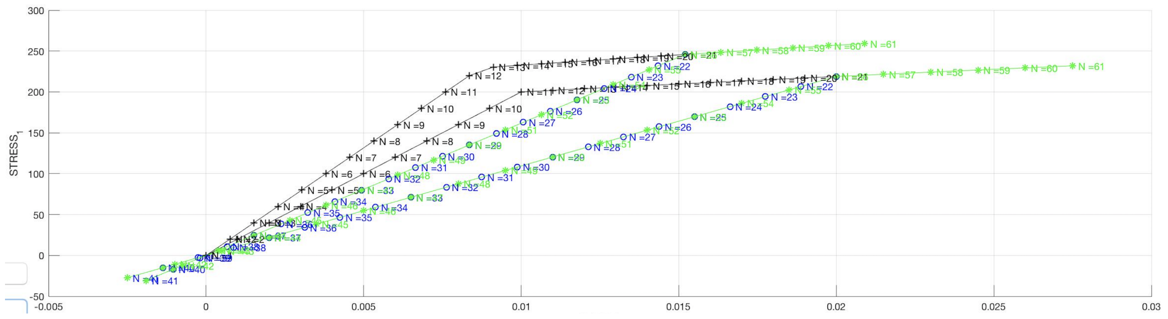


Figure 12: Comparison with poisson 0 and 0.49

2 Rate dependent 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 implementation of the viscous damage model add the η viscous parameter, the time integration alpha methods α. In order to add the option to compute viscous damage models on the code, the functions *Modelo de daño 1* and *rmap daño 1*.

In order to evaluate in each step if there is an evolution of the damage parameter, the *rtrial* to be evaluated is $\tau_{\epsilon_{n+\alpha}}$. In order to compute it, we need the information from the previous step and decide the integration method (α) to apply $\tau_{\epsilon_{t+\alpha}} = (1-\alpha)\,\tau_{\epsilon_n} + \alpha\,\tau_{\epsilon_{n+1}}$. Furthermore, it arises a new definition of r_{n+1} for loading condition.

In the following lines, it will be carried different tests for the symmetric damage model under viscous conditions and keeping fixed the values of Poisson ratio and hardening/softening parameter.

2.1.1 Assess the correctness of the implementation: different viscosity parameters η .

For the following section: $Poisson = 0.3$ and $H = 0.5$. The path load it is uniaxial tensile load for better illustration of the results. As it can be appreciate on the figure 13, different values of η arise to modification of the damage steepness line (once, over the values of $\sigma_y = 100$).

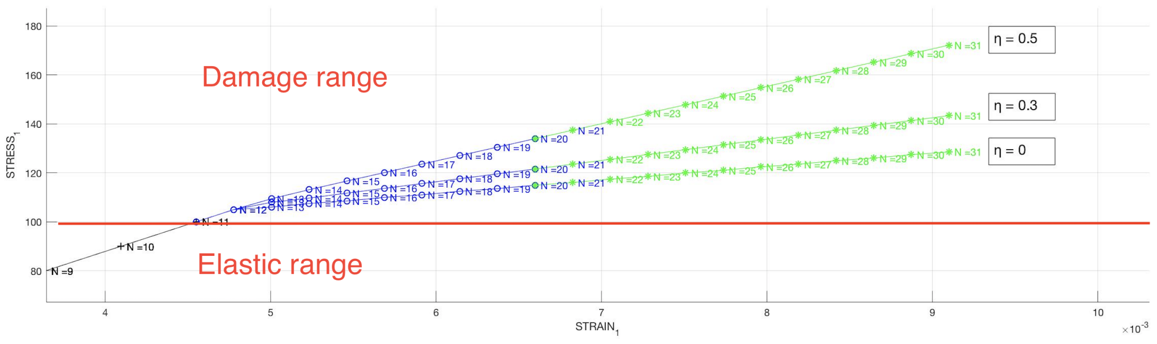


Figure 13: Response of different values of η

The correctness of the code it is ensured from the η response. The inviscid model it is recover with 0 viscosity, and as the viscosity increase the specimen is able to carry more stress with the same strain. That response result natural for material with hight viscosity.

2.1.2 Assess the correctness of the implementation: different strain rate $\dot{\varepsilon}$.

The computation of the figure 14 has fix the parameters $Poisson = 0.3$, $softening = -0.1$, $\eta = 0.5$ and tensile loading path.

The results shown on the figure 14, ensure the correctness of the code. At $\dot{\varepsilon} = 0$, we recover the same result as in the inviscid case, but when we increase the $\dot{\varepsilon} = 10$, the response against higher speed arise the stress response on the material/specimen as it is expected.

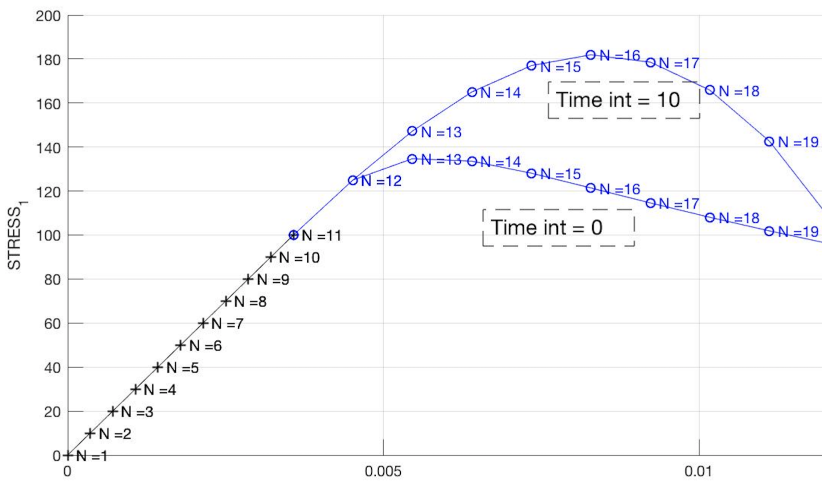


Figure 14: Response of different values of $\dot{\varepsilon}$

2.1.3 Assess the correctness of the implementation: different α values .

The numerical time integration of the constitutive equation is done by the family of alpha methods. On this stage, the intention it is study the influence of the α method on the strain-stress space. In order to obtain the best visualization of the results, it is considered a hight time interval.

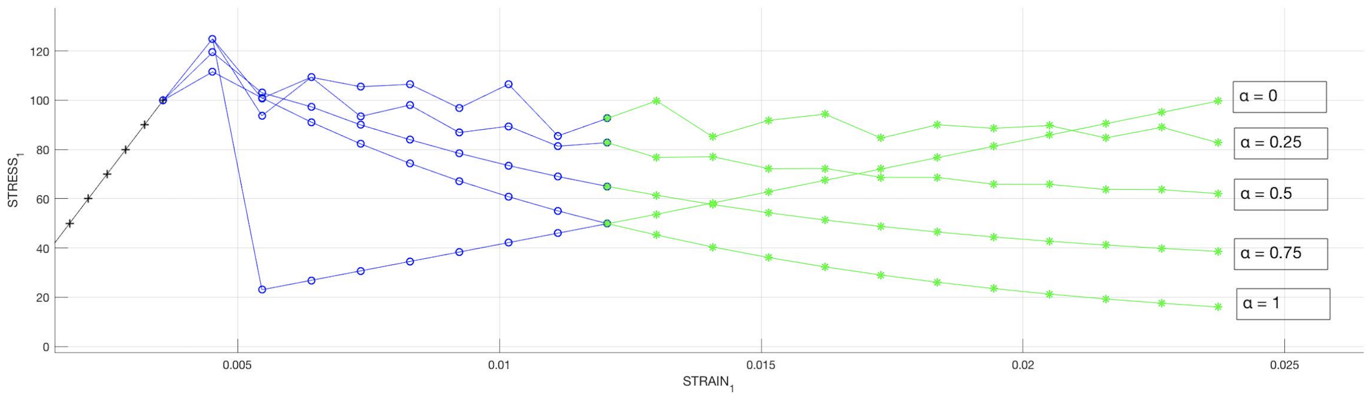


Figure 15: Response of different values of α

As it can be seen on the figure 15, for values of α smaller than or equal than 1/2 the stability compromise the results. On the other side, for values of α higher than 1/2, the stabilization is guaranteed. As as remark, the second order accuracy Crack-Nicholson method has instability at the damage surface, but it becomes smooth as evolves.

2.1.4 Assess the correctness of the implementation: The effects of the α values, on the evolution along time of the C11 component of the tangent and algorithmic constitutive operators.

As it can be seen on the figure 16, for hight values of α, the tangential component of the constitutive matrix decrease.

The values of the matrix is based on the damage variable for each step. For better comprehension, in the figure 17 it can be find the evolution of the internal damage variable along the time.

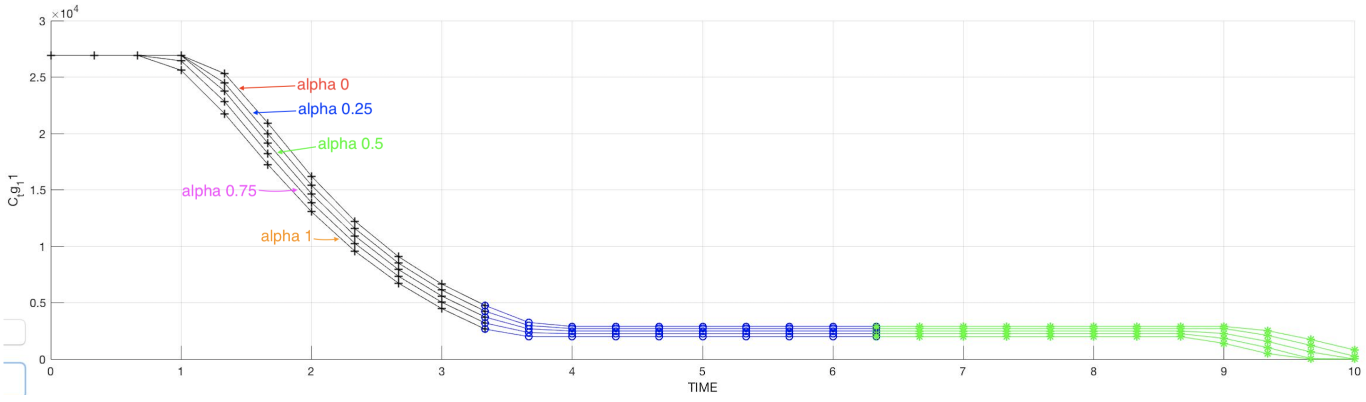


Figure 16: Response of different values of α on the C_{11} tangent constitutive operator

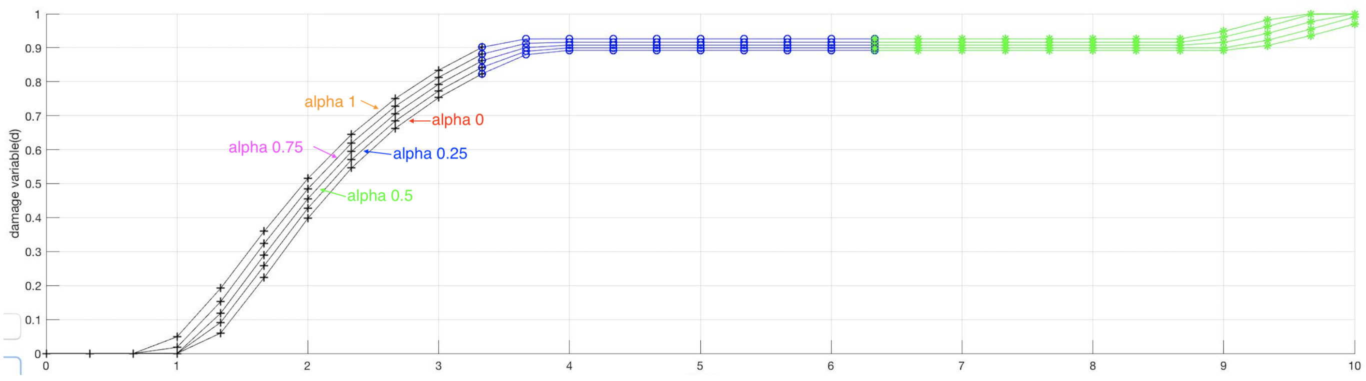


Figure 17: Evolution of the damage parameter along time

Form the figure 17, the damage variable increase with the α value. The correctness it is ensure, as the increase of the damage variable leads to decrease the values of the constitutive tangent matrix.

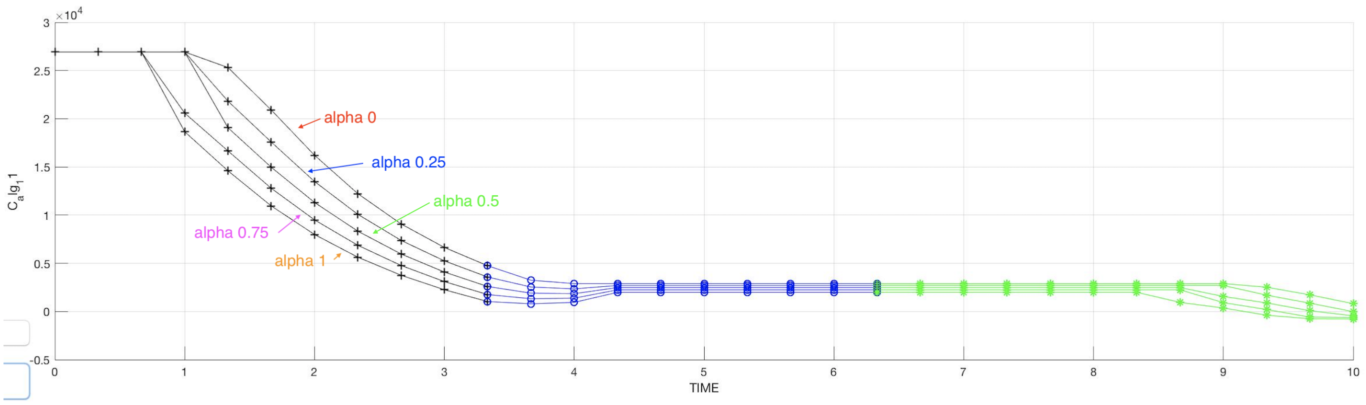


Figure 18: Response of different values of α on the C_{11} algorithmic constitutive operator

in the figure 18, as equal to the the figure 16, for hight values of α the algorithmic component decrease in accordance with the tangential one given the total matrix as a summation of both terms for the first component C_{11}

3 Conclusion

As a global evaluation, the implementation of the code plus the analysis for each stage proposed give as an global view of the continuum damage constitutive models.

The part of the rate independent models, doesn't present a good behavior of the general material due to the lack of the viscosity and the relaxation from a given strain. The independent part of the damage constitutive models, lead to have a better comprehension of the damage models when one can move to the rate dependent models.

On the other side, the convergence and accuracy on the numerical integration remark how the behavior of the computed results can be taken as a significant information or not.

To sum up, the clearance on the time independent part of the assignment give and overview of the real behavior of the materials.

4 Annex

4.1 Printed subroutines

```
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','damge variable d',↙
'C_alg_11', 'C_tg_11'};

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

% % if viscpr == 1
% %     % Comment/delete lines below once you have implemented this case
% %     % *******************************************************
% %     menu({'Viscous model has not been implemented yet. '; ...
% %         'Modify files "damage_main.m","rmap_dano1" ' ; ...
% %         'to include this option'},  ...
% %         'STOP');
% %     error('OPTION NOT AVAILABLE')
% % else
% % end
```

```
totalstep = sum(istep) ;


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


% Elastic constitutive tensor
% ----------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
eps_n  = 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
hvar_n(7) = 0; %d
hvar_n(8) = ce(1,1);
hvar_n(9) = ce(1,1);
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 = 5 ;
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) = hvar_n(7)  ; %  Damage variable (d)
vartoplot{i}(4) = hvar_n(8) ;
vartoplot{i}(5) = hvar_n(9) ;
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,:) ;
        eps_n = strain(i-1,:) ;
        %↙
*************************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,↙
delta_t);
        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6), 'r:',MDtype,n );
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1)                         ;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %*********************************************************************
        % 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) = hvar_n(7)  ; %  Damage variable (d)
        vartoplot{i}(4) = hvar_n(8) ;
        vartoplot{i}(5) = hvar_n(9) ;
    end
end
```

```
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¿½n ï¿½ltima↙
%*
%*                     ntype                                 %*
%*                                ntype=1  plane stress                             %*
%*                                ntype=2  plane strain                             %*
%*                                ntype=3  3D                                       %*
%*                     ce(4,4)     Constitutive elastic tensor  (PLANE S.       )    %↙
*
%*                     ce(6,6)                                ( 3D)                  %↙
*
%*************************************************************************************


%*************************************************************************************
%*        Inverse ce                                                                %*
ce_inv=inv(ce);
c11=ce_inv(1,1);
c22=ce_inv(2,2);
c12=ce_inv(1,2);
c21=c12;
c14=ce_inv(1,4);
c24=ce_inv(2,4);
%↙
*************************************************************************************







%↙
*************************************************************************************
% POLAR COORDINATES
if MDtype==1
    tetha=[0:0.01:2*pi];
    %↙
*************************************************************************************
    %* RADIUS
    D=size(tetha);                       %*  Range
    m1=cos(tetha);                       %*
    m2=sin(tetha);                       %*
    Contador=D(1,2);                     %*


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

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

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

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


elseif MDtype==2

     tetha=[-pi/2+0.01:0.01:pi-0.01];
    %↙
*************************************************************************************
    %* 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
        A = m1(i)*(m1(i)>0);
        B = m2(i)*(m2(i)>0);
        radio(i)= q/sqrt([A B 0 nu*( A + B )]*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
        A = m1(i)*(m1(i)>0);
        B = m2(i)*(m2(i)>0);
        o = (A + B)/ ((abs(m1(i))) + (abs(m2(i))));
        radio(i)= q/ ((o+(1-o)/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
```

```
function [rtrial,e_1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n,eps_n,viscpr,ALPHA)
%↙
****************************************************************************************
%*          Defining damage criterion surface                                          %↙
*
%*                                                                                     %↙
*
%*
%*                          MDtype=  1      : SYMMETRIC                                %↙
*
%*                          MDtype=  2      : ONLY TENSION                             %↙
*
%*                          MDtype=  3      : NON-SYMMETRIC                            %↙
*
%*                                                                                     %↙
*
%*                                                                                     %↙
*
%* OUTPUT:                                                                             %↙
*
%*                          rtrial                                                     %↙
*
%↙
****************************************************************************************



%↙
****************************************************************************************
if (MDtype==1) %* Symmetric
    % Define the tau epsilon at n+1
    e_1=sqrt(eps_n1*ce*eps_n1');
    % define the tau epsilon at n
    e_0=sqrt(eps_n*ce*eps_n');
    % Decide if the model is viscid or not
    if viscpr == 0                           % --Inviscid type-- %
    rtrial= e_1;
    else % compute tau epsilon + alpha       % ---Viscid type--- %
    rtrial = (1 - ALPHA)*e_0 + ALPHA*e_1;
    end

elseif (MDtype==2)  %* Only tension
    % Define the tau epsilon at n+1
    sigma_bar = eps_n1*ce;
    sigma_bar_plus(:) = sigma_bar(:).*(sigma_bar(:)>0);
    e_1 = sqrt(sigma_bar_plus*eps_n1');
    % Define the tau epsilon at n
    sigma_bar = eps_n*ce;
    sigma_bar_plus(:) = sigma_bar(:).*(sigma_bar(:)>0);
    e_0 = sqrt(sigma_bar_plus*eps_n');

   % Decide if the model is viscid or not
    if viscpr == 0                          % --Inviscid type-- %
    rtrial= e_1;
    else % compute tau epsilon + alpha
    rtrial = (1 - ALPHA)*e_0 + ALPHA*e_1;       % ---Viscid type--- %
    end


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

    % Define the tau epsilon at n+1
    sigma_bar = eps_n1*ce;
    sigma_bar_plus(:) = sigma_bar(:).*(sigma_bar(:)>0);
    o = ( sigma_bar_plus(1) + sigma_bar_plus(2) )/( abs(sigma_bar(1)) + ...
        abs(sigma_bar(2)) );
    e_1 = (o+(1-o)/n)*sqrt(eps_n1*ce*eps_n1');

    % Define the tau epsilon at n
    sigma_bar = eps_n*ce;
    sigma_bar_plus(:) = sigma_bar(:).*(sigma_bar(:)>0);
    o = ( sigma_bar_plus(1) + sigma_bar_plus(2) )/( abs(sigma_bar(1)) + ...
        abs(sigma_bar(2)) );
    e_0 = (o+(1-o)/n)*sqrt(eps_n*ce*eps_n');

     % Decide if the model is viscid or not
    if viscpr == 0                              % --Inviscid type-- %
    rtrial= e_1;
    else % compute tau epsilon + alpha
    rtrial = (1 - ALPHA)*e_0 + ALPHA*e_1;       % ---Viscid type--- %
    end

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

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

%↙
**************************************************************************************
%*                                          *
%*            Integration Algorithm for a isotropic damage model
%*
%*↙
*
%*             [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n1,hvar_n,Eprop,ce)↙
*
%*↙
*
%* INPUTS               eps_n1(4)   strain (almansi)    step n+1↙
*
%*                                  vector R4    (exx eyy exy ezz)↙
*
%*                      hvar_n(6)   internal variables , step n↙
*
%*                                  hvar_n(1:4) (empty)                          *
%*                                  hvar_n(5) = r  ; hvar_n(6)=q↙
*
%*                      Eprop(:)    Material parameters↙
*
%*
%*                      ce(4,4)     Constitutive elastic tensor↙
*
%*↙
*
%* OUTPUTS:             sigma_n1(4) Cauchy stress  , step n+1↙
*
%*                      hvar_n(6)   Internal variables , step n+1↙
*
%*                      aux_var(3)  Auxiliar variables for computing const. tangent↙
tensor  *
%↙
**************************************************************************************↙
*


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

```
%*************************************************************************************
%*       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                                                            %↙
*
[rtrial,e_1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n,eps_n,viscpr,ALPHA);
%*************************************************************************************


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

if(rtrial > r_n)
    if  (viscpr == 1)  %%%%%%%%%%%%%% Viscid %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %*   Loading
    fload=1;
    r_n1=  ((eta-delta_t*(1-ALPHA))/(eta+ALPHA*delta_t))*r_n + (delta_t/(eta +↙
ALPHA*delta_t))*rtrial;
    %delta_r=r_n1 - r_n;
    delta_r=rtrial - r_n;
    else %%%%%%%%%%%%%% Inviscid %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %*   Loading
    fload=1;
    delta_r=rtrial-r_n;
    r_n1= rtrial;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        % Exponential %
           q_inf = 2*(r0-zero_q)+zero_q;

           if H > 0 % Hardening
           q_n1 = q_n+((H*(q_inf-r0)/r0)*exp(H*(1-rtrial/r0)))*delta_r;

           H_n1 = ((H*(q_inf-r0)/r0)*exp(H*(1-rtrial/r0)));

           else     % Softening
```

```
            q_n1 = q_n+((H*(q_inf-r0)/r0)*(1/exp(H*(1-rtrial/r0))))*delta_r;

            H_n1 = ((H*(q_inf-r0)/r0)*(1/exp(H*(1-rtrial/r0))));

            end

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

%*************************************************************************************
% Tangent constitutive equatio
%*************************************************************************************

if viscpr == 1
    if rtrial > r_n

        Ce_alg_n1 = (1-dano_n1)*ce+((ALPHA*delta_t)/(eta+ALPHA*delta_t))*(1/e_1)*↙
((H_n1*r_n1-q_n1)/(r_n1^2))*...
            ((ce*eps_n1')'*(ce*eps_n1'));

        C_alg = Ce_alg_n1(1,1);

        Ce_tan_n1=(1-dano_n1)*ce;
        C_tan = Ce_tan_n1(1,1);
        hvar_n1(8)=C_alg;
        hvar_n1(9)=C_tan;
    elseif rtrial <= r_n

        Ce_alg_n1 = (1-dano_n1)*ce;
        C_alg = Ce_alg_n1(1,1);

        Ce_tan_n1 = Ce_alg_n1;
        C_tan = Ce_tan_n1(1,1);
        hvar_n1(8)=C_alg;
        hvar_n1(9)=C_tan;
    end
end



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