Computational Solid Mechanics

Assignment 1: "Damage constitutive models"

Student: Marcello Rubino

1. Introduction

The following report shows the implementation of MatLab algorithm for the inviscid and viscous damage constitutive model for Solid Mechanics. The main aim of this report is to show the different models, changing parameters and making different loading paths, and discussing the results obtained. All the models implemented are using the plane-strain hypothesis. The models shown in this topic use a Young modulus of 2000, a Yield stress of 200 and a Poisson ratio equal to 0.3.

2. Inviscid Model (rate-independent)

This model shows the inviscid behavior of a damage constitutive model. In the first part we discuss three different damage surfaces that have been implemented in the code, the second part pays attention to the hardening/softening laws while the third part show the results for three different loading paths.

2.1 - Damage surface models

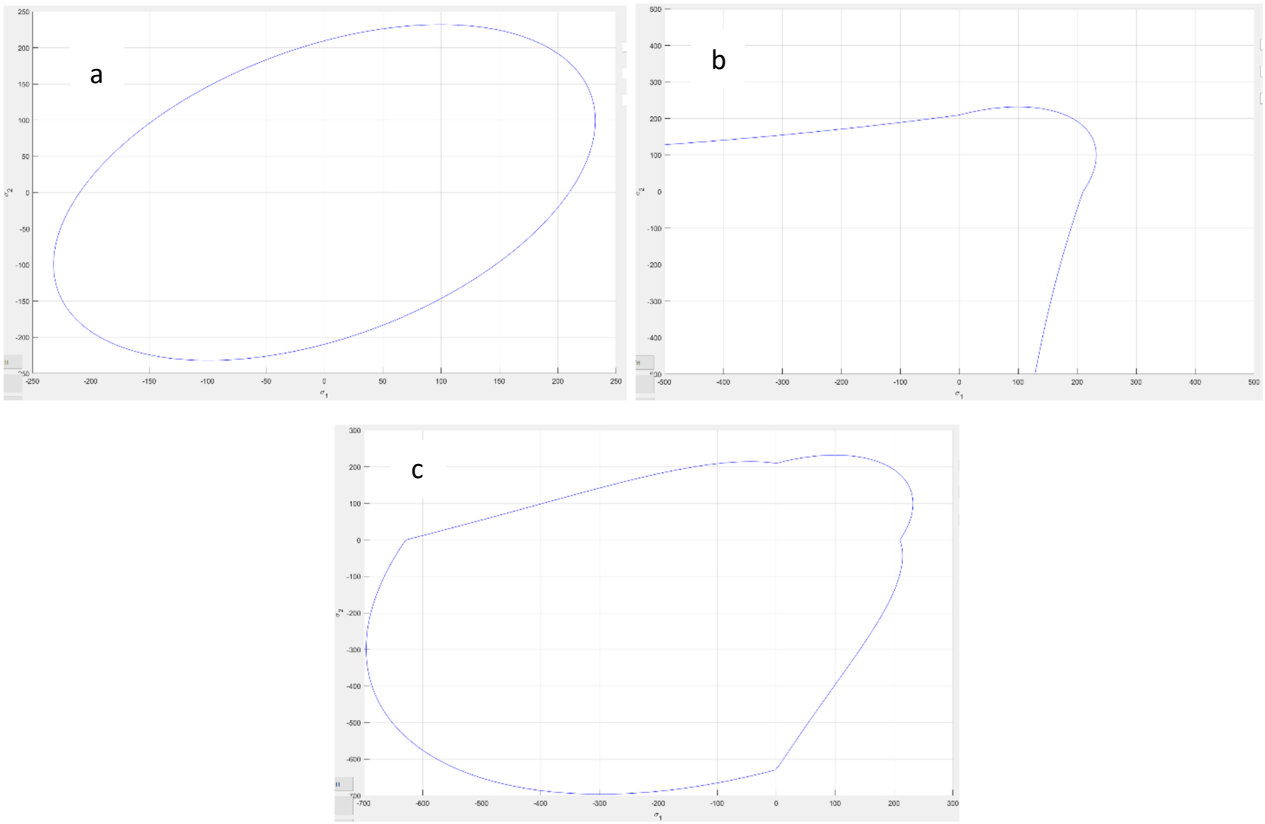

Figure 1 (a-b-c) – Damage surfaces (symmetric – only tension – non symmetric)

In the three figures above we can see the three different damage surfaces implemented. We can see that for the symmetric part we have same contribution of the elastic domain for tension and compression: the results is like an ellipse.
The only-tension model instead shows a completely different behavior: the tension part coincides with the symmetric case, while the compression area doesn't have a boundary. For the second and the fourth quadrants we have an asymptotic behavior of the surface.

The non-symmetric model (n = 3), shows the same tension part of the symmetric model, while, respect to the only-tension problem, the compression area has a "hat" that bounds the domain: the compression area is exactly n times bigger the traction (this is typical case for material like concrete, for which usually the n parameter is equal to 10).

2.2 - Hardening/Softening laws – plot of r-q(r)

In this part it is shown the representation of the hardening/softening laws implemented in the code. A very simple uniaxial loading path has been computed ($\Delta\sigma_1^{(1)}$ = +100 ; $\Delta\sigma_1^{(2)}$ = +100 ; $\Delta\sigma_1^{(3)}$ = +100). In particular, it's represented the different behavior in the graph of the internal variable r and the hardening variable $q(r)$ between the linear and exponential law respectively for hardening (H = 2) and softening case (H = -2).

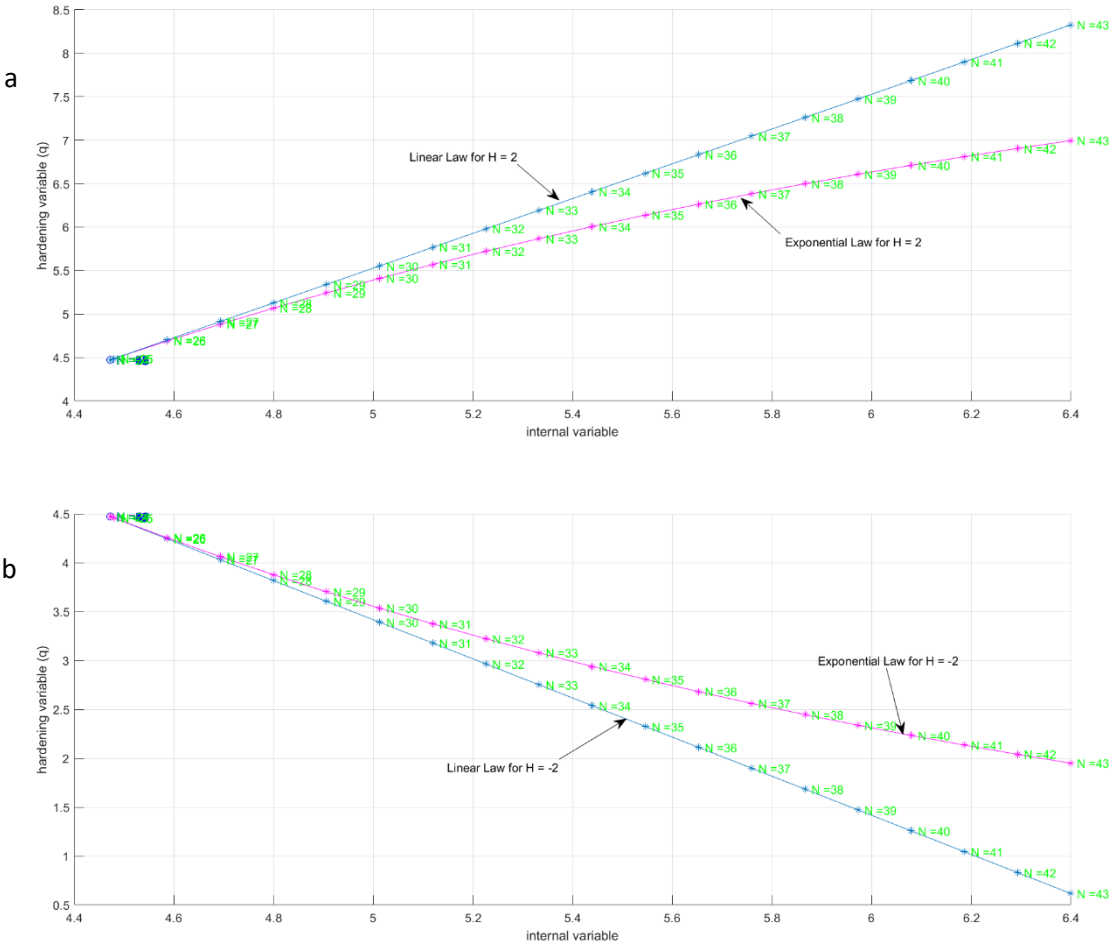

Figure 2 (a-b) – Hardening – Softening laws in the r-q(r) plot

In the two figures above it's possible to see that the exponential law, for both cases H>0 and H<0, is much slower than the linear law. In fact the H modulus is the initial tangent modulus of hard/soft for the exponential case. Lastly, it's possible to see one important behavior of the model, supported by theoretical framework: the linear or exponential development of the hardening variable starts only when the initial damage surface is reached by the loading path. Before this important state (the state when the damage starts to develop, and the damage variable d starts to be greater than zero) the r is equal to the initial value r_0 and the q=q_0.

2.3 - Results of the inviscid model (rate-independent) for different loading paths

In this part three different loading paths used for the three damage surfaces have been computed. The results represented in the stress-strain graph and in the damage model graph (σ_1-σ_2) show different behaviors depending on the loading path used and the damage surface that has been considered. All of them use an exponential law for the hardening/softening model and the same data already described and a H modulus equal to -1 (softening).

2.3.1 – Loading path 1 ($\Delta\sigma_1^{(1)}$ = +250 / $\Delta\sigma_2^{(1)}$ = 0 ; $\Delta\sigma_1^{(2)}$ = -600 / $\Delta\sigma_2^{(2)}$ = 0 / $\Delta\sigma_1^{(3)}$ = +350 ; $\Delta\sigma_2^{(3)}$ = 0)

A] Symmetric damage surface

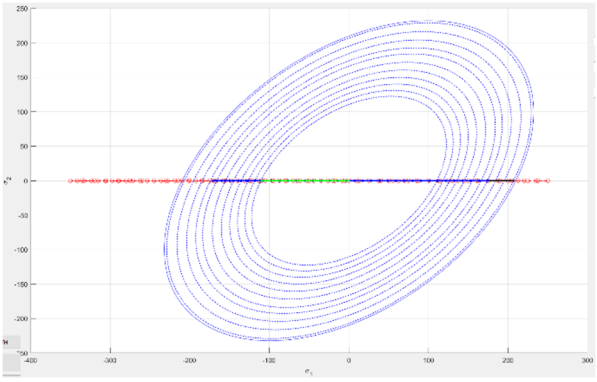

Figure 3a – Damage surface for the loading path 1 - Symmetric

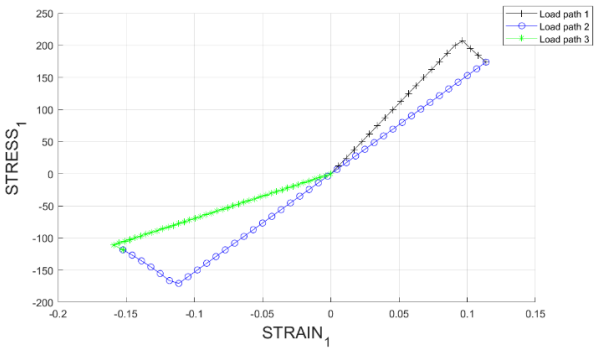

Figure 3b - Stress-Strain for the loading path 1 – Symmetric

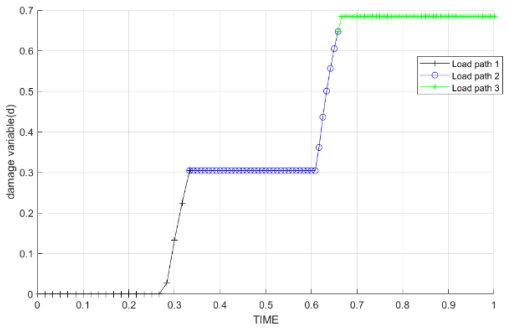

Figure 3c – Time-Damage Variable for the loading path 1 - Symmetric

This loading path represents a uniaxial tensile loading, followed by a uniaxial compression loading, and a uniaxial unloading. It's possible to see that the initial linear elastic behavior gets to the damage when it's reached the yielding stress (figure 3b): in this moment the damage variable starts to increase (figure 3c) and the damage surface start to shrink due to the softening property (figure 3a). Reached a certain level of strain (linked to the value of 250 of the loading path) there is an inversion of the loading and starts the unloading/compression part: this part is, for the most of the time linear elastic, but shows a decreased stiffness due to the damage already reached. Since the damage surface is now smaller, the new damaging in the compression part arrives "earlier" than before, but shows the same behavior. Again the damage increases another time and the damage surface becomes smaller. The last part of the path, the green one, is a unloading in the elastic region, but shows a stiffness even smaller: the sample described by the model now is really damaged.

B] Only-tension damage surface

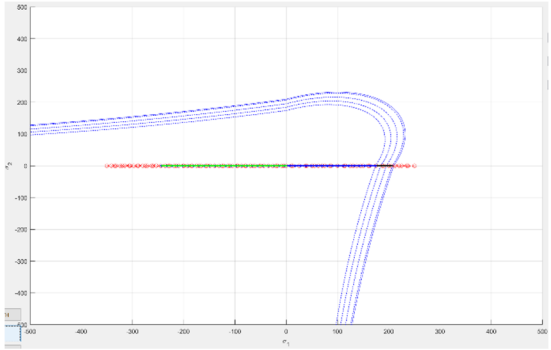

Figure 4a – Damage surface for the loading path 1 - Only-tension

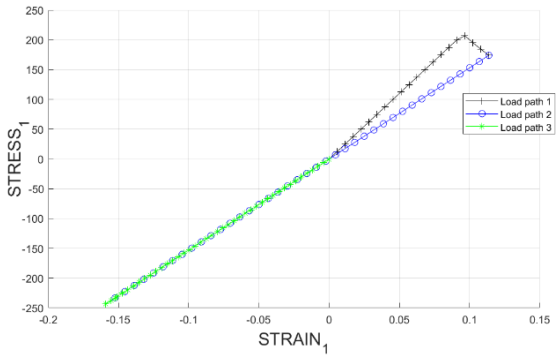

Figure 4b - Stress-Strain for the loading path 1 – Only-tension

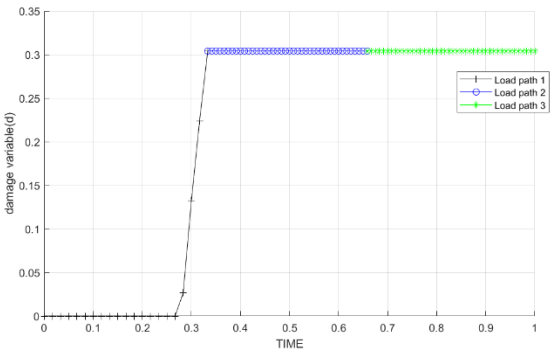

Figure 4c – Time-Damage Variable for the loading path 1 – Only-tension

This case shows for the first part the same behavior of the symmetric case (first damage, increase of *d* value until 0.3, decrease of the stiffness and the elastic domain shrinking). Since the domain in the compression area is infinite (figure 4a), the second load path (the blue one) doesn't reach again the damage surface (figure 4a), so the *d* remains constant (figure 4c) and the slope of the stress-strain plotting too (figure 4b).

C] Non-symmetric damage surface

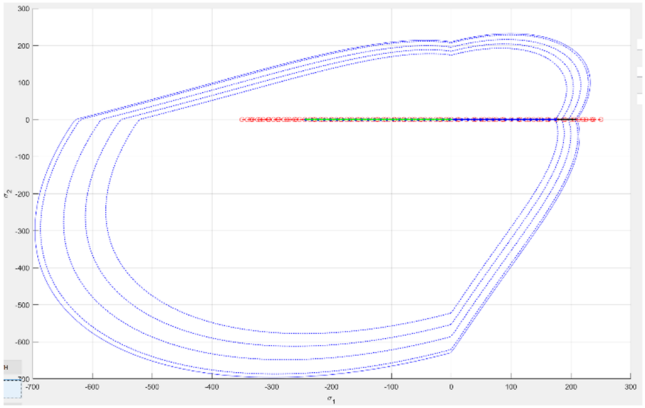

Figure 5a – Damage surface for the loading path 1 – Non-Symmetric

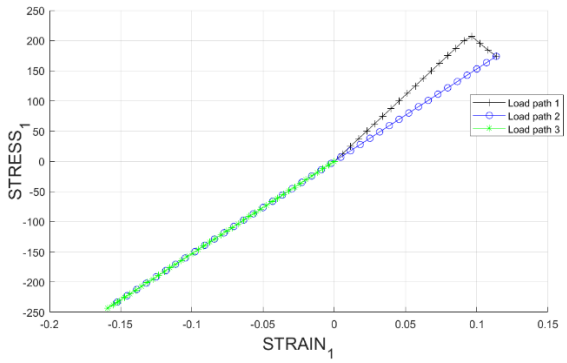

Figure 5b - Stress-Strain for the loading path 1 – Non-Symmetric

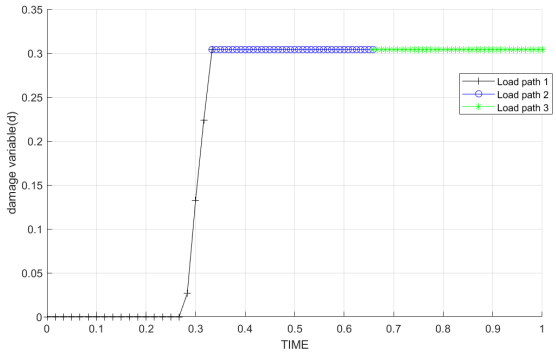

Figure 5c – Time-Damage Variable for the loading path 1 – Non-Symmetric

Since the damage surface for the non-symmetric case, is enough big (*n* = 3), the behavior of the model is the same of the only-tension model.

2.3.2 – Loading path 2 ($\Delta\sigma_1^{(1)}$ = +250 / $\Delta\sigma_2^{(1)}$ = 0 ; $\Delta\sigma_1^{(2)}$ = -250 / $\Delta\sigma_2^{(2)}$ = -250 / $\Delta\sigma_1^{(3)}$ = +100 ; $\Delta\sigma_2^{(3)}$ = +100)

This loading path has been computed to all three damage models, using the same data used for the first loading path. Since it's not totally a uniaxial path, the stress-strain graph used is the same of the first model (direction 1) but shows some particularities.

A] Symmetric damage surface

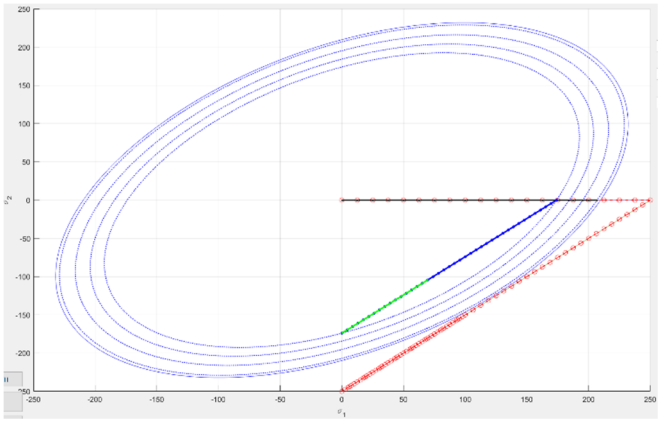

Figure 6a – Damage surface for the loading path 2 – Symmetric

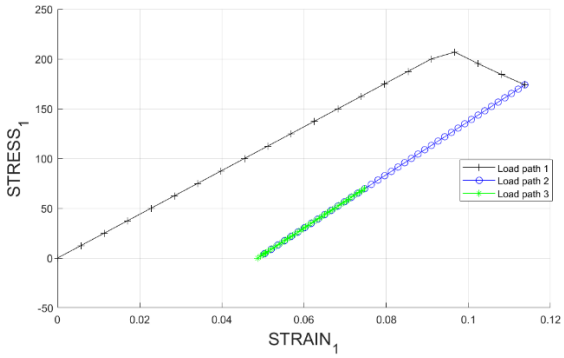

Figure 6b - Stress-Strain for the loading path 2 – Symmetric

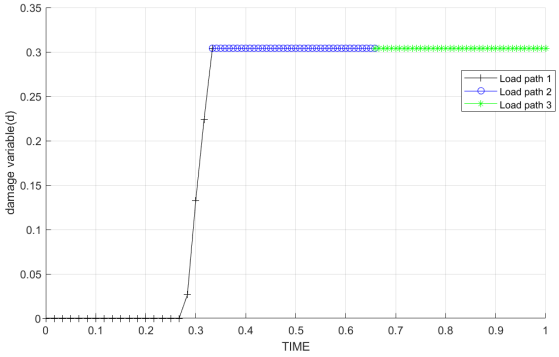

Figure 6c – Time-Damage Variable for the loading path 2 - Symmetric

It's possible to see that the first uniaxial loading shows the same behavior in the stress-strain and time-d graphs: indeed, when the damage surface is reached, the elastic domain starts to shrink because of the softening phenomenon (figure 6a). Since the second and the third paths are biaxial, now the loading path moves inside the elastic domain (once reached the damage surface, the strain-stress state must be inside or on it). Because of that there is no increase of the damage (figure 6c), and the process of unloading/loading is now completely elastic linear. In the stress-strain graph (figure 6a9 we can see that the slope of the blue and green line is higher than the black one (initial uniaxial loading): this is because the path doesn't reach the 0 again like the other path did, and the decreased constitutive tensor along the direction 1, is affected by the nature of the loading which is biaxial, and involves the Poisson coefficient.

B] Only-tension damage surface

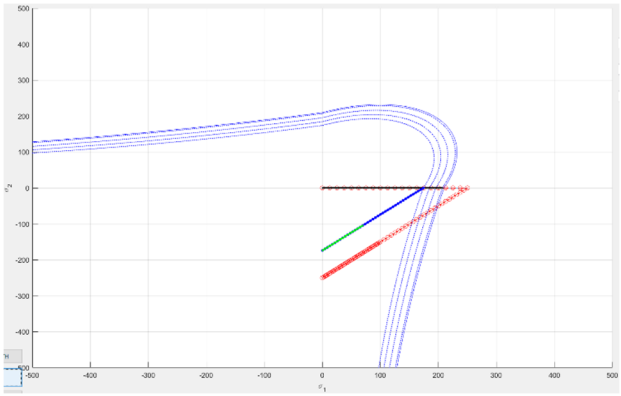

Figure 7a – Damage surface for the loading path 2 - Only-tension

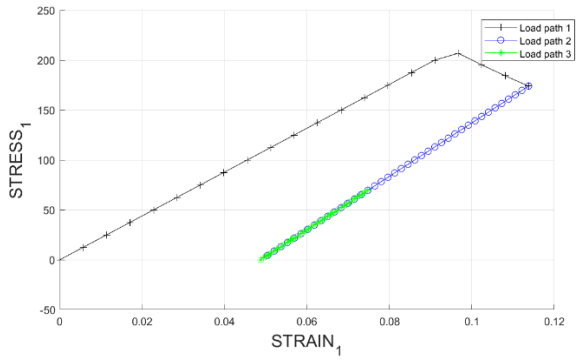

Figure 7b - Stress-Strain for the loading path 2 – Only-tension

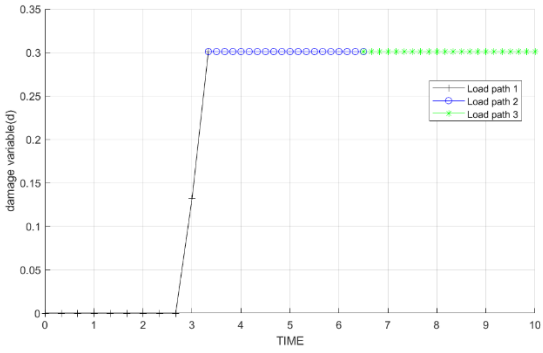

Figure 7c – Time-Damage Variable for the loading path 2 – Only-tension

For this model we can see the same results, since the load path, that involved in the second and third part the elastic domain, has the same behavior here, since here the compressive domain is infinite (figure 7a).

C] Non-symmetric damage surface

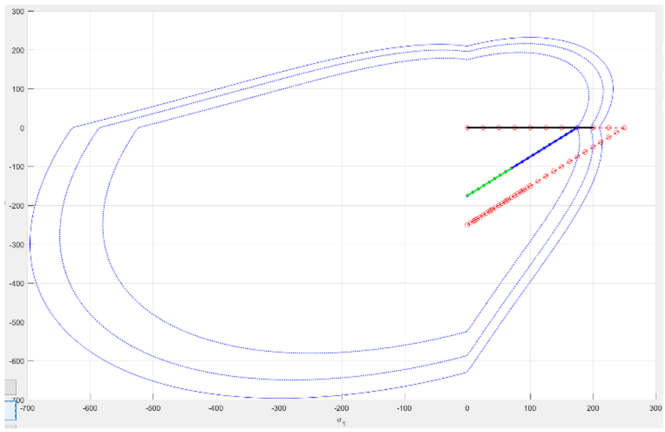

Figure 8a – Damage surface for the loading path 2 – Non-Symmetric

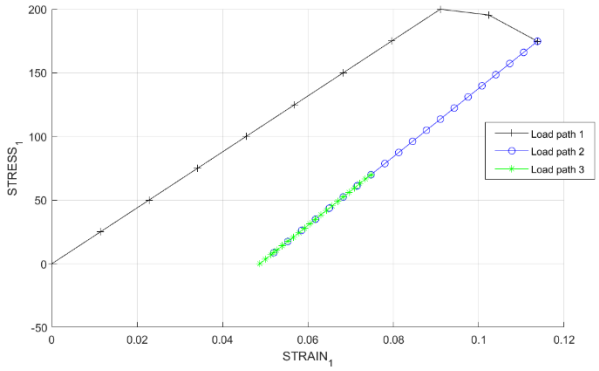

Figure 8b - Stress-Strain for the loading path 2 – Non-Symmetric

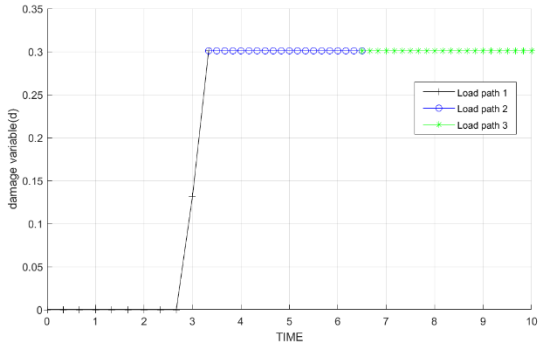

Figure 8c – Time-Damage Variable for the loading path 2 – Non-Symmetric

The non-symmetric damage surface has the same behavior of the other two surfaces in this case. Since the second and third part of the loading path is biaxial and involves the elastic domain in the symmetric model, here, even more so, the loading path is within the elastic domain (figure 8a) and the damage variable doesn't grow (figure 8c).

2.3.3 – Loading path 3 ($\Delta\sigma_1^{(1)}$ = +250 / $\Delta\sigma_2^{(1)}$ = +250 ; $\Delta\sigma_1^{(2)}$ = -600 / $\Delta\sigma_2^{(2)}$ = -600 / $\Delta\sigma_1^{(3)}$ = +350 ; $\Delta\sigma_2^{(3)}$ = +350)

This path is the biaxial version of the first loading path. In this case, since it involves always both directions (1 and 2), the stress-strain graph shows the norm of the strain and the norm of the stress, to have an idea

of the combination of the two directions and how the system develops. It's not possible to see in the right position the compression and the tension part in this graph since it's all in the first quadrant and the norms are always positive by definition.

A] Symmetric damage surface

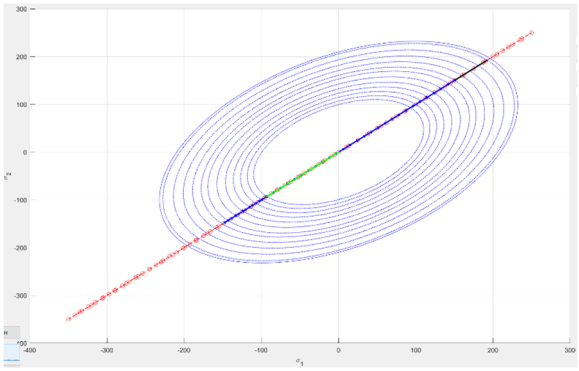

Figure 9a – Damage surface for the loading path 3 – Symmetric

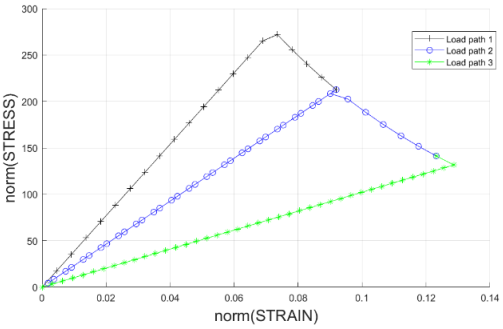

Figure 9b - Stress-Strain for the loading path 3 – Symmetric

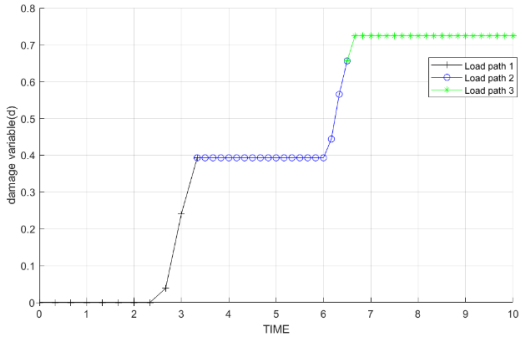

Figure 9c – Time-Damage Variable for the loading path 3 – Symmetric

The symmetric damage surface for this problem behaves in the same way of the symmetric in the case of the first load path. The first part of the path shows firstly a linear elastic behavior with the initial constitutive tensor (figure 9b). After reaching for the first time the damage surface, the elastic domain shrinks (figure 9a) and the damage increases (figure 9c); a process of biaxial unloading/compression shows an elastic behavior that passes by the origin: the slope of the curve stress-strain is smaller (blue line) since the system now show an important value of the damage (almost $d = 0.4$). When reaching again the damage surface, now the same surface is much smaller and the new increase of damage happens earlier. The last biaxial unload (green line) is still elastic and shows a very low slope, which represents a very bad condition of the material.

B] Only-tension damage surface

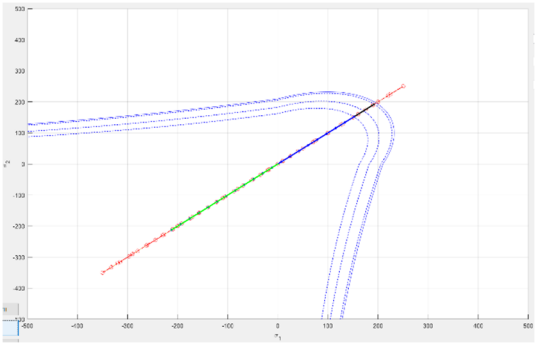

Figure 10a – Damage surface for the loading path 3 - Only-tension

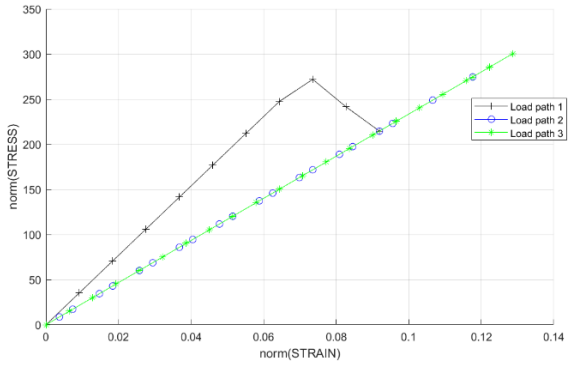

Figure 10b - Stress-Strain for the loading path 3 – Only-tension

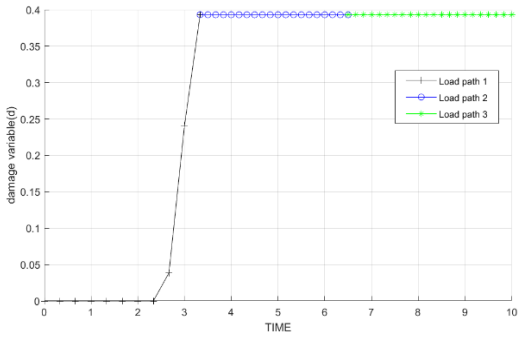

Figure 10c – Time-Damage Variable for the loading path 3 – Only-tension

In this case it's possible to see that the first part of the path reaches the damage surface (initial tension) (figure 10b). The elastic domain shrinks again (figure 10a) and the damage variable increases (figure 10c). The rest of the path stays always inside the elastic domain: in particular, we know that the compression in this case never gets to a damage, so the second part of the path is completely elastic. The third part is elastic too since it's only an unloading process and doesn't tend again to the tension part. The material keeps its state of damage (d = 0.4). In the norm(stress)-norm(strain) graph this behavior can be seen very well. The slope of the blue and the green lines remains the same, linked to the same value of the damage variable.

C] Non-symmetric damage surface

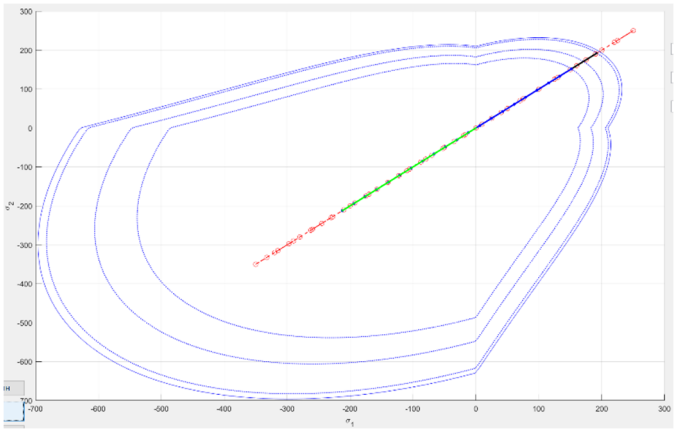

Figure 11a – Damage surface for the loading path 3 – Non-Symmetric

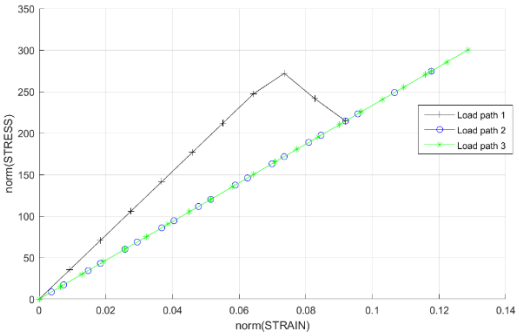

Figure 11b - Stress-Strain for the loading path 3 – Non-Symmetric

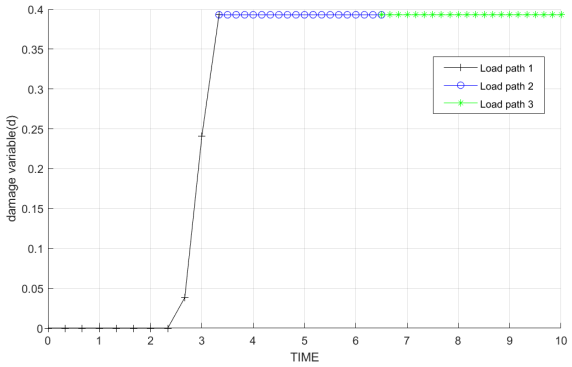

Figure 11c – Time-Damage Variable for the loading path 3 – Non-Symmetric

The same observations can be done for this model: the tension part reaches the damage surface and makes the damage develop. The second and the third part of the path remain inside the elastic domain since the "hat" for the compression is quite far (figure 11a). Although the softening behavior makes the damage surface gets smaller the damage surface is not reached again and there can't be any further develop for the damage (figure 11c).

2.4 – Conclusions for the implemented inviscid model

All the models implemented show the results as expected and follow the theoretical framework of the inviscid damage model. In particular, it's important to underline the utility of this inviscid model, since it can describe the properties of many useful materials. This model, though, leaks of the time dependence, which is very useful in many situation and can describe the natural late response of materials.

3. Viscous Model (rate-dependent)

This model shows the viscous behavior of a damage constitutive model. In the first part we see the different behaviors changing the viscous coefficient η, the second part studies the behavior for different strain rate $\dot{\varepsilon}$ while the third part pays attention to the response for different values of the integration constant α. In this case it's been computed a very simple uniaxial loading path ($\Delta\sigma_1^{(1)}$ = +100 ; $\Delta\sigma_1^{(2)}$ = +100 ; $\Delta\sigma_1^{(3)}$ = +300) only for the symmetric version of the damage surface.

3.1 – Different Viscosity coefficient η

The viscous model is influenced by the viscosity value, that describes, fixing the other parameters, the attitude of the material to be more (or less) fast to respond to the external solicitation. The fixed parameters are the Young Modulus (E = 2000), the Yielding stress (σ_y = 200), the Poisson ratio (ν = 0.3), the Hard/Soft modulus (H = 0), the Time Interval equal to 1 second and the coefficient α equal to 1 (forward-Eulerian integration). The different values used for plotting the results are η = 0,0.1,1,10. As such, the most interesting plots are the 'stress-strain' and the 'time-damage value' graphs.

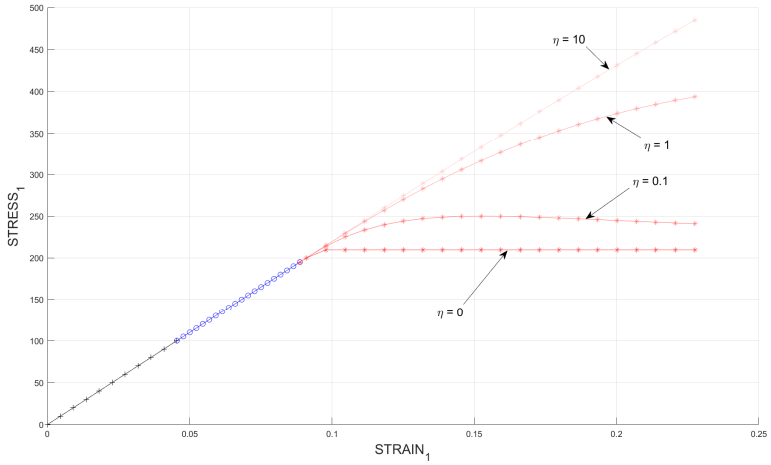

Figure 12 – Stress-Strain graph for different values of η

The stress-strain graph (figure 12) shows the different response of the same material that has different values for the viscosity. As expected and supported by the theory, bigger is the value of the coefficient η higher is the slope of the last part of the loading path. The path in this case gets out from the damage surface because the viscous response allows it. For very high levels of the viscosity (η = 10) the curve tends to remain linear, because it's much slower the response to the loading solicitation and the system dissipates a big quantity of energy. While the viscosity decreases, the response is faster, and the slope of the curve is lower. The opposite extreme case (η = 0) shows a rate-independent case, already discussed, for a hardening modulus equal to 0 (the curve doesn't show any slope). In this case the material response instantaneously to the external loading, and cannot resist to this increasing loading.

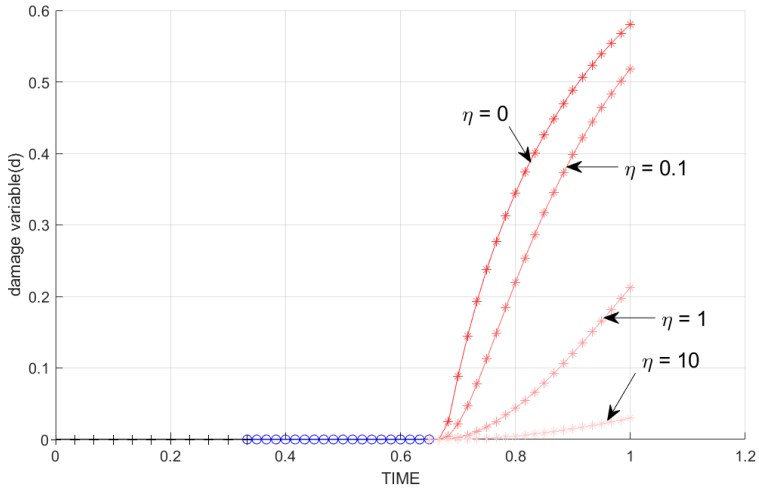

Figure 13 – Time-Damage Variable graph for different values of η

In the Time-Damage Variable graph (figure 13) we can see the develop of the damage from a more direct point of view. In fact the $\eta = 0$ shows the highest increase of damage, since it's equivalent to a rate-independent case. Increasing the value of η, the slope decreases very fast, until, for $\eta = 10$, it's almost horizontal, and the develop of the damage is really slow or almost doesn't show up.
This important result tests the validation of the implemented viscous model, because it's conforming to the theory and the expected curves for the given values of the parameters.

3.2 – Different strain rate $\dot{\varepsilon}$

The rate-dependent viscous model is influenced also by different values of the strain rate. In order to show this influence, a set of different values of time intervals are computed in the model, in order to see different values for the strain rate $\dot{\varepsilon}$. In this case the viscosity is fixed ($\eta = 1$) and the integration constant α is equal to 0.5, while the other parameters remains the same. The time intervals are: 0.1,1,10,100.

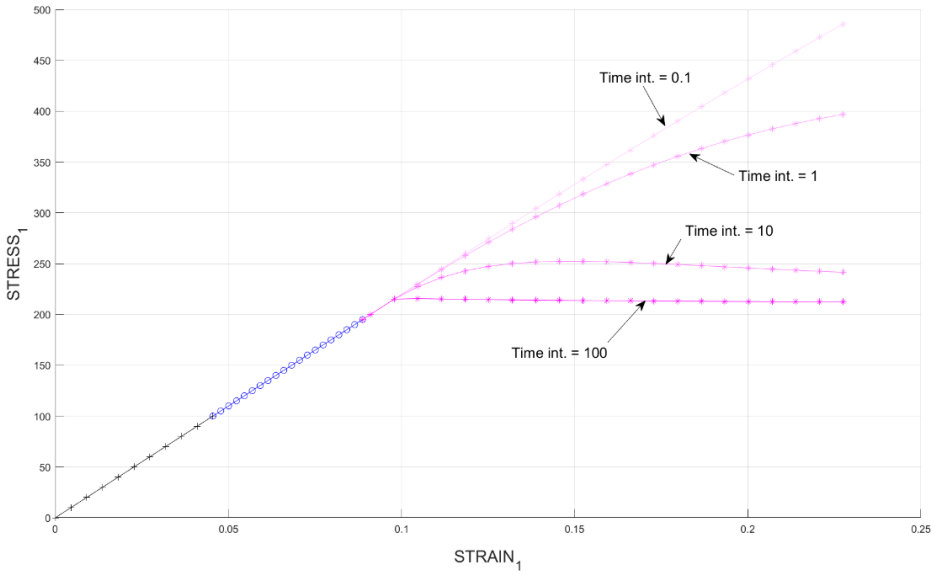

Figure 14 – Stress-Strain graph for different values of $\dot{\varepsilon}$

In this plot (figure 14) we can see a similar situation to the stress-strain graph for different values of the viscosity. For a time interval very small (which coincides to a high value of $\dot{\varepsilon}$), the increase of damage is very small and slow, and the curves remains almost linear. Increasing the time interval (so decreasing the value of $\dot{\varepsilon}$) we arrive at the opposite

extreme situation, where the response of the material to the external solicitation is almost instantaneous, and the rate-dependent models tends to the rate-independent inviscid model. In this case too, for very small number of $\dot{\varepsilon}$, the material cannot resist to the external loading, cannot dissipate energy and shows big increase of damage.

3.3 – Different integration constant α

In this last part of the analysis the same parameters have been used except for the hardening/softening modulus that is now equal to -0.1 (softening case). The time interval is fixed to 100 seconds. In the first part the influence of different values of α for the stress-strain graph will be studied, secondly the influence of the same parameter to the constitutive matrices. The set of values used for α are: 0,0.25,0.5,0.75,1.

A] Influence for the Stress-Strain

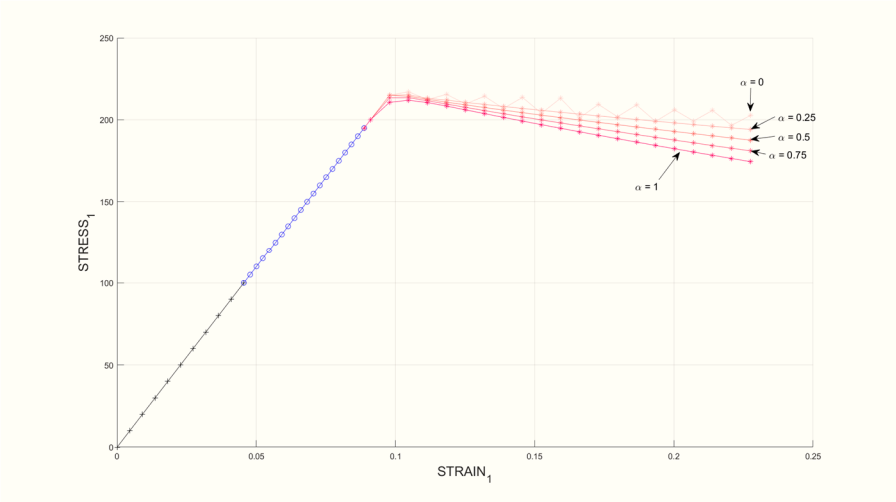

Figure 15 – Stress-Strain graph for different values of α

Changing the value of α, it's possible to see in figure 15 that for small values like 0 (backward-Eulerian integration method) or 0.25 the time integration method is unstable. For higher values of α, instead, the integration is more accurate and stable. In particular, since this is α second order problem, it's preferred to use the value of 0.5 which is referred to a second order accuracy.

B] Influence for the Constitutive matrices

In this part it's shown the influence of α over the 1,1 component of the constitutive matrices (C_{alg} and C_{tan}), plotted against time.

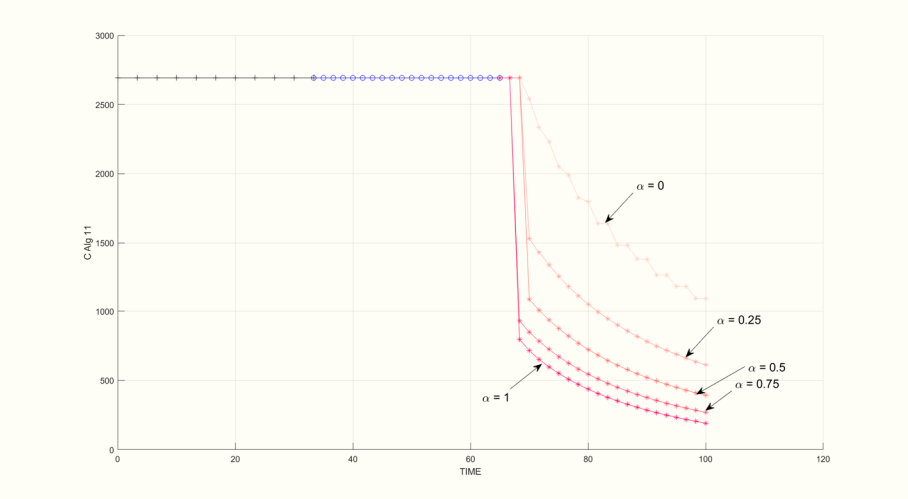

Figure 16 – C Algorithmic 11-Time graph for different values of α

In the figure 16 it's possible to see that the C_{alg} for a value of α equal to 0 is instable while for other values of a remains stable. An important result here is shown: the value of C_{alg} is smaller for values of a bigger, but this doesn't mean it's more accurate. The value of 0.5 that guarantee a second order accuracy is the best choice for this viscous rate-dependent problem.

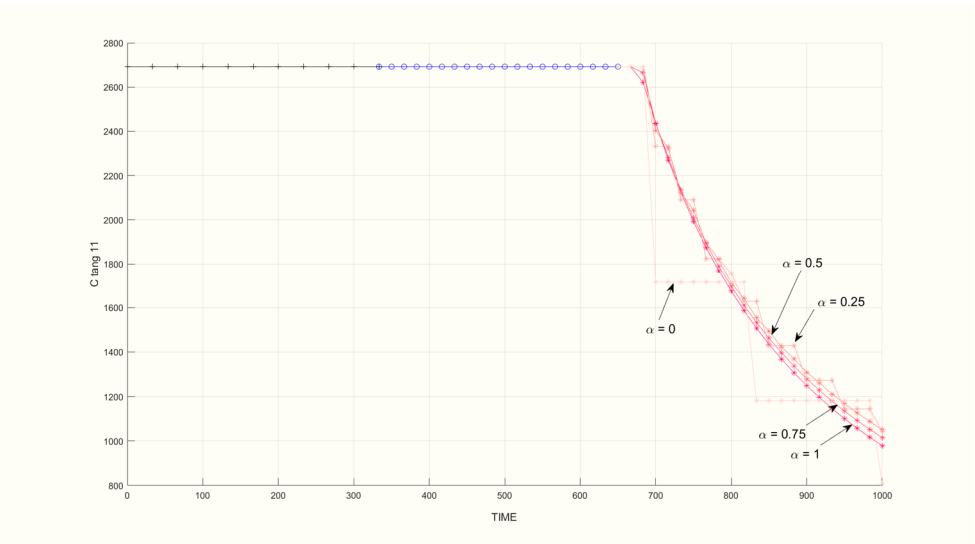

Figure 17 – C Tangent 11-Time graph for different values of α

In the figure 17, the time interval has been increased to 1000 seconds to show that both $\alpha = 0$ and $\alpha = 0.25$ are instable, for big time intervals. The tangent value of the constitutive matrix has the same behavior of the algorithmic one. Once reached the damage surface it starts to decrease very fast, as expected and supported by theory frame. In this case the influence of α is very much visible: in fact, the value of 0 and 0.25 show a completely instable trend and the system cannot catch the right value of the matrix at each iteration. For values bigger than 0.5 it's possible to see a stable trend, but that does not say anything about the accuracy of the integration. The theoretical support suggests us to use the value of 0.5. Moreover, another important aspect needs to be underlined: the constitutive tangent and algorithmic matrices, as already remarked in the theory frame, are the same until the loading path reaches the damage surface. Once the damage starts to develop $C_{alg} \neq C_{tan}$.
Only in case of $\alpha = 0$ (backward-Eulerian integration method) the two matrices are the same also after the damage, and both show an instable trend (in figure 18 and 19 it's shown this aspect, using again the same time interval (100 seconds) and $\alpha = 0$).

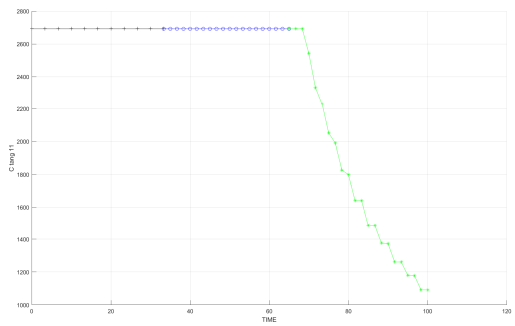

Figure 18 - C 11-Time graph for $\alpha = 0$

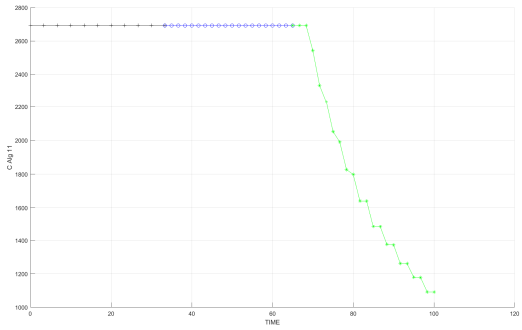

Figure 19 – C Algorithmic 11-Time graph for $\alpha = 0$

3.4 – Conclusions for the implemented viscous model

All the results obtained for the computation of the viscous model coincide with expectation and the theoretical framework. The utility of this mathematical model can vary for many different materials as concrete, steel or ceramics. This MatLab program gives the scientist/engineer the possibility to get some theoretical/modeling results for some materials and compare the results with the empirical results from laboratory tests on samples.

Annex of subroutines

Below there are all the changed subroutines for the implemented MatLab program.

- In Function "Dibujar_criterio_dano1.m" (implementation of static/initial damage surfaces):

A] Only – Tension damage surface:

```
elseif MDtype==2 %ONLY TENSION
    tetha=[-pi/2*(0.9999):0.01:pi*0.9999]; %working only in quadrants III,I,II
    %*************************************************************************************
    %* RADIUS
    m1=cos(tetha);                      %* COSIN(TETHA)
    m2=sin(tetha);                      % SIN(TETHA)
    m3=m1+m2;                           %* SUM OF SIN AND COSIN
    Contador=length(tetha);             %* NUMBER OF DISCRETE TETHA NUMBERS

    % INITIALIZING THE VALUES
    %radio = zeros(1,Contador) ; %RADIUS
    s1    = zeros(1,Contador) ; %SIGMA 1
    s2    = zeros(1,Contador) ; %SIGMA 2
    m1plus = zeros(1,Contador); %m1+
    m2plus = zeros(1,Contador); %m2+
    m3plus = zeros(1,Contador); %m3+
    a = zeros(1,Contador);
    for i = 1:Contador
        %Macaulay brackets on the three parameters
        if m1(i)>0
            m1plus(i) = m1(i);
        else
            m1plus(i) = 0;
        end
        if m2(i)>0
            m2plus(i) = m2(i);
        else
            m2plus(i) = 0;
        end
        if m3(i)>0
            m3plus(i) = m3(i);
        else
            m3plus(i) = 0;
        end
        % calculation of the radius
        radio(i)=q/sqrt([m1plus(i) m2plus(i) 0 nu*m3plus(i)]*ce_inv*[m1(i) m2(i) 0 nu*m3(i)]');
        % projection on the two directions, so to obtain sigma1 and sigma2
        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);
    end
    % plot the surface
    hplot =plot(s1,s2,tipo_linea);
```

B] Non-Symmetric damage surface

```
elseif MDtype==3 %TENSION-COMPRESSION

    tetha=[0:0.01:2*pi];
    %*************************************************************************************
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %* COSIN(TETHA)
    m2=sin(tetha);                          % SIN(TETHA)
    m3=m1+m2;                               %* SUM OF SIN AND COSIN
    Contador=D(1,2);                        %* NUMBER OF DISCRETE TETHA NUMBERS


    % INITIALIZING THE VALUES
    radio = zeros(1,Contador) ; %RADIUS
    s1    = zeros(1,Contador) ; %SIGMA 1
    s2    = zeros(1,Contador) ; %SIGMA 2
    m1plus = zeros(1,Contador); %m1+
    m2plus = zeros(1,Contador); %m2+
    par_theta = zeros(1,Contador); %par_theta
    coeff_tau_q = zeros(1,Contador);%coeff_tau_q

    for i=1:Contador
       %Macaulay brackets on the first two parameters
       if m1(i) > 0
            m1plus(i) = m1(i);
       else
            m1plus(i) = 0;
       end
       if m2(i) > 0
            m2plus(i) = m2(i);
       else
            m2plus(i) = 0;
       end

       % calculation of theta parameter
       par_theta(i) = (m1plus(i)+m2plus(i))/(abs(m1(i))+abs(m2(i)));
       % calculation of the coeffcient for the radius
       coeff_tau_q(i) = par_theta(i)+(1-par_theta(i))/n;
       % calculation of the radius
       radio(i)= q/(coeff_tau_q(i)*sqrt([m1(i) m2(i) 0 nu*m3(i)]*ce_inv*[m1(i) m2(i) 0 ...
            nu*m3(i)]'));

       % projection on the two directions, so to obtain sigma1 and sigma2
       s1(i)=radio(i)*m1(i);
       s2(i)=radio(i)*m2(i);
    end
    % plot the surface
    hplot =plot(s1,s2,tipo_linea);

    % Comment/delete lines below once you have implemented this case
    % *************************************************
end
%*************************************************************************************
```

- In function "Modelos_de_dano1.m": it's been implemented the two more damage surfaces for different models (it uses the n+1 time step variables):

```
elseif (MDtype==2)  %* Only tension
sigma_s = ce*eps_n1'; % vector of stresses at n+1
for i=1:length(sigma_s) %Macaulay brackets on sigma
    if sigma_s(i) > 0
    sigma_s_plus(i)=sigma_s(i);
    else
    sigma_s_plus(i)=0;
    end
end
rtrial= sqrt(sigma_s_plus*eps_n1'); % calculation of the damage surface

elseif (MDtype==3)  %*Non-symmetric
sigma_s = ce*eps_n1'; % vector of stresses at n+1
for i=1:2 %Macaulay brackets on sigma (only sigma1 and sigma2)
    if sigma_s(i) > 0
    sigma_s_plus(i)=sigma_s(i);
    else
    sigma_s_plus(i)=0;
    end
end
theta = (sigma_s_plus(1)+sigma_s_plus(2))/(abs(sigma_s(1))+abs(sigma_s(2))); %calculation of theta
rtrial= (theta + (1-theta)/n)*sqrt(eps_n1*ce*eps_n1');  % calculation of the damage surface
end
%*************************************************************************************
```

- In function "rmap_dano1.m" the inviscid and viscous models of hardening/softening have been implemented. Moreover, the calculation of the C_{11} for the Algorithmic and Tangent tensor has been added:

```
% added more parameters for viscous calculation
eta = Eprop(7);
alpha = Eprop(8);
%*************************************************************************************
```

```
%*************************************************************************************
%*       Damage surface                                                              %*
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n); % step n+1 damage surface
[rtrial_prev] = Modelos_de_dano1 (MDtype,ce,eps_n,n); % previous step (n) damage surface
rtrial_n_alpha = rtrial_prev*(1-alpha)+rtrial*alpha; % damage surface at n+alpha step
%*************************************************************************************


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

```
% here, with an "if" loop I consider both cases: if viscpr == 1 (viscous
% model) and if viscpr == 0 (inviscid model)

if viscpr == 0 %inviscid model
    if(rtrial > r_n)
        %*   Loading

        fload=1;
        delta_r=rtrial-r_n;
        r_n1= rtrial  ;
        if hard_type == 0
            %  Linear
            H_r = H;
            q_n1= q_n+ H_r*delta_r;
        else
            %Hardening/Softening exponential law
            q_inf = r0 + (r0-zero_q); %calulation of q_infinity
            if H > 0
                H_r = H*((q_inf-r0)/r0)*exp(H*(1-rtrial/r0)); %calculation of tangent hard modulus
            else
                H_r = H*((q_inf-r0)/r0)*1/(exp(H*(1-rtrial/r0)));%calculation of tangent soft modu
            end
            q_n1 = q_n + H_r*delta_r; %calculation of q(n+1)
        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 %viscous model
        if (rtrial_n_alpha > r_n)
            % loading
            fload=1;
            delta_r=rtrial_n_alpha-r_n;
            % computation of r at the step n+1
            r_n1 = (eta - delta_t*(1-alpha))/(eta + alpha*delta_t)*r_n + (delta_t/(eta + alpha*delta_t))*rtrial_n_alpha;
            if hard_type == 0
                %  Linear law
                H_r = H;
                q_n1= q_n+ H_r*delta_r;
            else
                %Hardening/Softening exponential law
                q_inf = r0 + (r0-zero_q); %calulation of q_infinity
                if H > 0
                    H_r = H*((q_inf-r0)/r0)*exp(H*(1-rtrial_n_alpha/r0)); %calculation of tangent hard modulus
                else
                    H_r = H*((q_inf-r0)/r0)*1/(exp(H*(1-rtrial_n_alpha/r0)));%calculation of tangent soft modulus
                end
                q_n1 = q_n + H_r*delta_r; %calculation of q(n+1)
            end

            if(q_n1<zero_q)
                q_n1=zero_q; %the q cannot be less than the given value q0
            end
        else
            % elastic load/unload
            fload=0;
            r_n1= r_n  ;
            q_n1= q_n  ;
        end
    end
```

```
% calculation of the Ce_algor_n1 and Ce_tang_n1
if viscpr == 1
   if rtrial_n_alpha > r_n %loading
        %constitutive algorithm matrix (loading)
       Ce_algor_n1 = (1.d0 - dano_n1)*ce+((alpha*delta_t)/(eta+alpha*delta_t))*(1/rtrial_n_alpha)*((H_r*r_n1-q_n1)/(r_n1^2)).*((ce*eps_n1')'*(ce*eps_n1'));
       Ce_tang_n1 = (1.d0 - dano_n1)*ce;  %constitutive tangent matrix (loading)
   else %elastic loading/unloading
       Ce_algor_n1 = (1.d0 - dano_n1)*ce;  %constitutive algorithm matrix (elastic)

       Ce_tang_n1 = Ce_algor_n1; %constitutive tangent matrix (elastic)
   end
end
```

```
%*********************************************************************************
%* Updating historic variables                                                   *
%  hvar_n1(1:4)  = eps_n1p;
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;
hvar_n1(7)= dano_n1;
if viscpr == 1
    hvar_n1(8)= Ce_algor_n1(1,1);
    hvar_n1(9)= Ce_tang_n1(1,1);
end
%*********************************************************************************
```

- In function "damage_main.m" some changes about the plotting of the damage surface and the extra variables (d – C_{11}_Alg – C_{11}_tang) have been implemented:

```
% 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 tang 11'};
```

```
% 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) = 1-hvar_n(6)/hvar_n(5); %initializing the damage
hvar_n(8) = ce(1,1); %ce_alg == ce at the beginning
hvar_n(9) = ce(1,1); %ce_tang == ce at the beginning
eps_n1 = strain(i,:) ;
sigma_n1 =ce*eps_n1'; % Elastic
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0  sigma_n1(4)];

nplot = 5 ;
vartoplot = cell(1,totalstep+1) ;
vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)
vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5)   ; %  Damage variable (d)
vartoplot{i}(4) = hvar_n(8)   ; % Constitutive matrix algorithm (1,1)
vartoplot{i}(5) = hvar_n(9) ; % Constitutive matrix tangent (1,1)
```

```
 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,:) ;
     % Total strain at the previous step
     eps_n = strain(i-1,:);
     %*************************************************************************************
     %*      DAMAGE MODEL
     % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     % change of variables needed in rmap_dano1 (considering the case of
     % viscous model) - new vars: eps_n,viscpr,delta_t
     [sigma_n1,hvar_n,aux_var] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,viscpr,delta_t);
     % PLOTTING DAMAGE SURFACE
     if viscpr == 0
         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
     else
     end
```

```
 % 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) = hvar_n(8) ; % Constitutive matrix algorithm (1,1)
 vartoplot{i}(5) = hvar_n(9) ; % Constitutive matrix tangent (1,1)
```