Computational solid mechanics

Pol Gomà

Assignment #1: Continuum damage models

Part I: Rate independent models

In the first part of this assignment the integration algorithms for the continuum isotropic damage non-symmetric tension-compression damage model and the tension-only damage model have been implemented. In addition, both linear and exponential hardening/softening cases have been implemented for both models as well.

After said implementations were done, we assessed the correctness of the results obtained. This way, for each model we have obtained the path at the stress space and the stress-strain curve corresponding to appropriate loading paths starting at the point $\sigma_1^{(0)} = 0\,;\,\sigma_2^{(0)} = 0$ and described by three-segment paths in the strain space defined in terms of their corresponding effective stress increments.

We are going to divide this initial part in 3 cases depending on the values of said effective stress increments, and for each case we are going to obtain the path at the stress space and the stress-strain curve for each model and for linear and exponential hardening/softening case. When dealing with a hardening case the value of the continuum hardening/softening modulus H is set to 0.6, whereas when dealing with a softening case H has a value of -0.6.

For all the cases that have been studied we have used the following material properties:

$$\begin{cases} E = 20000 \\ \nu = 0.3 \\ \sigma_y = 200 \\ n = 3\,(non - symmetric\,damage\,model) \end{cases}$$

Case 1

The first case is characterised by the following effective stress increments:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0 \quad (uniaxial\,tensile\,loading) \\ \Delta\bar{\sigma}_1^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = 0 \quad (uniaxial\,tensile\,unloading/compressive\,loading) \\ \Delta\bar{\sigma}_1^{(3)} = \gamma \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 0 \quad (uniaxial\,compressive\,unloading/tensile\,loading) \end{cases}$$

This way, we have chosen the following values for these effective stress increments:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = 400 \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0 \\ \Delta\bar{\sigma}_1^{(2)} = -600 \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = 0 \\ \Delta\bar{\sigma}_1^{(3)} = 300 \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 0 \end{cases}$$

0.0.1 The tension-only damage model

In this subsection the results obtained for the tension-only damage model are going to be presented. These are the results obtained for the hardening case:

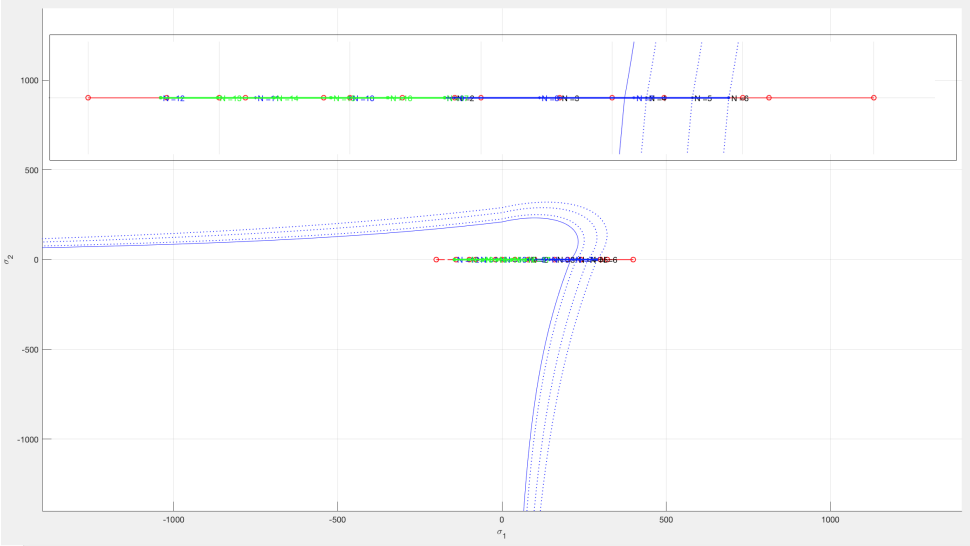
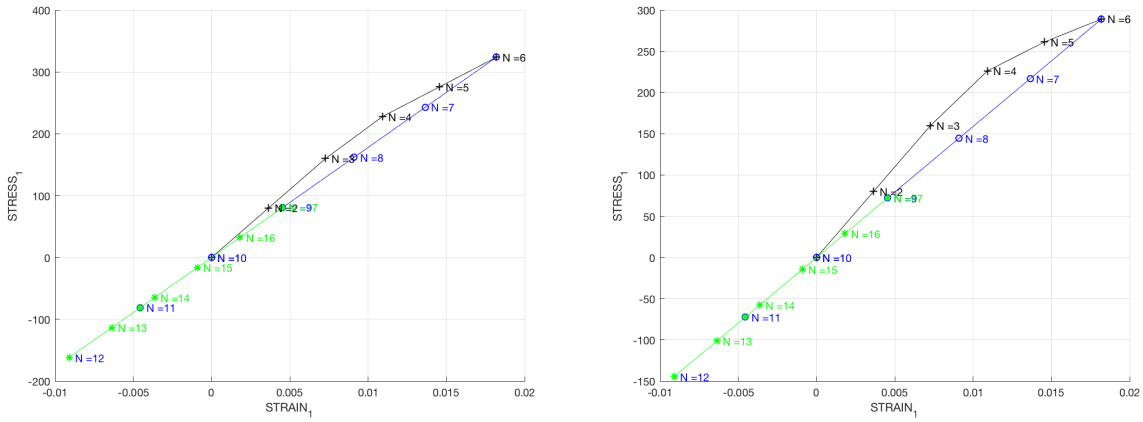


Figure 1: Stress path for the tension-only model with exponential hardening



(a) Linear hardening

(b) Exponential hardening

Figure 2: Stress-strain curve for the tension-only damage model with linear and exponential hardening

We can see that in the first stress increment (uniaxial tensile loading) we have surpassed the yield strength, and consequently damage occurs. This is why we can see an expansion of the damage surface, since we are in the hardening case, in order to keep the points in the border, as it is not admissible to be outside the damage surface. This is also why in this same interval the slopes after the elastic region in the stress-strain curve are positive.

In the second stress increment we performed an uniaxial tensile unloading and a compressive loading afterwards. Since damage has occurred we are not unloading with the same slope, but with a less steep one. Since we are in a uniaxial case we know that this less steep slope is $E^{sec\_d} = (1-d)E$. It is important to remark that unloading does not produce healing. We have finished the unloading inside the elastic region, and since we are dealing with the only tension damage model, we remain inside the elastic region during compression, as this model does not take into account failure by compression.

Eventually we have performed an uniaxial compressive unloading and a tensile loading afterwards. The former follows the less steep slope caused by the damage mentioned before, and since the tensile loading does not surpass the yield strength, so does it, and we remain therefore in the elastic region in both cases. We can also see the different behaviour of the linear and exponential hardening law in the last figure. It may be adequate to comment that the more steps we use, the more the exponential behaviour can be captured.

0.0.2 The non-symmetric damage model

In this section the results obtained for the non-symmetric damage model are going to be presented. These are the results obtained for the hardening case:

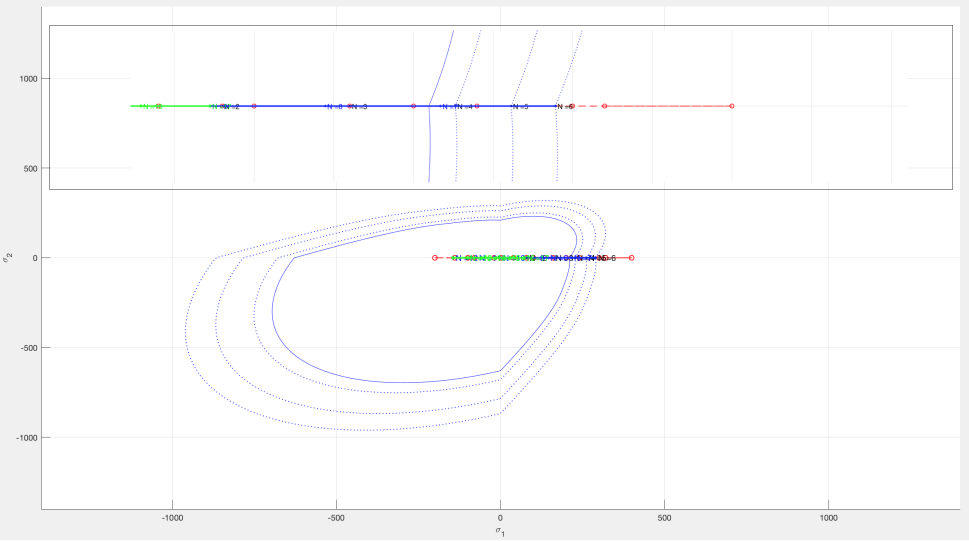
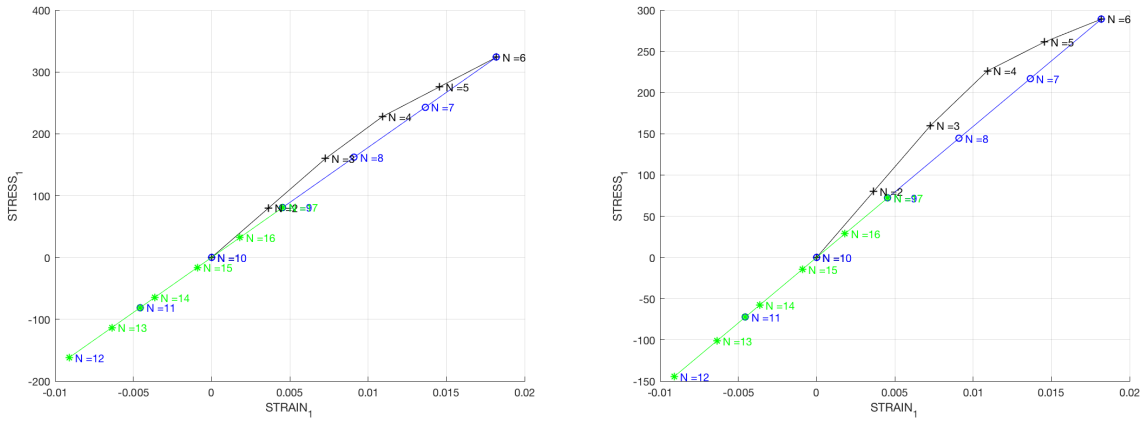


Figure 3: Stress path for the non-symmetric damage model with exponential hardening



(a) Linear hardening

(b) Exponential hardening

Figure 4: Stress-strain curve for the non-symmetric damage model with linear and exponential hardening

In this first case the results obtained for this model are the same as the ones obtained for the tension-only model, which was to be expected due to the loading path selected. Even though the non-symmetric damage model does take into account failure by compression, conversely to the tension-only model, in both cases we exit the initial damage surface in the first stress increment, but remain inside the damage surface for the other two.

Case 2

The second case is characterised by the following effective stress increments:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0 \quad (uniaxial\,tensile\,loading) \\ \Delta\bar{\sigma}_1^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = -\beta \quad (biaxial\,tensile\,unloading/compressive\,loading) \\ \Delta\bar{\sigma}_1^{(3)} = \gamma \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = \gamma \quad (biaxial\,compressive\,unloading/tensile\,loading) \end{cases}$$

Thus, the following values for these effective stress increments have been chosen:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = 400 \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0 \\ \Delta\bar{\sigma}_1^{(2)} = -600 \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = -600 \\ \Delta\bar{\sigma}_1^{(3)} = 300 \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 300 \end{cases}$$

0.0.3 The tension-only damage model

In this subsection the results obtained for the tension-only damage model are going to be presented. These are the results obtained for the softening case:

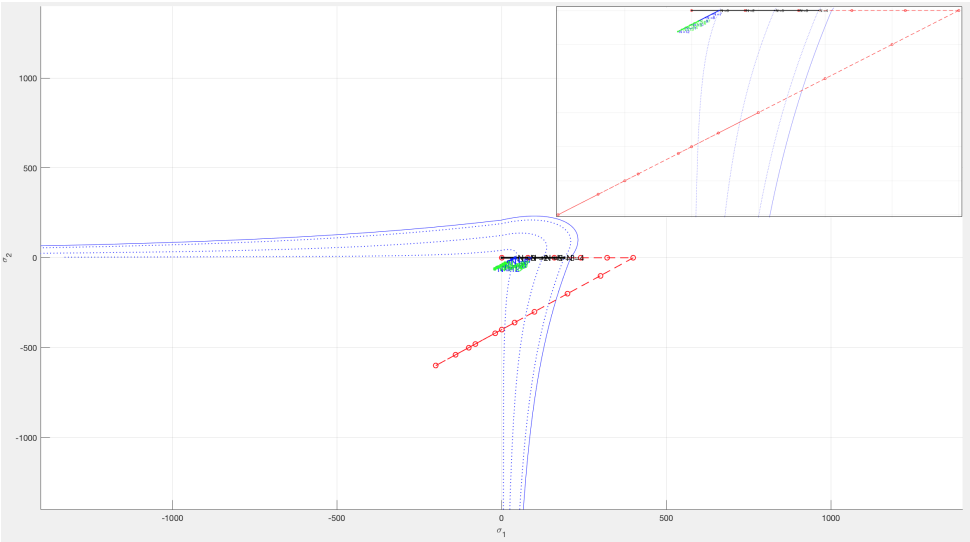
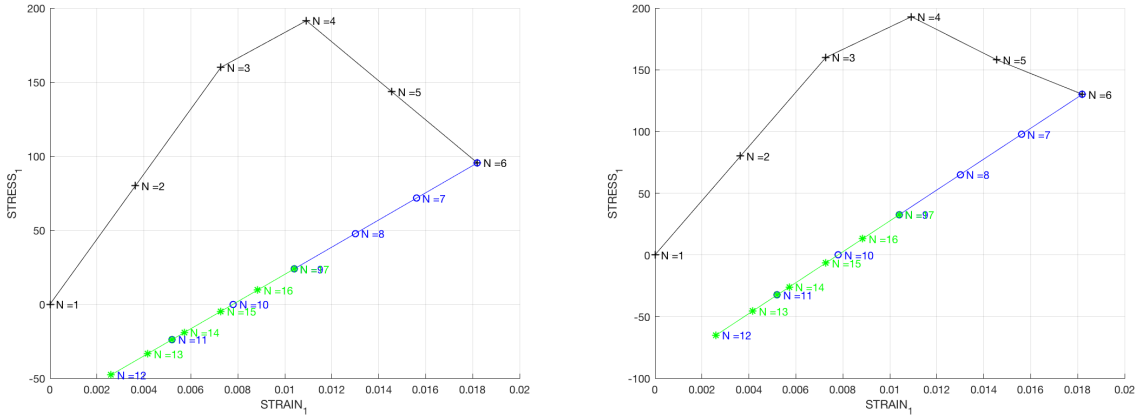


Figure 5: Stress path for the tension-only damage model with exponential softening



(a) Linear softening

(b) Exponential softening

Figure 6: Stress-strain curve for the tension-only model with linear and exponential softening

We can see that in the first stress increment we surpass the yield strength again, exiting this way the damage surface, and that is why it contracts, since we are in the softening case now. Then we perform a biaxial tensile unloading and a compressive loading. Since we have left the damage surface and therefore damage has occurred, the unloading is done with a less steep slope again. The tensile unloading and the compressive loading follow the same slope, as we remain in the elastic region in both cases. Eventually we perform a biaxial compressive unloading and tensile loading, but again this is done inside the elastic region, so no more damage occurs and we follow the same slope in the stress-strain curve. Again it can be noted the difference in the behaviour of the softening when using a linear softening law or an exponential one.

0.0.4 The non-symmetric damage model

In this subsection the results obtained for the non-symmetric damage model are going to be presented. These are the results obtained for the linear and exponential softening case:

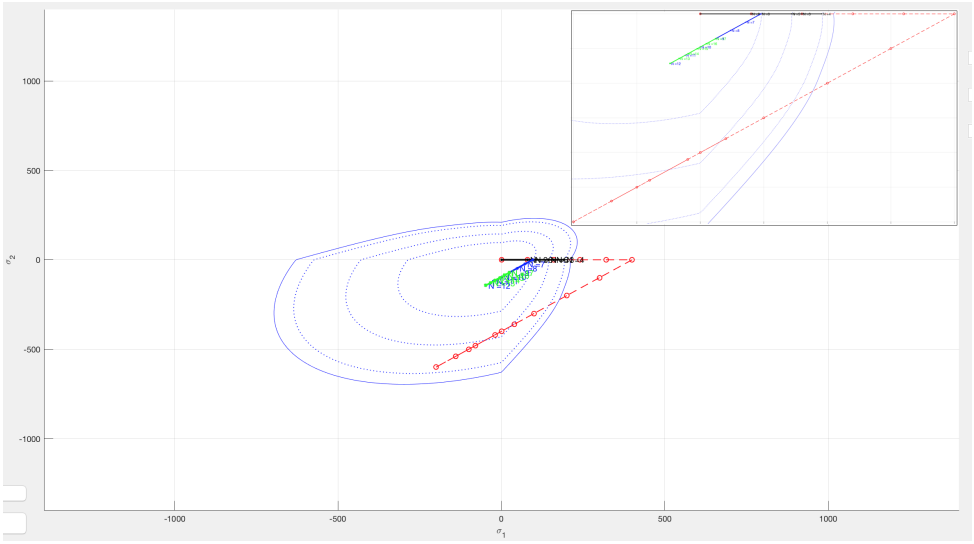
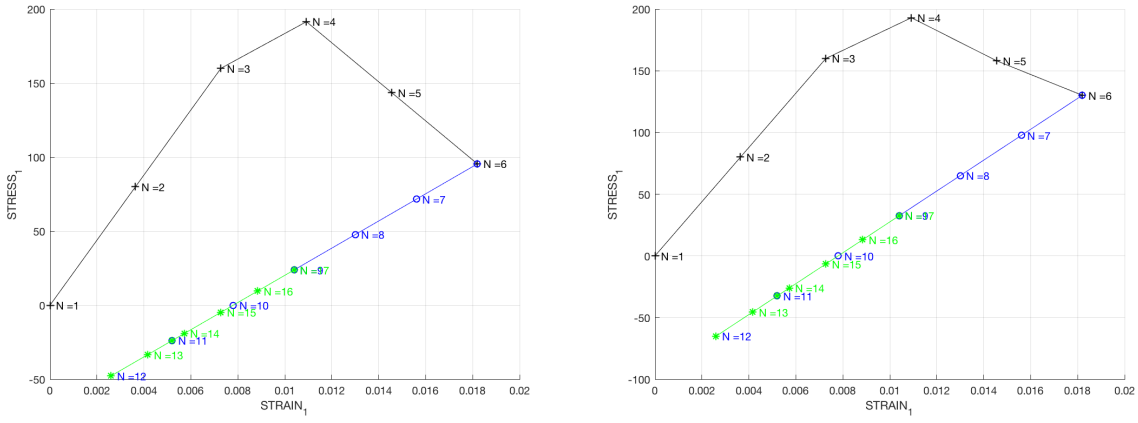


Figure 7: Stress path for the non-symmetric damage model with exponential softening



(a) Linear softening

(b) Exponential softening

Figure 8: Stress-strain curve for the non-symmetric damage model with linear and exponential softening

Again the results obtained are the same as in the tension-only case due to the loading path chosen. They would have differed if in the last stress increment we had surpassed the damage surface again, for instance, but since we remain in the elastic region we obtain the same results as with the previous model.

Case 3

The third case is characterised by the following effective stress increments:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = \alpha \quad (biaxial\, tensile\, loading) \\ \Delta\bar{\sigma}_1^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = -\beta \quad (biaxial\, tensile\, unloading/compressive\, loading) \\ \Delta\bar{\sigma}_1^{(3)} = \gamma \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = \gamma \quad (biaxial\, compressive\, unloading/tensile\, loading) \end{cases}$$

This way, the following values for these effective stress increments have been chosen:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = 400 \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 400 \\ \Delta\bar{\sigma}_1^{(2)} = -600 \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = -600 \\ \Delta\bar{\sigma}_1^{(3)} = 1200 \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 1200 \end{cases}$$

0.0.5 The tension-only damage model

In this subsection the results obtained for the tension-only damage model are going to be presented. These are the results obtained for the hardening case:

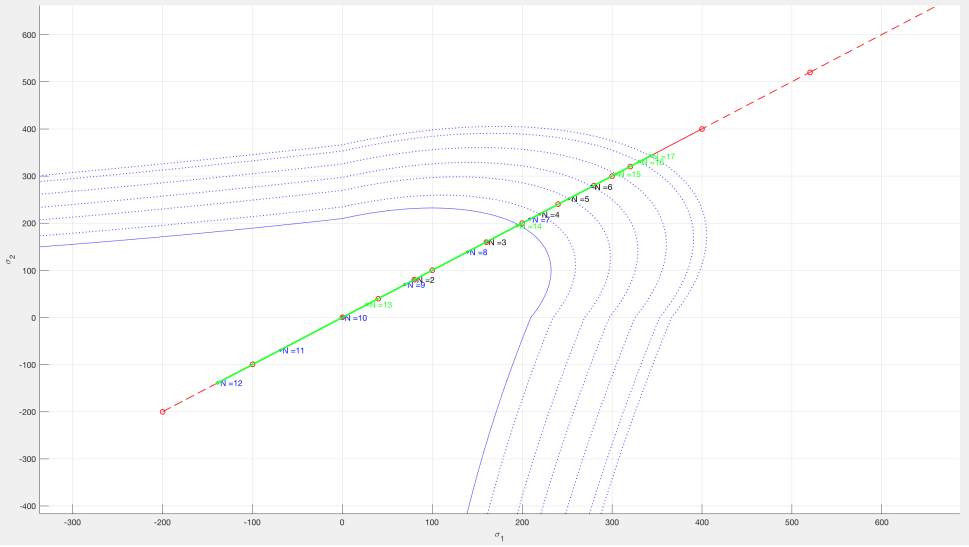
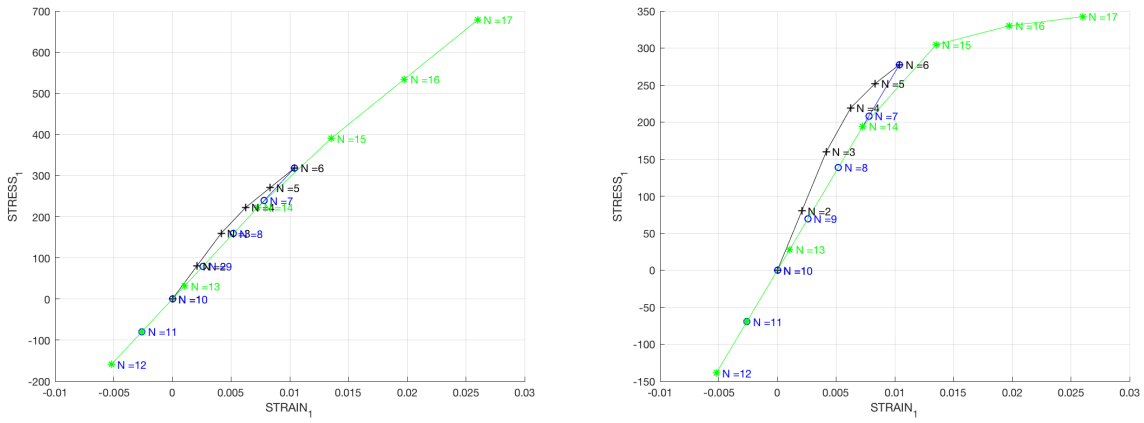


Figure 9: Stress path for the tension-only model with exponential hardening



(a) Linear hardening

(b) Exponential hardening

Figure 10: Stress-strain curve for the tension-only damage model with linear and exponential hardening

In this last case we have chosen a loading path such that we exit the damage surface in the first stress increment, come back in the second, and exit again in the third one, as to see more clearly the effects of damage.

We have first carried out a biaxial tensile loading, in which we surpass the yield strength and therefore an expansion of the damage surface occurs, since we are in the hardening case now. This is also the reason why the slopes after the initial elastic part of the stress-strain curve are positive. Afterwards a biaxial tensile unloading and a compressive loading take place, and since damage has occurred we find again a less step slope in the stress-strain curve. We are dealing with the only-tension model now, so no matter how much we load in compression we are always going to descend with the same slope (we remain in the elastic region and therefore no damage occurs). Eventually we perform a biaxial compressive unloading and a tensile loading, and in the first one we go up with the last slope we had obtained because we do not exit the damage surface (with the 'damaged' slope, as unloading does not produce healing), but with the tensile loading we eventually exit the damage surface again, and this is why we can see the hardening behaviour again. This way, had we unloaded again we would have descended with an even less steep slope, since more damage has occurred.

In this last case the difference in dealing with a linear hardening law or a hardening exponential one can be seen with much more clarity.

0.0.6 The non-symmetric damage model

In this subsection the results obtained for the non-symmetric damage model are going to be presented. These are the results obtained for the hardening case:

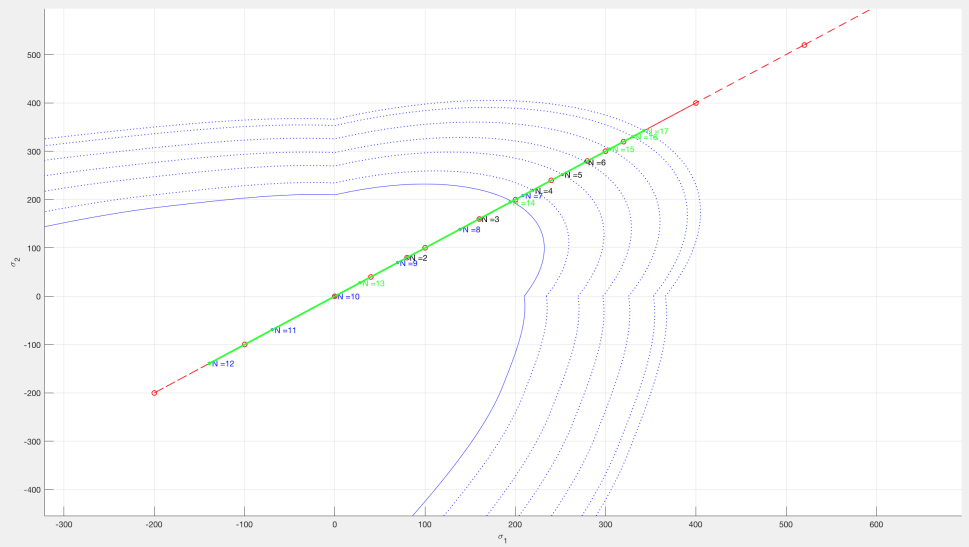
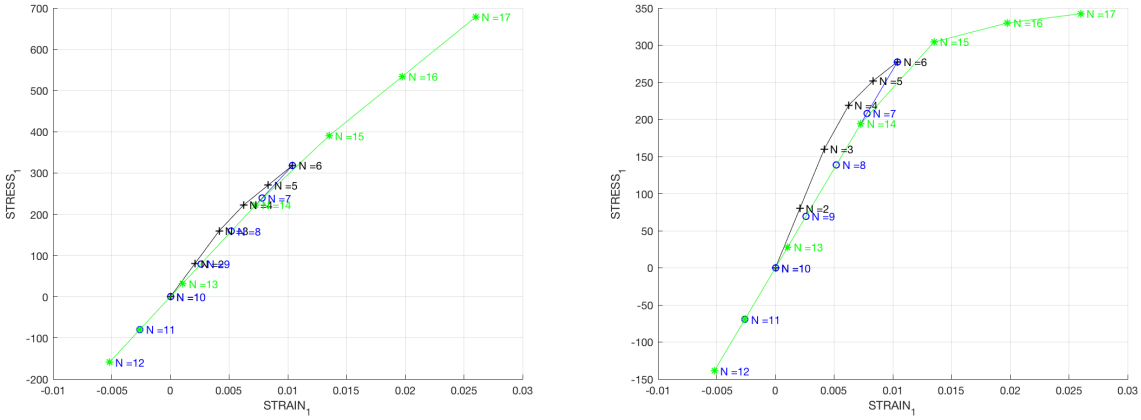


Figure 11: Stress path for the non-symmetric damage model with exponential hardening



(a) Linear hardening

(b) Exponential hardening

Figure 12: Stress-strain curve for the non-symmetric damage model with linear and exponential hardening

We have obtained again the same results as with the only-tension model, which was to be expected due to the same reason as in the other cases, the loading path chosen. In both cases we leave the initial damage surface in the first stress increment, come back in the second, and exit again in the last one.

Part II: Rate dependent models

0.1 Part II-1

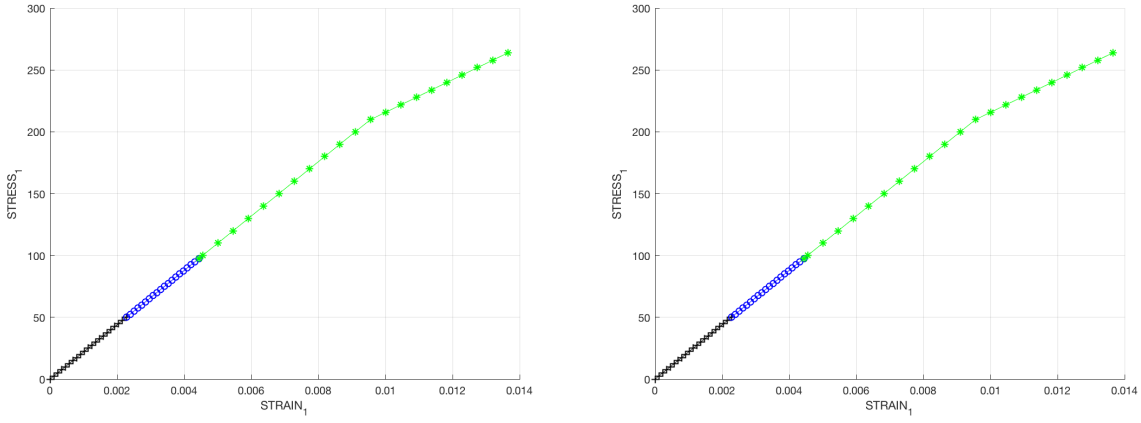
In the second part of this assignment the integration algorithm for the the continuum isotropic viscodamage symmetric tension-compression model for the plane strain case has been implemented in the Matlab code provided. In order to validate the implementations that have been done we have analysed the influence of the viscosity parameter η, the strain rate $\dot{\epsilon}$ and the α parameter of the time-integration method in the stress-strain curve for a specific case for the symmetric damage model with:

$$\begin{cases} E = 20000 \\ \nu = 0.3 \\ \sigma_y = 200 \\ \textit{Linear hardening with } \text{H=0.6} \end{cases}$$

And the following loading path:

$$\begin{cases} \Delta\bar{\sigma}_1^{(1)} = 50 \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0 \\ \Delta\bar{\sigma}_1^{(2)} = 50 \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = 0 \\ \Delta\bar{\sigma}_1^{(3)} = 200 \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 0 \end{cases}$$

But first another analysis has been carried out as to check that the code that has been implemented is correct. We know that when dealing with the viscodamage model, if the viscosity parameter is set to 0 and we use implicit methods for the time integration we recover the inviscid model. We know that in the α-family methods we obtain explicit schemes for $\alpha < 0.5$, and implicit schemes otherwise. This way, we have analysed the case that has just been introduced using first the viscid model with $\eta = 0$, total time $T = 10\,s$ and $\alpha = 1$, and then with the inviscid model. These are the results obtained:



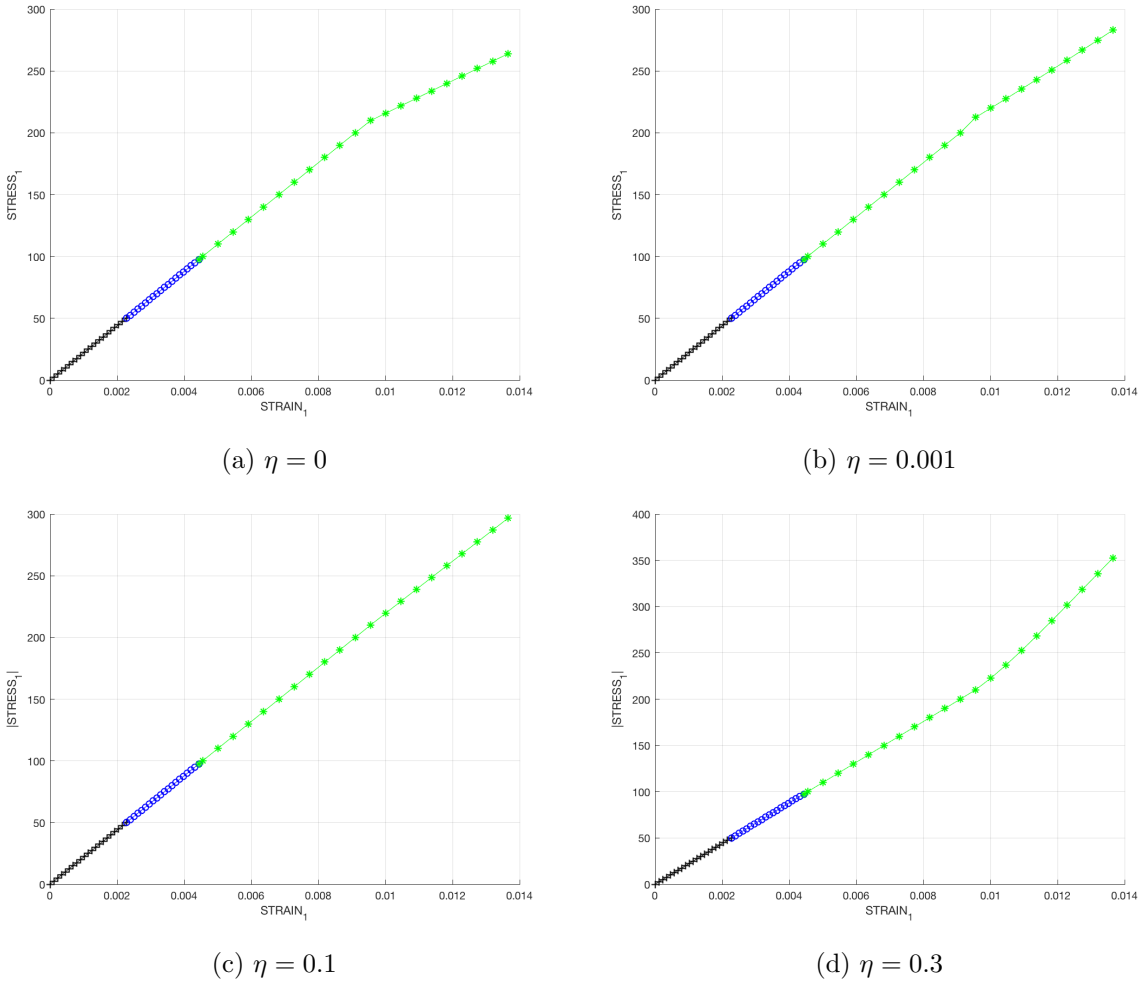
(a) Stress-strain curve for the viscous model

(b) Stress-strain curve for the inviscid model

Figure 13: Stress-strain curves for the non-symmetric damage model with linear and exponential hardening

It is easy to see that the results obtained are exactly the same, which is a good indicator in order to validate our code. Now the results obtained for the analysis of the influence of the viscosity parameter η are going to be shown. We have chosen $\alpha = 1$, which corresponds to the Backward Euler scheme, an implicit method and unconditionally stable therefore. The total time has been set to 10 s.

Thus:



(a) $\eta = 0$

(b) $\eta = 0.001$

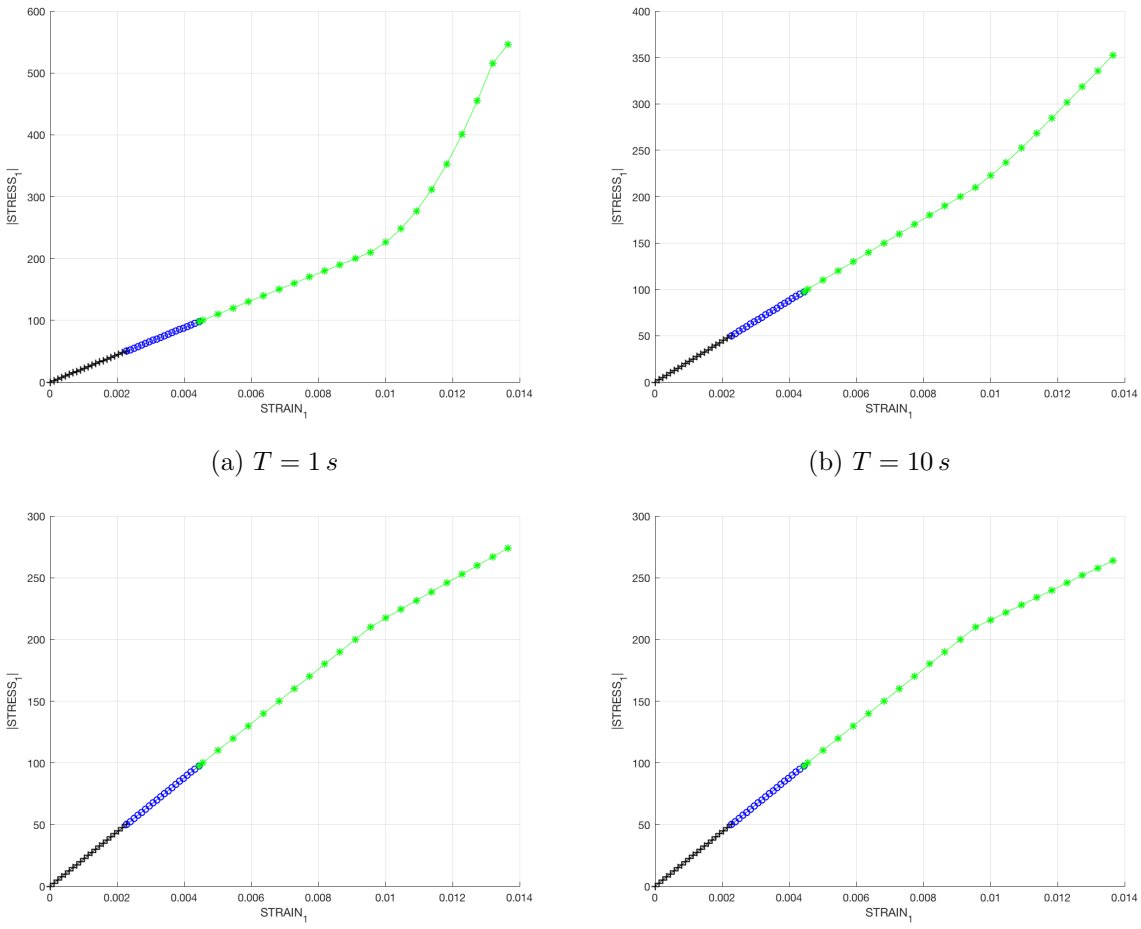
(c) $\eta = 0.1$

(d) $\eta = 0.3$

Figure 14: Stress-strain curves for the different of values of η

We know beforehand that the viscosity effects make the material able to achieve higher stresses when in inelastic behaviour. And this fact can be easily seen taking a look at the plots just presented. We can also see that the hardening behaviour begins later in time as η increases.

In order to see the influence of the strain rate $\dot{\epsilon}$ on the stress-strain curves we have to fix all the parameters except for the total time T. The results obtained for this analysis can be seen in the figure below. Again we have used $\alpha = 1$ and $\eta = 0.3$. This way:



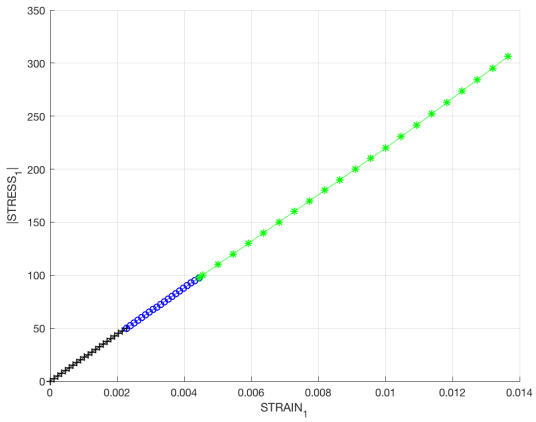
(a) $T = 1\,s$ (b) $T = 10\,s$

(c) $T = 100\,s$ (d) $T = 1000\,s$

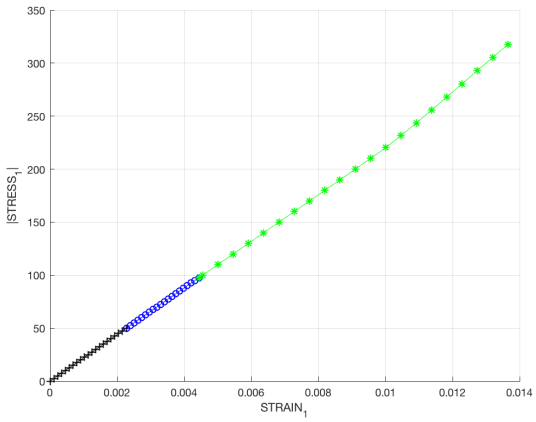
Figure 15: Stress-strain curves for the different of values of $\dot{\epsilon}$

The strain rate $\dot{\epsilon}$ depends on the total time, i.e the higher the total time is, the lower the strain rate will be. Analysing the figure above we can see that as we increase the total time (and decrease $\dot{\epsilon}$ consequently) we obtain lower stresses for the same values of strain. In addition, we can see that when dealing with this kind of models the stresses depend on the strain as well as on the strain rate. Eventually, we can consider that in the last plot of the figure above the total time is big enough for $\dot{\epsilon}$ to tend to 0, and we can see that in this case we have recovered the inviscid case (which can be seen in figure 13), which was to be expected.

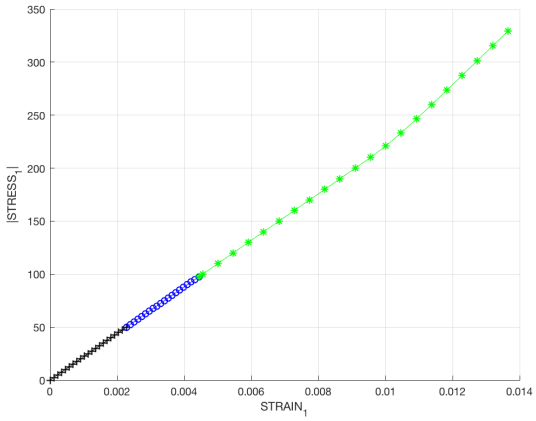
As far as α is concerned, the results obtained can be seen in the following figures. We have used $\eta = 0.3$ and $\dot{\epsilon} = 10\,s$:



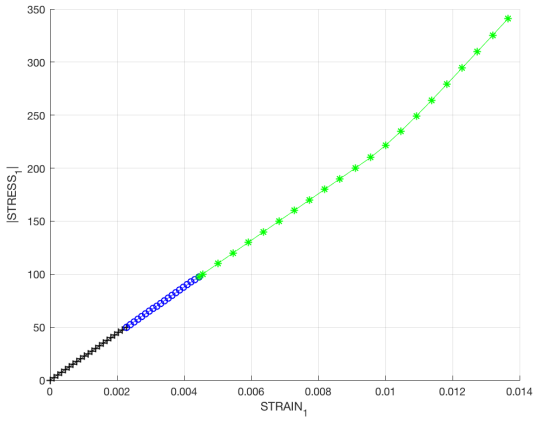
(a) $\alpha = 0$



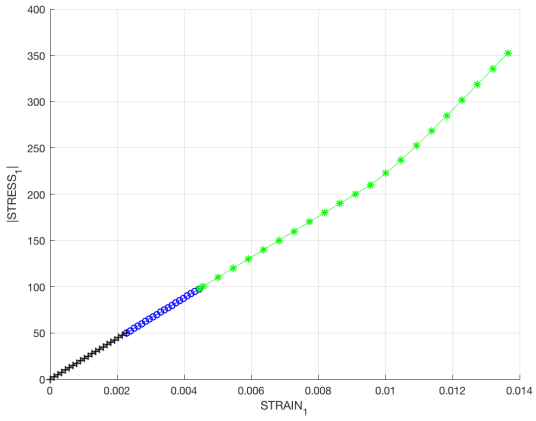
(b) $\alpha = 0.25$



(c) $\alpha = 0.5$



(d) $\alpha = 0.75$



(e) $\alpha = 1$

Figure 16: Stress-strain curves for the different of values of α

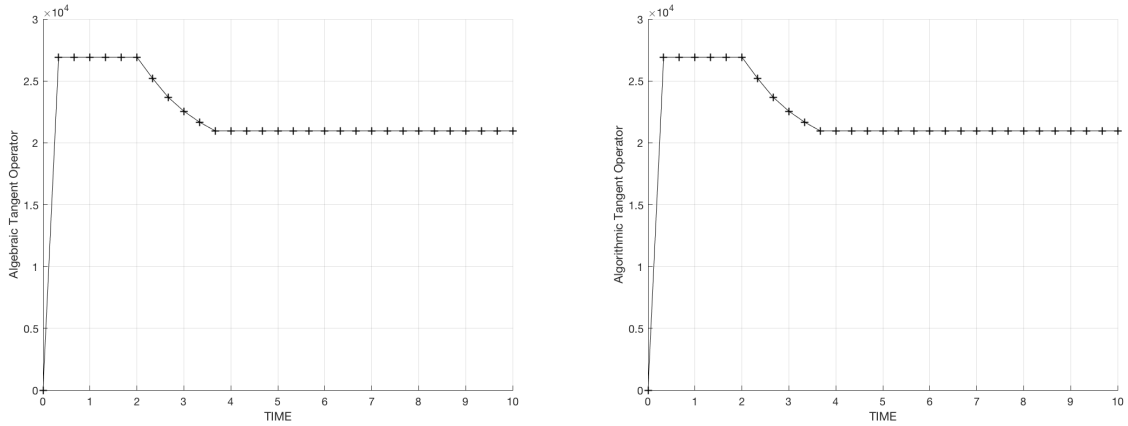
Analysing the results obtained we can see that the election of α plays an important role. We know that if the value of α is smaller than 0.5 (and we have therefore an explicit method) and the time step is not small enough the method fails. Eventually we know that the α family methods are unconditionally stable if $\alpha \geq 0.5$ (implicit methods), and conditionally stable otherwise (explicit methods).

0.2 Part II-2

In the last part of this assignment we have studied the effect of the α parameter from the time integration method on the evolution through time of the C_{11} component of the tangent and algorithmic constitutive operators. We have used the same loading path as in case 1 from the part of this assignment, the symmetric damage model, and:

$$\begin{cases} E = 20000 \\ \nu = 0.3 \\ \sigma_y = 200 \\ Linear\, hardening\, with\, H = 0.6 \\ \eta = 0.3 \\ \dot{\epsilon} = 10\, s^{-1} \end{cases}$$

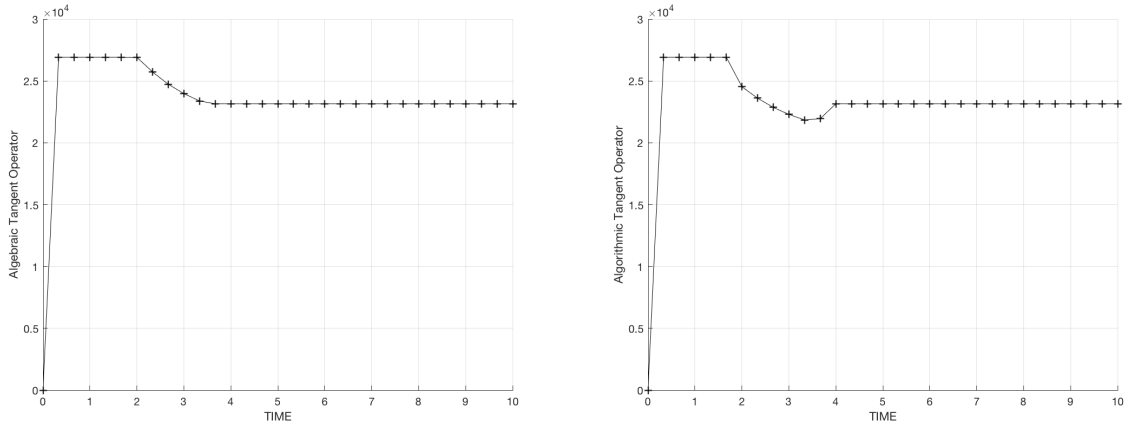
These are the results obtained for this analysis:



(a) C_{11} component of the tangent constitutive operator

(b) C_{11} component of the algorithmic constitutive operator

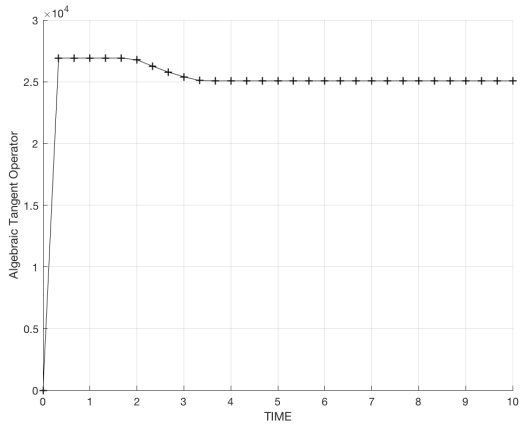
Figure 17: $\alpha = 0$



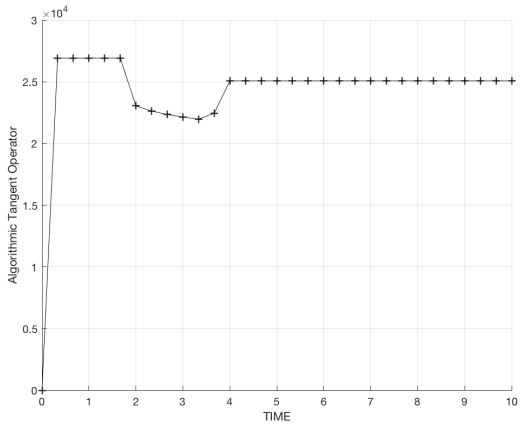
(a) C_{11} component of the tangent constitutive operator

(b) C_{11} component of the algorithmic constitutive operator

Figure 18: $\alpha = 0.25$

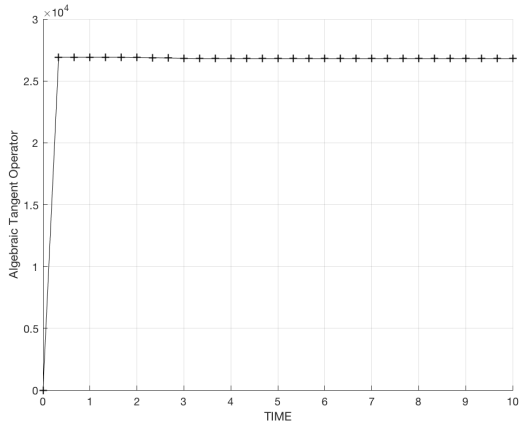


(a) C_{11} component of the tangent constitutive operator

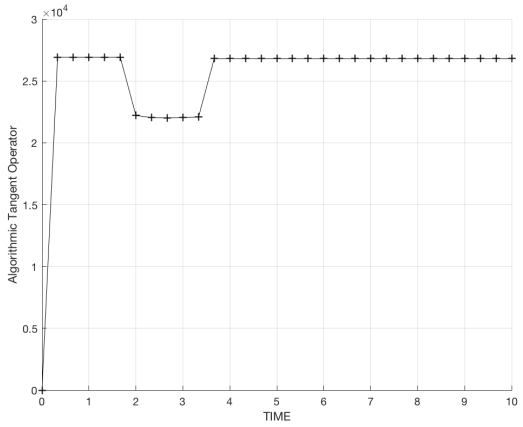


(b) C_{11} component of the algorithmic constitutive operator

Figure 19: $\alpha = 0.5$

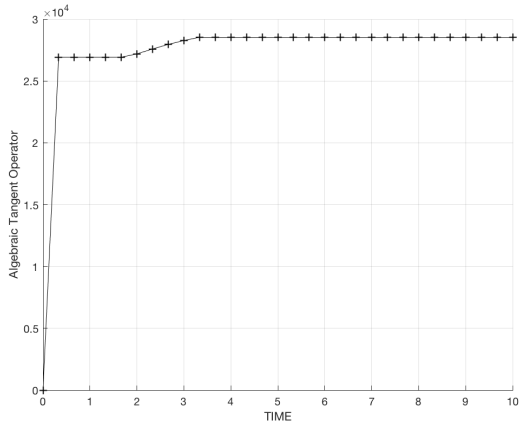


(a) C_{11} component of the tangent constitutive operator

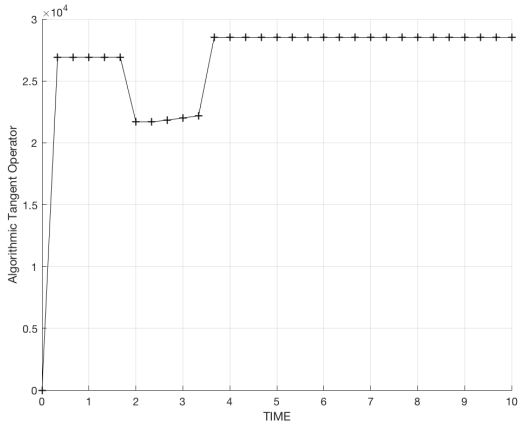


(b) C_{11} component of the algorithmic constitutive operator

Figure 20: $\alpha = 0.75$



(a) C_{11} component of the tangent constitutive operator



(b) C_{11} component of the algorithmic constitutive operator

Figure 21: $\alpha = 1$

Taking a look at these plots the first fact that can be mentioned is that for $\alpha = 0$ the C_{11} component of the analytical and the algorithmic tangent constitutive operators is exactly the same, which was definitely to be expected, since in this particular case the additional term in the definition of the algorithmic tangent constitutive operator for loading conditions is 0. This is not the case for any α different than 0, as there would be an additional term to add to the algorithmic tangent constitutive operator.

Another thing that one can realise is that in the elastic case and for loading conditions the C_{11} components of both tangent constitutive operators are the same, which was again utterly to be expected. This way, these C_{11} components just differ in loading conditions, which also matches the theory. It can also be seen that as we keep increasing the value of α the C_{11} components of the algorithmic and analytical constitutive tangent operators differ more from one another.

Appendix

In this appendix the code that has been written in order to implement the features needed to carry out this assignment is going to be shown. It is important to remark that just the routines modified as to solve the assignment are included in this appendix.

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
%*************************************************************************
%*                  PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODEL
                        %*
%*
              %*
%*      function [ce] = tensor_elastico (Eprop, ntype)
     %*
%*
              %*
%*      INPUTS
 %*
%*
              %*
%*                     Eprop(4)    vector de propiedades de material
              %*
%*                                          Eprop(1)=  E------>modulo de
 Young           %*
%*                                          Eprop(2)=  nu----->modulo de
 Poisson         %*
%*                                          Eprop(3)=  H----->modulo de
 Softening/hard. %*
%*                                          Eprop(4)=sigma_u----->tensi???
n ???ltima         %*
%*                        ntype                                    %*
%*                                    ntype=1  plane stress
              %*
%*                                    ntype=2  plane strain
              %*
%*                                    ntype=3  3D
              %*
%*                        ce(4,4)     Constitutive elastic tensor  (PLANE
 S.        )     %*
%*                        ce(6,6)                                   ( 3D)
 %*
%*************************************************************************


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

```
%*************************************************************************
% POLAR COORDINATES
if MDtype==1
    tetha=[0:0.01:2*pi];

  %*************************************************************************
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        % Number of points


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

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

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

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


elseif MDtype==2

    tetha=(-pi/2)*0.9999:0.01:pi*0.9999;

    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        % Number of point

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

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

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

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

elseif MDtype==3

    tetha=0:0.01:2*pi;
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        % Number of points


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

     for i=1:Contador

         sigma = [m1(i) m2(i) 0 nu*(m1(i)+m2(i))];
         sigmapos = sigma.*(sigma>0);
         sigmaabs= abs(sigma);
         theta= sum(sigmapos)/sum(sigmaabs);
         coef= theta + (1-theta)/n;

        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))]'))/coef;

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

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


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



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

```
Not enough input arguments.

Error in dibujar_criterio_dano1 (line 25)
ce_inv=inv(ce);
```

Published with MATLAB® R2017a

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



%*************************************************************************
if (MDtype==1)      %* Symmetric

rtrial= sqrt(eps_n1*ce*eps_n1');

elseif (MDtype==2)  %* Only tension

    sigma_bar = eps_n1*ce;
 sigma_barpos = sigma_bar.*(sigma_bar>0);
       rtrial = sqrt(sigma_barpos*eps_n1');

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

        sigma = eps_n1*ce;
     sigmapos = sigma.*(sigma>0);
     sigmaabs = abs(sigma);
        theta = sum(sigmapos)/sum(sigmaabs);
         coef = theta + (1-theta)/n;
       rtrial = coef*sqrt(eps_n1*ce*eps_n1');

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

Not enough input arguments.

Error in Modelos_de_dano1 (line 18)
```

```
if (MDtype==1)      %* Symmetric
```

Published with MATLAB® R2017a

```
function [sigma_n1,hvar_n1,aux_var,C_algoritmico,C_algebraico] =
 rmap_dano1 (eps_n1,eps_n0,hvar_n,Eprop,ce,MDtype,n,viscpr,delta_t)

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

%*
                *
%* OUTPUTS:            sigma_n1(4) Cauchy stress  , step n+1
                *
%*                     hvar_n(6)   Internal variables , step n+1
                    *
%*                     aux_var(3)  Auxiliar variables for computing
 const. tangent tensor  *
%*************************************************************************

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

```
%*************************************************************************
%*       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);
[rtrial_n] = Modelos_de_dano1 (MDtype,ce,eps_n0,n); % (for viscous)
[rtrial_nplusalpha] = (1-ALPHA_COEFF)*rtrial_n
+ALPHA_COEFF*rtrial ; %(for viscous)
%*************************************************************************


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

if viscpr == 1
    if (rtrial_nplusalpha > r_n)
        %*   Loading
        fload =1;
        delta_r = rtrial_nplusalpha-r_n;
        r_n1 =((eta-delta_t*(1-ALPHA_COEFF))/(eta-
ALPHA_COEFF*delta_t))*r_n+...
            (delta_t*rtrial_nplusalpha)/(eta+ALPHA_COEFF*delta_t);
        if hard_type == 0
            %  Linear
         q_n1 = q_n+ H*delta_r;
        else
            %Exponential
            qinf = r0+(r0-zero_q);
            if HARDSOFT_MOD > 0
            q_n1 = q_n+((H*(qinf-r0)/r0)*exp(H*(1-rtrial_nplusalpha/
r0)))*delta_r;
            else
            q_n1 = q_n+((H*(qinf-r0)/r0)*(1/exp(H*(1-
rtrial_nplusalpha/r0))))*delta_r;
            end
        end
         if(q_n1 < zero_q)
```

```
          q_n1 = zero_q;
          end
    else
        %*     Elastic load/unload
        fload=0;
        r_n1 = r_n  ;
        q_n1 = q_n  ;
    end
else
    if(rtrial > r_n)
    %*   Loading
    fload = 1;
  delta_r = rtrial-r_n;
     r_n1 = rtrial  ;
    if hard_type == 0
        %  Linear
        q_n1 = q_n+H*delta_r;
    else %  Exponential
        qinf = r0+(r0-zero_q);
        if HARDSOFT_MOD > 0
        q_n1 = q_n+((H*(qinf-r0)/r0)*exp(H*(1-rtrial/r0)))*delta_r;
        else
        q_n1 = q_n+((H*(qinf-r0)/r0)*(1/exp(H*(1-rtrial/
r0))))*delta_r;
        end
    end

    if(q_n1 < zero_q)
        q_n1 = zero_q;
    end

else

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

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

if viscpr == 1
    if (rtrial_nplusalpha > r_n)
    sigma_bar = ce*eps_n1';
```

```
    C_algoritmico = (1-dano_n1)*ce + ...
    (ALPHA_COEFF*delta_t)/((eta+ALPHA_COEFF*delta_t)*r_n1)*((H*r_n1-
q_n1)/r_n1^2)*(sigma_bar'*sigma_bar);
    else
    C_algoritmico = (1-dano_n1)*ce;
    end
    C_algebraico = (1-dano_n1)*ce;

else
    C_algoritmico = (1-dano_n1)*ce;
 C_algebraico = (1-dano_n1)*ce;
end
```

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

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

```
Not enough input arguments.

Error in rmap_dano1 (line 26)
hvar_n1 = hvar_n;
```

Published with MATLAB® R2017a

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

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

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

% SET LABEL OF "vartoplot" variables  (it may be defined also outside
 this function)
% ---------------------------------
 LABELPLOT = {'hardening variable (q)','internal variable'};

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



if ntype == 1
    menu('PLANE STRESS has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
else
    mstrain = 4    ;
    mhist   = 6    ;
end

totalstep = sum(istep) ;
```

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


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

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

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

for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ delta_t(iload) ;
        % Total strain at step "i"
        % ------------------------
        eps_n1 = strain(i,:);
        eps_n0 = strain(i-1,:);

  %*************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
```

```
        [sigma_n1,hvar_n,aux_var,C_algoritmico,C_algebraico] =
 rmap_dano1(eps_n1,eps_n0,hvar_n,Eprop,ce,MDtype,n,viscpr,delta_t);
        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu,
 hvar_n(6), 'r:',MDtype,n );
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1)
              ;
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

  %*********************************************************************
        % GLOBAL VARIABLES
        % ***************
        % Stress
        % ------
        m_sigma=[sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2)
 0 ; 0 0  sigma_n1(4)];
        sigma_v{i} =  m_sigma ;

        % VARIABLES TO PLOT (set label on cell array LABELPLOT)
        % -----------------
        vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
        vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
        vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5)  ; %  Damage variable
 (d)
        vartoplot{i}(4) = C_algoritmico(1,1);
        vartoplot{i}(5) = C_algebraico(1,1);
    end
end
```

```
Not enough input arguments.

Error in damage_main (line 77)
E      = Eprop(1) ; nu = Eprop(2) ;
```

Published with MATLAB® R2017a