Computational Solid Mechanics

Masters in Numerical Methods

Assignment 1

Numerical Integration of Damage Models

Shardool Kulkarni

March 30, 2020

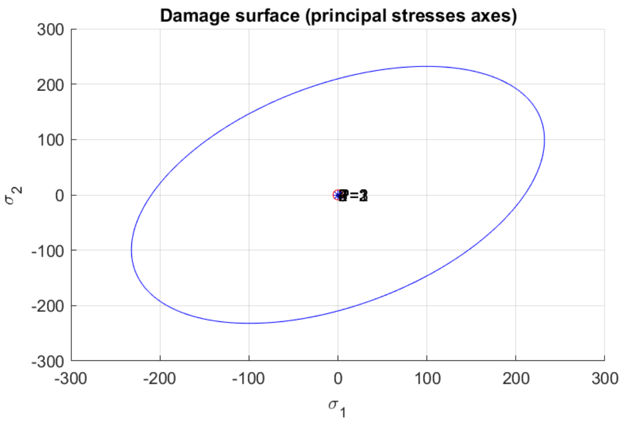
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Contents

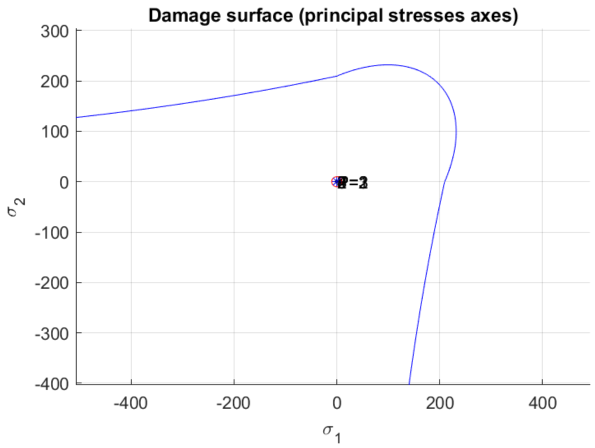
1 Part 1: Rate Independent Damage Models

1.1 Characterisation of the Elastic Domain

The code initially has the the symmetric tension compression model, we have further implemented the Tension only model and the symmetric Tension compression model. All the models in this report are run for a Plane Strain case. All of the models are represented in the High-Westergaard stress space.



(a) Symmetric Tension Compression



(b) Tension only model

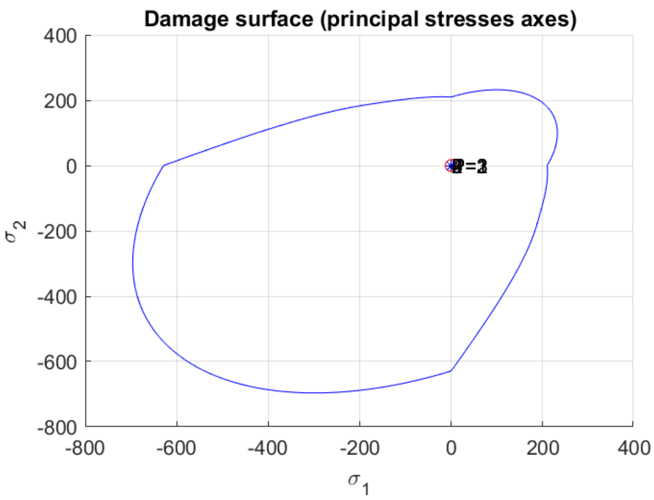
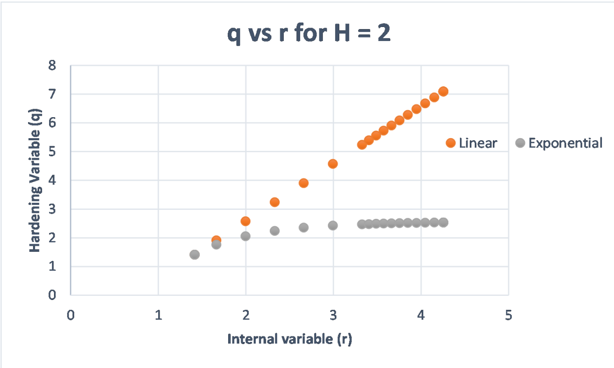


Figure 2: Tension Compression Model

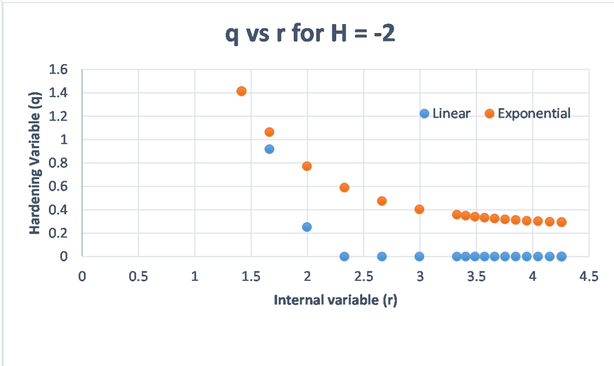
The behaviour for the curves is similar for the tension only model and the symmentric tension compression model in the first quadrant. The tension only model approaches the horizontal and the vertical axes asymptotically. Compressive loading which happens in the third quadrant is naturally absent. Whereas it is postulated that the compressive loading occurs at much higher stresses for tension compression model and that can be seen in fig 2 . This leads to a bigger elastic domain in the third quadrant compared to the first quadrant.

1.2 Characterization of the Hardening and Softening law

The MATLAB program already had provisions for linear hardening, exponential hardening was programmed where the hardening variable q has an exponential response with respect to the damage variable r. The exponential and linear hardening / softening is observed by plotting the hardening variable q vs internal variable r for a uniaxial tensile loading case.It was also observed that the evolution of the damage variable with time was slower in the exponential case compared to the linear case.

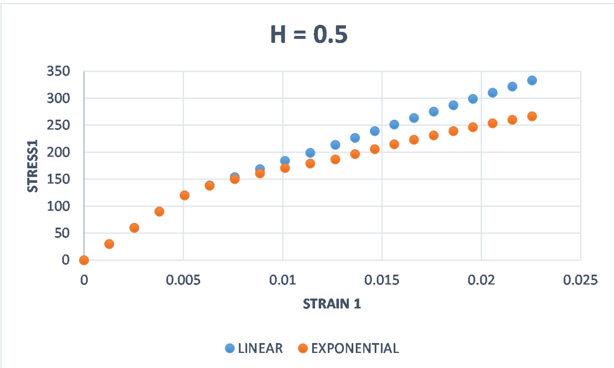


(a) Hardening Linear vs exponential

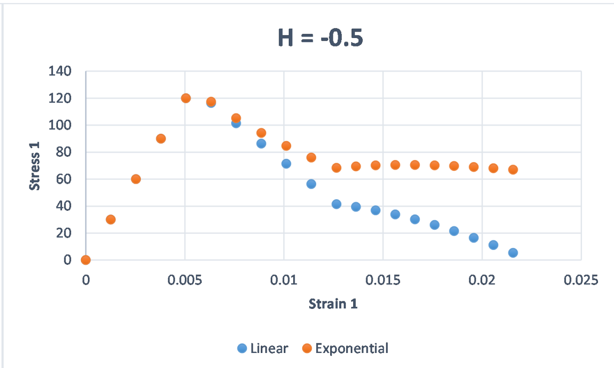


(b) Softening Linear vs Exponential

Figure 3: q vs r plots for Hardening and Softening cases



(a) Hardening Linear vs exponential



(b) Softening Linear vs Exponential

Figure 4: Stress vs Strain plots for Hardening and Softening cases

1.3 Case 1: Uniaxial Tension Compression loading

The robustness of the code was correctly verified by testing it under several conditions, all the cases are analyzed using the exponential hardening softening law. The linear case has already been implemented as stated earlier

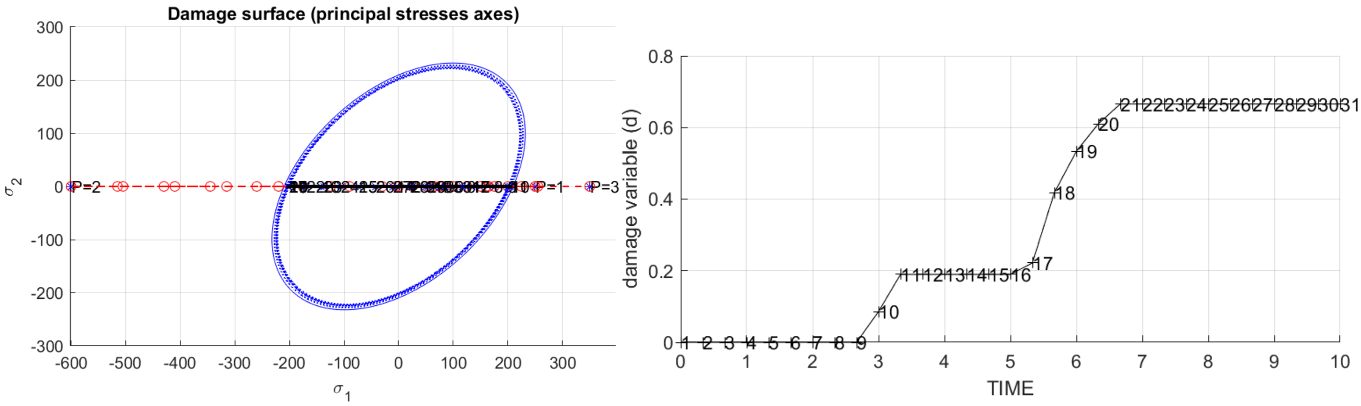
The loading path for this case is given by:

$$\Delta\bar{\sigma}_1^{(1)} = 250 \qquad \Delta\bar{\sigma}_2^{(1)} = 0 \tag{1}$$

$$\Delta\bar{\sigma}_1^{(2)} = -600 \qquad \Delta\bar{\sigma}_2^{(2)} = 0 \tag{2}$$

$$\Delta\bar{\sigma}_1^{(3)} = 300 \qquad \Delta\bar{\sigma}_2^{(3)} = 0 \tag{3}$$

Exponential Hardening of $H = -0.5$ and poisson ratio $\nu = 0.3$ were selected to illustrate all the cases in this section.



(a) Symmetric Model loading path and damage surface

(b) Damage variable evolution

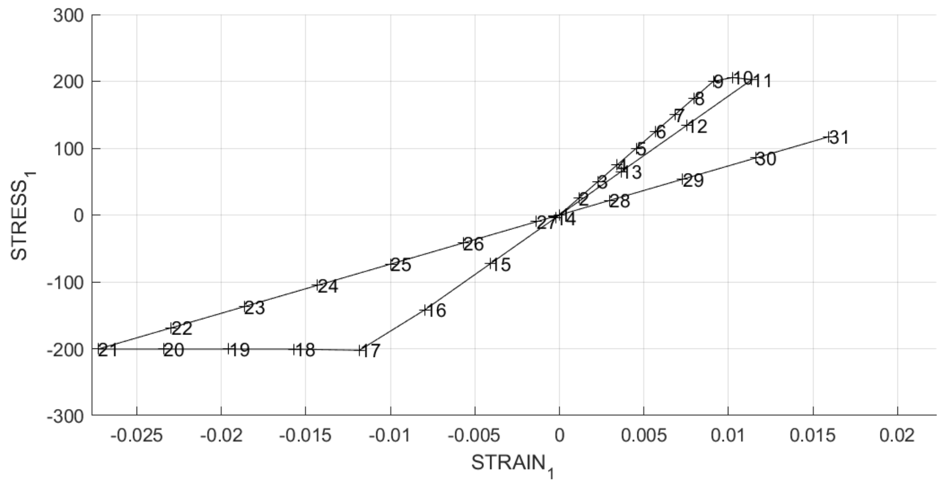
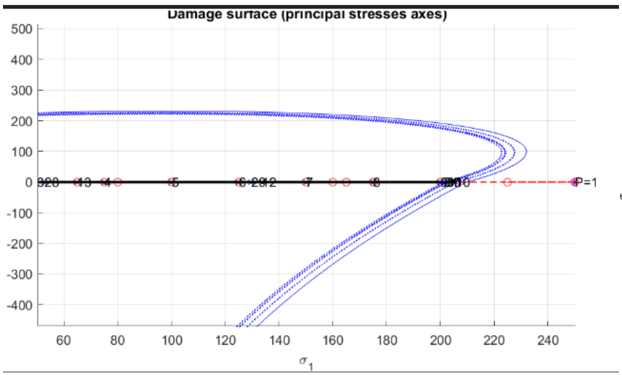


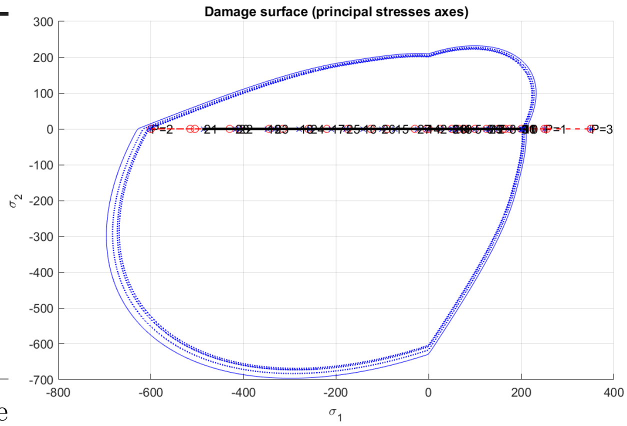
Figure 6: Stress vs Strain curve

Loading paths were chosen to illustrate the response of tensile and compressive damage. The damage plot fig 5a shows that the material deforms the damage surface in order to contain the applied tension and compressive stresses. The stress-strain curve 6 shows the path followed by the material over the application of successive cycles of pressure, first we see an elastic regime under tension, then the pure loading and the corresponding damage occurs,we then see tensile unloading and elastic deformation, then pure loading happens again in order to increase its damage variable. Finally the material is unloaded. The evolution of the damage variable with time is also plotted which illustrates the evolution of the damage variable only during the tensile damage and the compressive damage.

Similar conditions are selected and the tension only model and the non-symmetric tension compression models are simulated.



(a) Tension only model loading path and damage surface



(b) Tension Compression Model

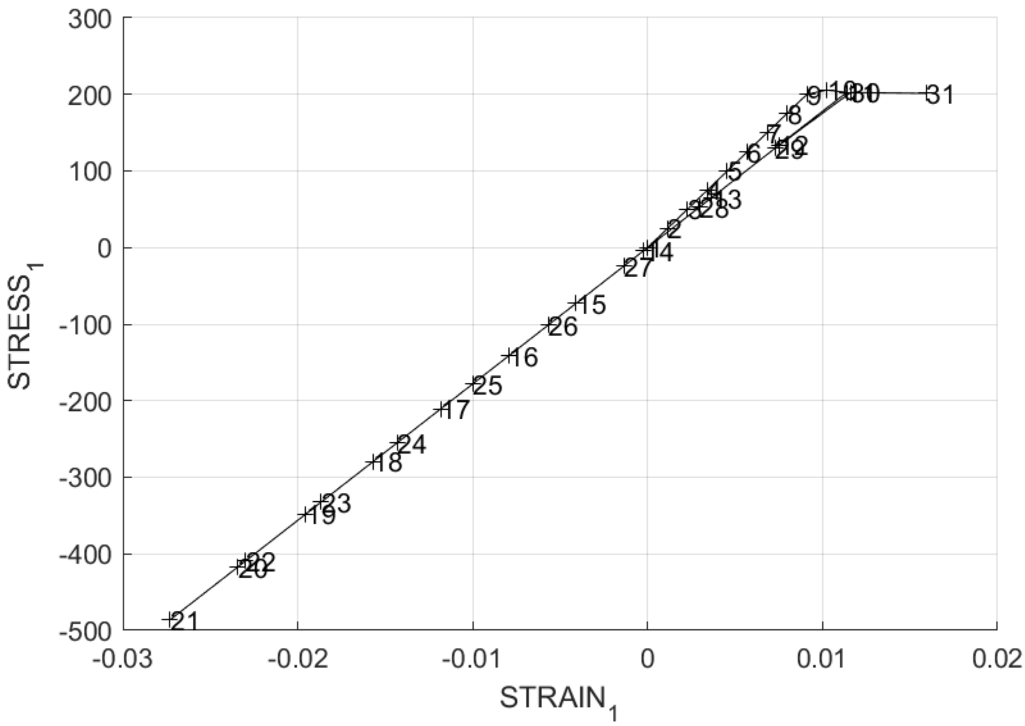


Figure 8: Stress vs Strain curve

According to the theory both the Tension only model and the non symmetric tension compression model behave the same way under these loading conditions. The first load step is similar as seen for the symmetric case where we see elastic response followed by a damage under the tensile load. In the second step the tensile loading and the compressive unloading occurs, the load step damage for compression does not occur as we do not encounter the damage surface during compression. The compressive unloading step also provides an elastic response.

1.4 Case 2: Biaxial tension compression loading

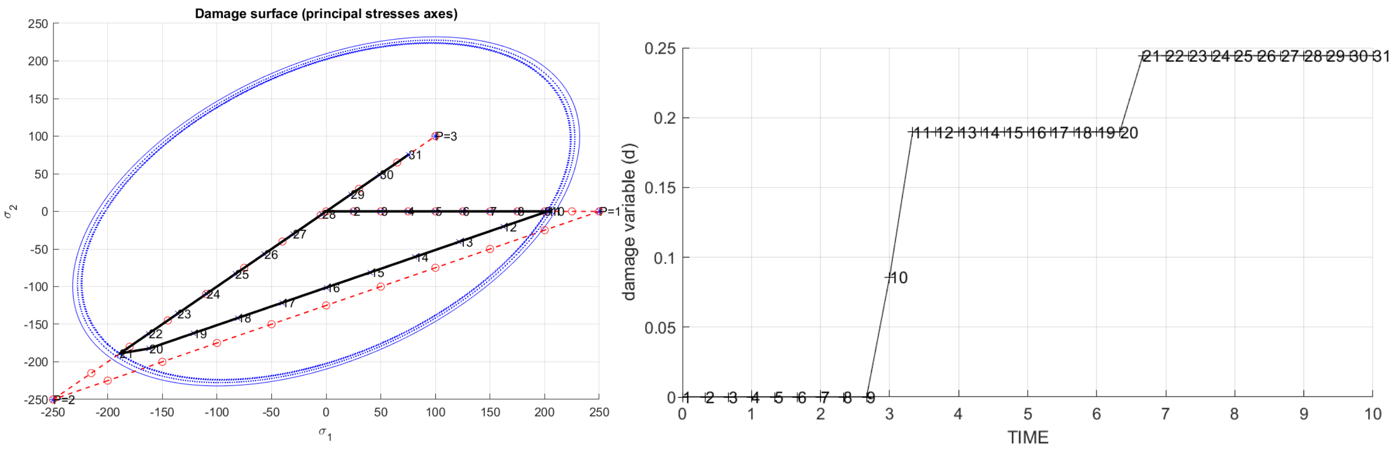
The loading path for this case is given by :

$$\Delta\bar{\sigma}_1^{(1)} = 250 \qquad \Delta\bar{\sigma}_2^{(1)} = 0 \tag{4}$$

$$\Delta\bar{\sigma}_1^{(2)} = -250 \qquad \Delta\bar{\sigma}_2^{(2)} = -250 \tag{5}$$

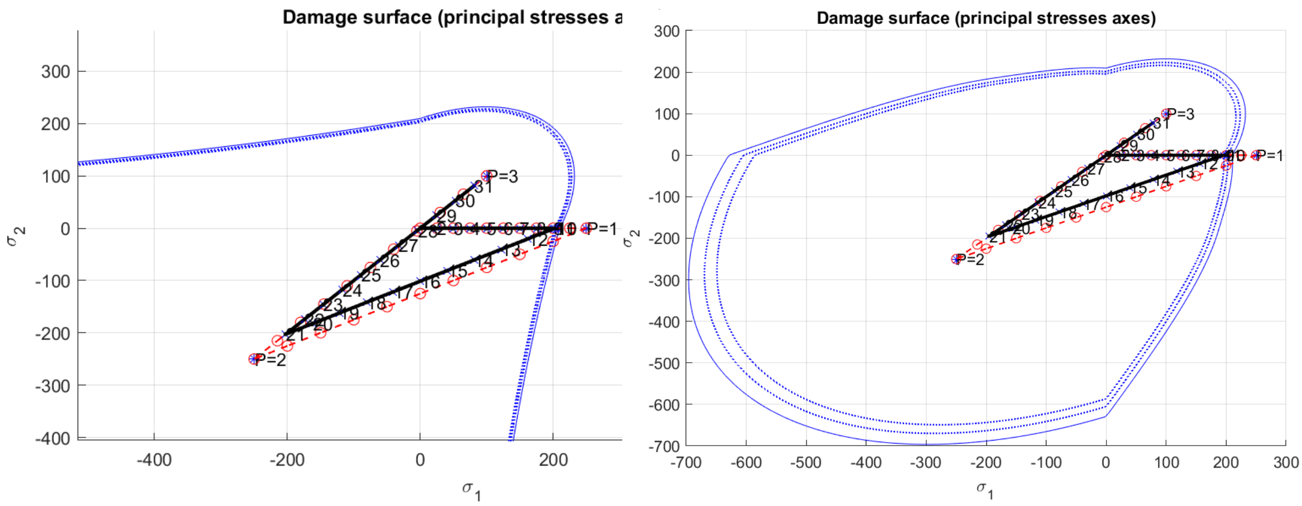
$$\Delta\bar{\sigma}_1^{(3)} = 100 \qquad \Delta\bar{\sigma}_2^{(3)} = 100 \tag{6}$$

Exponential Hardening of $H = -0.5$ and poisson ratio $\nu = 0.3$ were selected to illustrate all the cases in this section.



(a) Symmetric Model loading path and damage surface

(b) Damage variable evolution



(a) Tension Only Model

(b) Tension Compression model

The second loading case is characterized by a first tensile loading on the x-axis, sufficiently large to overcome the elastic limit of the material, with an identical response in the three

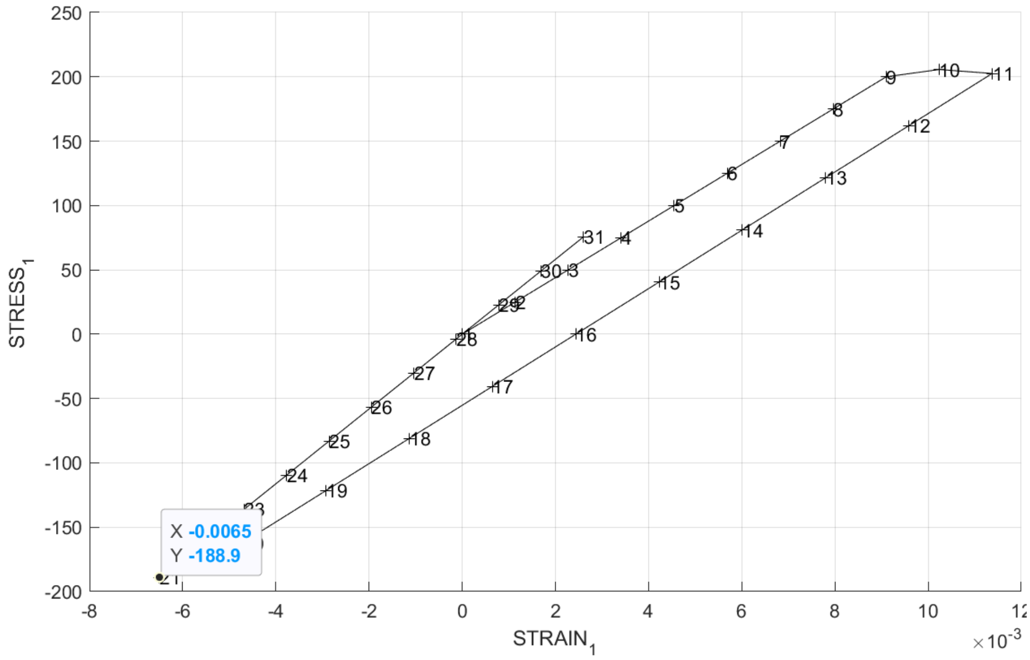


Figure 11: Stress vs Strain curve

models. This causes and damage and due to the negative hardening surface causes contraction of the damage surface in the stress space. Due to this contraction, the second and the third steps of biaxial loading gives out an elastic response.

The stress-strain curve illustrates the three load cases sufficiently well, it is fairly similar for all the three types of loading cases. Also, the damage happens in two load steps, we can see that the damage variable takes two jumps in step 10 and step 21 in fig 9b. It is constant otherwise.

1.5 Case 3: Biaxial Tension Compression Loading

The loading path for this case is given by :

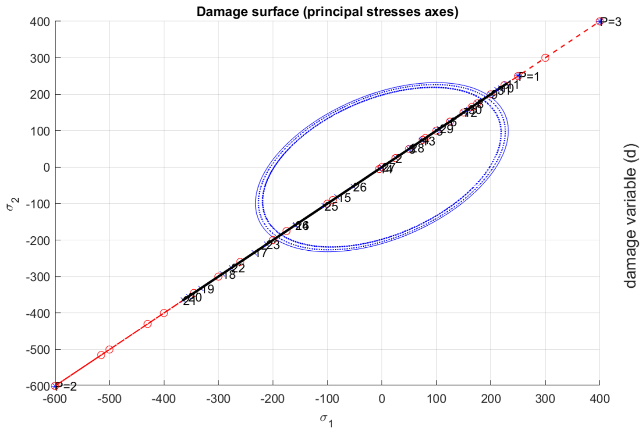
$$\Delta\bar{\sigma}_1^{(1)} = 250 \qquad \Delta\bar{\sigma}_2^{(1)} = 250 \tag{7}$$

$$\Delta\bar{\sigma}_1^{(2)} = -600 \qquad \Delta\bar{\sigma}_2^{(2)} = -600 \tag{8}$$

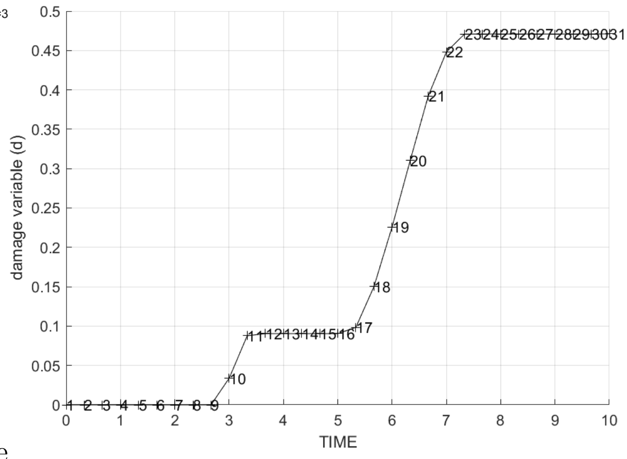
$$\Delta\bar{\sigma}_1^{(3)} = 400 \qquad \Delta\bar{\sigma}_2^{(3)} = 400 \tag{9}$$

Exponential Hardening of $H = -0.5$ and poisson ratio $\nu = 0.3$ were selected to illustrate all the cases in this section.

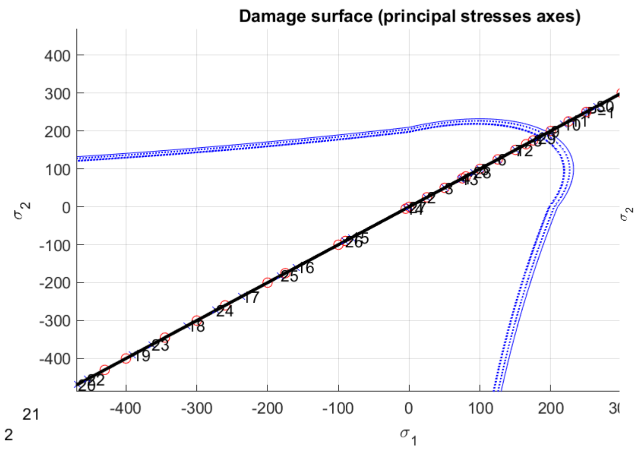
The first load step is similar to the symmetric case, where the elastic behaviour is observed until the damage surfac is reached and then damage occurs. The second step is biaxial unloading followed by biaxial compressive loading exhibits elastic like behaviour unlike the symmetric case where damage under compressive biaxial loading was observed. The Stress vs Strain cvurve is plotted in figure 14



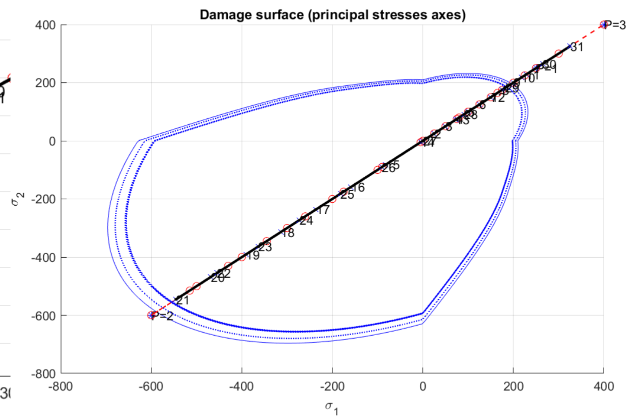
(a) Symmetric Model loading path and damage surface



(b) Damage variable evolution



(a) Tension Only Model



(b) Tension Compression model

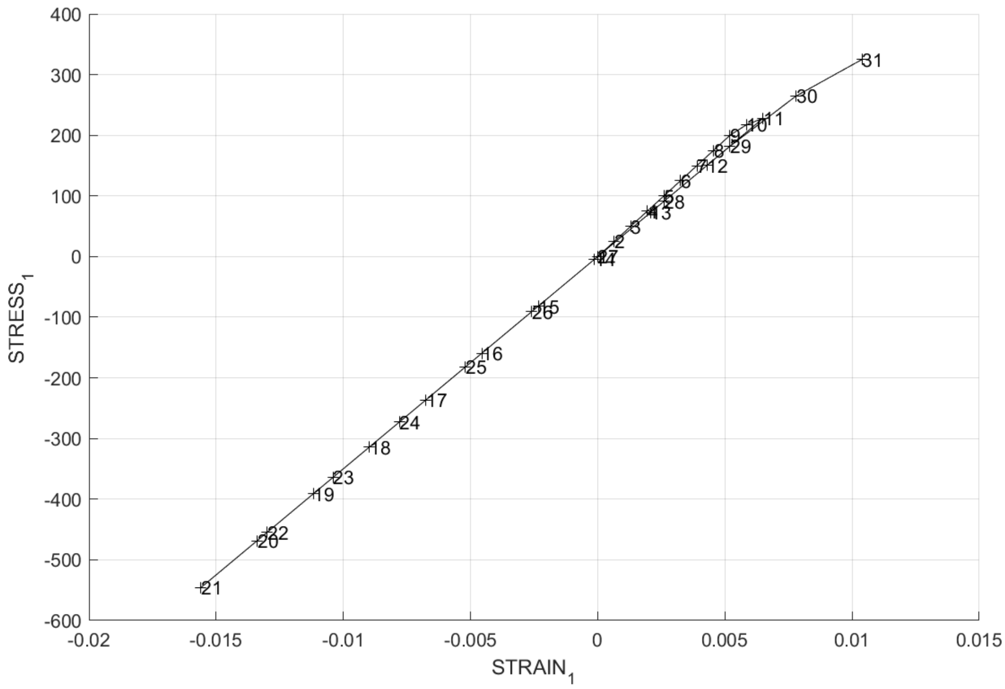


Figure 14: Stress vs Strain curve

2 Part 2: Rate Dependent Damage Models

2.1 Effect of variation of viscosity parameters

The continuum isotropic visco-damage model was implemented for the plane strain symmetric tension compression model. In this subsection the problem is subjected to uniaxial tension with the following loading paths and parameters.

$$\Delta\bar{\sigma}_1^{(1)} = 100 \qquad \Delta\bar{\sigma}_2^{(1)} = 0 \tag{10}$$

$$\Delta\bar{\sigma}_1^{(2)} = 100 \qquad \Delta\bar{\sigma}_2^{(2)} = 0 \tag{11}$$

$$\Delta\bar{\sigma}_1^{(3)} = 300 \qquad \Delta\bar{\sigma}_2^{(3)} = 0 \tag{12}$$

additional conditions are as follows $H = 0$, Time Interval $= 1$ $\nu = 0.3$ and the viscosity parameter $\eta = 0, 0.1, 1$.

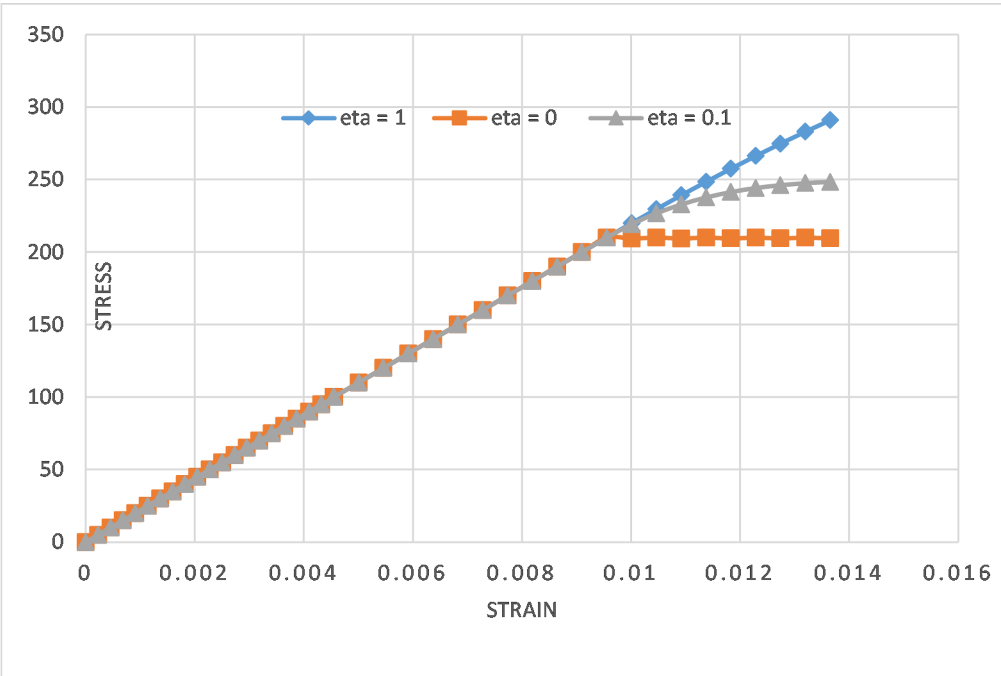


Figure 15: Viscosity variation

The stress vs strain curve 14 shows that the most viscous case offers more elastic behavior due to lesser damage as compared to the inviscid case. The behavior for viscosity rates $\eta > 1$ follows a similar trend to $\eta = 1$.

2.2 Effects of variation of strain rate

The continuum isotropic visco-damage model was implemented for the plane strain symmetric tension compression model. The strain rate is inversely proportional to the time interval in which the problem is being solved. In this section we vary the Time interval by keeping the other parameters same.

$$\Delta\bar{\sigma}_1^{(1)} = 100 \qquad \Delta\bar{\sigma}_2^{(1)} = 0 \tag{13}$$

$$\Delta\bar{\sigma}_1^{(2)} = 100 \qquad \Delta\bar{\sigma}_2^{(2)} = 0 \tag{14}$$

$$\Delta\bar{\sigma}_1^{(3)} = 300 \qquad \Delta\bar{\sigma}_2^{(3)} = 0 \tag{15}$$

additional conditions are as follows $H = 0$, Time Interval $= 0.1, 1, 10, 100$ $\nu = 0.3$ and the viscosity parameter $\eta = 1$. The integration coefficient was $\alpha = 0.5$.

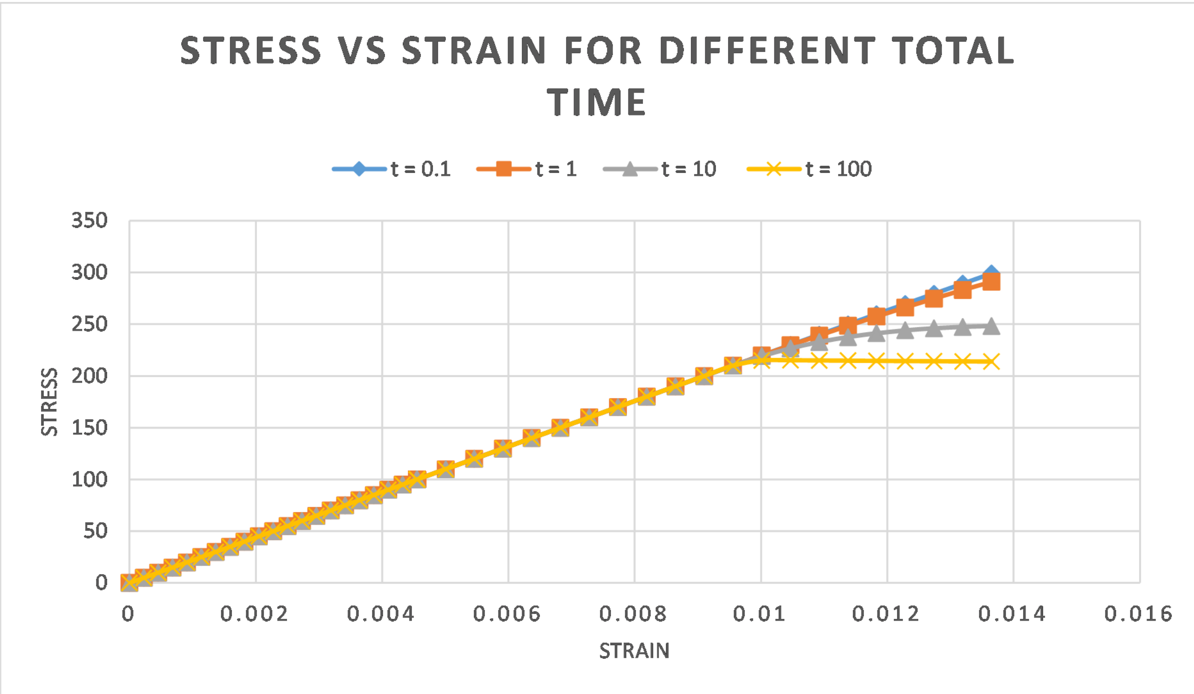


Figure 16: Strain rate variation

As the rheological model states that $F = \eta\dot{\delta}$ the force is proportional to velocity and viscosity. The viscosity is a constant for all the cases. We can observe that stress is proportional to the strain rate.

2.3 Effect of Variation of α

In this subsection we vary the parameter α involved in the integration algorithm and analyze the effects on the stress strain curve and the evolution of the fi

rst component of the $\mathbb{C}_{alg}$ and $\mathbb{C}_{tang}$ constitutive tensors keeping all the other parameters same.We once again chose a plane strain symmetric tension compression model with similar load paths as before. The additional parameters are as follows $H = 0.1$, Time Interval $= 100$ $\nu = 0.3$ and the viscosity parameter $\eta = 1$. The integration coefficient was $\alpha = 0, 0.25, 0.5, 0.75, 1$.

A large value of time is chosen to show the effects of the parameter on the integration algorithm. We can see that for $\alpha < 0.5$ we get oscillatory solutions since the stability is conditional in that interval. Unconditional stability is obtained when $\alpha \in [0.5, 1]$. The valus of $\alpha = 0$ corresponds to forward euler which is a first order explcit method, $\alpha = 1$ corresponds to backward euler, which is first order implicit method and $\alpha = 0.5$ is Crank-Nicolson method which is second order in nature.

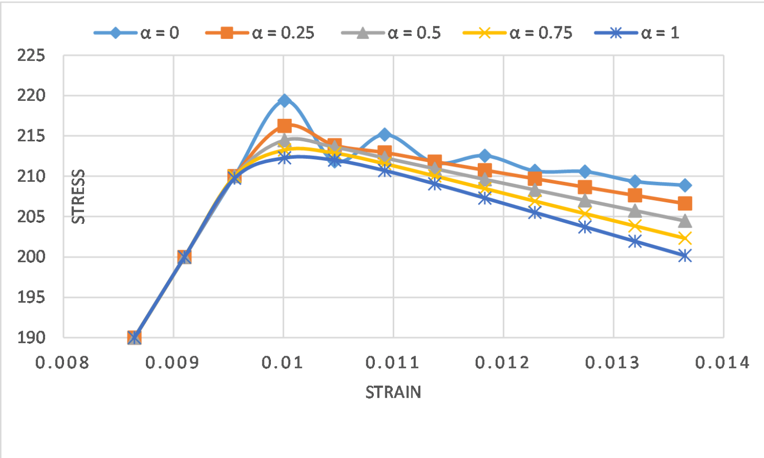


Figure 17: Stress vs Strain for different Integration Coefficient

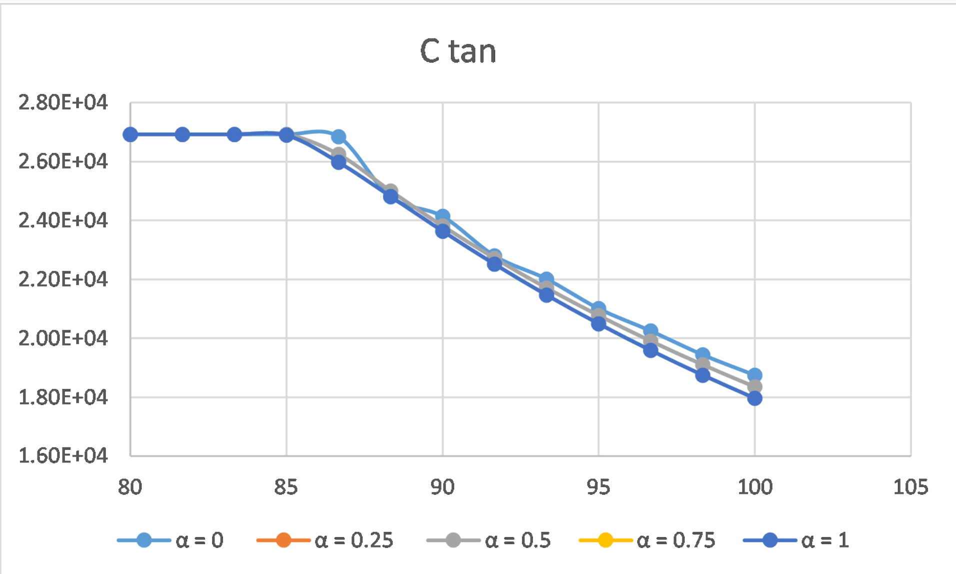


Figure 18: Variation of $\mathbb{C}^{11}_{tang}$

As stress-strain curves tend to be less inclined as time goes by, the value of the algorithmic constitutive operator will decrease, the lower values of α make the process more similar to the inviscid case.

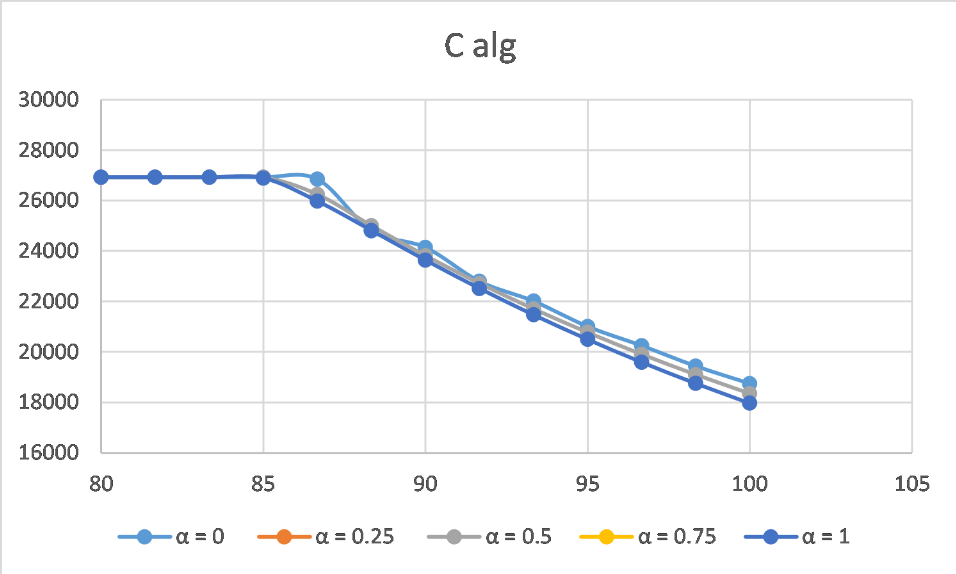


Figure 19: Variation of $\mathbb{C}_{alg}^{11}$

3 Appendix

A MATLAB code for symbolic matrix multiplication written for this assignment.

Matlab function *dibujarcriteriodano*1

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

%
    *************************************************************************
% DRAWING OF THE DAMAGE SURFACE OF THE MATERIAL
%
    *************************************************************************
%* Definition of the polar coordinates variables
     tetha=[0:0.01:2*pi];
     D=size(tetha);
     m1=cos(tetha);
     m2=sin(tetha);
     Contador=D(1,2);
     radio = zeros(1,Contador);
```

```
    s1 = zeros(1,Contador); %Principal stress in x axis
    s2 = zeros(1,Contador); %Principal stress in y axis

%* Evaluation of the elastic domain in terms of the MDtype variable
if MDtype==1     % Symmetric(tension-compression) model case
    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     % Tensile-damage model case
    for i=1:Contador
        radio(i)= q/sqrt([mcauley(m1(i)) mcauley(m2(i)) 0 mcauley(nu*(m1(i)+m2(i)))]*ce_inv*[m1(i) m2(i) 0 nu*(m1(i)+m2(i))]');
        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);
    end

    hplot =plot(s1,s2,tipo_linea);

elseif MDtype==3     % Non-symmetric tension-compression model case
    for i=1:Contador
        % Definition of new variable TETHA
        TETHA = (mcauley(m1(i))+mcauley(m2(i))+mcauley(nu*(m1(i)+m2(i))))/(abs(m1(i))+abs(m2(i))+abs(nu*(m1(i)+m2(i))));
        % Definition of new variable COEFF
        Q = TETHA + (1 - TETHA)/n;
        radio(i) = q/(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);

else
    error('WRONG INPUT (MDtype) FOR DAMAGE MODEL SELECTION')
end

```

```
return
```

Matlab function *Modelosdedano*1

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



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

    r_n = sqrt(eps_n*ce*eps_n');
    r_n1 = sqrt(eps_n1*ce*eps_n1');
    rtrial = (1 - ALPHA)*r_n + ALPHA*r_n1;

elseif (MDtype==2)  %* Only tension

    sigmaB = ce * eps_n' ;
```

```
        sigmaB_n1 = ce * eps_n1' ;
        for i = 1 : length(sigmaB)
            for j = 1 : length(sigmaB_n1)
                sigmaB(i) = mcauley(sigmaB(i));
                sigmaB_n1(j) = mcauley(sigmaB_n1(j));
            end
        end
        r_n = sqrt(eps_n * sigmaB);
        r_n1 = sqrt(eps_n1 * sigmaB_n1);
        rtrial = (1 - ALPHA)*r_n + ALPHA*r_n1;

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

        sigmaB = ce * eps_n' ;
        TETHA_n = (mcauley(sigmaB(1))+mcauley(sigmaB(2))+mcauley(sigmaB(3))+mcauley(sigmaB(4)))/(abs(sigmaB(1))+abs(sigmaB(2))+abs(sigmaB(3))+abs(sigmaB(4)));
        Q_n = TETHA_n + (1 - TETHA_n)/n;
        r_n = Q_n * sqrt(eps_n * ce * eps_n');
        sigmaB_n1 = ce * eps_n1' ;
        TETHA_n1 = (mcauley(sigmaB_n1(1))+mcauley(sigmaB_n1(2))+mcauley(sigmaB_n1(3))+mcauley(sigmaB_n1(4)))/(abs(sigmaB_n1(1))+abs(sigmaB_n1(2))+abs(sigmaB_n1(3))+abs(sigmaB_n1(4)));
        Q_n1 = TETHA_n1 + (1 - TETHA_n1)/n;
        r_n1 = Q_n1 * sqrt(eps_n1*ce*eps_n1');
        rtrial = (1 - ALPHA)*r_n + ALPHA*r_n1;

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

Matlab function rmap dano1

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



hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
H_n     = hvar_n(7);
E       = Eprop(1);
nu      = Eprop(2);
% H       = Eprop(3); Moved to historic variables vector
sigma_u = Eprop(4);
hard_type = Eprop(5) ;
viscpr = Eprop(6) ;
% Definition of the viscous parameter eta and the integration parameter
% ALPHA, which will be in use in the rest of the subroutine, even for the
% inviscid model. If eta=0 and ALPHA=1, the inviscid model is recovered. It
% is done so in order to use only one code able to cover the two options
if viscpr == 1;
    eta = Eprop (7);
    ALPHA = Eprop (8);
else
    eta = 0;
    ALPHA = 1;
end

% *****************************************************************************
%*      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_n,eps_n1,n,ALPHA);
% ******************************************************************


% ******************************************************************
%*                                                          %*
%*    -------------->     fload=0 : elastic unload
                                           %*
%*    -------------->     fload=1 : damage (compute algorithmic
    constitutive tensor)          %*
fload=0;

if(rtrial > r_n)
    %*   Loading
    fload=1;
    delta_r = rtrial - r_n;
    r_n1 = ((eta - var_t*(1-ALPHA))/(eta + ALPHA*var_t))*r_n + (
        var_t/(eta + ALPHA*var_t))*rtrial;
    if hard_type == 0
        %  Linear
        q_n1 = q_n + H_n*delta_r;
        q_r_n1 = q_n + H_n*(r_n1 - r_n); % For the following
            computation of algorithmic constitutive operator
        H_n1 = H_n;
    else
        %  Exponential
        A = 10; %Positive value that defines the shape of the
            curve
        q_inf = q_n + H_n*delta_r;
        q_n1 = q_inf-(q_inf-q_n)*exp(A*(-delta_r/q_n));
        q_r_n1 = q_inf-(q_inf-q_n)*exp(A*(1-r_n1/q_n)); % For the
             following computation of algorithmic constitutive
            operator
        H_n1 = A*((q_inf - q_n)/q_n)*exp(A*(-delta_r/q_n));
    end

```

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


else
     %*       Elastic load/unload
     fload=0;
     r_n1= r_n   ;
     q_n1= q_n   ;
     H_n1= H_n   ;

end
% Computing damage variable
% ****************
dano_n1    = 1.d0-(q_n1/r_n1);

%  Computing stress
%  ***************
sigma_n1   =(1.d0-dano_n1)*ce*eps_n1';
hold on
plot(sigma_n1(1),sigma_n1(2),'bx')



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


% ************************************************************************
%* Updating historic variables                                          %*
hvar_n1(1:4)   = eps_n1;
hvar_n1(5)= r_n1  ;
hvar_n1(6)= q_n1  ;
hvar_n1(7)= H_n1  ;
% ************************************************************************




```

```
%
    ***********************************************************************

%* Auxiliar variables

    %*
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;

fload = 1
% Computing tangent and algorithmic constitutive operators
% ****************
if fload == 0 %Elastic loading / unloading
     ce_tan = (1 - dano_n1)*ce;
     ce_alg = ce_tan;
else %Pure loading
     ce_tan = (1 - dano_n1)*ce;
     ce_alg = ce_tan + ((ALPHA*var_t*(H_n1*r_n1-q_n1))/((eta+ALPHA
        *var_t)*r_n1^3))*(eps_n1*eps_n1');
end
% Storing C11 component of the tangent and algorithmic
    constitutive operators
aux_var(3) = ce_tan(1,1);
aux_var(4) = ce_alg(1,1);
%
    ***********************************************************************
```