Computational Solid Mechanics. Assignment 1

Adrià Galofré

April 7th 2017

1 Introduction

The present work explains the modifications done to the MATLAB code given in order to implement the features requested. The original code that reproduces a Constitutive Isotropic Damage model was able to simulate the following cases:

- Plane Strain
- Symmetric Tension
- Linear damage model
- Inviscid case

The objective has been to extend the code to asses:

- Non-Symmetric Tension damage model
- Tension only
- Linear/Exponential damage evolution
- Plane Strain Viscous case

2 Code features implemented

In this section the most relevant code implementations to extend the code are shown and explained.

2.1 Rate independent model

2.1.1 Implementation of the Continuum Isotropic damage "non-symmetric tension-compression damage" model and the "tension-only" damage model.

In order to extend the code, "non-symmetric tension-compression" damage model and "tension-only" had been implemented. This models use different energy norms that had been implemented in the file *Modelos_de _dano1*. For the non-symmetric damage model the energy norm is defined as:

$$\tau_\sigma = (\theta + \frac{1-\theta}{n})\sqrt{\sigma : C : \sigma}$$

this θ parameter is introduced, which is a weight factor dependant on the stress state σ defined as:

$$\theta = \frac{\sum_{i=1}^{3} \langle \sigma_i \rangle}{\sum_{i=1}^{3} |\sigma_i|}$$

And for the tension only one:

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

Also file *dibujar_criterio _dano1* has been modified to plot the damage surfaces.

2.1.2 Implementation of Linear and Exponential Hardening/Softening (H<0 and H>0).

The hardening/softening law defines how the damage surface will increase or shrink due to the damage model. Linear damage law was already implemented and the exponential one has been code. This two laws are located inside the rmap dano1.m file in the code.
Linear Hardening/softening:

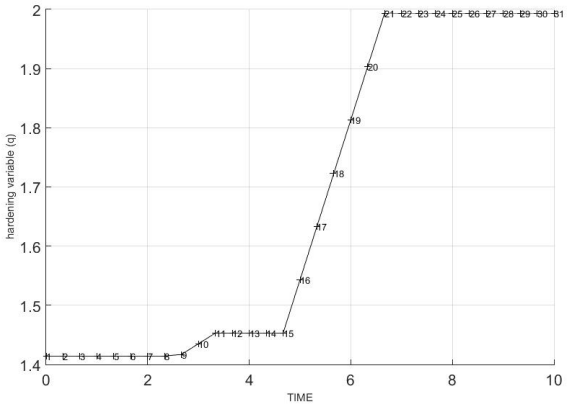
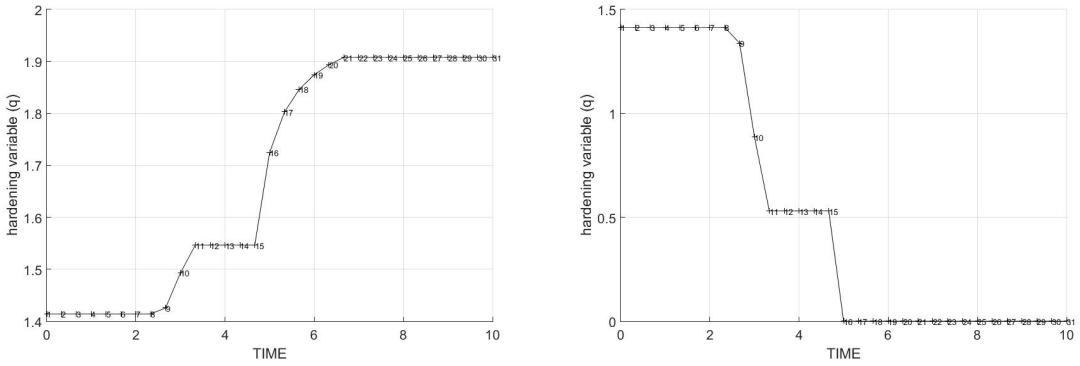


Figure 1: Linear Hardening Law

Exponential Hardening/softening law:
It can be observed when r becomes bigger than r_0 an exponential law drives the response of the model:



(a) Exponential Law for $q_\infty > r_0$

(b) Exponential Law for $q_\infty < r_0$

Figure 2: Exponential Hardening/Softening Law

2.2 Rate dependent model

2.2.1 Continuum Isotropic visco-damage

On this second part on Rate Dependent models, the integration code for a continuum isotropic visco-damage model has been implemented.Now, viscosity and the integration will appear on the model. First thing that had

been modified from the code is the way in which it is decided if we are in loading or in elastic unloading. Instead of the τ_ε used in Part I, in order to take into account the α integration parameter, $\tau_{\varepsilon_{n+\alpha}}$ is used:

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

The integration parameter and the η viscosity parameter introduced play an important role when loading as can be seen in the next equation:

$$r_{n+1} = \frac{[\eta - \triangle t(1-\alpha)]}{\eta + \alpha\triangle t} + \frac{\triangle t}{\eta + \alpha\triangle t}\tau_{\varepsilon_{n+\alpha}}$$

3 Validation of the code

3.1 Rate independent models

Some different stress tests had been run in order to analyse the results and assure that the implementation of the different damage models had been done correctly. All the cases presented below have their start in the non-stress state ($\sigma_1 = 0, \sigma_2 = 0$) and the next material and model properties:

$$\begin{aligned} YOUNG_M &= 20000 \\ POISSON &= 0.3 \\ HARDSOFT\_MOD &= 0.1 \\ YIELD\_STRESS &= 200 \\ HARDTYPE &= LINEAR \end{aligned}$$

Cases 1-3 correspond to "non-symmetric tension-compression" damage model and cases 4-5 to "tension-only" damage model.

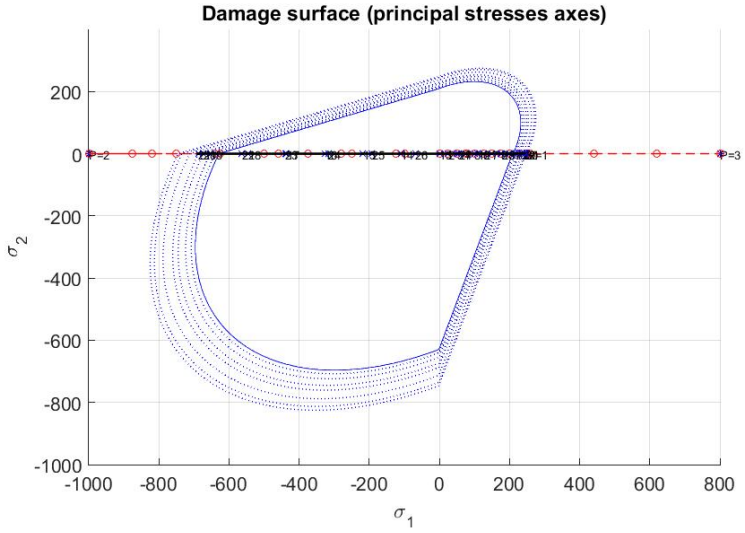
3.1.1 Case 1

Stress Path:

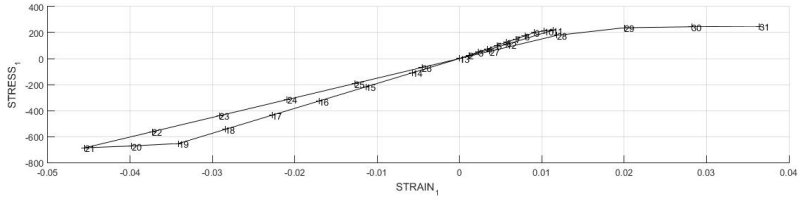
$$\begin{aligned} \triangle\sigma_1 &= 250 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= -1000 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= 800 & \triangle\sigma_2 &= 0 \end{aligned}$$

In this case all the loading/unloading is done uniaxially. As it can be observed in Figure 3, on the three different increments of stress applieds the

damage surface is reached. As consequence of reaching the end of this damage surface on the Stress-Strain Graphic some horitzontal plateau appears either when the stress is in the first quadrant and second one.



(a) Stress-Stress Graphic



(b) Stress-Strain Graphic

Figure 3: Stress-Stress and Stress-Strain representations

3.1.2 Case 2

Stress Path:

$$\begin{aligned} \triangle\sigma_1 &= 250 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= -1000 & \triangle\sigma_2 &= -1250 \\ \triangle\sigma_1 &= 800 & \triangle\sigma_2 &= 550 \end{aligned}$$

Now only the first increment of loading is uniaxial. It can be observed in both Figure 4 and Figure 5. In the first we see clearly, how as consequence of reach the damage surface, it is no possible to follow the first loading path. The damage evolution is always positive or equal to zero, that means, when damage is done it can not be recovered by unloading or loading in the opposite way.

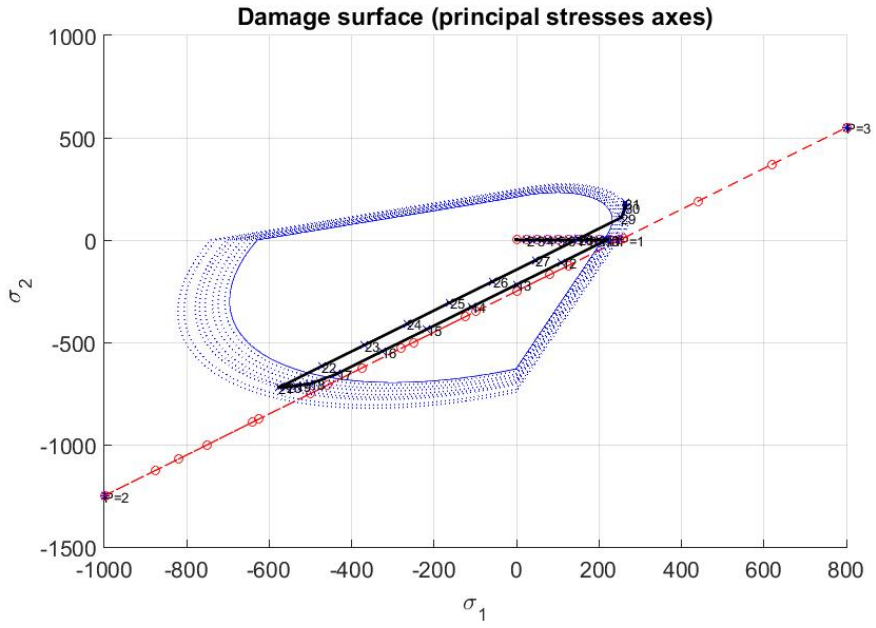
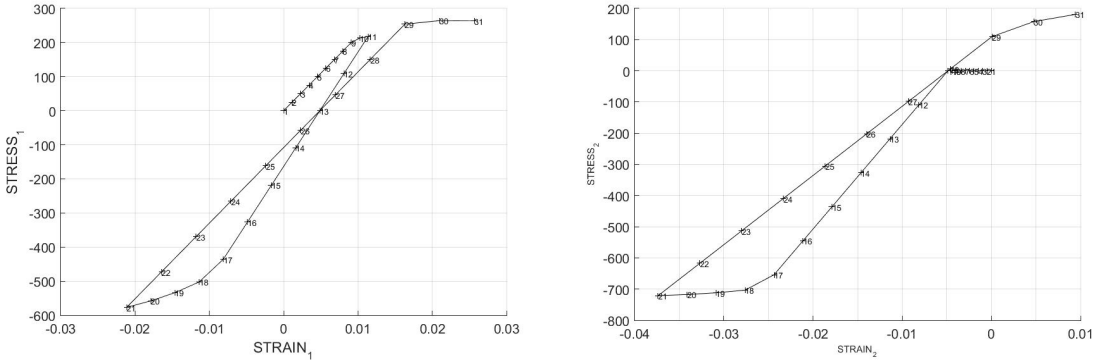


Figure 4: Stress-Stress Graphic



(a) $Stress_1 - Strain_1$ Graphic (b) $Stress_2 - Strain_2$ Graphic

Figure 5: Stress-Strain representations

3.1.3 Case 3

Stress Path:

$$\begin{aligned} \triangle\sigma_1 &= 250 & \triangle\sigma_2 &= 250 \\ \triangle\sigma_1 &= -1000 & \triangle\sigma_2 &= -1000 \\ \triangle\sigma_1 &= 800 & \triangle\sigma_2 &= 800 \end{aligned}$$

Now, a fully biaxial case is tested. As in Case 2, Figure 7 shows how in damage surface the strain increases more with less stress increment.

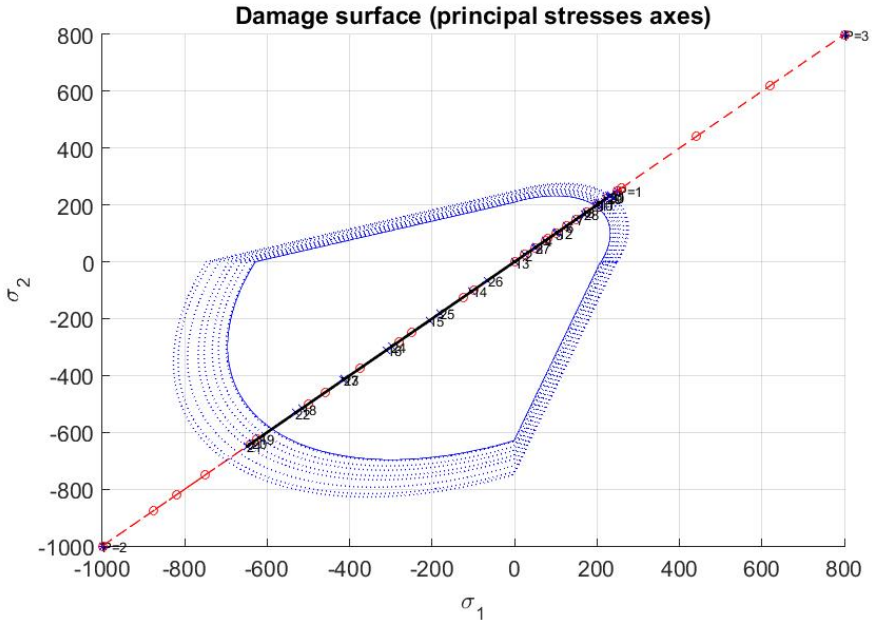
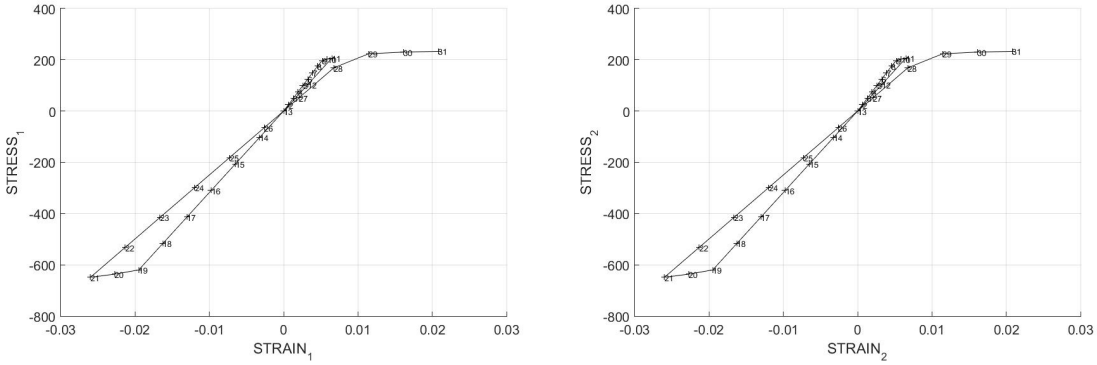


Figure 6: Stress-Stress Graphic



(a) $Stress_1 - Strain_1$ Graphic

(b) $Stress_2 - Strain_2$ Graphic

Figure 7: Stress-Strain representations

3.1.4 Case 4

Stress Path:

$$\begin{aligned} \triangle\sigma_1 &= 250 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= -1000 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= 800 & \triangle\sigma_2 &= 0 \end{aligned}$$

This is the first of the "Tensile-only" damage model. In Figure 8, the damage surface of this model is represented. As it can be deduced from that figure, the damage surface can be only reached for tensile stresses.

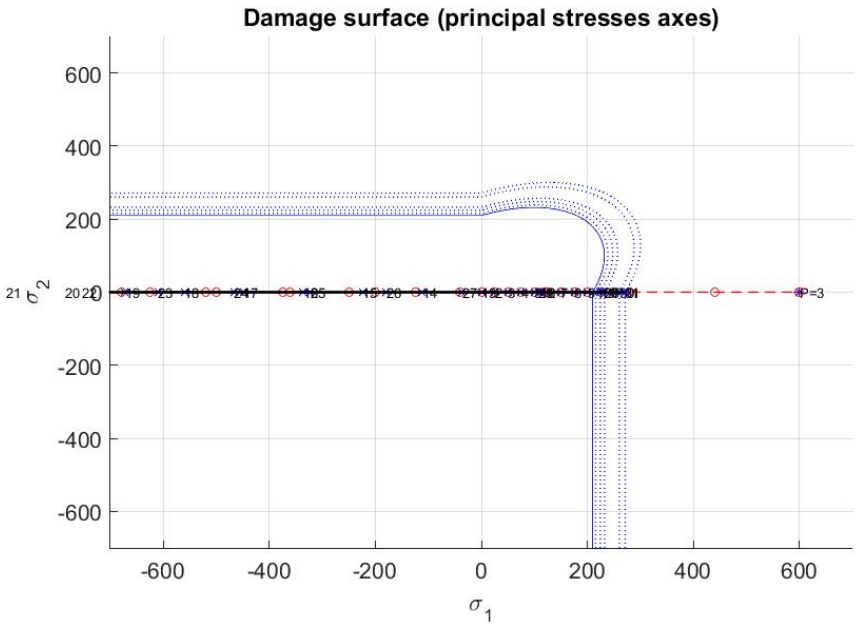


Figure 8: Stress-Stress Graphic

In figure below, it can be ho the strain keep grow for negative stress values but, as said in paragraph before, how for a high enough tensile, damage surface is reached.

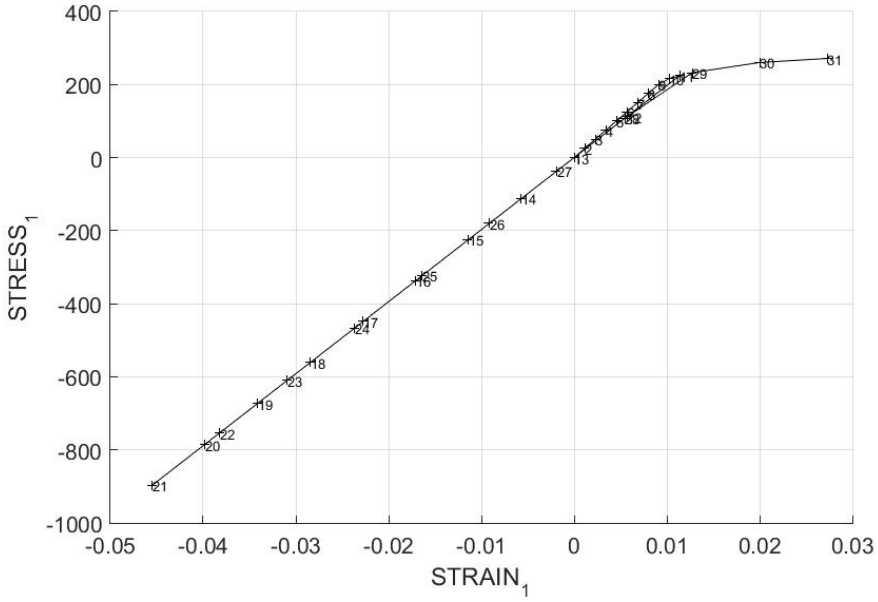


Figure 9: $Stress_1 - Strain_1$ Graphic

3.1.5 Case 5

Stress Path:

$$\begin{aligned} \triangle\sigma_1 &= 250 & \triangle\sigma_2 &= 0 \\ \triangle\sigma_1 &= -1000 & \triangle\sigma_2 &= -1250 \\ \triangle\sigma_1 &= 800 & \triangle\sigma_2 &= 550 \end{aligned}$$

As done with the "nonsymetric" damage model, for this second case of "Tension-only", the path is described with a first uniaxial loading followed by one biaxial loading step and a third biaxial loading. As Figure 10 shows, the first uniaxial loading and the third biaxial loading reach the damaged surface. Alternatively, a path where an horizontal or a vertical damage surface were reached could be done. The conclusions would have been the same than for this case. Those surfaces can only be reached when a positive increment of stress is applied, and once reached, we can see that the strain increases faster.

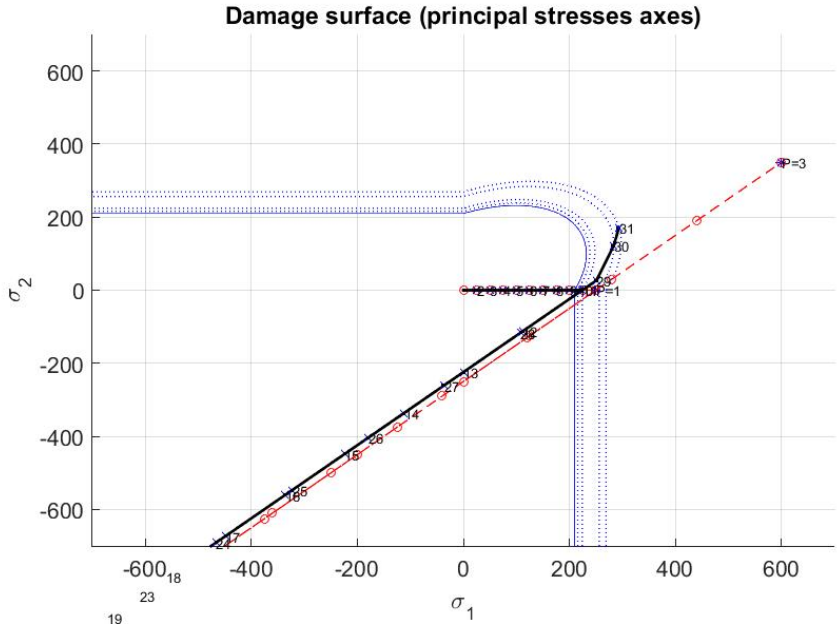
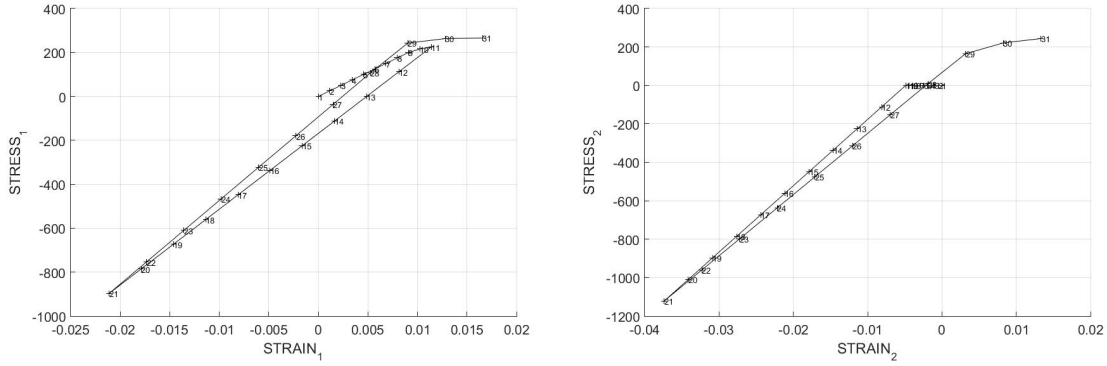


Figure 10: Stress-Stress Graphic



(a) $Stress_1 - Strain_1$ Graphic

(b) $Stress_2 - Strain_2$ Graphic

Figure 11: Stress-Strain representations

3.2 Rate dependent models:

As done for testing the different damage models implemented, some cases are presented blow to test the correct implementation of the viscosity. In

this case, an analysis of the influence of some parameters such as η or α is presented. All the simulations doone to analyse the role of this new parameters has been done with the symetric tension-compresion damage model.

1. Material behaviour for different viscosity parameters η: The different values for η viscosity parameter where used:

$$\eta = 0(black)$$
$$\eta = 0.25(blue)$$
$$\eta = 0.5(green)$$
$$\eta = 1(orange)$$

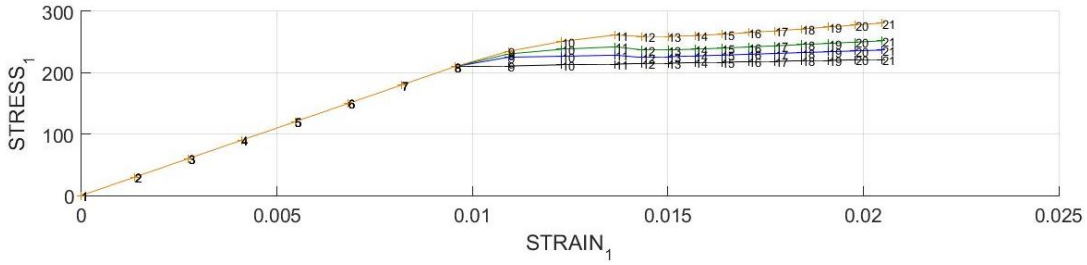


Figure 12: Stress-Stress Graphic. Viscosity.

As Figure 12 shows, when increasing the viscosity parameter, more stress is needed to get the same strain.

2. Influence in the response of different Strain rates ($\dot{\varepsilon}$):

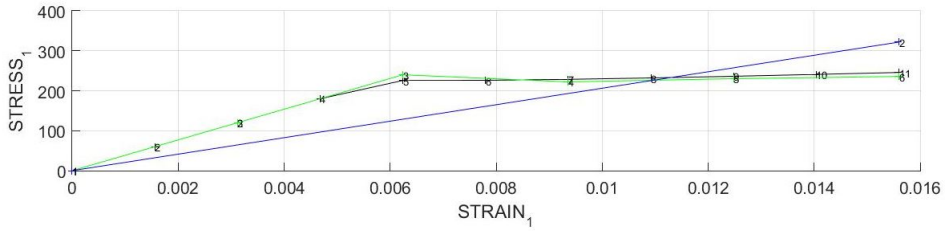
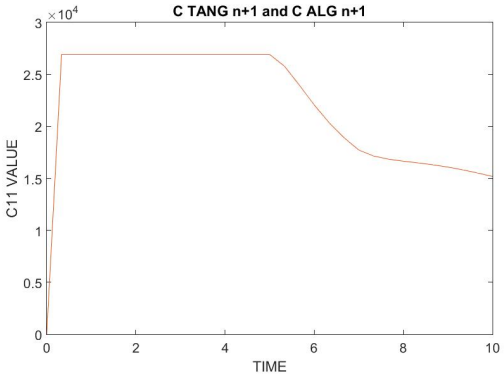


Figure 13: Stress-Stress Graphic. Strain rate($\dot{\varepsilon}$)

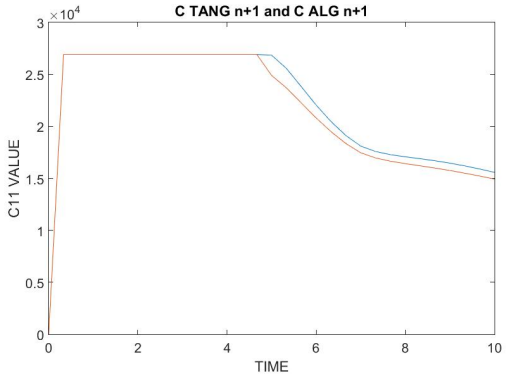
Figure 13 shows for the same loading process three different strain rates. The blue curve uses only one step to load, while the green and the black uses 5 and 10 respectively. So, for strain rates too big we are not able to represent accurately the behaviour of the material.

3. Different α values for the time integration method:
   The integration parameter (α) has been studied too. Figure 14 shows for each value used the evolution of the C_{11} of the C_{tang}(always in blue) and the C_{alg}(in red) tensors. It can be observed that increasing α the difference between the two tensors increases due to for α and η different from zero an extra term appears when calculating C_{alg} when the viscous efects and the Hardening parameter play and influence role:

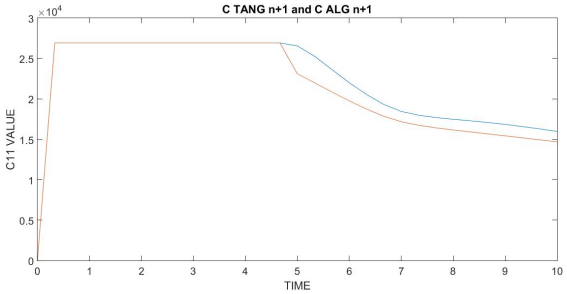
$$C_{alg} = (1 - d_{n+1})C + \frac{\alpha \triangle t}{\eta + \alpha \triangle t} \frac{1}{\tau_{\varepsilon_{n+1}}} \frac{H_{n+1} r_{n+1} q(r_{n+1}}{(r_{n+1})^2} (\bar{\sigma_{n+1}} x \bar{\sigma_{n+1}})$$



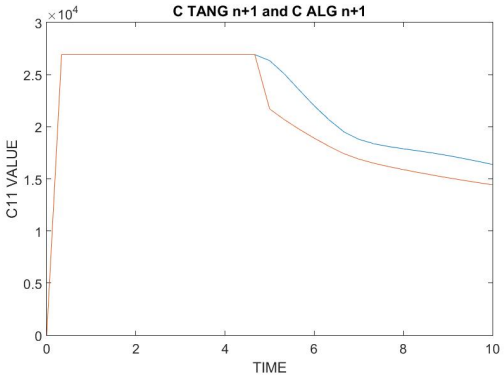
(a) $\alpha = 0$: C_{11} component for C_{alg} and C_{tang}



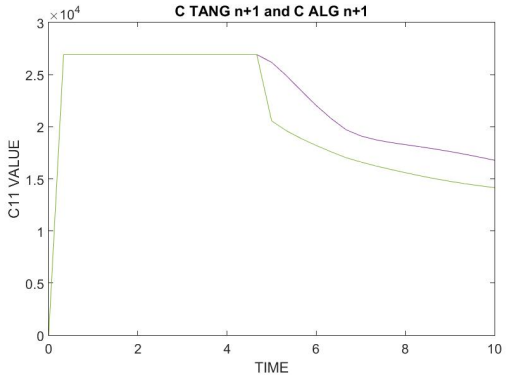
(b) $\alpha = \frac{1}{4}$: C_{11} component for C_{alg} and C_{tang}



(c) $\alpha = \frac{1}{2}$: C_{11} component for C_{alg} and C_{tang}



(d) $\alpha = \frac{3}{4}$: C_{11} component for C_{alg} and C_{tang}



(e) $\alpha = 1$: C_{11} component for C_{alg} and C_{tang}

Figure 14: α parameter influence on the C_{11} component

4 Conclusions

After running the examples proposed to asses the correctness of the code extension implemented the behaviour of the strain-stress paths and the evolution of the damage surface are as expected from the theory.
Regarding the rate dependent models it is observed that with viscosity the stress paths can exceed the damage surface. This effect of the inertial forces is in accordance with the viscous theory. Finally it is proved that increasing time integration parameter makes the tangent and algorithmic constitutive operator differ.
To sum up we can conclude that the code is capable to describe the expected response to the different cases of the isotropic continuum damage model.

5 Anex. Matlab Code

5.1 main_nointeractive

```
clc
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program for modelling damage model
% (Elemental gauss point level)
% ------------------
% Developed by J.A. Hdez Ortega
% 20-May-2007, Universidad Polit cnica de Catalu a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%profile on

% ---------------------------
% ****************
% INPUTS
% ****************

% YOUNG's MODULUS
% ----------------
YOUNG_M =cinput('Young Modulus:',20000);
%YOUNG_M = 20000 ;
% Poisson's coefficient
% ------------------------
POISSON = cinput('Poisson ratio:',0.3);
%POISSON = 0.3 ;
% Hardening/softening modulus
% ------------------------------
HARDSOFT_MOD = cinput ('HARDSOFT_MOD:',0.1);
%HARDSOFT_MOD = 0.1 ;
% Yield stress
% ------------
YIELD_STRESS= cinput ('Yield Stress:',200);
%YIELD_STRESS = 200 ;
% Problem type  TP = {'PLANE STRESS','PLANE STRAIN','3D'}
% ------------------------- = 1              =2          =3
% -------------
ntype= 2 ;
```

```
% Model      PTC = {'SYMMETRIC','TENSION','NON-SYMMETRIC'} ;
%                          = 1          = 2          = 3
% ------------------------------------------------------
MDtype =cinput('MDtype:',1);
% Ratio compression strength / tension strength
% ------------------------------------------------
n = 3 ;
% SOFTENING/HARDENING TYPE
% ------------------------

HARDTYPE = 'LINEAR'   %LINEAR,EXPONENTIAL

% VISCOUS/INVISCID
% ------------------------
q_inf=2;

VISCOUS = 'YES'
% Viscous coefficient ----
% ------------------------
eta = 0.3 ;
% TimeTotal (initial = 0) -----
% ------------------------
TimeTotal =10;
% Integration coefficient ALPHA
% ------------------------
ALPHA_COEFF = 0.5 ;
% Points ---------------------
% ---------------------------
nloadstates = 3 ;

SIGMAP = zeros(nloadstates,2) ;
%SIGMAP(1,:) =[0  0];
SIGMAP(1,:) =[250  0];
SIGMAP(2,:) =[-1000  0];
SIGMAP(3,:) =[800  0];

% Number of time increments for each load state
% ---------------------------------------------
istep = 10*ones(1,nloadstates) ; %rate dependent
```

```
% VARIABLES TO PLOT
vpx = 'STRAIN_1'  % AVAILABLE OPTIONS: 'STRAIN_1', 'STRAIN_2'
%                       '|STRAIN_1|', '|STRAIN_2|'
% 'norm(STRAIN)', 'TIME', 'r'
vpy = 'STRESS_1'              % AVAILABLE OPTIONS: 'STRESS_1',
%'STRESS_2'
%                       '|STRESS_1|', '|STRESS_2|'
% 'norm(STRESS)', 'TIME', 'DAMAGE VAR.',
%'hardening variable (q)','damage variable (d)'
%  'internal variable (r)'

%  3) LABELPLOT{ivar}
-%-> Cell array with the label string for
%                                       variables of "varplot"
%
LABELPLOT = {'hardening variable (q)','internal variable (r)',
'damage variable (d)'};

%%%%%%%%%%%%%%%%%%%%55 END INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Plot Initial Damage Surface and effective stress path
strain_history = PlotIniSurf(YOUNG_M,POISSON,YIELD_STRESS,SIGMAP,
ntype,MDtype,n,istep);



E       = YOUNG_M       ;
nu      = POISSON       ;
sigma_u = YIELD_STRESS  ;



switch  HARDTYPE
    case 'LINEAR'
        hard_type = 0   ;
    otherwise
        hard_type = 1   ;
end
switch  VISCOUS
    case 'YES'
```

```
        viscpr = 1       ;
    otherwise
        viscpr = 0
                 eta=0
                 ALPHA_COEFF=1;
end


Eprop    = [E nu HARDSOFT_MOD sigma_u hard_type viscpr eta
ALPHA_COEFF]                ;



% DAMAGE MODEL
% ------------
[sigma_v, vartoplot, LABELPLOT_out, TIMEVECTOR, CALG, CTANG]=
damage_main(Eprop, ntype, istep, strain_history, MDtype, n, TimeTotal, q_inf);



try; LABELPLOT; catch; LABELPLOT = LABELPLOT_out ; end ;



% PLOTTING
% --------

ncolores = 3 ;
colores =  ColoresMatrix(ncolores);
markers = MarkerMatrix(ncolores) ;
hplotLLL = [] ;

figure(2)

for i = 2:length(sigma_v)
    stress_eig  = sigma_v{i} ; %eigs(sigma_v{i}) ;
    tstress_eig = sigma_v{i-1}; %eigs(sigma_v{i-1}) ;
    hplotLLL(end+1) = plot([tstress_eig(1,1)
    stress_eig(1,1) ],[tstress_eig(2,2) stress_eig(2,2)],'LineWidth',2,'co
```

```
    'Marker',markers{1},'MarkerSize',2);
    plot(stress_eig(1,1),stress_eig(2,2),'bx')
    text(stress_eig(1,1),stress_eig(2,2),num2str(i))


    % SURFACES
    % ------

end




% % SURFACES
% % ------
% 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



DATA.sigma_v     = sigma_v      ;
DATA.vartoplot   = vartoplot    ;
DATA.LABELPLOT   = LABELPLOT    ;
DATA.TIMEVECTOR = TIMEVECTOR   ;
DATA.strain = strain_history ;



plotcurvesNEW(DATA,vpx,vpy,LABELPLOT,vartoplot) ;

%% Connect to Excel, make it visible and add a worksheet
%xl = actxserver('Excel.Application'); set(xl,'Visible',1);
%xl.Workbooks.Add(1); xls = xl.ActiveSheet;

%% Paste in the MATLAB figures
%print(figure(1), '-dbitmap'); xls.Range('E3').PasteSpecial;
%print(figure(2), '-dbitmap'); xls.Range('I3').PasteSpecial;
```

```
%figure(3)

%plot(TIMEVECTOR,CTANG)

%figure(4)

%plot(TIMEVECTOR,CALG)
```

5.2 damage$_main$

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR
,CALG,CTANG]=damage_main(Eprop,ntype,istep,strain,
MDtype,n,TimeTotal,q_inf)
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
```

```
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);
% 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 = 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) = 1-hvar_n(6)/hvar_n(5); %Damage variable (d)

CTANG=zeros(totalstep+1,1);
CALG=zeros(totalstep+1,1);
for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;

            % Total strain at step "i"
        % ------------------------
            if viscpr==1
            eps_n1=[strain(i-1,:);strain(i,:)];
            else
            eps_n1 = strain(i,:) ;
            end


        %************************************************
        %*       DAMAGE MODEL
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var,Calg,Ctang] = rmap_dano1(eps_n1,sigma_v{i
        MDtype,n,q_inf,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
                C11a=Calg(1,1);
                C11t=Ctang(1,1);
                CTANG(i)=C11t;
                CALG(i)=C11a;



        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %**************************************************
        % 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 ;
                %C11 to plot

        % VARIABLES TO PLOT (set label on cell array LABELPLOT)
        % ----------------------
        vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
        vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
        vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5);%Damage variable (d)
                end

end
```

5.3 rmap_dano1

```
function [sigma_n1,hvar_n1,aux_var,Calg,Ctang] = rmap_dano1 (eps_n1,sigma_v

%***************************************************%*
*
%* 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);
ALPHA_COEFF=Eprop(8);
```

```
%Viscous parameters
viscpr=Eprop(6);
eta=Eprop(7);
%*****************************************************************


%*****************************************************************
%*        initializing
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;

 %comment
 if(r_n<=0.d0)
      r_n=r0;
      q_n=r0;
 end
 %comment

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


%*****************************************************************
f viscpr==1
     [rtrial_n]=Modelos_de_dano1 (MDtype,ce,eps_n1(1,:),
     sigma_v,n);
     [rtrial_n1]= Modelos_de_dano1 (MDtype,ce,eps_n1(2,:),
     sigma_v,n);
     [rtrial]=(1-ALPHA_COEFF)*rtrial_n+ALPHA_COEFF*rtrial_n1;
     eps_n1=eps_n1(2,:);
else
     [rtrial]=Modelos_de_dano1(MDtype,ce,eps_n1,sigma_v,n);
end

%*****************************************************************
%*    Ver el Estado de Carga
```

```
%*
%*    ---------->     fload=0 : elastic unload
%*
%*    ---------->     fload=1 : damage

fload=0;

if(rtrial > r_n)
    %*   Loading

    fload=1;
    delta_r=rtrial-r_n;
    r_n1= ((eta-delta_t*(1-ALPHA_COEFF))/...
    (eta+ALPHA_COEFF*delta_t))*r_n ...
              +(delta_t/(eta+ALPHA_COEFF*delta_t))*rtrial
;

    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
                A=1;

      _n1=q_inf-(q_inf-q_n)*exp(A*(1-rtrial/r_n));


        % 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')
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    end
```

```
        dano_n1    = 1.d0-(q_n1/r_n1);
        sigma_n1   =(1.d0-dano_n1)*ce*eps_n1';
        Ctang=(1-dano_n1)*ce;
        delta_t
        ALPHA_COEFF
        eta
        Calg=Ctang-(ALPHA_COEFF*delta_t)/...
        (eta+ALPHA_COEFF*delta_t)*(1/rtriale)*...
        ((q_n1-H*r_n1)/(r_n1)^2)*(sigma_n1'*sigma_n1);


else

    %* Elastic load/unload
    fload=0;

    r_n1= r_n  ;
    q_n1= q_n  ;
        dano_n1    = 1.d0-(q_n1/r_n1);
        sigma_n1   =(1.d0-dano_n1)*ce*eps_n1';
        Calg=(1-dano_n1)*ce;
        Ctang=Calg;


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