Computational Solid Mechanics
MSc in Computational Mechanics
Universitat Politecnica de Catalunya

Assignment 1

Federico Parisi

20-03-2020

Contents

1. Abstract

In this report is going to be illustrated the implementation of a code simulating the damage behaviours of a material, in terms of its main characteristics.

It will be briefly given an overview over the damage models and its features, then will be shown the implementations done. The code has been written in MatLab. The code has been tested with different situations, in which the loads will increase or decrease in different way, with different damage models. Then the results will be commented in order to check the correctness of the implementations.

The situations will be analyzed under the inviscid and viscous hypothesis with different values of the integration parameter α, viscosity η and time of load application.

2. Rate independent models

The Rate independent models are models in which the deformation rate doesn't depend on the rate at which loads are applied.

In this part will be considered the inviscid case: the *Eprop*(6) in the MatLab code will be set equal to zero.

2.1 Damage models

The damage models are the symmetric, only-tension and non-symmetric. The damage surface is different for any of these models, the code has been implemented in order to be able to plot the right damage surfaces for each model. Each surface will change after applying a load, depending of the hardening/softening modulus that will describe if the material is going to harden or soften. This part will be described in section 2.2.

Here are shown the plots of the different models. The elastic region is defined by the plotted damage surface.

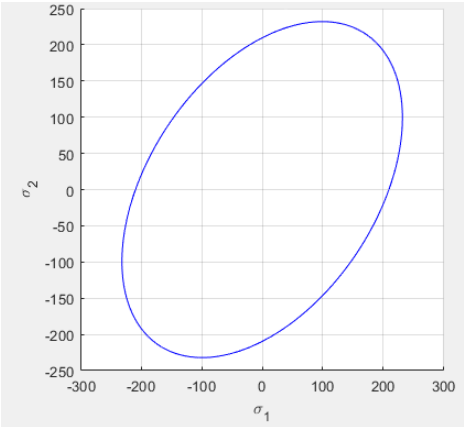

Figure 2.1: Symmetric

In the symmetric model (2.1), the elastic region is the same for compression and traction. This model was already implemented in the code, so no changing has been done.

In the only tension model (2.2), the compression stresses are set equals to zero. That's why the plot presents two asymptotes close to the zeros of σ_1 and σ_2. The implementations done are shown between lines 69 and 91 in the Appendix 8.1 in which is shown the code. At the beginning, the characteristic sizes of the domain are defined, as in the symmetric model. Then the radius is computed considering only the σ positives and the negative ones equals to zero (lines 84-85 in 8.1).

In the non symmetric model (2.3), the damage surface is lead by the compression/traction ratio, defined equal to three. As a consequence, it can be appreciated in the plot that the compression elastic domain is three times bigger than the traction one.

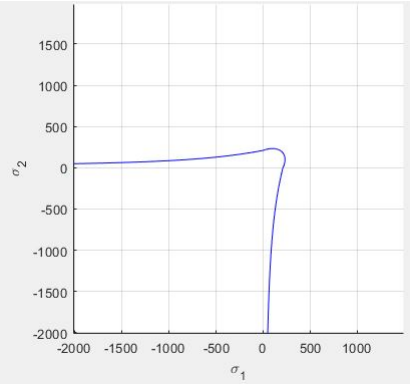

Figure 2.2: Only tension

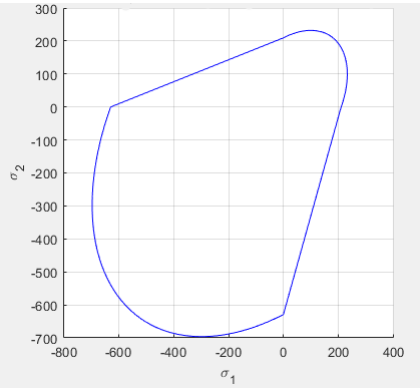

Figure 2.3: Non symmetric

The implementations of the code are in the Appendix A 8.1 between lines 95 and 124. After initializing all the variables in the same way as the symmetric and only tension case, it computes the sum of the positives stresses and the sum of the absolute values in order to compute the coefficient θ. This coefficient is needed to compute the τ, and so the radius, in this way:

$$\tau_\sigma = \theta + \frac{1-\theta}{n}]\sqrt{\sigma : C^{-1} : \sigma} \tag{2.1}$$

For the application of the load paths, in Appendix C 10.1 is computed the *rtrial* for each step.

2.2 Hardening/Softening

The hardening or softening process after applying a load can be linear or exponential. It is the process of a material of changing its yield stress after reaching the previous one and it depends on the material properties, especially on the hardening/softening modulus. Below there is a simulation done considering the hardening process, exaggerating the value of the hard/soft modulus $H = 1$. The softening case will be the opposite.

In Figure 2.4 and Figure 2.5 are presented the new damage surfaces at each iteration. As can be noticed, in the linear case (Figure 2.4) the damage surface is increasing time by time constantly and the damage surface

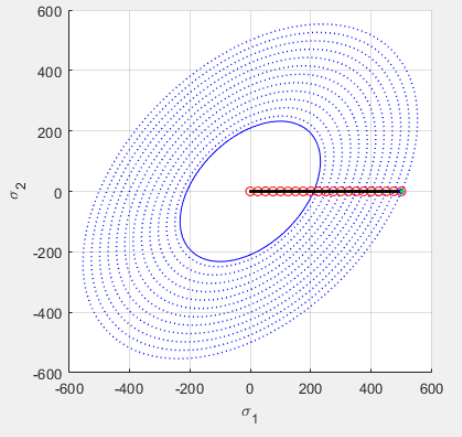

Figure 2.4: Linear

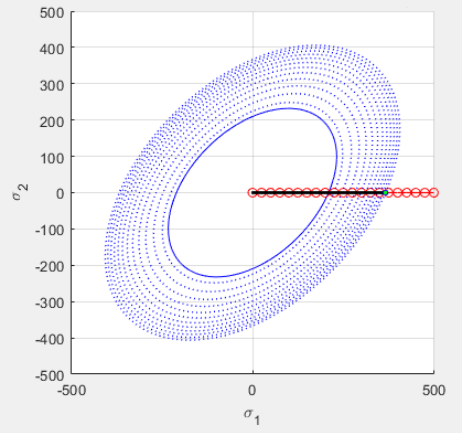

Figure 2.5: Exponential

follows the load path. In the exponential case (Figure 2.5, this increment of the σ_y won't follow completely the load path and the damage surface increases less every step.

These behaviours are well describe by the comparison plot in Figure 2.6. This plot shows the changing of the hardening variable during the load application compared to the internal variable. The red one is the exponential behaviour, and it coincides with the Figure 2.5 in which the damage surface increases less than the linear behaviour. In fact the hardening variable of the linear increases constantly, as the damage surface.

The code has been implemented in the function in Appendix 9.1 between lines 78 to 85. It has been defined the q-n1 for the exponential law following the theory and the q-infinite in order to compute the requested variable. These values are defined as in Figure 2.7.

To assess the correctness of the code, it has been tested for each type of damage model defined above with three different series of loading path both for the linear and exponential hardening/softening.

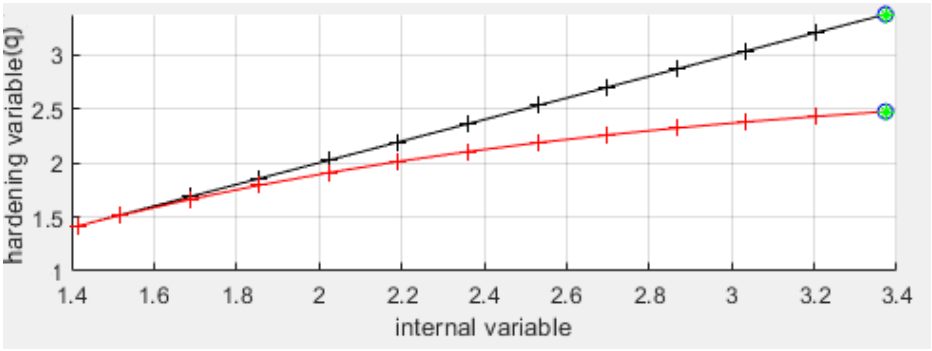

Figure 2.6: Comparison

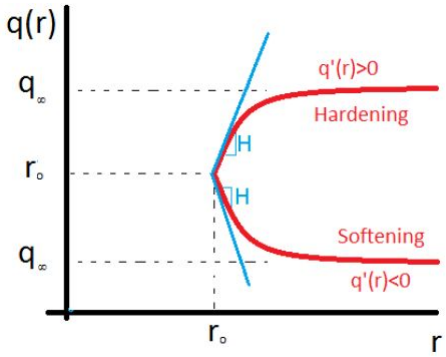

Figure 2.7: q-r

3. Uniaxial loading/unloading

The values chosen are:

$$\begin{aligned} E &= 2000 \\ Yield\ stress\ (\sigma_y) &= 200 \\ Poisson\ ratio\ (\nu) &= 0.3 \\ Harde/soft\ ratio\ (H) &= -0.3\ (softening) \\ ratio\ compression/traction\ (n) &= 3 \end{aligned} \tag{3.1}$$

while the increments α, β, γ are defined as following:

$$\begin{aligned} \alpha &= 300 \\ \beta &= 1350 \\ \gamma &= 800 \end{aligned} \tag{3.2}$$

Being uniaxial, the increment on σ_2 will be always equal to zero. The increment of σ will be the following:

$$\begin{aligned} \Delta\sigma_1^1 &= (\alpha; 0) \\ \Delta\sigma_1^2 &= (-\beta; 0) \\ \Delta\sigma_1^3 &= (\gamma; 0) \end{aligned} \tag{3.3}$$

Symmetric Model: The differences in the symmetric model can be seen very clearly. Figure III in 3.1 shows the difference in softening of the exponential (red line) from the linear. The correctness can be seen analyzing the strain-stress plot (Figure III 3.1). In the first part the load overcome the yield stress and, because of the softening situation, the stress goes down. Same situation in the opposite side, but with compression. Here it happens that for the linear behaviours the material is going to be completely damaged, as it can't hold any σ. On the other hand, the exponential shows a better behaviour.

Only Tension Model: In the only tension model is in evidence (Figure II 3.2) the difference in the tension field in which the linear softening (red line) is lower than the exponential, as expected. In compression it will always remain in the elastic field. In fact, in the plot in Figure III 3.2 can be noticed that the load path cross the damage surface only in one point.

Non Symmetric Model: In the non symmetric damage model (Figure 3.3), the exponential (red line) behaves in the correct way respect to the linear.

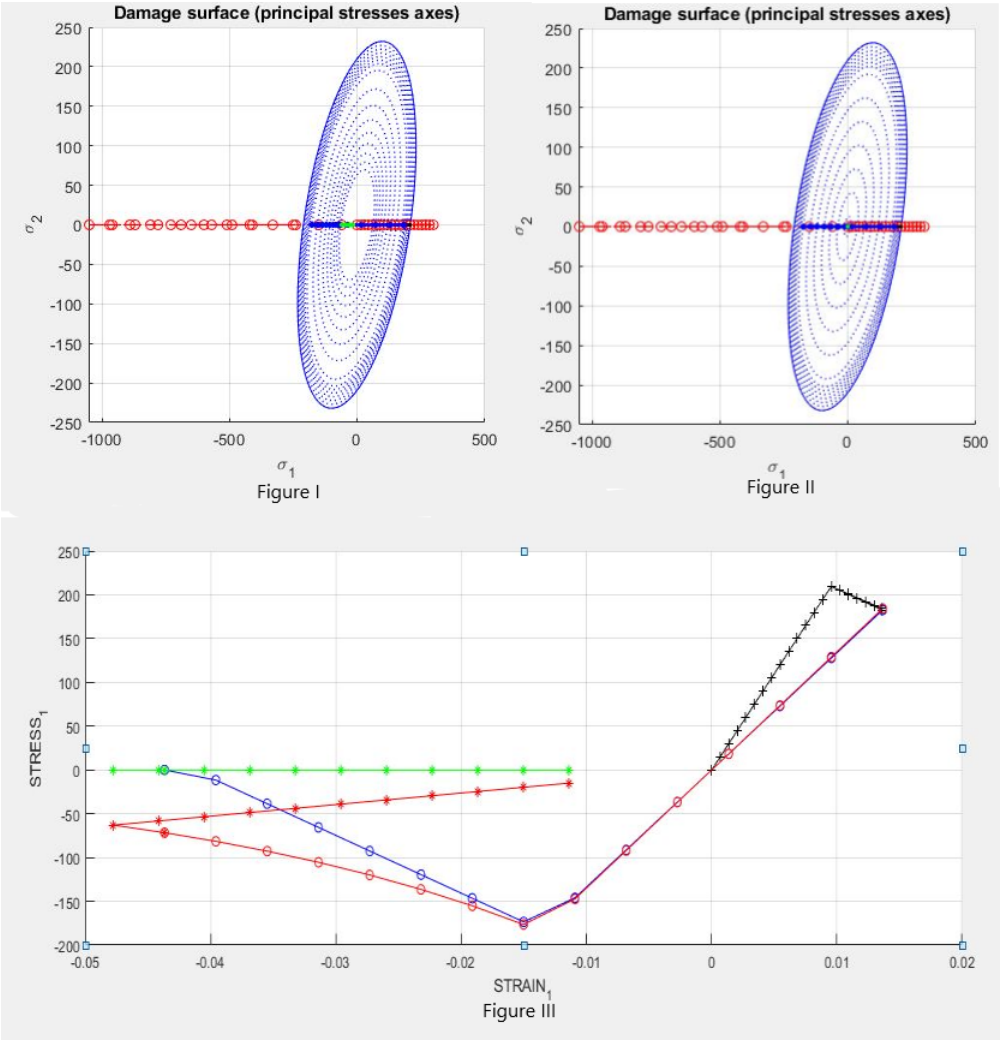

Figure 3.1: Symmetric Model, Case 1

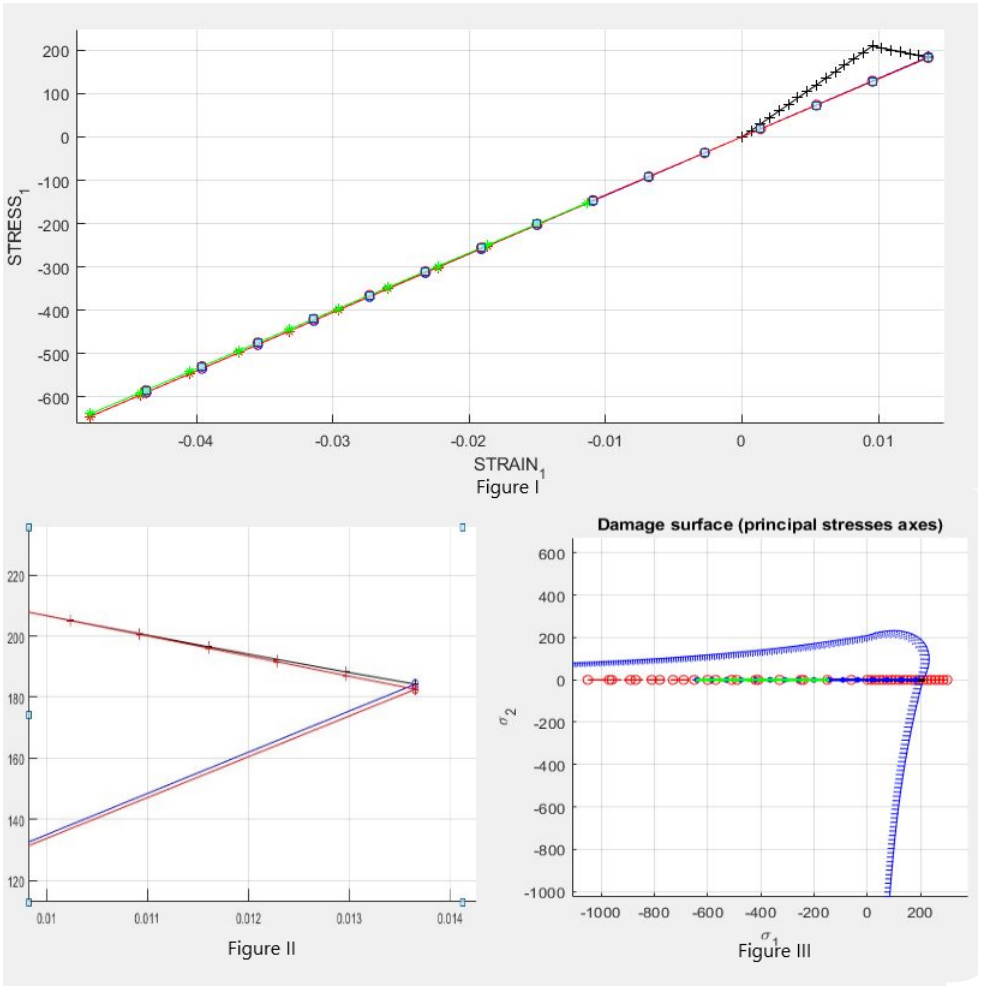

Figure 3.2: Only tension Model, Case 1

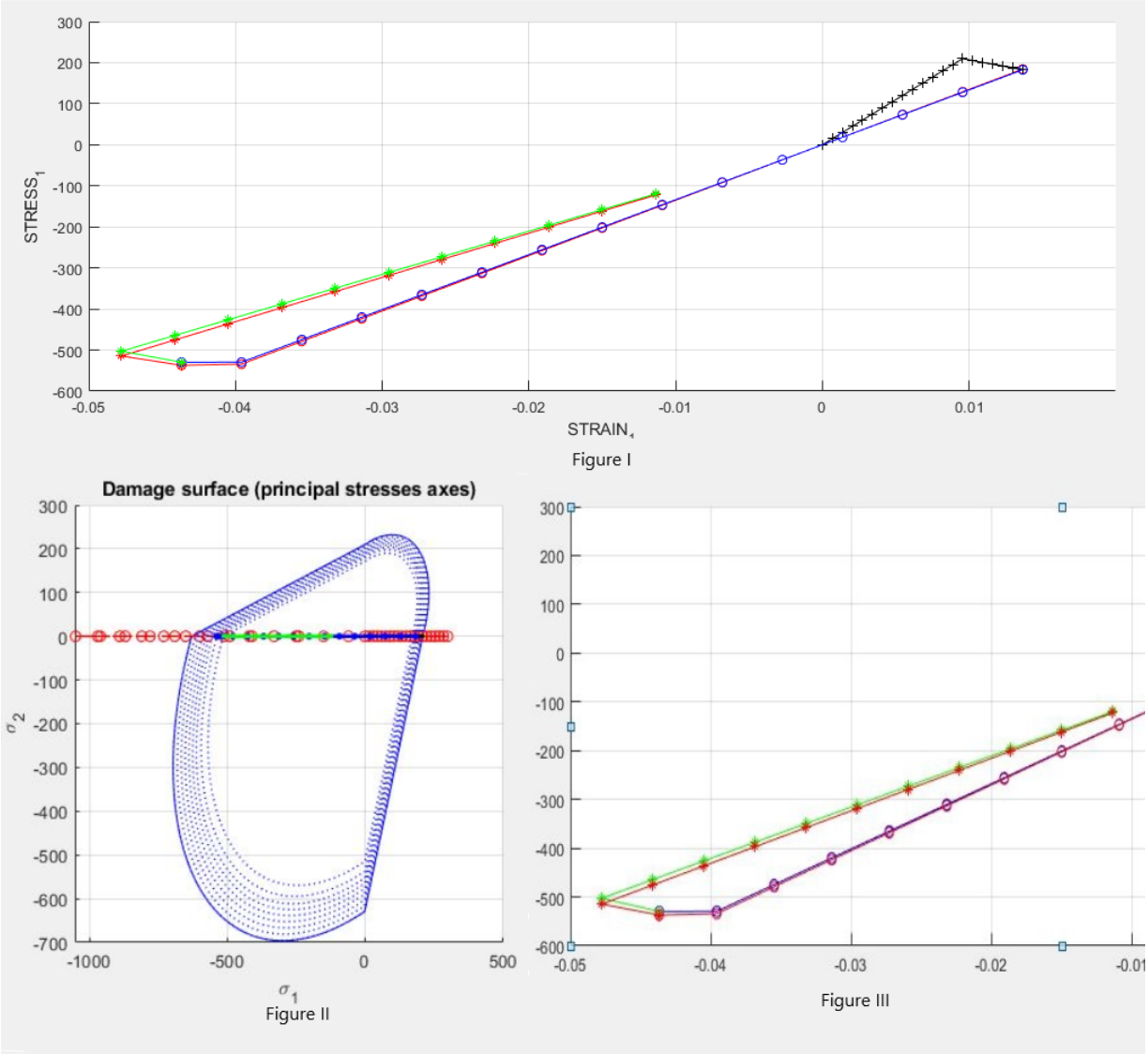

Figure 3.3: No symmetric Model, Case 1

4. Biaxial asymmetric loading/unloading

The values chosen are:

$$\begin{aligned} E &= 2000 \\ Yield\ stress\ (\sigma_y) &= 250 \\ Poisson\ ratio\ (\nu) &= 0.3 \\ Harde/soft\ ratio\ (H) &= 0.3\ (hardening) \\ ratio\ compression/traction\ (n) &= 3 \end{aligned} \tag{4.1}$$

while the increments α, β, γ are defined as following:

$$\begin{aligned} \alpha &= 400 \\ \beta &= 1000 \\ \gamma &= 800 \end{aligned} \tag{4.2}$$

The increment of σ are defined as:

$$\begin{aligned} \Delta\sigma^1 &= (\alpha; 0) \\ \Delta\sigma_2^2 &= (-\beta; -\beta) \\ \Delta\sigma_2^3 &= (\gamma; \gamma) \end{aligned} \tag{4.3}$$

Symmetric Model: In Figure III of 4.1 is shown the plot in which the red line is the exponential hardening law. In this case it is an hardening situation as the hard/soft modulus is positive. In fact it can be appreciate that the σ is increasing after reaching the yield stress. During the second load path, there are two σ. It can be seen the effect of the increment of σ_2 that shows the different behaviours of the linear and exponential case, previously explained in 2.2.

Only Tension and No Symmetric Model: The attention has to be focused on the similarity of the two cases described in 4.2 and 4.3. The load paths has been chosen small enough in order to show that in the non symmetric case, considering the compression/traction ratio defined in 2.1, the behaviours should be the same. In fact, looking at the Figure II of 4.2 and 4.3 the behaviours are almost the same. In each figure, the red line is the exponential.

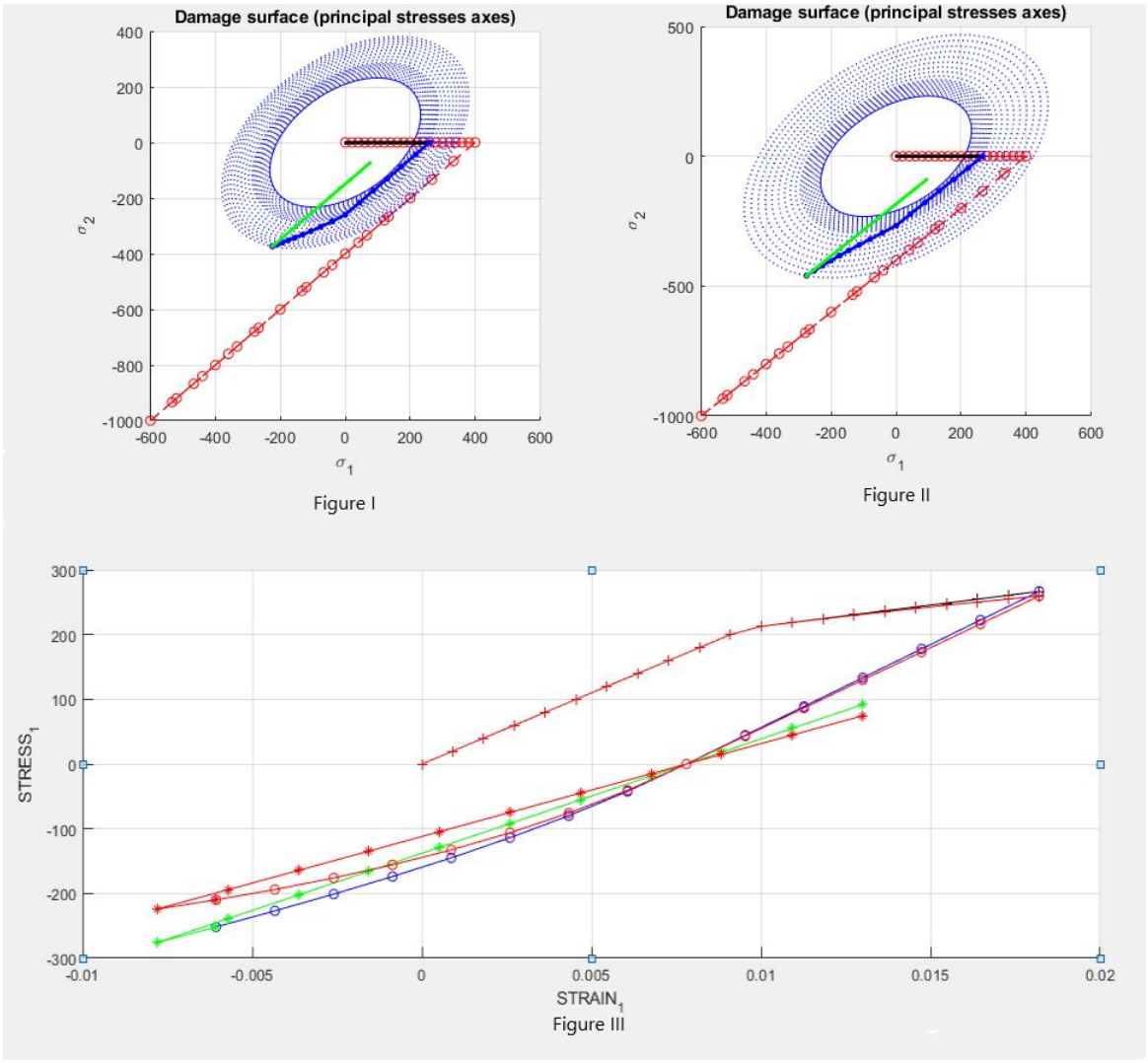

Figure 4.1: Symmetric Model, Case 2

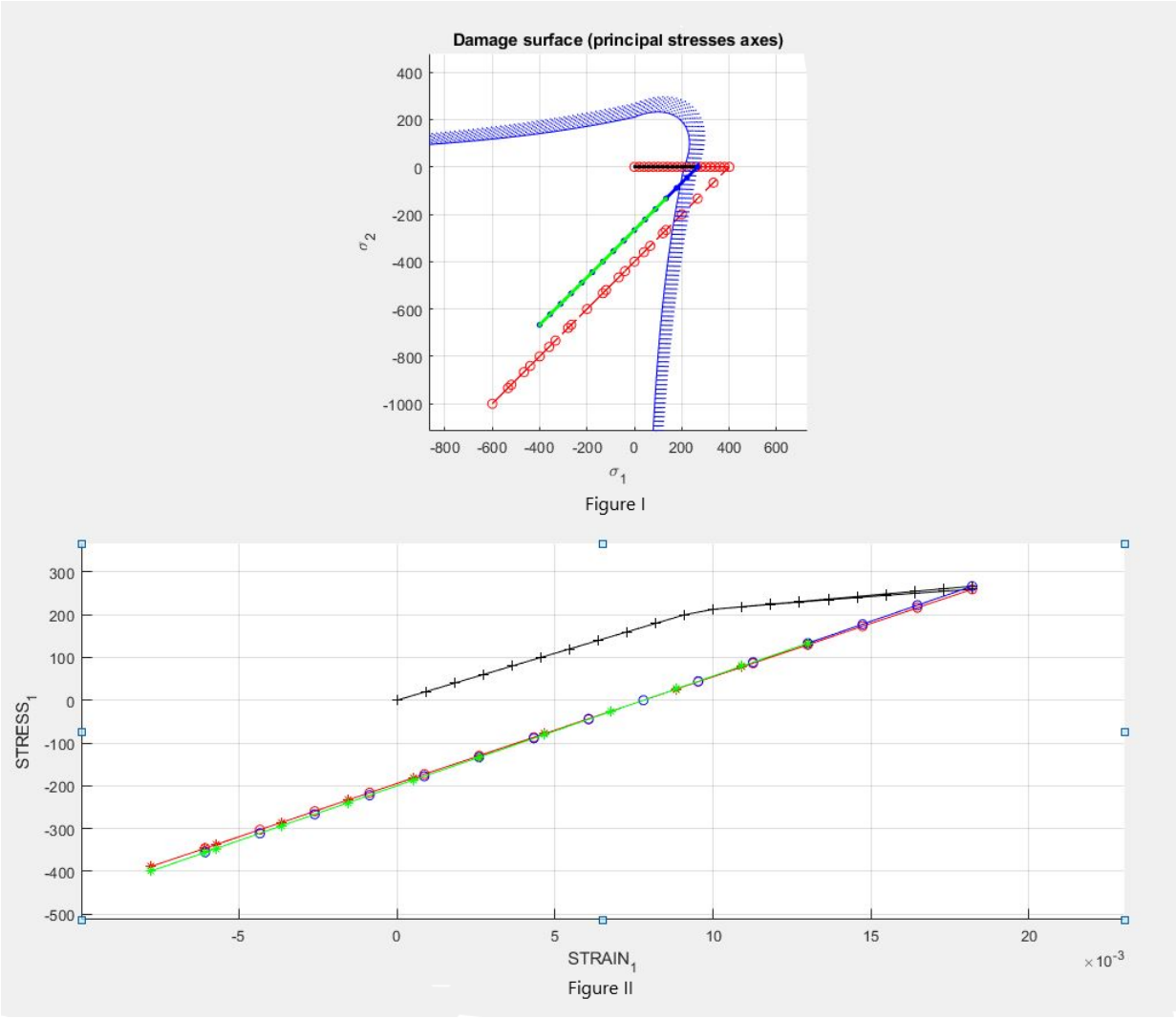

Figure 4.2: Only Tension Model, Case 2

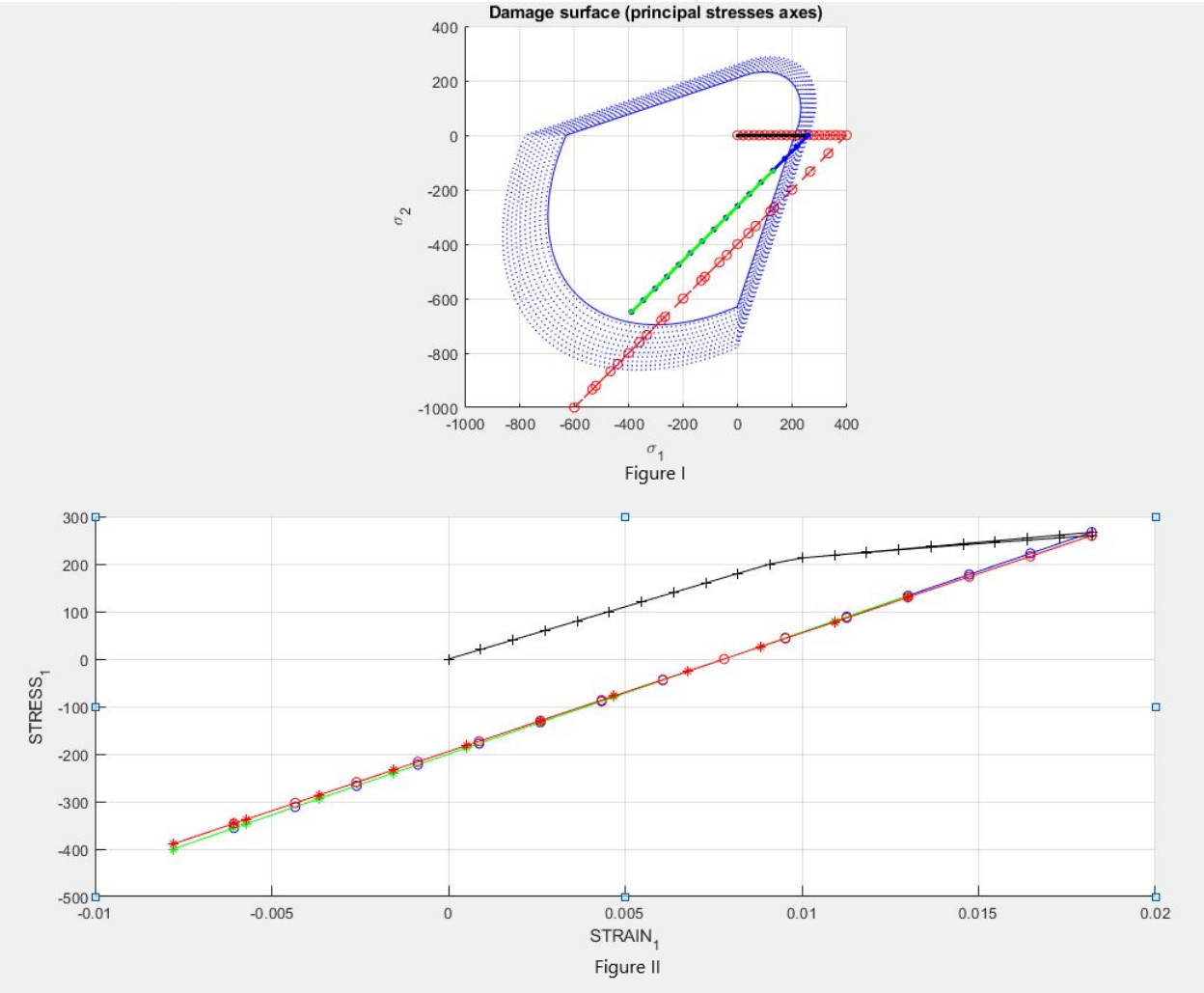

Figure 4.3: No symmetric Model, Case 2

5. Biaxial symmetric loading/unloading

The values chosen are:

$$\begin{aligned} E &= 2000 \\ Yield\ stress\ (\sigma_y) &= 250 \\ Poisson\ ratio\ (\nu) &= 0.3 \\ Harde/soft\ ratio\ (H) &= -0.2\ (softening) \\ ratio\ compression/traction\ (n) &= 3 \end{aligned} \tag{5.1}$$

while the increments α, β, γ are defined as following:

$$\begin{aligned} \alpha &= 350 \\ \beta &= 1100 \\ \gamma &= 850 \end{aligned} \tag{5.2}$$

The increment of σ are defined as:

$$\begin{aligned} \Delta\sigma_2^1 &= (\alpha; \alpha) \\ \Delta\sigma_2^2 &= (-\beta; -\beta) \\ \Delta\sigma_2^3 &= (\gamma; \gamma) \end{aligned} \tag{5.3}$$

Symmetric Model: In figure II of 5.1 is show the plot in function of the norm of the stress and strain, in order to take into account both of the σ. In this case can be better appreciate that the yield stress is the same after the load cycle. In Figure III of 5.1 it can be seen the changing in time of the hardening variable, that makes sens considering the softening situation.

Only Tension Model: It can be clearly seen from Figure 5.2 that the material is damaged only during the first load path. That was expected as is an only tension model and in compression will be always in the elasticity field. The red line is the exponential law, that is less damaged than the linear. The difference is so small that for the plot strain-stress will be plotted only the one for exponential. The difference of behaviours can be better appreciated in Figure II of 5.2.

Non Symmetric Model: As it can be seen, the behaviours are almost the same as the only tension model. The material doesn't get really damaged.

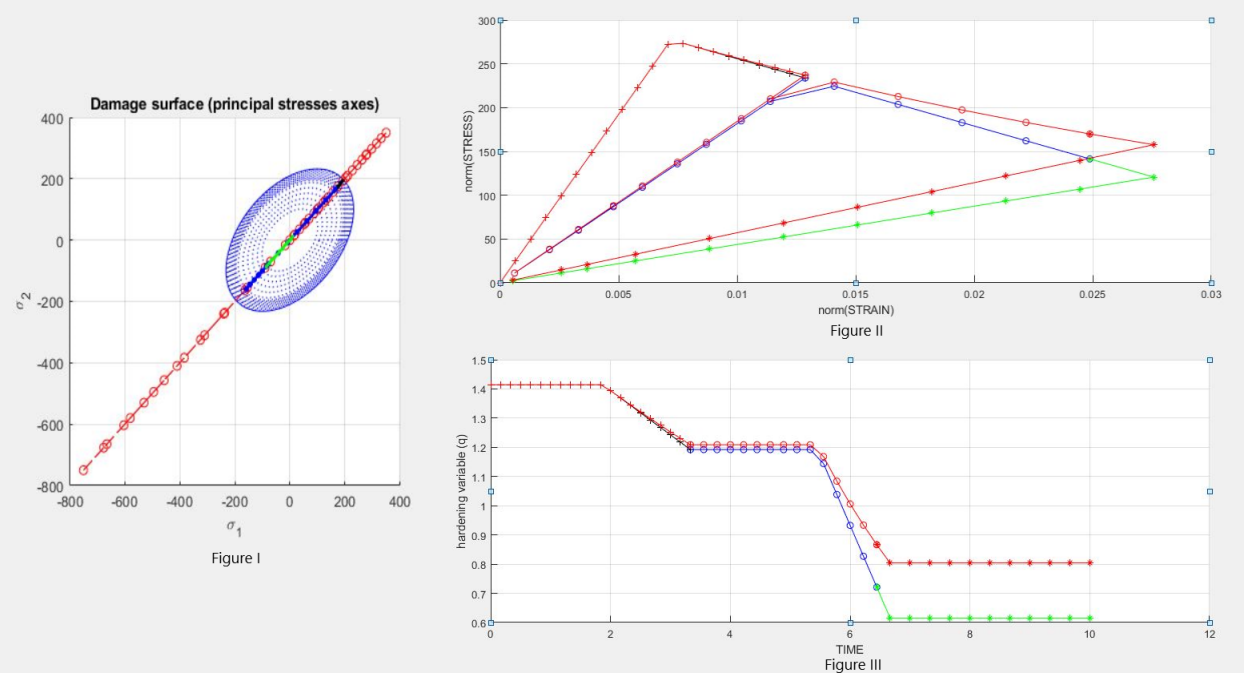

Figure 5.1: Symmetric Model, Case 3

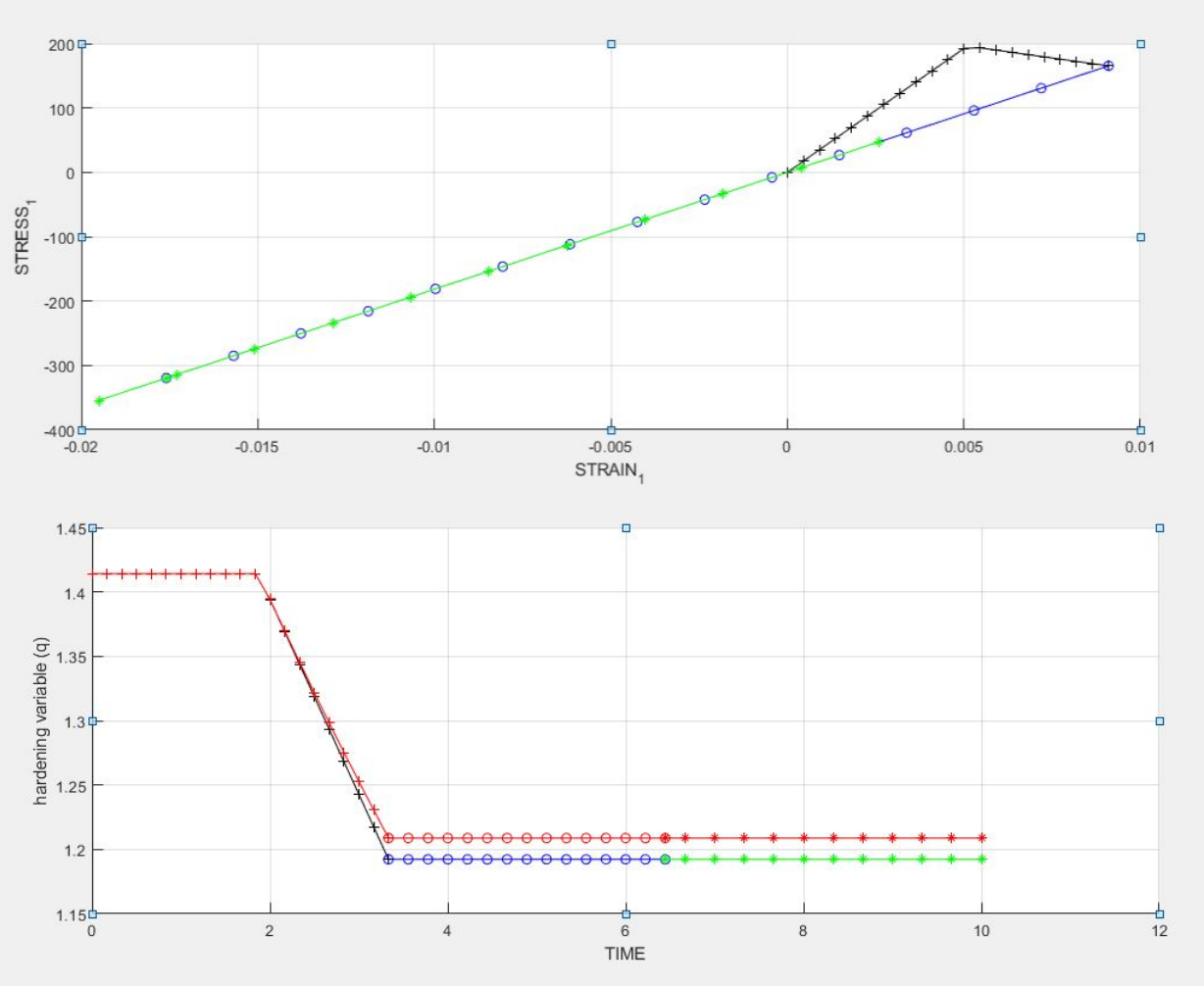

Figure 5.2: Only Tension Model, Case 3

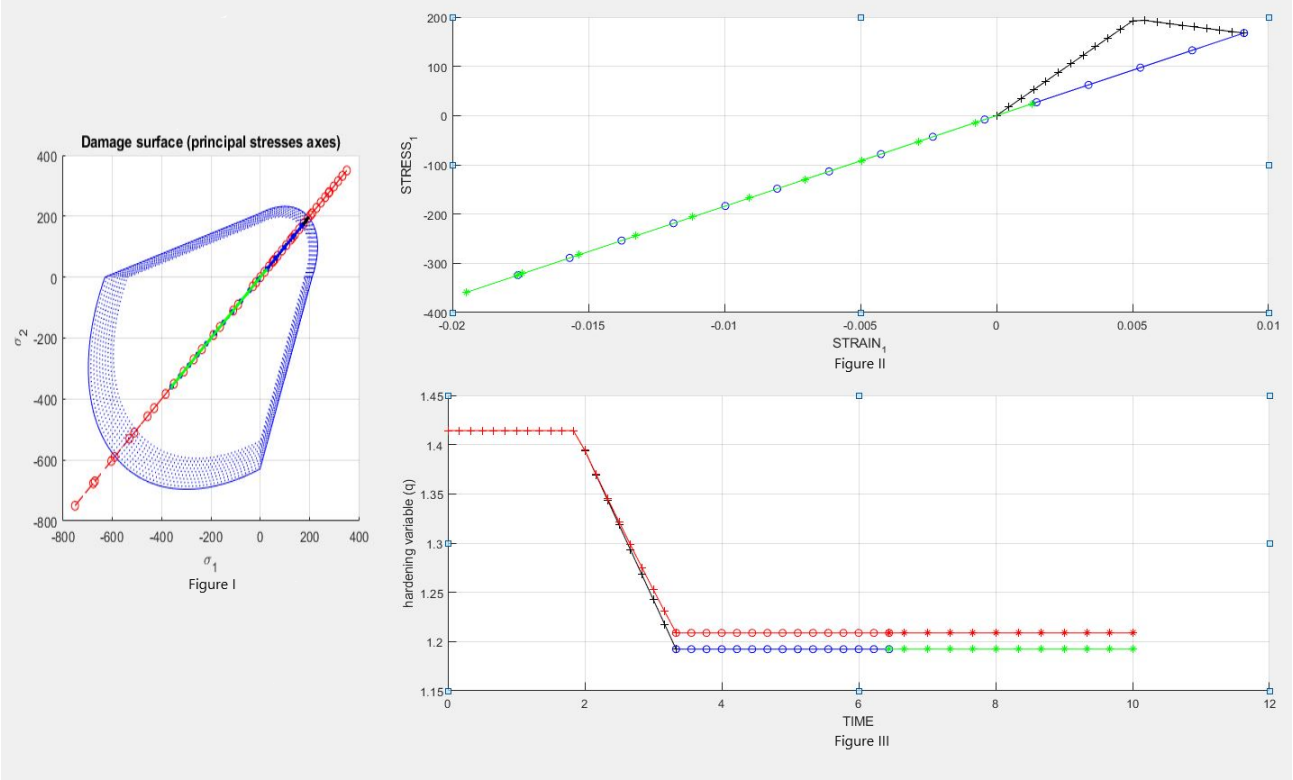

Figure 5.3: No symmetric Model, Case 3

6. Rate dependent models

The rate dependent models are the ones in which the deformation rate depends on the rate at which the load is applied. This models had to be implemented, and the implementations done are shown in the Appendix B 9.1 and D 11.1.

In the function $rmap - dano1$ 9.1 the changing has been done in lines 51 to 61. The code computes the needed values or r_{n1} in order to compute the values for the next iteration. Depending if in visco-damage situation or not, the code will compute this value with the respective law (lines 51-54 if inviscid, lines 55-61 if viscous). To do so, in the viscous case, the code is taking the values of $rtrial$ from the function $Modelos - de - dano1$ in 10.1 but now it considers two different steps as different inputs: $eps - n$ and $eps - n1$ to define $rtrial$ and $rtrial - n$ for computing the requested value $rtrial - an$ depending on the integration coefficient factor α and the time $delt$.

The $eps - n$ written above is computed (line 143, Appendix D 11.1) as the eps at the step $i - 1$, so it has been defined as $eps - n1$ but for the previous step. To compute the σ and the other variables, the code calls the function $rmap - dano1$ as said before. To call it, it needs to give as an input also the new $eps - n$ and the time interval $delt$ defined in line 103 of Appendix D 11.1.

In order to make everything works, the new variables has been given as an input of the function $rmap - dano1$ as can be appreciated in line 148 of Appendix D 11.1 and in line 1 of Appendix B 9.1.

To check the correctness of the implementation, the code has been tested with different values of the viscosity (6.1), different values of the strain rate (6.2) and different values of the integration factor α (6.3).

6.1 Viscosity η

In the plot in Figure 6.1 is shown the behaviours of the material while changing the viscous coefficient η. The simulation has been run with the linear symmetric damage model with the following values:

$$\begin{aligned} E &= 2000 \\ \nu &= 0.3 \\ H &= 0.2(hardening) \\ \alpha &= 1 \\ t &= 20 \\ n &= 3 \end{aligned} \tag{6.1}$$

And it has been considered only one increment of σ:

$$\sigma = (400; 400) \tag{6.2}$$

Has been chosen three different values of η:

$$\begin{aligned} \eta_1 &= 0.3 \\ \eta_2 &= 3 \\ \eta_3 &= 9 \end{aligned} \tag{6.3}$$

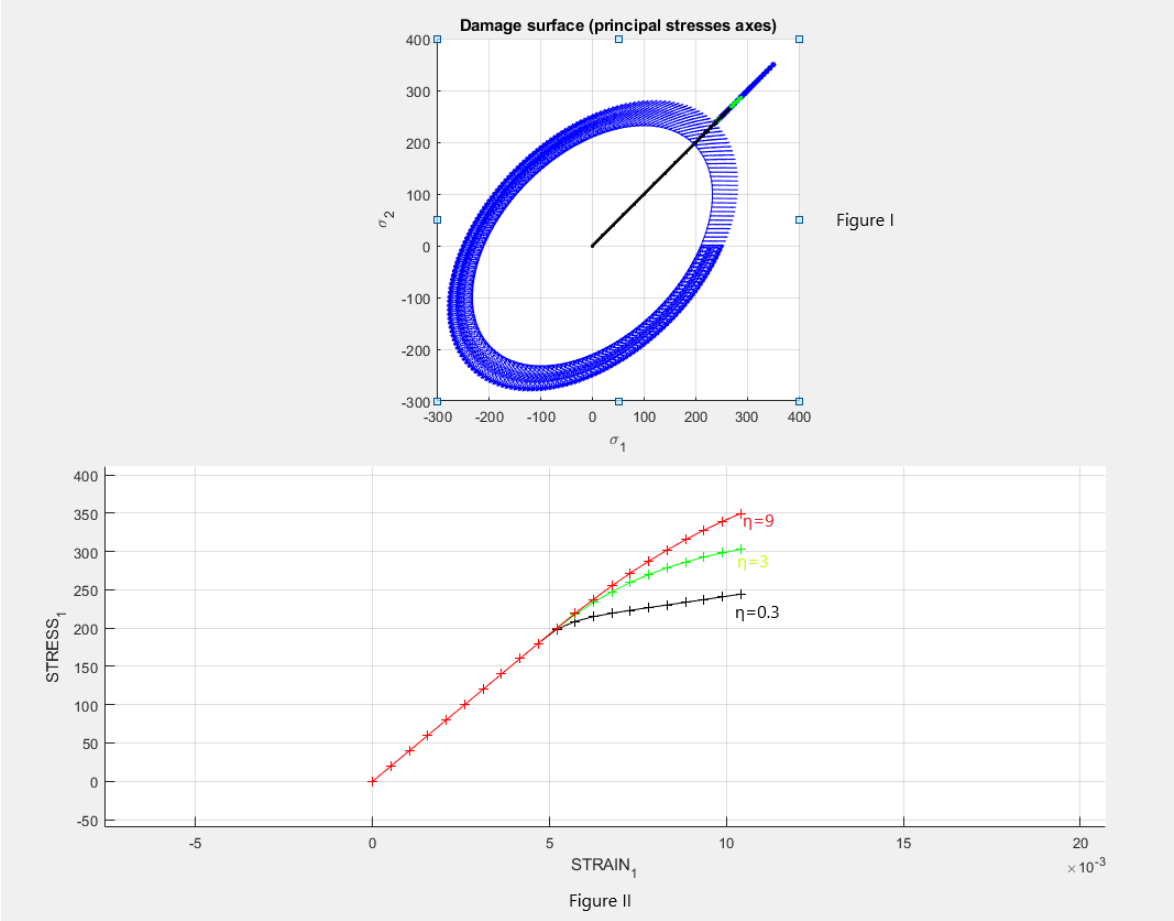

Figure 6.1: Effect of viscosity η

The correctness of the code can be seen as, for $\eta \to 0$ the behaviours of the plot tends to the inviscid linear one. While for $\eta \to \infty$, in the inelastic domain, the stress increases proportionally to the viscosity. As a results, increasing η the damage will be smaller.

6.2 Strain rate

The strain rate can be defined as the strain ϵ divided on time. So to test the code it has been chosen different values of total time in order to simulate different strain rate. The simulation has been run with the linear symmetric damage model with the following values:

$$\begin{aligned} E &= 2000 \\ \nu &= 0.3 \\ H &= 0.2(hardening) \\ \alpha &= 1 \\ \eta &= 0.3 \\ n &= 3 \end{aligned} \tag{6.4}$$

And it has been considered only one increment of σ:

$$\sigma = (400; 400) \tag{6.5}$$

Has been chosen three different values of time:

$$\begin{aligned} t_1 &= 1 \\ t_2 &= 10 \\ t_3 &= 20 \end{aligned} \tag{6.6}$$

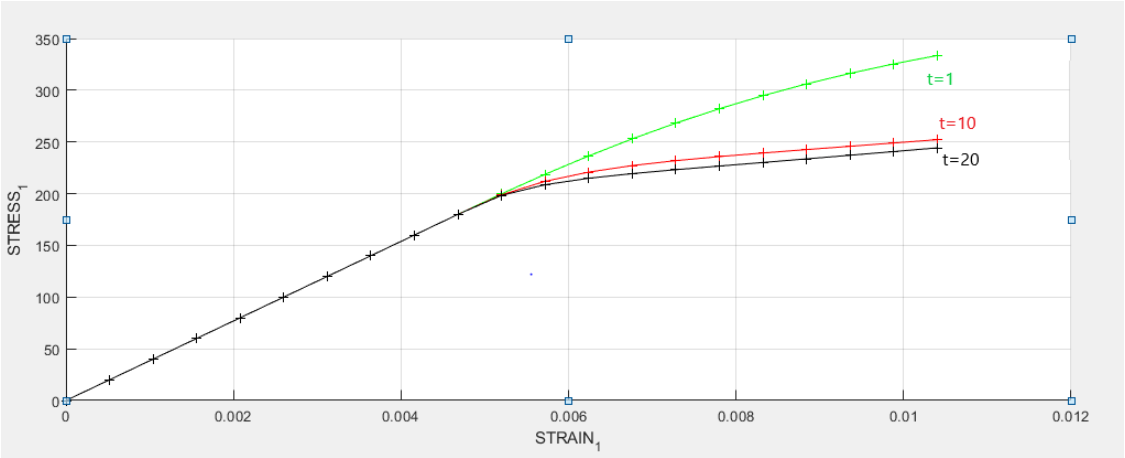

Figure 6.2: Effect of the strain rate

In the plot can be clearly seen how the stress increasing change with the strain rate. For t $\to 0$ the strain rate goes to ∞ and the material behaved as a fully elastic one. While when time increases, the strain rate decreases and consequently the σ is going to be smaller according to the theory.

6.3 α parameter

The α factor influences the accuracy of the method. From the theory it is stable between $[\frac{1}{2}; 1]$ and if equal to $1/2$ it is second order using crank-nicholson scheme. So it will be expected the best behaviour with $\alpha = 1/2$ and the worst one going to zero.

The simulation has been run with the linear symmetric damage model with the following values:

$$\begin{aligned} E &= 2000 \\ \nu &= 0.3 \\ H &= -0.2(softening) \\ t &= 15 \\ \eta &= 0.3 \\ n &= 3 \end{aligned} \tag{6.7}$$

And it has been considered only one increment of σ:

$$\sigma = (400; 400) \tag{6.8}$$

Has been chosen five different values of α:

$$\begin{aligned} \alpha_1 &= 0 \\ \alpha_2 &= 1/4 \\ \alpha_3 &= 1/2 \\ \alpha_4 &= 3/4 \\ \alpha_5 &= 1 \end{aligned} \tag{6.9}$$

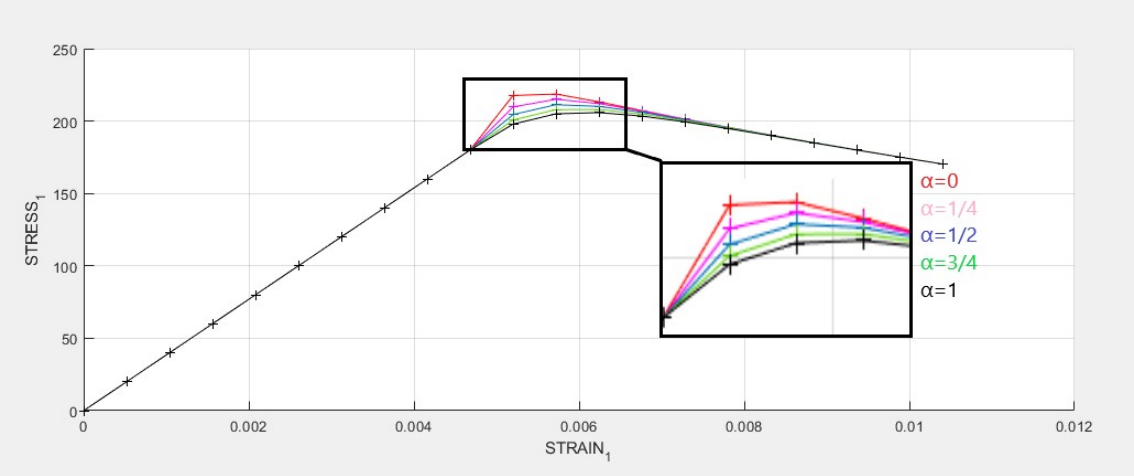

Figure 6.3: Effect of α parameter

As it can be noticed in Figure 6.3, when $\alpha \to 0$ the plot has an unusual slope that doesn't perfect suit the reality. On the other hand, going to the values of $1/2$, it makes sense because is starting from the slope of the elastic domain. While going to 1, the method can be seen is stable according to the theory, but less accurate than the blue line because the slope is smoothing a lot.

7. Conclusions

In conclusion it can be said that the code well simulate the reality. It has been implemented well and in every situation it gave reasonable results.

While hardening/softening, the damage surface behaved as expected following the linear or exponential laws, according to the interface window. It gave similar results with similar load path series, even if the model were different, according to the physical meanings of the model (case in 4 for only-tension and no symmetric).

While evaluating the hardening variable, it was computed in function of time according with the theory and the hard/soft laws, as well as the damage. The viscous implemented case, showed reasonable results while changing the parameters, from the material to the integration methods.

8. Appendix A

8.1 Dibujar-criterio-dano1

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
axis equal
axis square
%*****************************************************************************************
%*                  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----->tensii¿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
    ml=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*


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

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

        sl(i)=radio(i)*ml(i);
        s2(i)=radio(i)*m2(i);

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


elseif MDtype==2
    eps=0.01;
    tetha=[-pi/2+eps:eps:pi-eps];

    D=size(tetha);                          %*  Range
    ml=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*

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

```

```
    for i=1:Contador

        radio(i)= q/sqrt([max(ml(i),0) max(m2(i),0) 0 max(nu*(ml(i)+m2(i)),0)]*ce_inv*[ml(i) m2(i) 0 ...
            nu*(ml(i)+m2(i))]');

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

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



elseif MDtype==3
    eps=0.01;

    tethatens=[0:eps:pi/2];
    tethacomp=[pi:eps:pi*3/2];
    tetha=[tethatens tethacomp 2*pi];

    D=size(tetha);                          %*  Range
    ml=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
        s_pos=sum([max(ml(i),0) max(m2(i),0) 0 max(nu*(ml(i)+m2(i)),0)]);
        s_abs=sum(abs([(ml(i)) (m2(i)) 0 nu*(ml(i)+m2(i))]));

        tetha_nosym=s_pos/s_abs;
        radio(i)= q/((tetha_nosym+(1-tetha_nosym)/n)*sqrt([ml(i) m2(i) 0 nu*(ml(i)+m2(i))]*ce_inv*[ml(i) m2(i) 0
            nu*(ml(i)+m2(i))]'));

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

```

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

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



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

9. Appendix B

9.1 rmap-dano1

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

%*****************************************************************************************
%*                                    *
%*           Integration Algorithm for a isotropic damage model
%*
%*                                                                                        *
%*            [sigma_n1,hvar_n1,aux_var] = rmap_danol (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) ;
%*****************************************************************************************


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

```
alpha=Eprop(8);
eta=Eprop(7);
%*****************************************************************************************


%*****************************************************************************************
%*       Damage surface
if Eprop(6) == 0 %inviscid
    [rtrial] = Modelos_de_danol (MDtype,ce,eps_nl,n);
    r_nl=rtrial;
    %*************************************************************************************
else
    [rtrial] = Modelos_de_danol (MDtype,ce,eps_nl,n);
    [rtrial_n]=Modelos_de_danol (MDtype,ce,eps_n,n);
    rtrial_an= (1-alpha)*rtrial_n +alpha*rtrial;
    delt=delta_t(1);
    r_nl= ((eta-delt*(1-alpha))*r_n+delt*rtrial_an)/(eta+alpha*delt);
end

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

if(rtrial > r_n)
    %*   Loading

    fload=1;
    delta_r=r_nl-r_n;

    if hard_type == 0
        %  Linear
        q_nl= q_n+ H*delta_r;
    else
        if H <0
            q_infinite=zero_q;
        else
            q_infinite=2*r0 -zero_q;
        end
        q_nl= q_infinite - ((q_infinite - r0))*exp(abs(H)*(1-r_nl/r0));
    end
```

```
87 -        if(q_n1<zero_q)
88 -            q_n1=zero_q;
89 -        end
90
91
92 -    else
93
94          %*     Elastic load/unload
95 -        fload=0;
96 -        r_n1= r_n  ;
97 -        q_n1= q_n  ;
98
99
100 -   end
101     % Damage variable
102     % ---------------
103 -   dano_n1   = 1.d0-(q_n1/r_n1);
104     %  Computing stress
105     %  ****************
106 -   sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
107     %hold on
108     %plot(sigma_n1(1),sigma_n1(2),'bx')
109
110     %*****************************************************************************************
111
112
113     %*****************************************************************************************
114     %* Updating historic variables                                                  %*
115     %  hvar_n1(1:4)  = eps_n1p;
116 -   hvar_n1(5)= r_n1 ;
117 -   hvar_n1(6)= q_n1 ;
118     %*****************************************************************************************
119
120
121
122
123     %*****************************************************************************************
124     %* Auxiliar variables                                                                 %*
125 -   aux_var(1) = fload;
126 -   aux_var(2) = q_n1/r_n1;
127     %*aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
128     %*****************************************************************************************
```

10. Appendix C

10.1 Modelos-de-dano1

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

%*****************************************************************************************
if (MDtype==1)      %* Symmetric
    rtrial= sqrt(eps_nl*ce*eps_nl');

elseif (MDtype==2)  %* Only tension
    sigten=ce*eps_nl';

    for i=1:4
        if sigten(i)<0
            sigten(i)=0;
        end
    end
    rtrial=sqrt(eps_nl*sigten);

elseif (MDtype==3)  %*Non-symmetric
    s_pos=0;
    signosym=ce*eps_nl';
    for i=1:4
        if signosym(i)>0
            s_pos=s_pos+signosym(i);
        end
    end
    s_abs=sum(abs(signosym));
    tetha_nosym=s_pos/s_abs;
    rtrial=(tetha_nosym+(1-tetha_nosym)/n)*sqrt(eps_nl*ce*eps_nl');

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

11. Appendix D

11.1 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  (it may be defined also outside this function)
% ----------------------------------
 LABELPLOT = {'hardening variable (q)','internal variable'};

E      = Eprop(1) ; nu = Eprop(2) ;
viscpr = Eprop(6) ;
sigma_u = Eprop(4);



if ntype == 1
    menu('PLANE STRESS has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
```

```
    error('OPTION NOT AVAILABLE')
else
    mstrain = 4    ;
    mhist   = 6    ;
end


totalstep = sum(istep) ;


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


% Elastic constitutive tensor
% ---------------------------
[ce]    = tensor_elasticol (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_nl  = 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_nl = strain(i,:) ;
sigma_nl =ce*eps_nl'; % Elastic
sigma_v{i} = [sigma_nl(1)  sigma_nl(3) 0;sigma_nl(3) sigma_nl(2) 0 ; 0 0  sigma_nl(4)];

nplot = 3 ;
```

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

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;
        % Total strain at step "i"
        % ------------------------
        eps_n=strain(i-1,:); %define eps_n at the iteration i-1
        eps_n1 = strain(i,:) ;
        %*******************************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] = rmap_dano1(delta_t,eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n);
        % 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)
    end
end
%end
```