Computational Solid Mechanics: Assignment 1

Juan Pedro Roldán Blasco

March 2018

1 Inviscid

1.1 Input data

The material parameters used in the three sets of tests can be found in table 1. The material follows a softening law with no Poisson effect considered.

1.2 Case 1: Full uniaxial test

For the loading-unloading-loading case, stress increments are set in table 2. We can see that the first increment is set to be outside the damage surface (as $\Delta\sigma_1$ > Yield stress), so as to start having damage at the end of the first load. The model is behaving as expected. Figures 1 to 4 show the evolution of the damage surfaces and the strain - stress relation. We can see that the full uniaxial test does not show any difference between damage models: as the region in stress space in which the stress path is drawn is the same for both models (the first quadrant), the evolution of said damage surface will be, consequently, the same. Differences can be seen when we consider linear or exponential softening. For this set of hardening parameters, softening develops more rapidly in the exponential case. It can be observed that the exponentially softened damage surface is smaller than in the linear case.

Yield stress	150
Linear hardening H	-1
Exp hardening A	1
Young modulus	200
Poisson ratio ν	0
Ratio comp/trac n	1.5
Hardening limit q_{inf}	$10^{-6} r_0$

Table 1: Material parameters for inviscid (in consistent units)

step	$\Delta\sigma_1$	$\Delta\sigma_2$
1	160	0
2	-50	0
3	60	0

Table 2: Stress increments in full uniaxial test

1.2.1 Plots

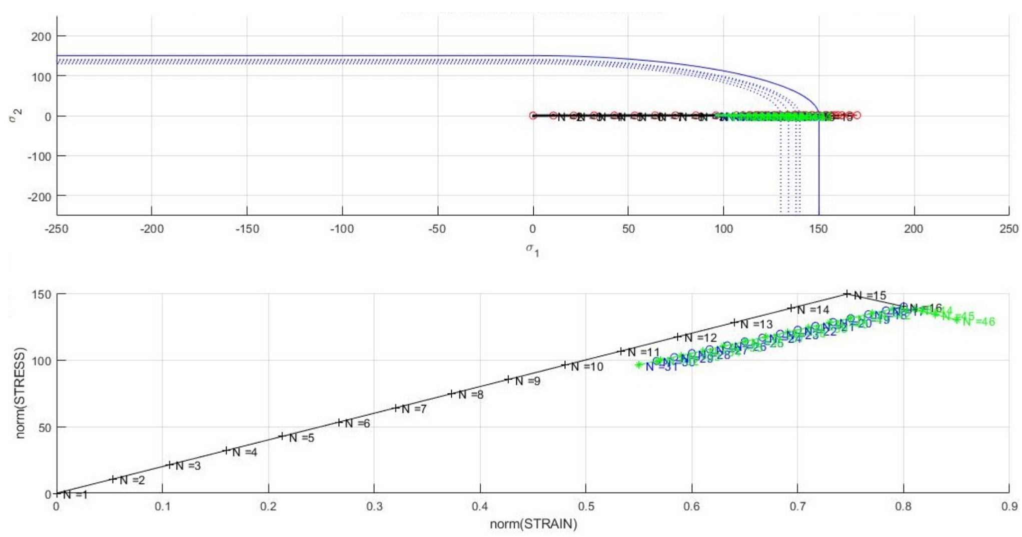

Figure 1: Damage surface (above) and strain-stress plot for the only-tension damage model with linear softening, case 1

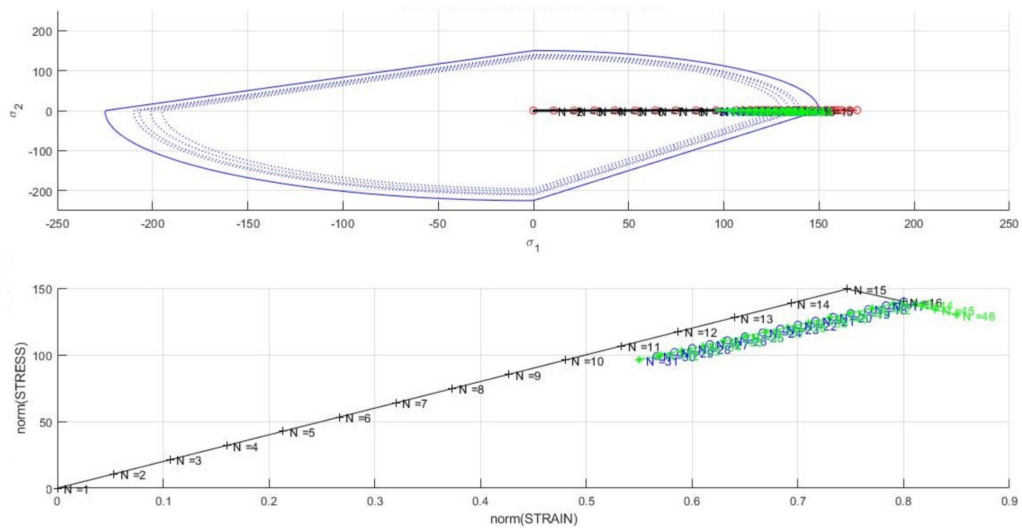

Figure 2: Damage surface (above) and strain-stress plot for the non-symmetric damage model with linear softening, case 1

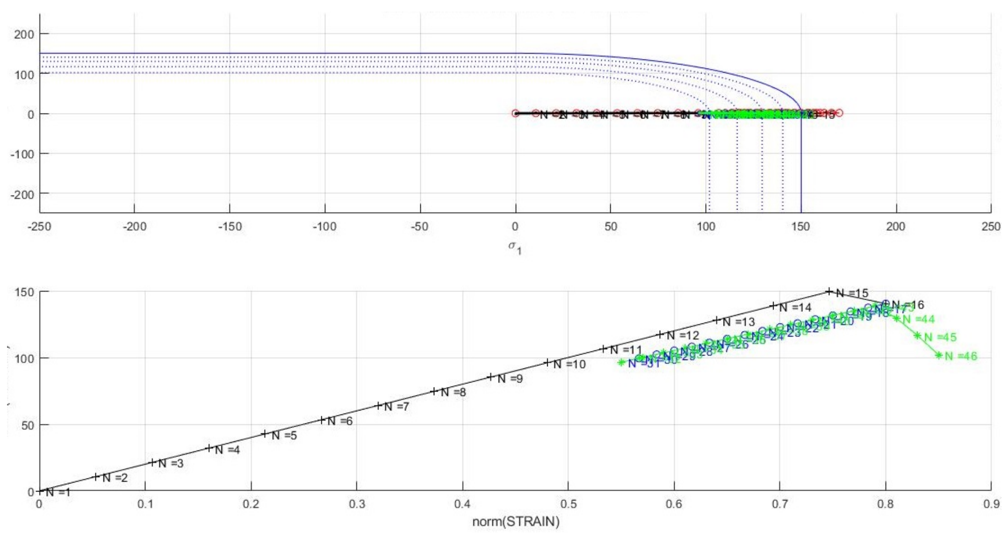

Figure 3: Damage surface (above) and strain-stress plot for the only-tension damage model with exponential softening, case 1

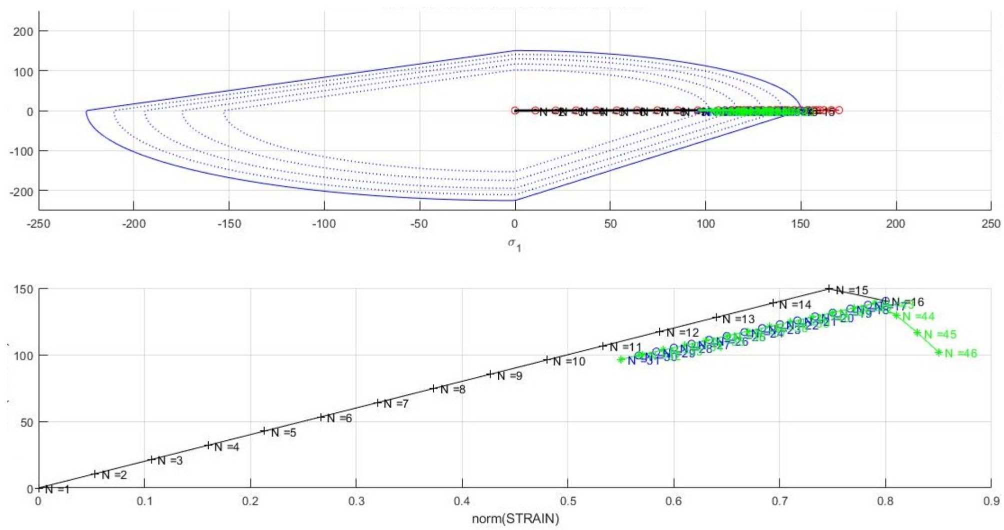

Figure 4: Damage surface (above) and strain-stress plot for the non-symmetric damage model with exponential softening, case 1

1.3 Case 2: Uniaxial - biaxial test

The initial stress increments can be found in table 3. The path chosen to finish every step outside of the damage surface in the non-symmetric case. The only tension model does not account for any limit in the compression zone, and that difference can be seen in the plots. Comparing figures 5 and 6, for instance, shows that while the biaxial loading and unloading stress paths are along the same line up while inside the elastic region (in red), for the non-symmetric model this is not case. For the latter, there exists a limit for the compression, and it is indeed surpassed (see the path in the stress space). This leads to the degradation of the material, and then, the change in the damage surface. In figure 6 this change is marked in red. Is the beggining of the third path (green line). If we consider exponential softening, the stress-strain trajectories show the corresponding exponential curve during inelastic loading, and (in this case, and given the hardening parameter value) the damage surface is reached earlier.

step	$\Delta\sigma_1$	$\Delta\sigma_2$
1	155	0
2	-230	-230
3	300	300

Table 3: Stress increments in uniaxial - biaxial test

1.3.1 Plots

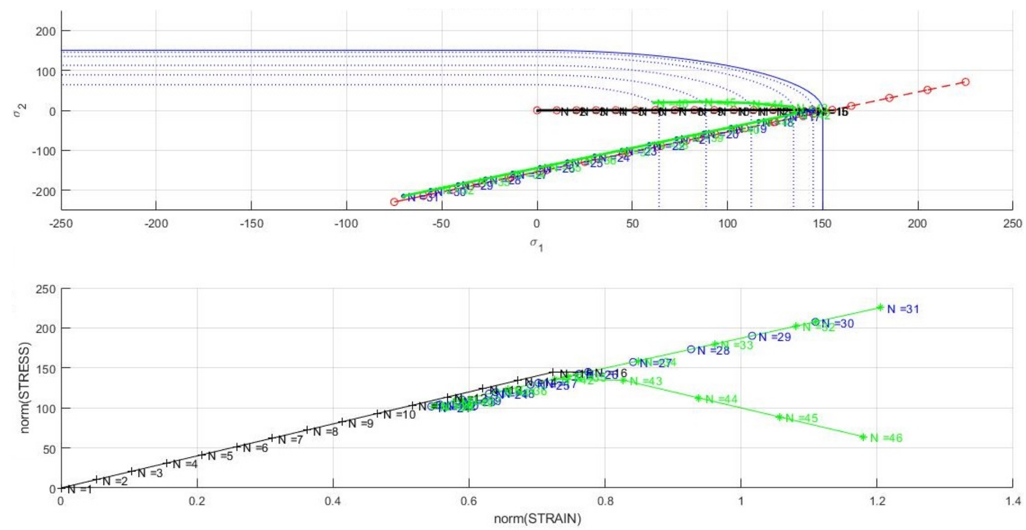

Figure 5: Damage surface (above) and strain-stress plot for the only-tension damage model with linear softening, case 2

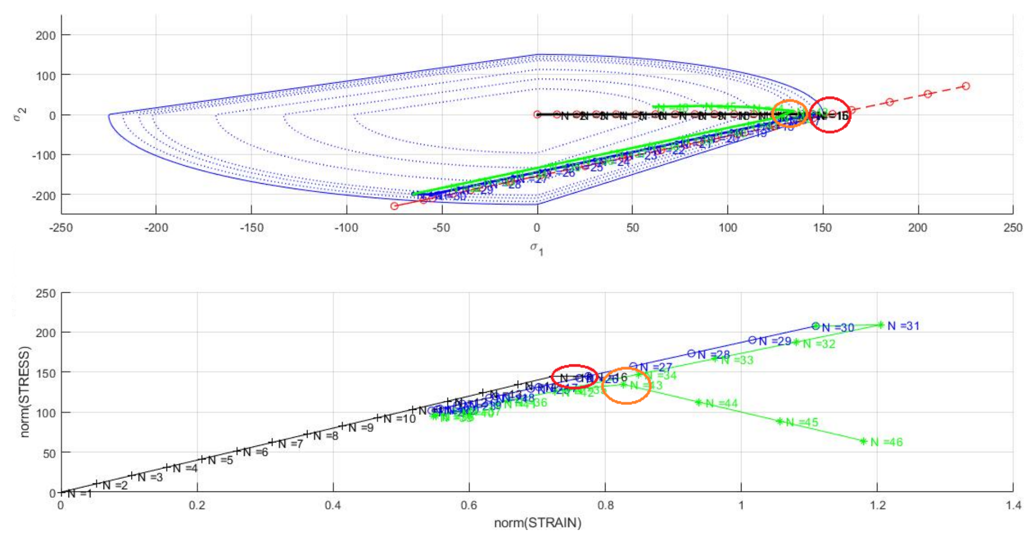

Figure 6: Damage surface (above) and strain-stress plot for the non-symmetric damage model with linear softening, case 2

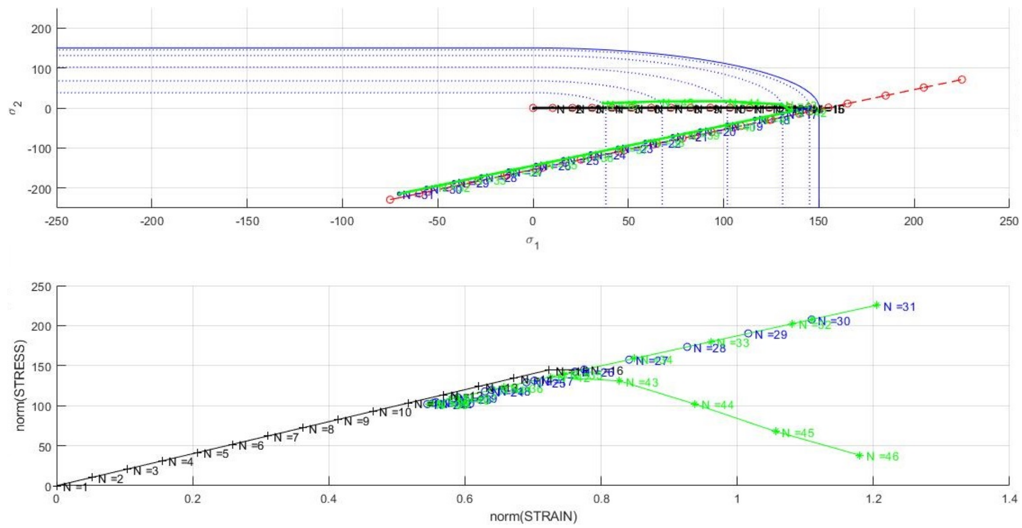

Figure 7: Damage surface (above) and strain-stress plot for the only-tension damage model with exponential softening, case 2

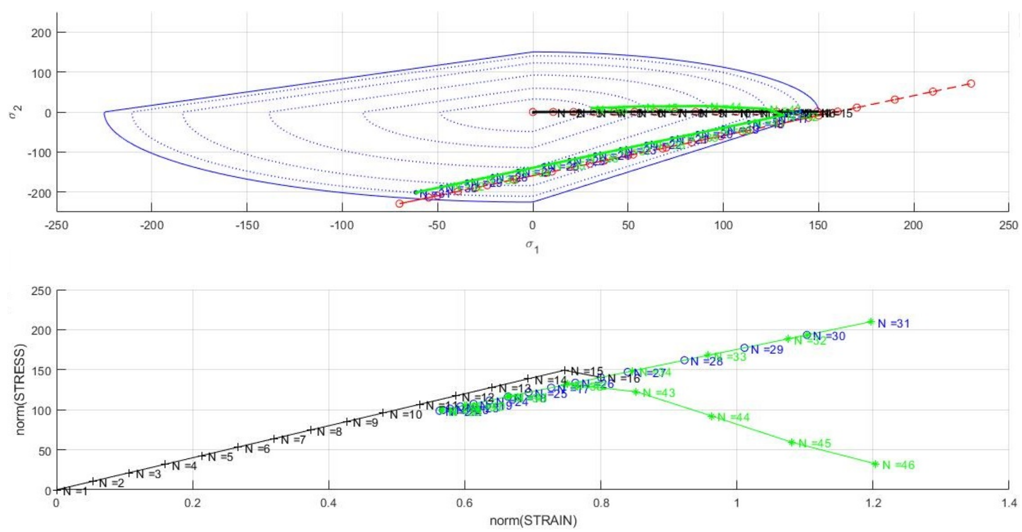

Figure 8: Damage surface (above) and strain-stress plot for the non-symmetric damage model with exponential softening, case 2

1.4 Case 3: Full biaxial test

Stress increments are set in table 4. As with the previous cases, the first stress increment is greater than the yield stress ($|120, 120| > 150$), so as to have damage since the first step.

step	$\Delta\sigma_1$	$\Delta\sigma_2$
1	120	120
2	-40	-40
3	50	50

Table 4: Stress increments in full biaxial test

The models behave in a completely different way. Both of them experience damage in the first step and during the second step reach the zero stress point only to rise again due to the fact that the situation has changed from tensile unloading to compressive loading. The non-symmetric model (figures 10 and 12) is damaged two times, one during the tensile loading and another one during the compressive loading. In this case, it can be assesed in all the —strain—-—stress— graphs that for complete unloading, the model returns to the origin (zero strain for zero stress).

1.4.1 Plots

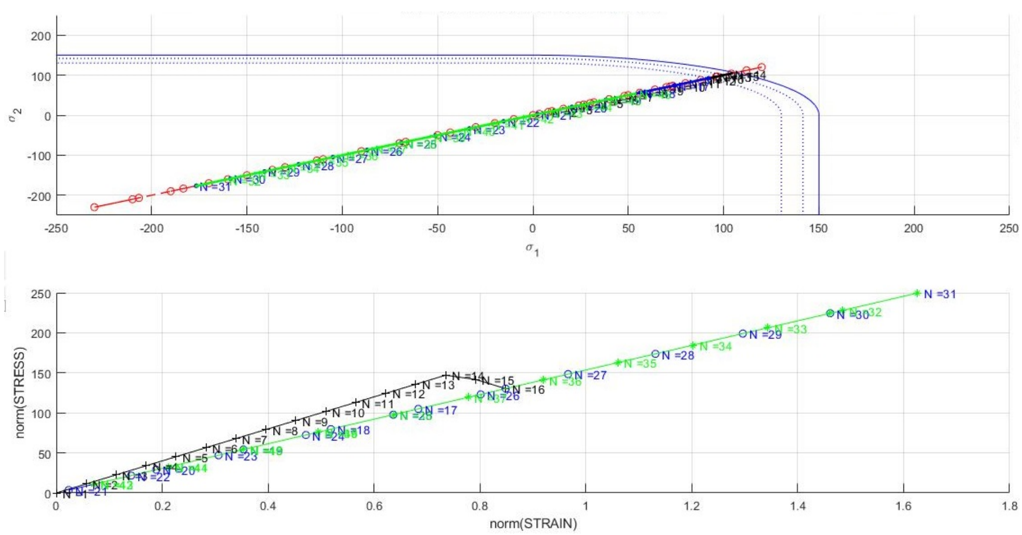

Figure 9: Damage surface (above) and strain-stress plot for the only-tension damage model with linear softening, case 3

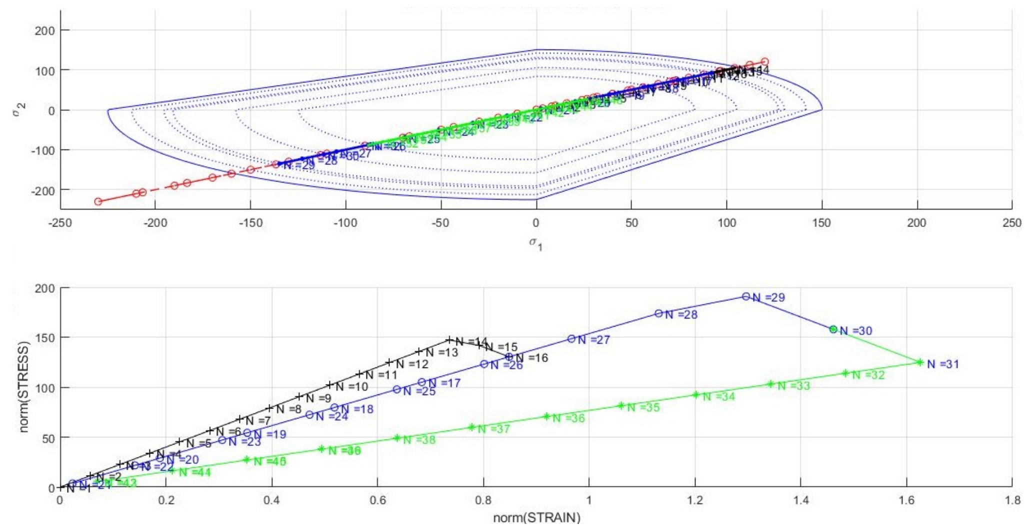

Figure 10: Damage surface (above) and strain-stress plot for the non-symmetric damage model with linear softening, case 3

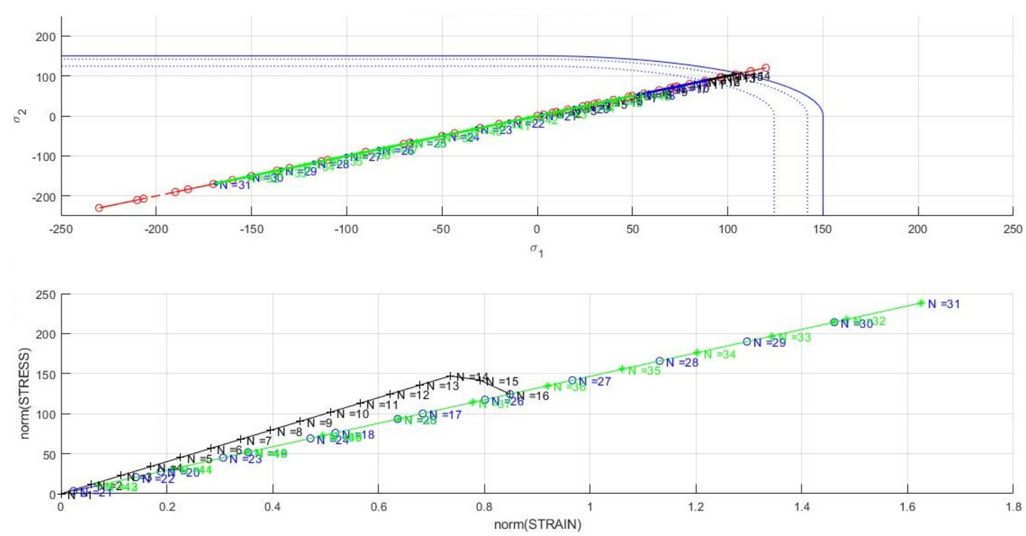

Figure 11: Damage surface (above) and strain-stress plot for the only-tension damage model with exponential softening, case 3

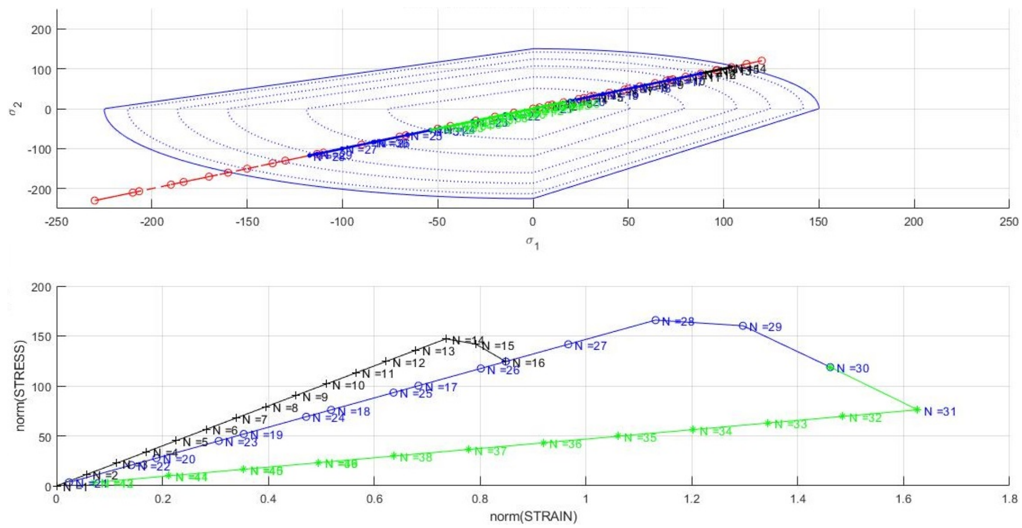

Figure 12: Damage surface (above) and strain-stress plot for the non-symmetric damage model with exponential softening, case 3

2 Viscous

In order to compare how the model's response is affected by viscosity and strain rate, we will first select one particular set of values as the standard (see table 5). When we want to see how the change of one particular parameter affects the behaviour of the solid, we will modify the value of said parameter in both directions (higher and lower). This way we will be able to, at least, see the trend of change in the solid's response: when it's getting stiffer and when it's getting softer. The stress path will be the same as in case 3 (full biaxial loading-unloading-loading).

Yield stress	150
Linear hardening H	-1
Young modulus	200
Poisson ratio ν	0
Viscous coefficient η	10
Hardening limit q_{inf}	$10^{-6}r_0$
Total time	100
α coefficient	0.5

Table 5: Standard configuration for the viscous symmetric model

Results for this standard example can be seen in figure 13. We will refer to this one everytime that we modify the viscosity, the strain rate (or time), and the α coefficient. We can see the general feature of the viscous solids: our

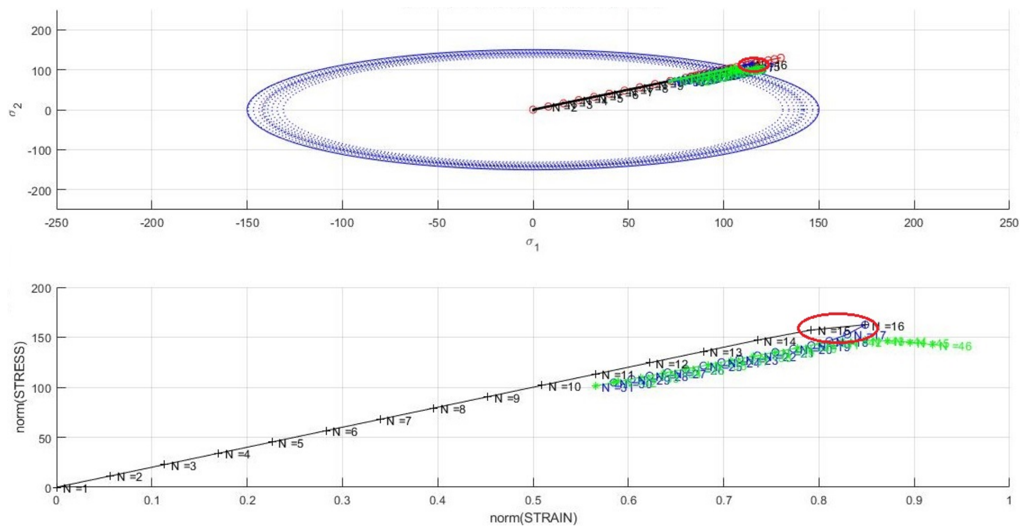

Figure 13: Standard results for the viscous case

stress path can be outside the damage zone, so we find that the norm of the stresses can be higher than the yield stress (points marked in red). The model can be damaged nonetheless, as the trajectories out of the damage surface and the change of its size suggests and the downwards (not linear due to viscosity) trajectory of the stress - strain path shows.

2.1 Effect of different viscosities

In order to asses the influence of the viscosity parameter η, we tried two new values: $\eta = 1$ and $\eta = 100$. For the former, the solid viscous response is almost negligible. We can see that the maximum stress is only slightly higher than the yield stress of 150. For the latter we find the oposite: the model increased viscosity stiffens it. Not only the maximum stresses are higher, but also the material is less damaged

2.1.1 Plots

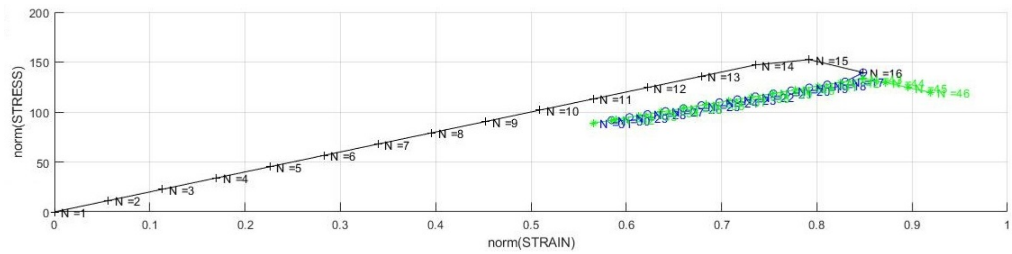

Figure 14: Damage surface (above) and strain-stress plot for viscous test with viscosity $\eta = 1$, the other values as in the standard case

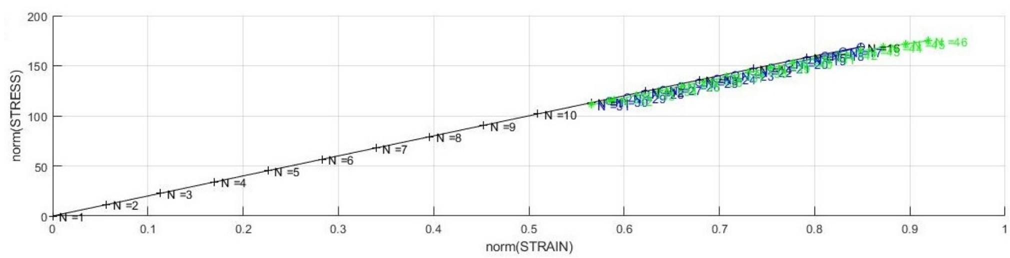

Figure 15: Damage surface (above) and strain-stress plot for viscous test with viscosity $\eta = 100$, the other values as in the standard case

2.2 Effect of different strain rates

The strain rate can be controlled via the total time used in the computation. High values of time means low strain rates (same strain applied over a longer period of time), and vice versa. For this reason, we used as extra values $t = 10$ and $t = 1000$ (one order of magnitude more and one less). As the strain rate increase, the material stiffens: damage appears later and the stress-strain relationshp is linear for a longer period of time. We can see that if the relationship between strain rate and viscosity is kept the same, the model's behaviour is also the same. The response for $\eta = 1$ and total time (indirect measure of strain rate) $t = 100$ is the same that the response for $\eta = 10$ and total time $t = 1000$. We can see that in the previous example the relation viscosity/time was the same $\frac{\eta}{t} = 0.01$. If we now reduce the time from the standard value of 100 to 10 (one order of magnitude less) with the rest of the parameters at their standard, we would have a relationship $\frac{\eta}{t} = 1$. This is the same as considered in 15, and that is why the model's response is the same. As expected, the model's response depends on the relationship strain rate / viscosity.

2.2.1 Plots

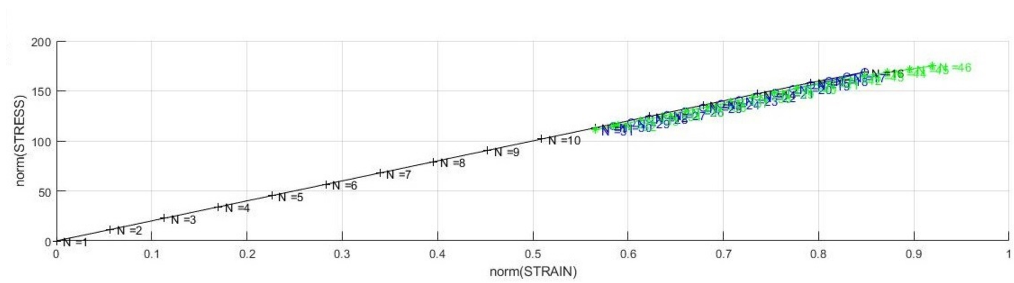

Figure 16: Damage surface (above) and strain-stress plot for viscous test with total time $t = 10$, the other values as in the standard case

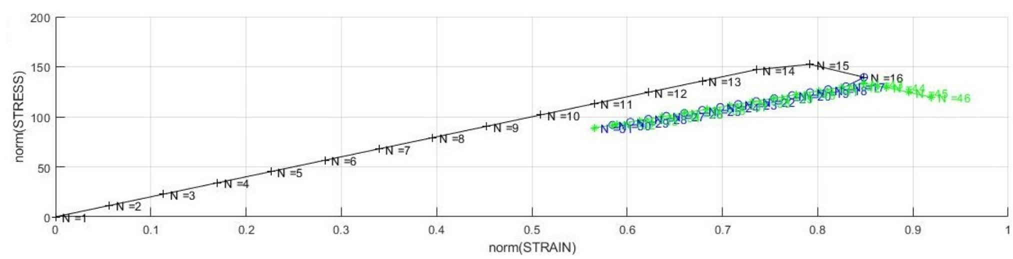

Figure 17: Damage surface (above) and strain-stress plot for viscous test with total time $t = 1000$, the other values as in the standard case

2.3 Effect of different alpha (time integration methods)

As the time integration method can lead to instabilities, we will try a combination of parameters such that said isntabilities can appear. The stress path considered differs from the used in the previous tests (see table 6), as in this case we want to be always outside of the damage zone to asses the behaviour during inelastic loading. As we want to use a low Δt, the total time and the time-steps per path used in the computations were $t = 1000000$ and $istep_1 = istep_2 = istep_3 = 10$, all the other parameters (except α, that we will modify) as standard. Our model then will work under $\frac{\Delta t}{\eta} = \frac{10000}{3}$. We can see in figure 18 that for the pure explicit method (Euler time integration) the results are completely spoiled: the solid is experiencing compression when we are inducing tension. For $\alpha = 0.25$, results are not as bad as with Euler, but we find instabilities: see that during inelastic loading (green and blue lines) the solution is going up and down. This was expected, as we are using a time-unstable integration coefficient for $\alpha \in [0\ 0.5)$. For $\alpha > 0.5$ (figures 20 to 22) these instabilities disappear. As we approach $\alpha = 1$, and given in this particular case the low viscosity that we are considering, the model behaves more like if it was inviscid. Again, this was expected from the theory.

step	$\Delta\sigma_1$	$\Delta\sigma_2$
1	120	120
2	20	20
3	30	30

Table 6: Stress increments in tests with different α

2.3.1 Plots

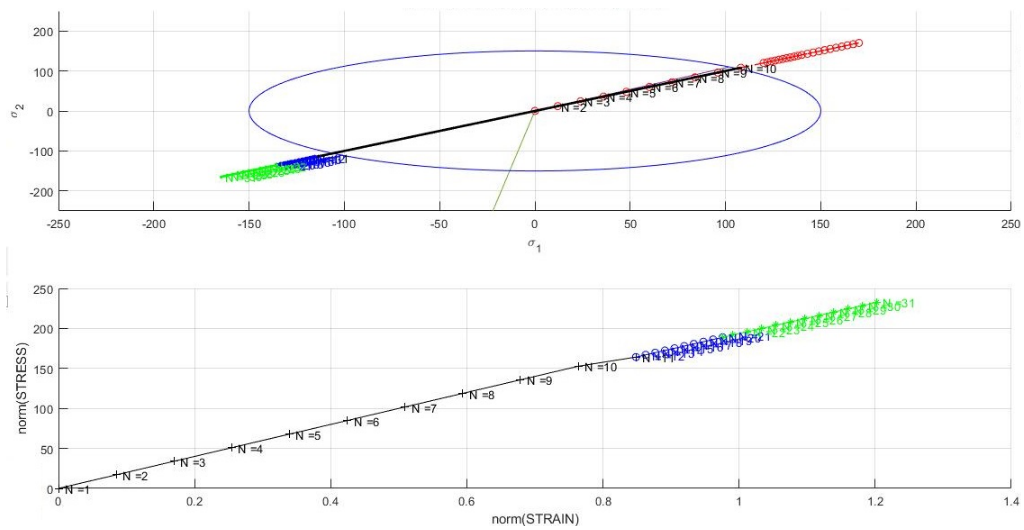

Figure 18: Damage surface (above) and strain-stress plot for viscous test with $\alpha = 0$. The results are spoiled and make no sense

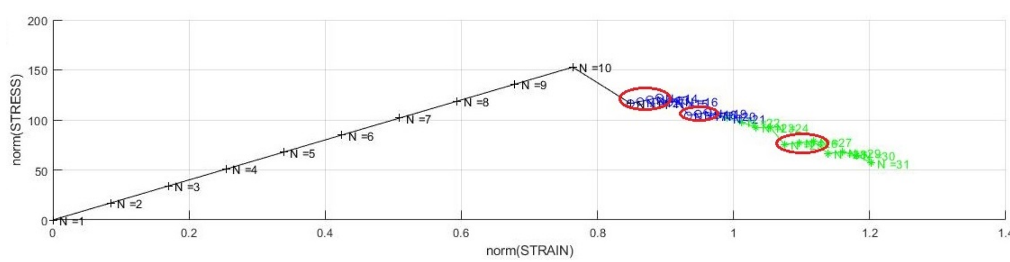

Figure 19: Strain-stress plot for viscous test with $\alpha = 0.25$. It shows oscillations (in red) due to instabilities

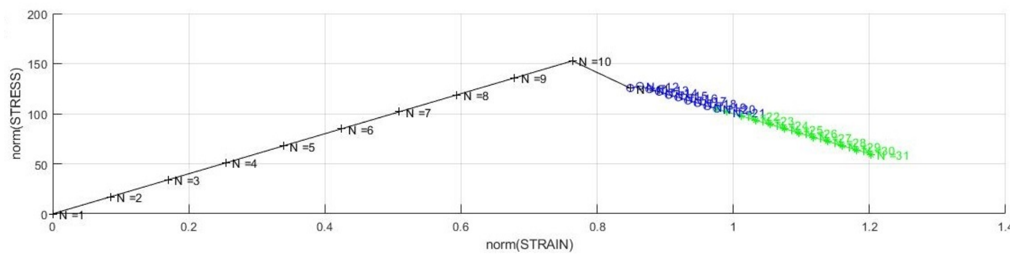

Figure 20: Strain-stress plot for viscous test with $\alpha = 0.5$, Crank-Nicholson method. The method for this α and higher are unconditionally stable

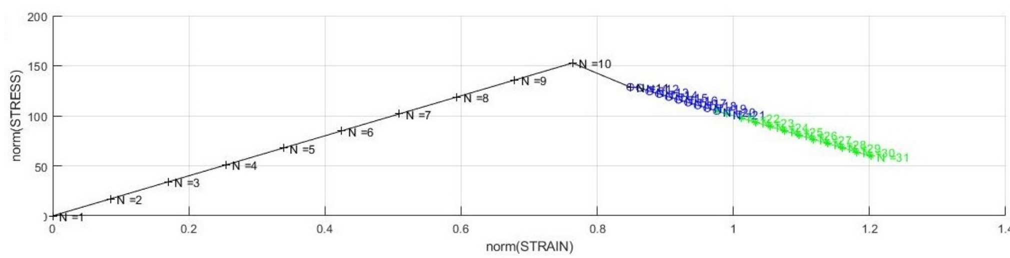

Figure 21: Strain-stress plot for viscous test with $\alpha = 0.75$. The inviscid part plays a higher role than with $\alpha = 0.5$

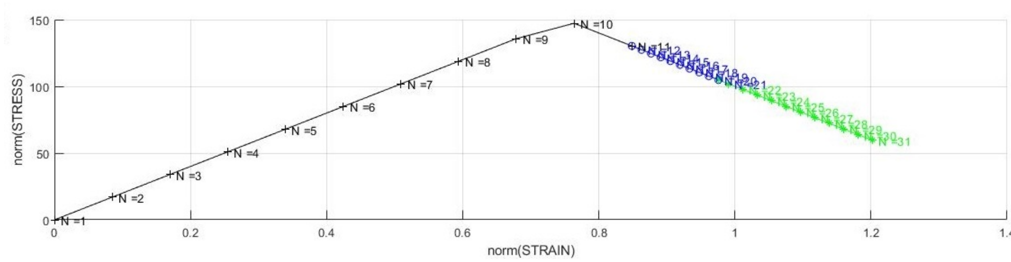

Figure 22: Strain-stress plot for viscous test with $\alpha = 1$. Inviscid case has been recovered

2.4 Constitutive operator evolution

Using the same input data from the previous tests we now take a look to the 11 component of the C operator in its analytical and algorithmic versions. We find similar results depending on α. For values smaller than 0.5, some oscillations due to instabilities are found. For greater than 0.5 values, the results are practicly the same, as it is stable (and, in this case, the time-viscosity relation is not very important). For the case $\alpha = 0$, both results coincde, although in this test we can see that the time integration method fails to compute realistic results.

2.4.1 Plots

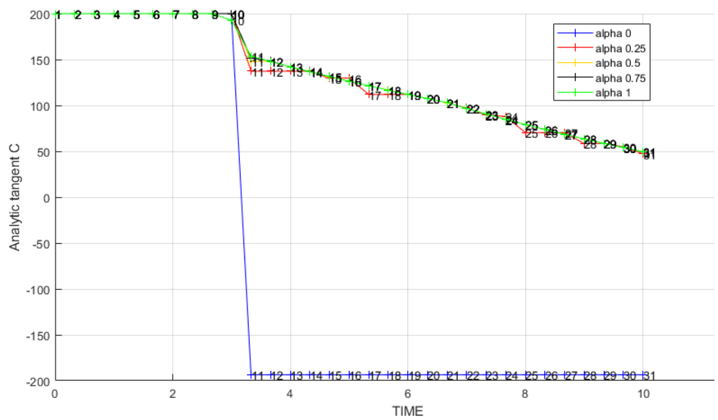

Figure 23: Evolution of the component 11 of the analytical tangent constitutive vs time for several values of α

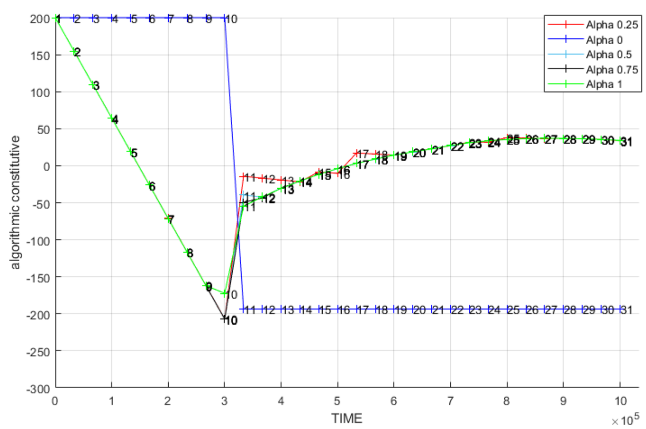

Figure 24: Evolution of the component 11 of the algorithmic tangent constitutive vs time for several values of α

3 Annex: Code

damage_main

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

```
%  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%  -----------------------------------------------------------------
%  1) sigma_v{itime}(icomp,jcomp) --> Component (icomp,jcomp) of the cauchy
%                                     stress tensor at step "itime"
%                                     REMARK: sigma_v is a type of
%                                     variable called "cell array".
%
%
%  2) vartoplot{itime--> Cell array containing variables one wishes to plot
%                                      --------------------------------------
%   vartoplot{itime}(1) =   Hardening variable (q)
%   vartoplot{itime}(2) =   Internal variable (r)%

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

% SET LABEL OF "vartoplot" variables
% ----------------------------------
 LABELPLOT = {'hardening variable (q)','internal variable',...
'Analytic tangent C','Algorithmic tangent C'};

E      = Eprop(1) ; nu = Eprop(2) ;
H = Eprop(3);
viscpr = Eprop(6) ;
sigma_u = Eprop(4);
eta = Eprop(7);
    alpha = Eprop(8);



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) ;
∆_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)
if viscpr
    vartoplot{i}(4) = (hvar_n(6)/hvar_n(5) )*ce(1,1);
    vartoplot{i}(5) = (hvar_n(6)/hvar_n(5) )*ce(1,1);
end

% LOOP over states (over the three paths)
for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ ∆_t(iload) ;
        % Total strain at step "i"
```

```
        % ------------------------
        eps_n1 = strain(i,:);
        %*****************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if (iload*iloc == 1)
            % For inviscid case, 1st rtrial must be done explicitly
            Eprop(8) = 0;
            rtrial_o = 0;
            [sigma_n1,hvar_n,aux_var,rtrial_o] = rmap_dano1...
                (eps_n1,hvar_n,Eprop,ce,MDtype,n,Δ_t,rtrial_o);
        else
            Eprop(8) = alpha;
            [sigma_n1,hvar_n,aux_var,rtrial_o] = rmap_dano1(eps_n1...
                ,hvar_n,Eprop,ce,MDtype,n,Δ_t,rtrial_o);

        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)
        if viscpr
            cet = sigma_n1'*sigma_n1;
            vartoplot{i}(4) = (hvar_n(6)/hvar_n(5))*ce(1,1);
            % Tangent analytical operator
            vartoplot{i}(5) =  ...
                (hvar_n(6)/hvar_n(5))*ce(1,1) - ((alpha*Δ_t)/...
        (eta+alpha*Δ_t))*(1/rtrial_o)*...
        ((hvar_n(6)-hvar_n(5)*H)/(hvar_n(5)^2))...
        *(cet(1,1));
        end
    end
end
```

rmap_dano1

```
function [sigma_n1,hvar_n1,aux_var,rtrial] = rmap_dano1 ...
    (eps_n1,hvar_n,Eprop,ce,MDtype,n,dt,rtrial_o)

%*************************************************************************************
%*                                          *
%*            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
*
%*                     dt          Time step
%*
%*                     rtrial_o    previous value of rtrial (inviscid case)
%
%* OUTPUTS:            sigma_n1(4) Cauchy stress   , step n+1
*
%*                     hvar_n(6)   Internal variables , step n+1
*
%*                     aux_var(3)  Auxiliar variables for computing
%                       const. tangent tensor  *
%**********************************************************************


hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5) ;
visc = Eprop(6);
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] = Modelos_de_dano1(MDtype,ce,eps_n1,n,r_n,q_n);
%**********************************************************************

% rtrial_o = rtrial previous timestep
%************************************************************************
%*   Ver el Estado de Carga
%*
%*   --------->    fload=0 : elastic unload
%*
%*   --------->    fload=1 : damage
%       (compute algorithmic constitutive tensor)         %*
fload=0;
if visc
    ralpha = alpha*rtrial + (1-alpha)*rtrial_o;
    if ralpha > r_n
        %*  Loading
        fload = 1;
        Δ_r = (ralpha - r_n)*dt/(eta+alpha*dt);
        r_n1 = r_n + Δ_r;
        if hard_type == 0
            % Linear
            q_n1= q_n+ H*Δ_r;
        else
            error('EXPONENTIAL LAW not implemented for inviscid case');
        end


    else
        %*  Unloading
        fload=0;
        r_n1= r_n  ;
        q_n1= q_n  ;

    end
else
    if(rtrial > r_n)
        %*   Loading

        fload=1;
        Δ_r=rtrial-r_n;
        r_n1= rtrial  ;
        if hard_type == 0
```

```
            %  Linear
            q_n1= q_n+ H*Δ_r;
        else
            % Comment/delete lines below once you have implemented this case
            % *******************************************************
    % %         menu({'Hardening/Softening exponential law has not
    % been implemented yet. '; ...
    % %             'Modify file "rmap_dano1" ' ; ...
    % %             'to include this option'},  ...
    % %             'STOP');
    % %         error('OPTION NOT AVAILABLE')
   %q_rate = 0.05;
            q_inf = zero_q; %
            q_n1 = q_inf - (q_inf-q_n)*exp(H*(1-(r_n1/r0)));
        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
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;
%**********************************************************************
```

Modelos_de_dano1

```
function [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n,r,q)
%*****************************************************************************************
%*          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
    s_eff = eps_n1*ce; % Efective stress
    s_eff_p = 0.5*(s_eff + abs(s_eff)); % Effective stress for only tension
    % As seen in Notes in continuum damage models page 18
rtrial = sqrt(s_eff_p * eps_n1');

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

    s =  eps_n1*ce; % Effective stress
    theta = 0; % Initialize theta = sum <sigma> / sum (|sigma|)
        for i=1:length(s)
            theta = theta + mac(s(i)); % Numerator
        end
    theta = theta/sum(abs(s)); %Denominator
    rtrial = ((theta + (1-theta)/n)) * sqrt(eps_n1*ce*eps_n1'); % strain r
% %     else
% %         if s(1) > 0 % 4th  quadrant
% %
% %         else % 2nd quadrant
% %
% %         end

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

dibujar_criterio_dano1

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)



%*******************************************************************
%*         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
        % Radius is the tau_sig = q/sqrt(sigma_zeta*C-1*sigma_zeta)
        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))]');

        %Polar projection: r cos, r sin
        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);

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


elseif MDtype==2

```

```

    tetha=[-0.5*pi:0.01:pi]; % Span the angle for onlt tension model
    %Implementing McAuley bracket: x*(x>0) = x if x>0, 0 if x =< 0
    % sigma+ = <sigma>
    %***********************************************************
    %* 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([mac(m1(i)) mac(m2(i)) 0 mac(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
    % Comment/delete lines below once you have implemented this case
    % *******************************************************
% %     menu({'Damage surface "NON-SYMMETRIC" has not been implemented yet. '; ...
% %         'Modify files "Modelos_de_dano1" and "dibujar_criterio_dano1"' ; ...
% %         'to include this option'},  ...
% %         'STOP');
% %     error('OPTION NOT AVAILABLE')
theta= [0:0.01:2*pi]; % Span the angle for non-symmetric model
%* RADIUS
    D=size(theta);                         %*  Range
    m1=cos(theta);                         %*
    m2=sin(theta);                         %*
    Contador=D(1,2);                       %*


    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;
    for i=1:Contador
        % Radius is the tau_sig = q/sqrt(sigma_zeta*C-1*sigma_zeta)
        if (theta(i) ≥ 0.5*pi) && (theta(i) ≤ pi) % If in second quadrant
            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))]'))/(m2(i) - (m1(i)/n));
        elseif (theta(i) > pi) && (theta(i) < 1.5*pi) % If in third quadrant
            radio(i)= n*q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*...
                ce_inv*[m1(i) m2(i) 0 ...
            nu*(m1(i)+m2(i))]');
```

```
        elseif (theta(i) >= 1.5*pi) % If in fourth quadrant
            radio(i) = n*(q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*...
                ce_inv*[m1(i) m2(i) 0 ...
            nu*(m1(i)+m2(i))]'))/(n*m1(i) - m2(i));
        else % If in first quadrant
            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))]');
        end
        %Polar projection: r cos, r sin
        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);

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



end

return
```