# Computational Solid Mechanics

Assignment 1

Zahra Rajestari

## ***Table of Contents***

## 1. PART I

### 1.1. Part I_a

The aim of this part of the assignment is to implement the rate dependent models. We have three types of damage model. The first type, symmetric model, has already been implemented, the second and third type, only-tension and non-symmetric damage models, are to be coded according to the theory and formulations provided in the slides.

In order to implement the "Tension-only" and "Non-symmetric" case, we have to define the model parameters in the corresponding codes.

The first function to change is "Modelos_de_dano1.m" in which we can define the models equations. For all cases, the parameter "eps_n1" defined in this function is the strain tensor at time step n+1 which we have to use in order to define $\bar{\sigma}^{+}$ for only-tension and non-symmetric cases. This variable is defined in time step n+1 as "sigma_e_n1" as the following:

```
sigma_e_n1 = eps_n1*ce;
sigma_e_plus_n1(:) = sigma_e_n1(:).*(sigma_e_n1(:)>0);
```

The McCauley brackets is defined using the aforementioned notion in MATLAB. In this function, "rtrial" is the so-called $\tau_{n+1}$ in the formulation of each model and is defined based on the formulas provided by the slides.

After implementing all the formulas in "Modelos_de_dano1.m" function, we have to modify "dibujar_criterio_dano1.m" in order to get the plots for only-tension and non-symmetric cases. For this purpose, a parameter called "radio" is defined which represents the radial distance of each point of the curves from the (0,0) coordinates as it is defined based on the formula provided in the slides. This value differs from one damage model to another because of the changes in $\tau_{n+1}$ for each model. Then, the x and y coordinate of this curve which are basically $\sigma_1$ and $\sigma_2$ are computed by changing the coordinated from radial to Cartesian. The figures are then plotted by changing the values of theta from 0 to 2π and computing the principle stresses for each angle.

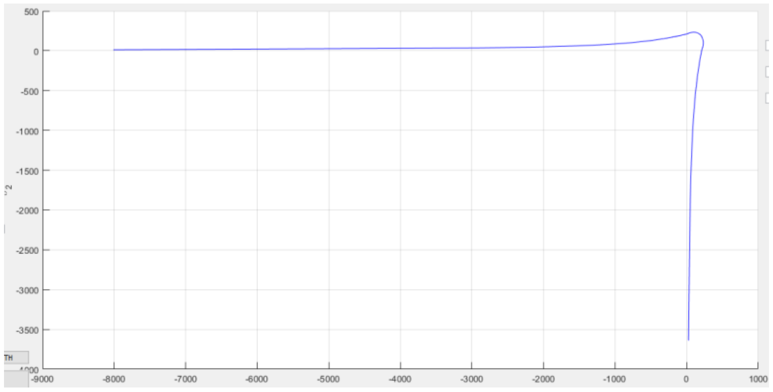


*Figure 1 Only-tension damage model*

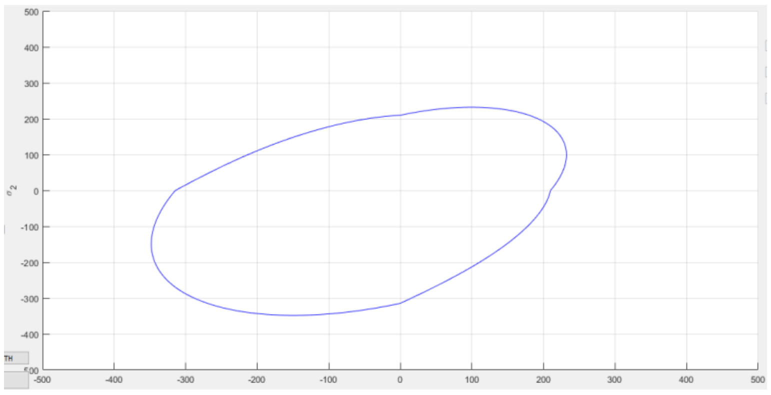


*Figure 2 Non-symmetric damage model*

Figure 1 and Figure 2above are the results of the implementation which look similar to the ones provided by the theory.

## 1.2. Part I_b

The aim of this part is to implement linear and exponential hardening/softening for each of those models. In order to implement the linear and exponential hardening and softening case, we have to modify the function called “rmap_dano1.m”. In this function, we define the variable “r_n1” and “q_n1” which are the so-called internal variable and hardening variable at step n+1. The internal variable r is equal to the rtrial found from function “modelos_de_dano1” in case of inviscid model, and the hardening variable q is found from the following formula:

$$q_{n+1} = q_n + H(r_{n+1} - r_n)$$

In this formula H is defined based on if we have linear or exponential case. In linear case, H is directly equal to the hardening coefficient which is entered in the function as input. In case of exponential, H is found based on the formula provided in the slides. In the formula, we need to define $q_\infty$ which is the upper bound in q-t curve for hardening. This value is found as:

$$q_\infty = r_0 + (r_0 - q_0)$$

It should also be noted that hardening or softening only happens in case of loading. However, when we have unloading case or elastic loading, the internal and hardening variables of step n+1 are equal to the ones from the previous step because basically in unloading or in elastic loading no hardening or softening in applied to the material.

## 1.3. Part I_c

In this part we have to run the code that we already implemented for each of the three cases provided in Table 1. The following cases are defined based on different values set for stress increments for evaluation of inviscid case. The stress is applied to the specimen linearly by the passing of time from the value equal to zero to the values provided in the table through stress increments. For all cases, the material is set to have a yield stress equal to 200 and Young modulus equal to 20000.

*Table 1 stress increments for each load path*

| Case | Point 1 | Point 2 | Point 3 |
|---|---|---|---|
| 1 | $\Delta\sigma_1^{(1)} = 400$ $\Delta\sigma_2^{(1)} = 0$ | $\Delta\sigma_1^{(2)} = -450$ $\Delta\sigma_2^{(2)} = 0$ | $\Delta\sigma_1^{(3)} = 600$ $\Delta\sigma_2^{(3)} = 0$ |
| 2 | $\Delta\sigma_1^{(1)} = 400$ $\Delta\sigma_2^{(1)} = 0$ | $\Delta\sigma_1^{(2)} = -700$ $\Delta\sigma_2^{(2)} = -300$ | $\Delta\sigma_1^{(3)} = 300$ $\Delta\sigma_2^{(3)} = 300$ |
| 3 | $\Delta\sigma_1^{(1)} = 400$ $\Delta\sigma_2^{(1)} = 400$ | $\Delta\sigma_1^{(2)} = -700$ $\Delta\sigma_2^{(2)} = -700$ | $\Delta\sigma_1^{(3)} = 300$ $\Delta\sigma_2^{(3)} = 300$ |

The results for each case include the figure of load path and the stress-strain curve. All cases are studied for both hardening and softening case which can be found in the following:

### 1.3.1. Part I_c_Case 1

The load path for this case is defined in three steps. The first step is tensile loading which crosses the damage surface. The 2nd step is compression or tensile unloading and the third step is again tensile loading. All steps are uniaxial loading which means the second component of stress is equal to zero. The load path for only-tension model is shown in Figure 3.

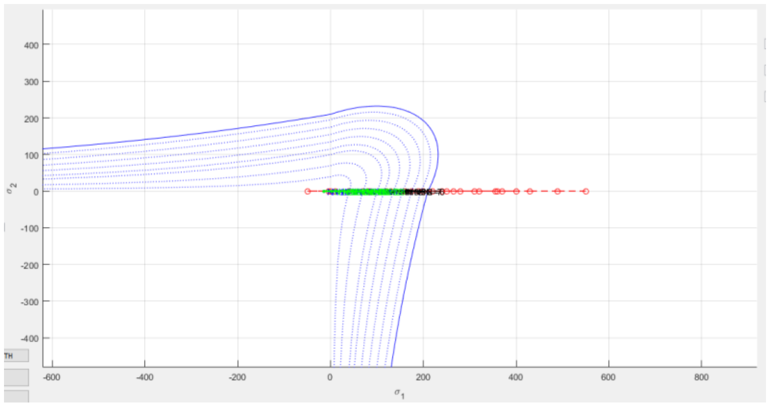


*Figure 3 Load path for only-tension case after computing the results*

The following figures show the stress-strain curve for hardening and softening cases.

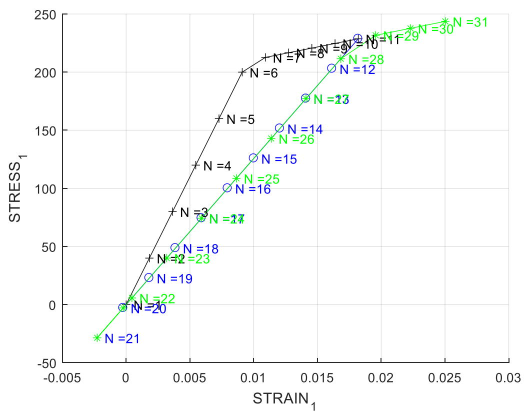


*Figure 4 Only-tension H=0.1, linear hardening, inviscid case*

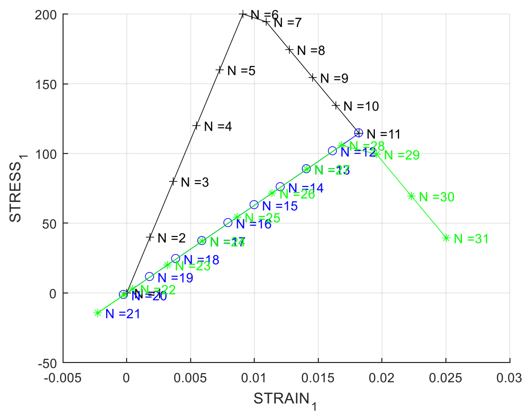


*Figure 5 Only-tension H= -0.5, linear softening, inviscid case*

Figure 4 shows the results in terms of stress against strain curve for linear hardening case with H=0.1. As it is seen, the black line which is related to the tensile loading step is changing in its slope. The transition point is where we are crossing the damage surface. The blue line is related to the elastic unloading which keeps the same slope through the whole unloading. And since we have the hardening case (H>0), the yield stress increases in value, at N=6 we have the yield stress from step 1 and at N=28 we have the yield stress for step 3 which is higher.

Figure 5 shows the results for linear softening case with H= -0.5. The trend is the same as the hardening case for each step but the difference is that because it is a softening case (H<0), the slope of the line decreases when crossing the damage surface.

The load path for non-symmetric model is shown in *Figure 6*

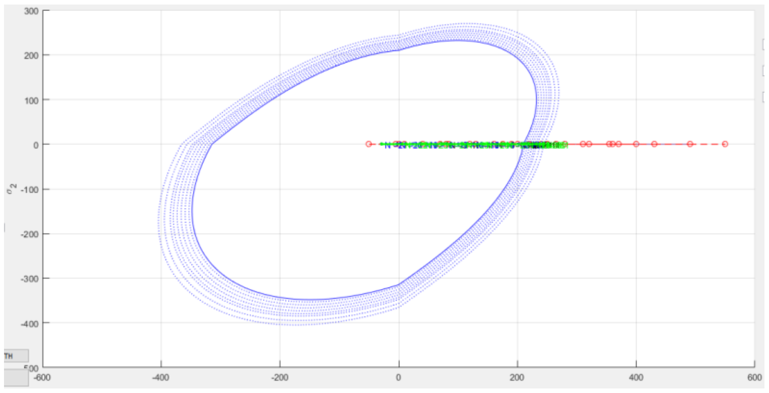
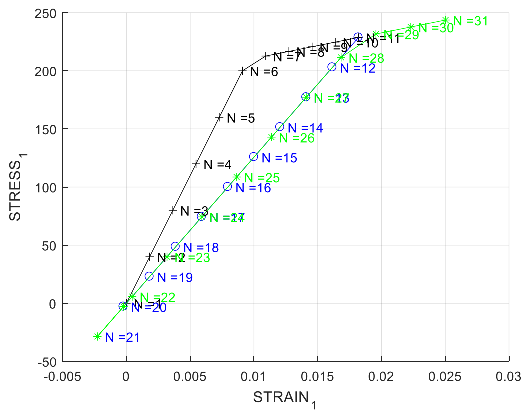


*Figure 6 Load path for non-symmetric case after computing the results*



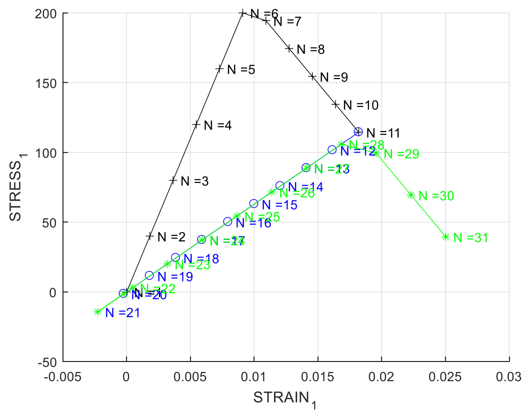


*Figure 7 Non-symmetric H=0.1, linear hardening, inviscid case*

*Figure 8 Non-symmetric H=-0.5, linear softening, inviscid case*

Figure 7 and Figure 8 show the results for non-symmetric hardening and softening case. The same trend as the only-tension case is obtained for non-symmetric case for linear hardening and softening.

The hardening coefficient is assumed to be 0.1 for all cases. As this coefficient becomes larger, the material obtains higher values for the yield stress as the damage surface is crossed. The following graph is provided to make a comparison between results of two different hardening coefficients. For softening case, the same trend is followed.

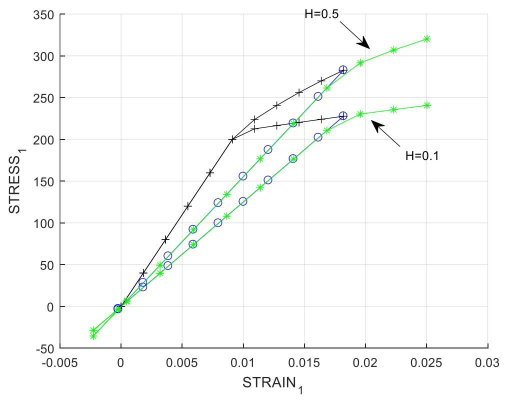


*Figure 9 Hardening case for H=0.1 and H=0.5*

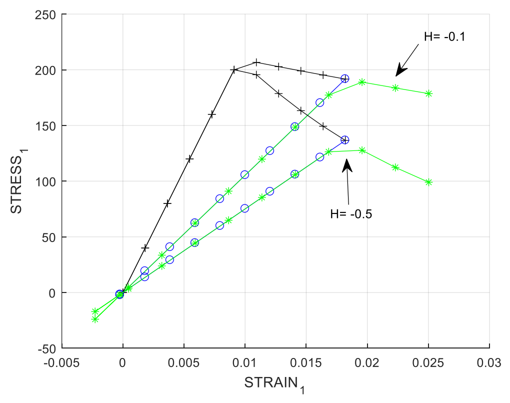


*Figure 10 Softening case for H= -0.1 and H= -0.5*

### 1.3.2. Part I_c_Case 2

The load path for this case is also defined in three steps. The first step is uniaxial tensile loading which crosses the damage surface. The 2nd step is biaxial compression or tensile unloading and the third step is again biaxial tensile loading. The load path for only-tension model is shown in Figure 11.

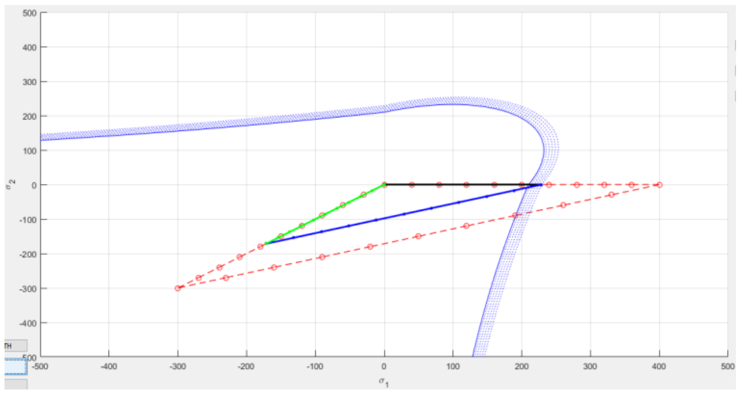


*Figure 11 Load path for only-tension after computing the results*

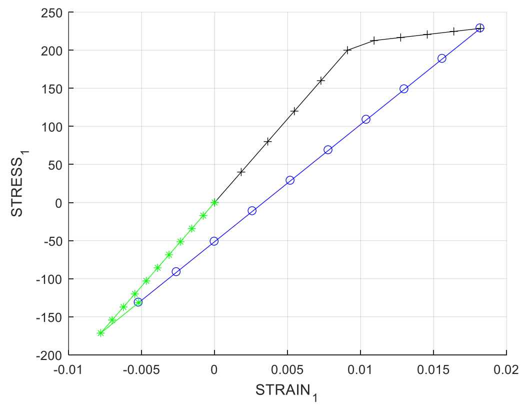


*Figure 12 Only-tension H=0.1, linear, inviscid case*

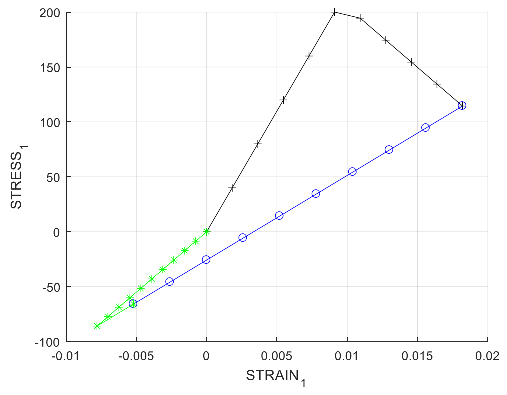


*Figure 13 Only-tension H= -0.5, linear, inviscid case*

Figure 12 shows the only-tension model stress versus strain curve the second loading case. In the first step for uniaxial loading the material has an elastic behavior until it crosses the damage surface and starts the transition. In the second step the material undergoes biaxial compression which is totally elastic. In the third step we are applying tension on the material until the stress components are zero. So the material starts to go back to zero in compression region.

Figure 13 shows the result for softening of only-tension case. It behaves the same as explained before for each loading step and since the hardening coefficient is negative the curve goes down as we cross the damage surface which means the yield stress in the next curve starts to decrease.

The load path for non-symmetric model is shown in Figure 14 for this case.

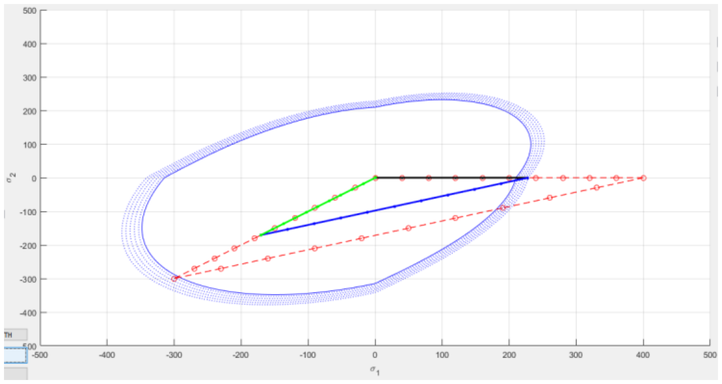


*Figure 14 Load path for non-symmetric after computing the results*

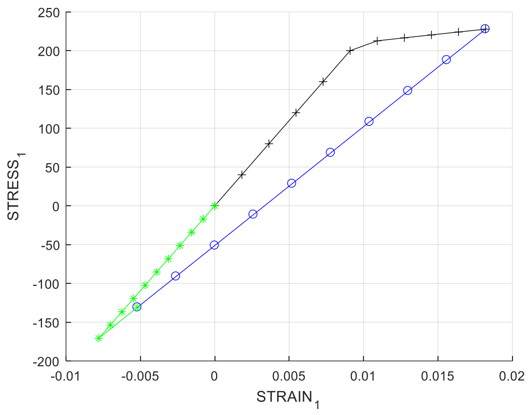


*Figure 15 non-symmetric H=0.1, linear hardening, inviscid case*

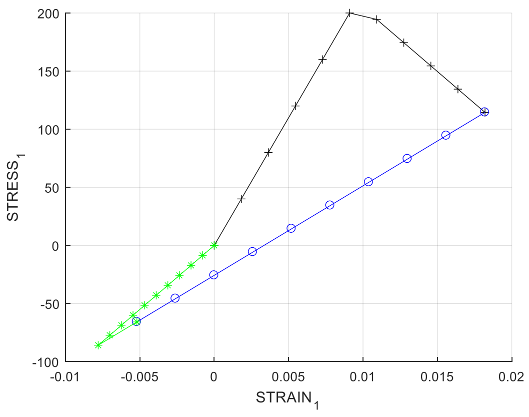


*Figure 16 non-symmetric H= -0.5, linear softening, inviscid case*

The trend for this model is the same as only-tension case for hardening and softening for each load step.

### *1.3.3. Part I_c_Case 3*

In this case the load path is defined to be biaxial tensile loading for the first step which crosses the damage surface, biaxial compression or tensile unloading for the 2nd step and is again biaxial tensile loading for the third step. The load path for only-tension model is shown in Figure 17.

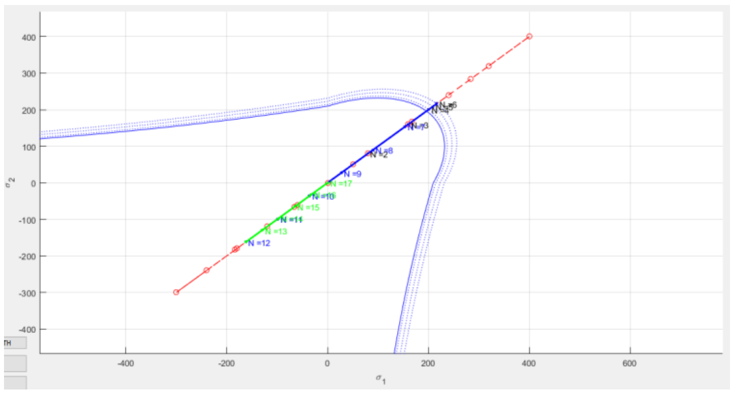


*Figure 17 Load path for only-tension after computing the results*

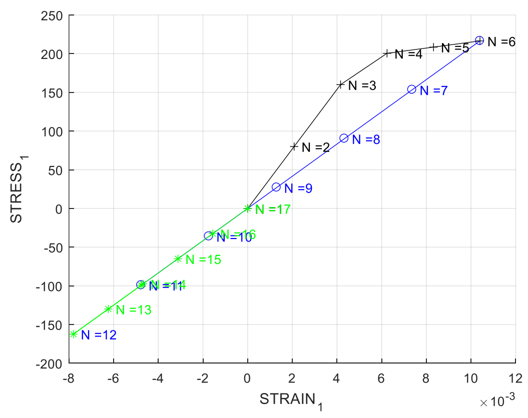


*Figure 18 only tension H=0.1, linear hardening, inviscid case*

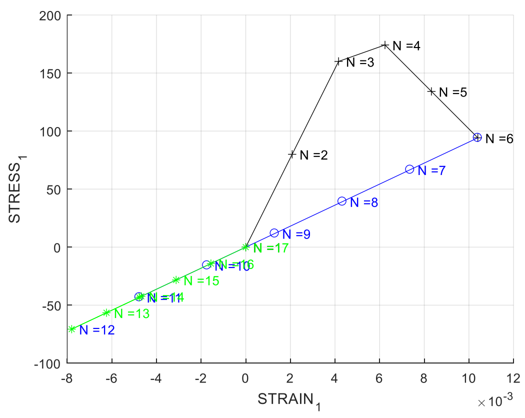


*Figure 19 only-tension H= -0.5, linear softening, inviscid case*

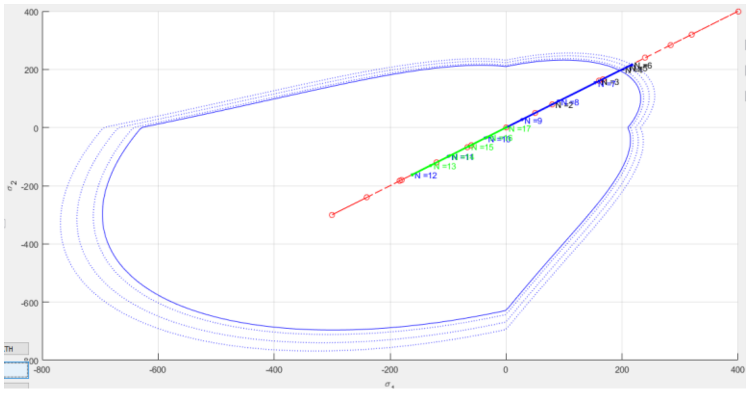


*Figure 20 Load path for non-symmetric after computing the results*

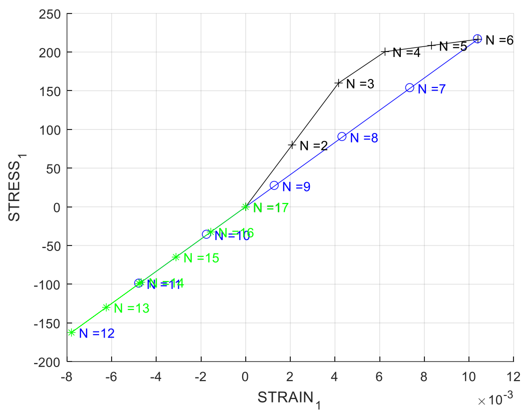


Figure 21 non-symmetric H=0.1, linear hardening, inviscid case

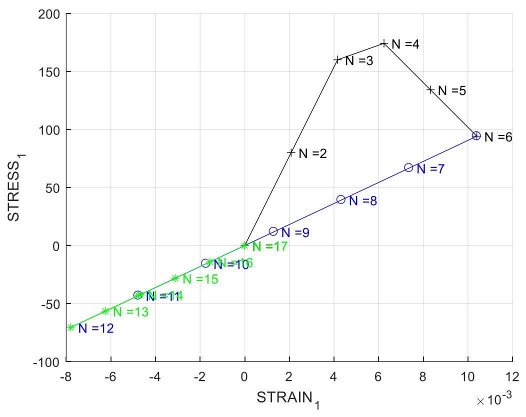


Figure 22 non symmetric H= -0.5, linear softening, inviscid case

As it is obvious from the graphs provided, the material has an elastic behavior as far as we are staying in elastic region and the curve has a transition of slope as it goes further the damage surface. Changing the load path yields to different behavior of the curve and different value for damage variable but at the end the whole logic behind the interpretation of the graphs stays the same.

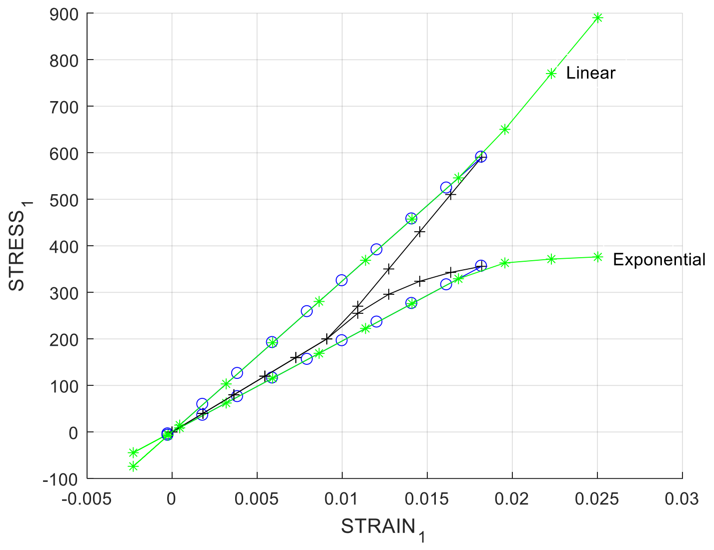


Figure 23 Comparison between linear and exponential hardening

In order to study the linear and exponential hardening/softening case, a sample load path as case 1 is applied to only-tension inviscid model. Figure 23 shows this difference when choosing the option linear or exponential hardening for a hardening coefficient of H=2. As it can be seen the material behaves differently, in linear case the stress-strain curve remains linear as the damage surface is reached and in exponential case we are observing exponential behavior. In addition, depending on the hardening coefficient, in exponential case the damage zone grows rapidly. It should be mentioned that linear and exponential case only have difference when increasing the hardening and softening coefficient.

## 2. Part II

### 2.1. Part II_d

In this part of the assignment we are supposed to implement the viscous model for symmetric type. According to the theory provided by the slides, to implement the viscous model we need the strain tensor at type step n apart from the strain tensor at time step n+1 which we already had. So, we have to create this tensor in function "damage_main.m" and then extract it as the input of the function "rmap_dano1.m" and "modelos_de_dano1.m". we also have to define the corresponding constants that we need which hasn't already been defined. In each of the functions, we have to define the condition of viscosity setting the variable "viscpr" equal to one and for inviscid case equal to zero.

After implementing the formulation of viscous model, we shall compare the viscous and inviscid behavior for a simple load path as uniaxial tension. The results are shown below:

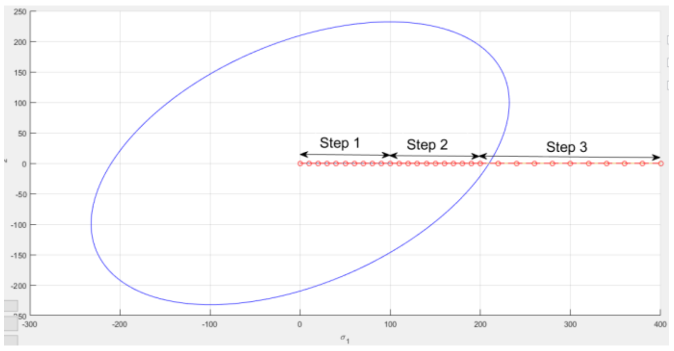


*Figure 24 Sample load path*

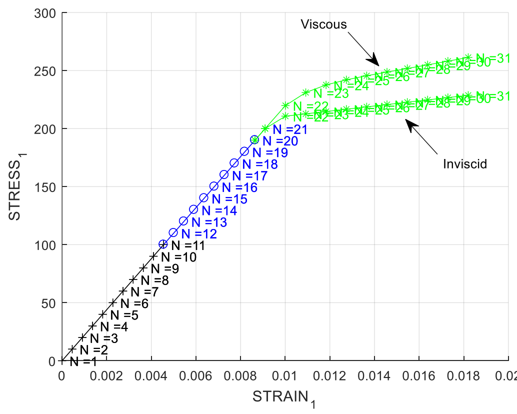


*Figure 25 Comparison between viscous and inviscid case for symmetric model*

As it can be seen from Figure 25, both viscous and inviscid models have the same behavior as far as we are in the elastic region. But when we cross the damage surface, viscous model has a steeper slope which means a higher value for$\frac{\partial\sigma}{\partial\varepsilon}$. This can be justified through the formulas provided for $\dot{\sigma}$ in the slides since in viscous case, for $\frac{\partial\sigma}{\partial\varepsilon}$ we have one extra positive term added to $\dot{\sigma}$ which doesn't exist in case of inviscid.

### 2.2. Part II_e

In this part, we are going to study the behavior of the viscous model through stress-strain keeping the Poisson ratio constant equal to 0.3 and the hardening coefficient equal to -0.5 for the sample load path of Figure 24. The study has been done by changing the value of viscous coefficient, strain rate and constant of time integration method. The results can be found below:

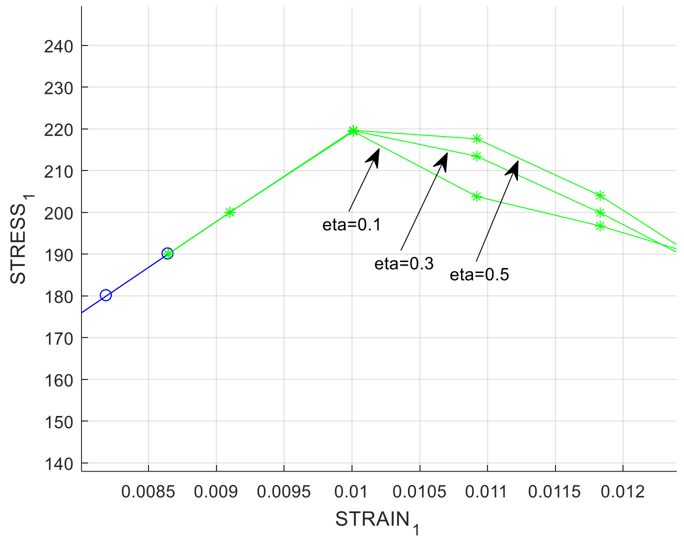


*Figure 26 stress-strain curve for viscous case for different viscous coefficients*

Figure 26 shows how the stress-strain curve is changing for different viscous coefficients (η). As it can be seen, as η increases the graphs shows higher values. This is in agreement with the theory since we know that the stress rate is directly related to the increase or decrease of the viscous coefficient.

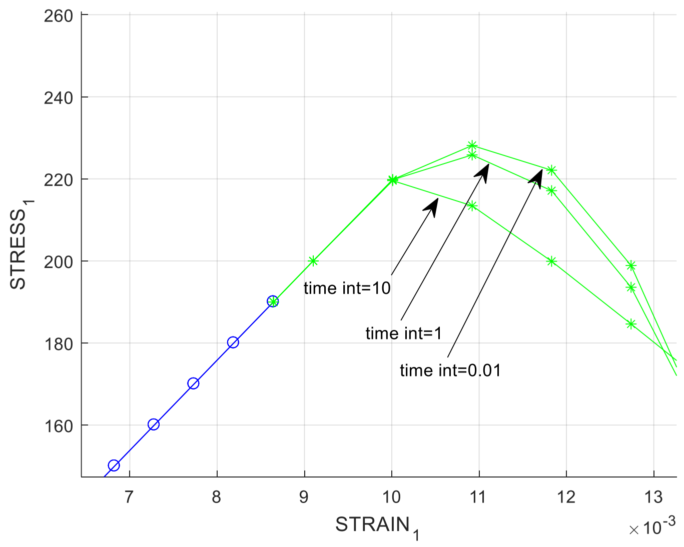


*Figure 27 stress-strain curve for viscous case for different strain rates*

Figure 27 shows the changes in stress-strain curve for different values of strain rate. In order to change the value of strain rate we have to change the variable “time_int”. As this variable decreases, the strain rate increases and the curve goes higher in value, which is in agreement with the theory since the stress rate is directly related to strain rate.

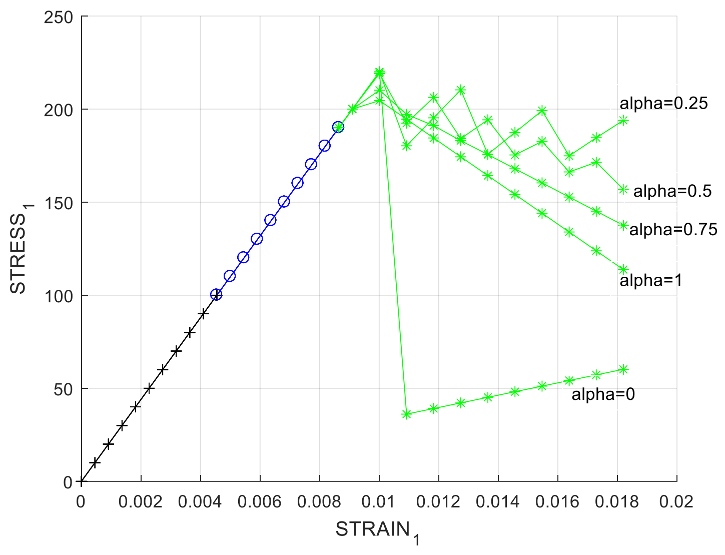


*Figure 28 stress-strain curve for viscous case for different alpha*

Figure 28 shows the stress-strain graph for different values of time integration constant (α) keeping the viscous coefficient equal to 0.3 and time interval equal to 1000. As it can be see, the method starts to have instabilities as α decreases. In fact, this method is stable in interval of [0.5,1] in which α=1 corresponds to Backward Euler (implicit) and α=0.5 corresponds to Crank-Nicolson. As we move away from this interval to zero, we have instabilities. This fact can easily be observed from the graph.

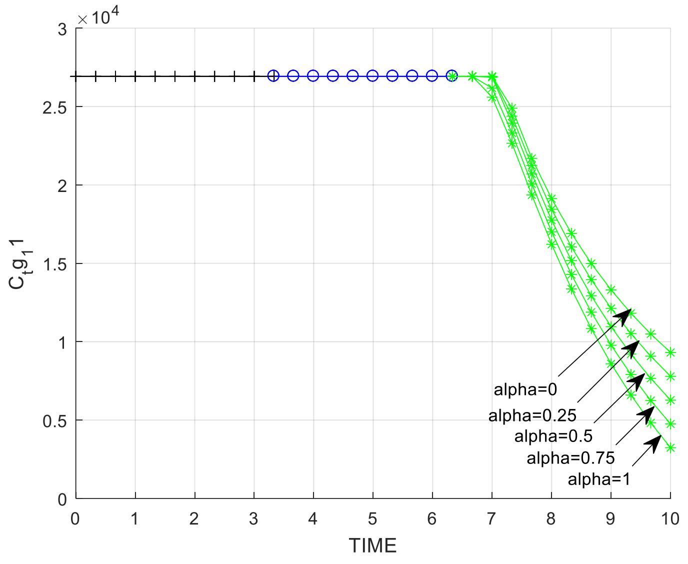


*Figure 29 C_tangent for viscous model for different values of alpha*

Figure 29 shows the variation in the first component of tangent constitutive matrix for different values of integration constant, α. As it can be seen from the figure, increasing the value of α yields a decrease in the first component of tangent constitutive matrix. This matrix depends on the value of the damage variable, d, at each time step. Therefore, in order to justify this behavior, we shall study how the damage variable changes by changing α.

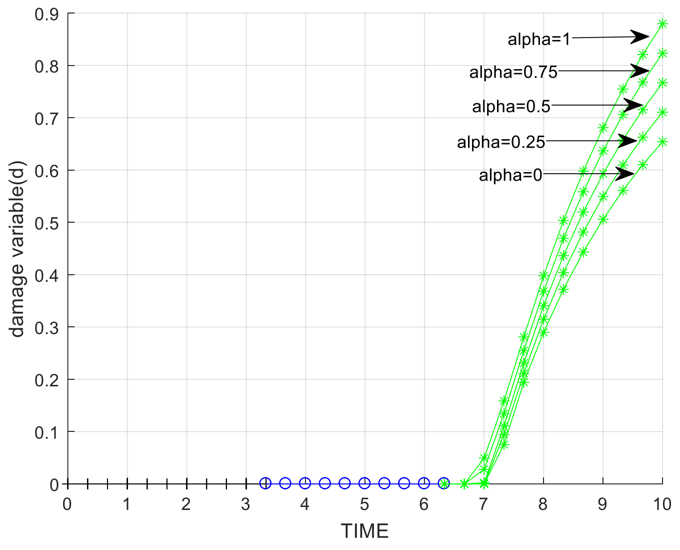


*Figure 30 variation of damage variable by changing alpha*

Figure 30 shows that the damage variable increases by increasing the value of α. According to the formula provided, the tangent constitutive matrix is related to minus of the damage variable. Therefore, increase in damage variable results in decrease in the constitutive tangent matrix, which is in agreement with the results shown in Figure 29.

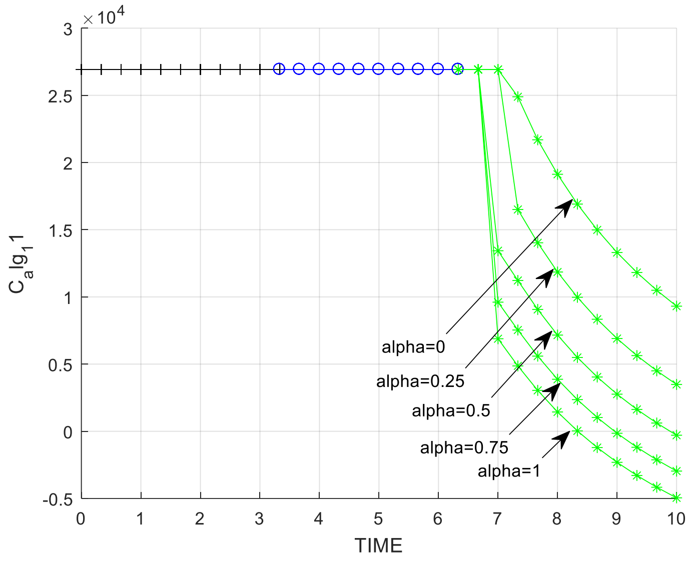


*Figure 31 C_algorithm for viscous model for different values of alpha*

Figure 31 shows the variation of the first component of algorithmic constitutive matrix with respect to time by changing the value of α. The formula provided for algorithmic constitutive matrix is the summation of two terms. The first term is the tangent constitutive matrix and the second term is a function of α. In

Figure 29 we have seen that by increasing the value of α, the value of first component of tangent constitutive matrix decreases. The second term in algorithmic constitutive matrix also decreases by increasing α. Therefore, the algorithmic matrix, in total, decreases by increasing α, which is in accordance by the results obtained.

In addition, we can observe that for value of $\alpha=0$ both the algorithmic and the tangent constitutive matrices obtain same results as it is said in the theory as well.

## 3. Conclusion

The aim of this assignment is to study different damage models for inviscid and viscous case and evaluate how these models behave under different conditions corresponding to different load path, viscosity parameters, strain rates and constant of time integration method.

The models for inviscid case have been evaluated under different load path including uniaxial and biaxial tensile and compression load. The behavior of the model for each of these load path depends on different parameters such as yield stress of the material, Poisson ratio and young modulus. However, in this study all parameters are kept the same and different load path is applied for different models in order to provide the context for better comparison.

A comparison has been done for linear and exponential hardening and softening. As the results show, in case of higher hardening/softening coefficients, exponential formulation yields to fast growth/shrink of damage zone apart from the exponential behavior of stress-strain curve.

After studying the inviscid model, viscous model has been implemented for all damage types. The influence of different parameters of viscous case on the behavior of the model has been evaluated keeping other parameters as a constant. In detail, the stress-strain curve, the tangent and algorithmic constitutive model and the damage variable have been observed for different values of viscosity parameter, strain rate and time-integration constant. The results for each case has been included in the report. The code has been added as the appendix to give more details on implementation.

## 4. Appendix

### 4.1. Function dibujar_criterio_dano1

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
%*************************************************************************************
%*                   PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODEL
%*
%*                                                                                 %*
%*       function [ce] = tensor_elastico (Eprop, ntype)                  %*
%*                                                                                 %*
%*       INPUTS                                                         %*
%*                                                                                 %*
%*                    Eprop(4)    vector de propiedades de material              %*
%*                                      Eprop(1)=  E------>modulo de Young          %*
%*                                      Eprop(2)=  nu----->modulo de Poisson        %*
%*                                      Eprop(3)=  H----->modulo de Softening/hard. %*
%*                                      Eprop(4)=sigma_u----->tensiï¿½n ï¿½ltima        %*
%*                     ntype                                    %*
%*                                ntype=1  plane stress                           %*
%*                                ntype=2  plane strain                           %*
%*                                ntype=3  3D                                     %*
%*                     ce(4,4)     Constitutive elastic tensor  (PLANE S.       )    %*
%*                     ce(6,6)                                  ( 3D)                %*
%*************************************************************************************


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



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


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

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

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

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


elseif MDtype==2         %%second type:only-tension damage model
```

```
tetha=[-pi/2+0.01:0.01:pi-0.01];
D=size(tetha);                      %*  Range
m1=cos(tetha);                      %*
m2=sin(tetha);                      %*
Contador=D(1,2);                    %*


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

for i=1:Contador
% McAuly Braket x*(x>0) if x>0 I obtain 1, Otherwise I obtain a 0
A = m1(i)*(m1(i)>0);
B = m2(i)*(m2(i)>0);

radio(i)= q/sqrt([A B 0 nu*(A+B)]*ce_inv*[m1(i) m2(i) 0 ...
nu*(m1(i)+m2(i))]');

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

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




elseif MDtype==3    %%third type:non_symmetric damage model

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


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

for i=1:Contador
% McAuly Braket x*(x>0) if x>0 I obtain 1, Otherwise I obtain a 0
A = m1(i)*(m1(i)>0);
B = m2(i)*(m2(i)>0);
alpha_N =A+B;
alpha_D =abs(m1(i))+abs(m2(i));
alpha = alpha_N/alpha_D;
radio(i)= q/((alpha+(1-alpha)/n)*(sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i) 0
...
nu*(m1(i)+m2(i))]')));
s1(i)=radio(i)*m1(i);
s2(i)=radio(i)*m2(i);

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


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

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

## 4.2. Function modelos_de_dano1

```
function [rtrial,taw_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n,eps_n,viscpr,ALPHA)
%*************************************************************************************
%*          Defining damage criterion surface                                       %*
%*                                                                                  %*
%*
%*                         MDtype=  1      : SYMMETRIC                              %*
%*                         MDtype=  2      : ONLY TENSION                           %*
%*                         MDtype=  3      : NON-SYMMETRIC                          %*
%*                                                                                  %*
%*                                                                                  %*
%* OUTPUT:                                                                          %*
%*                         rtrial                                                   %*
%*************************************************************************************


%*************************************************************************************
if (MDtype==1)      %* Symmetric
    taw_n = sqrt(eps_n*ce*eps_n') ;
    taw_n1 = sqrt(eps_n1*ce*eps_n1') ;

    if viscpr == 0    %%Inviscid
        rtrial= sqrt(eps_n1*ce*eps_n1') ;
    elseif viscpr == 1    %%Viscous
        rtrial = (1-ALPHA)*taw_n + ALPHA*taw_n1;
    end

elseif (MDtype==2)  %* Only tension

    sigma_e_n = eps_n*ce;
    sigma_e_plus_n(:) = sigma_e_n(:).*(sigma_e_n(:)>0);
    taw_n = sqrt(sigma_e_plus_n*eps_n') ;

    sigma_e_n1 = ce*eps_n1';
    sigma_e_plus_n1(:) = sigma_e_n1(:).*(sigma_e_n1(:)>0);
    taw_n1 = sqrt(sigma_e_plus_n1*eps_n1') ;

    if viscpr == 0   %%inviscid
        rtrial= sqrt(sigma_e_plus_n1*eps_n1');
    elseif viscpr == 1   %%viscous
        rtrial = (1-ALPHA)*taw_n + ALPHA*taw_n1;
    end

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

    sigma_e_n = eps_n*ce;
    sigma_e_plus_n(:) = sigma_e_n(:).*(sigma_e_n(:)>0);
    theta_n = (sigma_e_plus_n(1)+sigma_e_plus_n(2))/(abs(sigma_e_n(1))+abs(sigma_e_n(2)));
    taw_n= (theta_n+(1-theta_n)/n)*sqrt(eps_n*ce*eps_n');

    sigma_e_n1 = eps_n1*ce;
    sigma_e_plus_n1(:) = sigma_e_n1(:).*(sigma_e_n1(:)>0);
    theta_n1 =
(sigma_e_plus_n1(1)+sigma_e_plus_n1(2))/(abs(sigma_e_n1(1))+abs(sigma_e_n1(2)));
    taw_n1= (theta_n1+(1-theta_n1)/n)*sqrt(eps_n1*ce*eps_n1');

    if viscpr == 0    %%inviscid
        rtrial= (theta_n1+(1-theta_n1)/n)*sqrt(eps_n1*ce*eps_n1');
    elseif viscpr == 1     %%viscous
        rtrial = (1-ALPHA)*taw_n + ALPHA*taw_n1;
    end

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

## 4.3. Function damage_main

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strain,MDtype,n,TimeTo
tal)
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','damage
variable(d)','C_alg_11','C_tg_11'};

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



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



totalstep = sum(istep) ;


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


% Elastic constitutive tensor
% ----------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
eps_n  = zeros(mstrain,1);

% Historic variables
% hvar_n(1:4) --> empty
% hvar_n(5) = q --> Hardening variable
% hvar_n(6) = r --> Internal variable
hvar_n  = zeros(mhist,1)  ;

% INITIALIZING  (i = 1) !!!!
% ***********i*
i = 1 ;
r0 = sigma_u/sqrt(E);
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_n
hvar_n(7) = 0; % d
hvar_n(8) = ce(1,1); % C_alg_11
hvar_n(9) = ce(1,1); % C_tg_11

eps_n1 = strain(i,:) ;
sigma_n1 =ce*eps_n1'; % Elastic
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0  sigma_n1(4)];

nplot = 5 ;
vartoplot = cell(1,totalstep+1) ;
vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
vartoplot{i}(3) = hvar_n(7)  ; %  Damage variable (d)
vartoplot{i}(4) = hvar_n(8) ; % Internal variable (r)
vartoplot{i}(5) = hvar_n(9) ; % Internal variable (r)

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


%*************************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,delta_t);
        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6), 'r:',MDtype,n );
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1)                          ;
        elseif (aux_var(1)<=0)
        end

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

        % VARIABLES TO PLOT (set label on cell array LABELPLOT)
        % ----------------
        vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
        vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
        vartoplot{i}(3) = hvar_n(7); %  Damage variable (d)
        vartoplot{i}(4) = hvar_n(8);
        vartoplot{i}(5) = hvar_n(9);
    end
end
```

### *4.4. Function rmap_dano1*

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

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


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


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


%*************************************************************************
%*      Damage surface                                                  %*
[rtrial,taw_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n,eps_n,viscpr,ALPHA);
%*************************************************************************


%*************************************************************************
%*   Ver el Estado de Carga                                               %*
%*   --------->    fload=0 : elastic unload                               %*
%*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)         %*
%% implementing exponential hardening for inviscid and viscous case

fload=0;

if(rtrial > r_n)
%     *   Loading
    fload=1;
    delta_r = rtrial-r_n;

    if viscpr == 0  %inviscid case
        r_n1= rtrial  ;
    elseif viscpr == 1 %viscous case
        Z = (eta-(1-ALPHA)*delta_t)/(eta+ALPHA*delta_t);
        Y = delta_t/(eta+ALPHA*delta_t);
        r_n1 = Z*r_n + Y*rtrial;
    end
    if hard_type == 0
%          Linear
        H_n1 = H;
        q_n1= q_n+ H*delta_r;
    elseif hard_type == 1
        q_inf = r0+(r0-zero_q);
%         exponential
        if H>0
            H_n1 = H*((q_inf-r0)/r0)*exp(H*(1-rtrial/r0));
            q_n1= q_n+ ((H*(q_inf-r0)/r0)*exp(H*(1-rtrial/r0)))*delta_r;
        elseif H<0
            H_n1 = H*((q_inf-r0)/r0)*(1/exp(H*(1-rtrial/r0)));
            q_n1 = q_n+ ((H*(q_inf-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

%%
% Damage variable
% ---------------
dano_n1 = 1-(q_n1/r_n1);  %damage parameter
sigma_n1  =(1-dano_n1)*ce*eps_n1';

%hold on
%plot(sigma_n1(1),sigma_n1(2),'bx')


%% tangent constitutive equation
if viscpr == 1
   if rtrial > r_n
       Ce_alg_n1 = (1-dano_n1)*ce+((ALPHA*delta_t)/(eta+ALPHA*delta_t))*...
           (1/taw_n1)*((H_n1*r_n1-q_n1)/(r_n1^2))*((ce*eps_n1')'*(ce*eps_n1'));
       C_alg = Ce_alg_n1(1,1);

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

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

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

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

%  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 ;
hvar_n1(7)=dano_n1;


%*************************************************************************************
%* 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;
%*************************************************************************************

end
```