UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Swansea University Prifysgol Abertawe

CENTRALE NANTES

Universität Stuttgart

MSc in Computational Mechanics

Computational Solid Mechanics

Assignment 1 - Damage modelling

Submitted By:
Mario Alberto Mendez Soto

Submitted To:
Prof. Joaquin A. Hernandez
Prof. Xavier Oliver

Spring Semester, 2019

1. Introduction

Damage models are used to simulate different materials that undergo degradation of their mechanical properties due to micro-defects and cracks that appear as a consequence of loading. These mathematical models allow engineers to predict damage-related phenomena without the need of a microscopic description that becomes impractical for many engineering applications [1].

A MATLAB implementation for rate-independent damage model at Gauss-point level was originally provided with the symmetric model fully operational under a plane strain hypothesis. This implementation allows the user to input the loading path and compute several properties such as stress-strain ($\sigma - \varepsilon$) curves, stress norm curves and evolution over time of damage(d), hardening variable (q), and internal variable (r).

Firstly, the code was modified to include the tension-only and non-symmetric models. Moreover, both linear and exponential laws of softening/hardening were implemented within the code. Subsequently, rate-dependent computations (alpha-integration method) were considered to study the viscous effects of a given material. The new implementation also includes computations for the algorithmic and tangent tensorial modulii. The modified routines can be found in the appendices of this report.

Please note that within the framework of the present report, a material with Young's modulus $E = 20,000$ Pa and Poisson ratio $\nu = 0.3$ was considered. The yield strength for this material was set equal to 200 Pa. In general, three load states were studied with 50 automatically-computed sub-steps (if not indicated otherwise).

2. Rate-independent damage model

After implementation of the tension-only model (refer to Appendix A), the following sample separate load paths were analysed:

$$\text{Biaxial tension loading} \begin{cases} \sigma_1 = [150\ ;\ 150] \\ \sigma_2 = [300\ ;\ 300] \\ \sigma_3 = [450\ ;\ 450] \end{cases}$$

$$\text{Biaxial compression loading} \begin{cases} \sigma_1^* = [-150\ ;\ -150] \\ \sigma_2^* = [-300\ ;\ -300] \\ \sigma_3^* = [-450\ ;\ -450] \end{cases}$$

For a perfect damage model with no hardening/softening, Figure (2.1) depicts the stress-strain and stress surface curves computed for the tension-only model. As expected, the material yields when the stress equals 200 Pa in tension, whereas, the behavior of the material in compression is perfectly elastic.

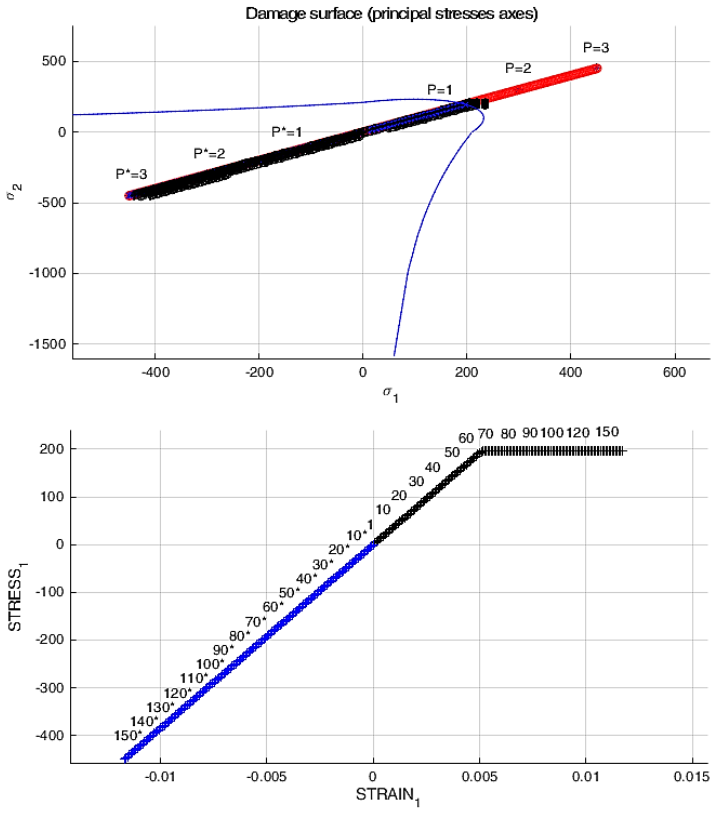


Fig. 2.1. – Damage surface and stress-strain curves for biaxial tension(—) and compression(—) loading for the tension-only model

To better illustrate the differences between the tension-only and symmetric model a comparison between the evolution of the damage variable can be considered (see Figure (2.2)). For this case, the material exhibits a linear softening law with $H = -0.2$ and the loading path is the following:

$$\begin{cases} \sigma_1 = [250\ ;\ 250] \\ \sigma_2 = [-300\ ;\ -300] \\ \sigma_3 = [0\ ;\ 0] \end{cases}$$

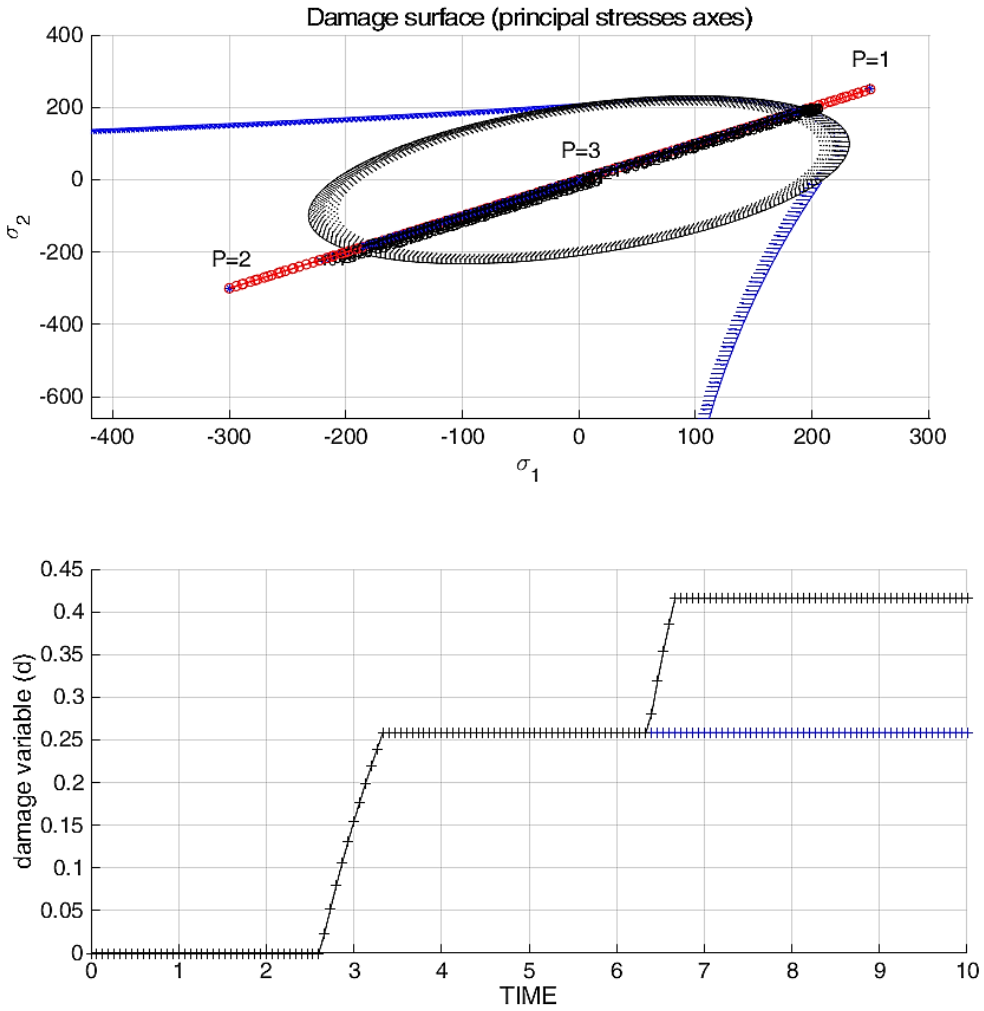


Fig. 2.2. – Comparison between symmetric (—) and only-tension (—) models

As described above, Figure (2.2) shows that during compression the model does not yield if the the tension-only model is used, therefore damage remains constant and does not change. Contrarily, for the symmetric model, damage starts under compression after the yield limit is reached causing an increase in the damage variable.

Similarly, after coding the non-symmetric model within the program (refer to Appendix A), the following two separate load states were input:

$$\text{Biaxial tension loading} \begin{cases} \sigma_1 = [100\ ;\ 100] \\ \sigma_2 = [200\ ;\ 200] \\ \sigma_3 = [300\ ;\ 300] \end{cases}$$

$$\text{Biaxial compression loading} \begin{cases} \sigma_1^* = [-200\ ;\ -200] \\ \sigma_2^* = [-400\ ;\ -400] \\ \sigma_3^* = [-600\ ;\ -600] \end{cases}$$

For the previously indicated load states and a linear softening law ($H = -0.8$), Figure (2.3) depicts the stress-strain and stress surface curves computed for the non-symmetric model ($n = 2$).

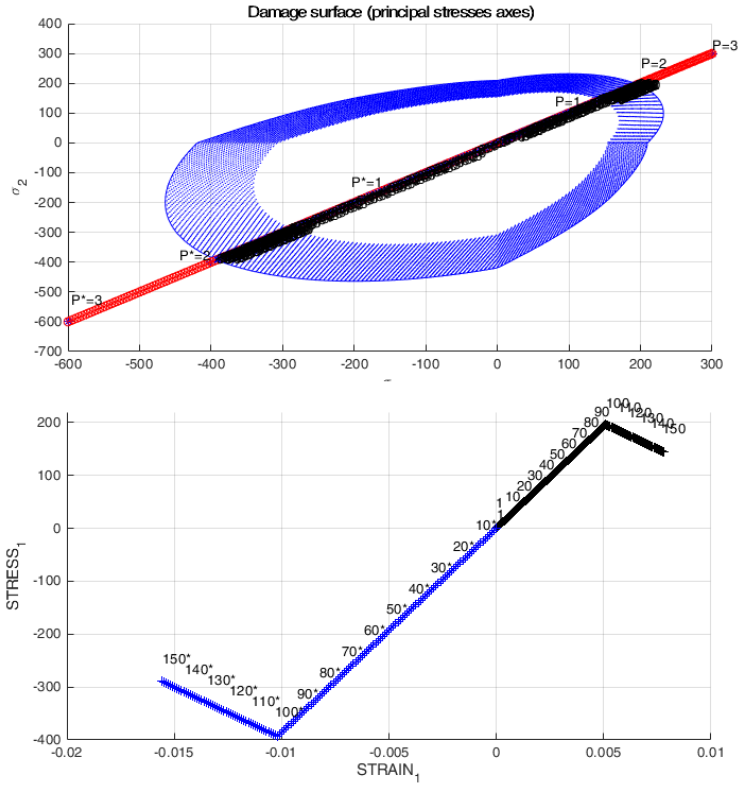
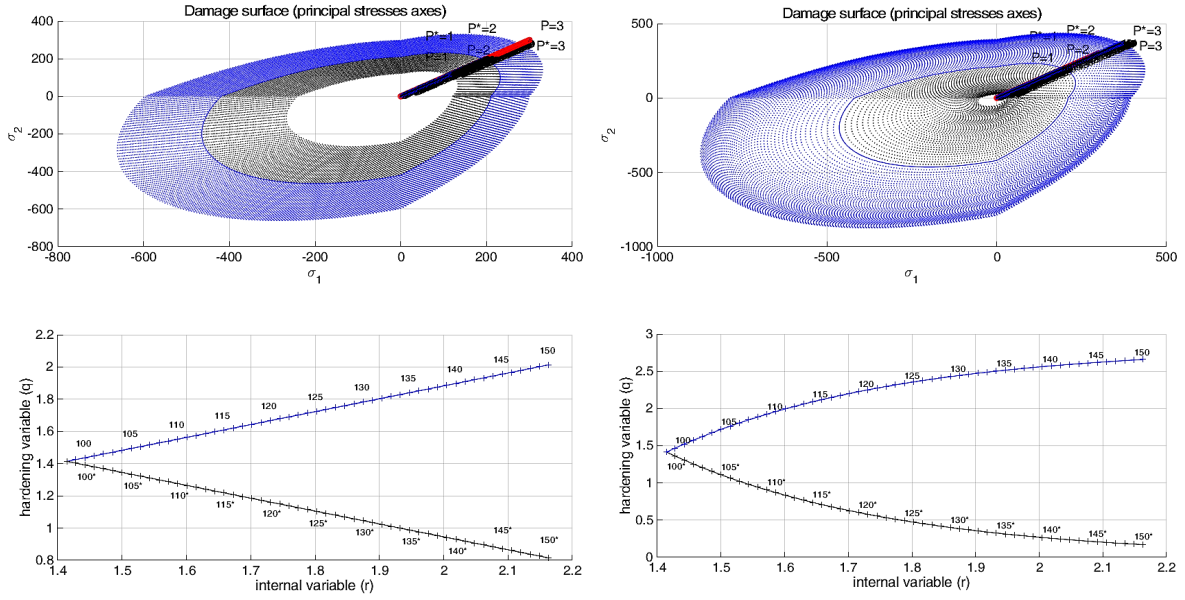


Fig. 2.3. – Damage surface and stress-strain curves for biaxial tension(—) and compression(—) loading for the non-symmetric model

Figure (2.3) shows how the material behaves differently depending on the type of loading: compression or tension. For positive axial loading, softening starts when $\sigma = 200$ Pa while if the sample is loaded in compression the material yields only after $\sigma = 400$ Pa is reached.

Hardening/softening linear and exponential laws were implemented by means of the modification of the original code. For the linear law, the parameter H mathematically defines the phenomenon while, in the exponential case, user-defined parameters q_∞ and A are used. The latter parameters describe the value q reached as r increases and the rate at which this value is approached, respectively. The MATLAB code can be seen in Appendix B.

Figure (2.4) shows the change in the damage surface for both laws and dependency between the hardening variable q and internal variable r. For this figure, an equal biaxial loading was considered with $P_1 = P_1^* = 150$ Pa, $P_2 = P_2^* = 200$, $P_3 = P_3^* = 300$. The model used was a non-symmetric one with $n = 2$ and the linear law has a parameter H equal to ± 0.8. For the exponential law, $A = 2$ and the hardening variable q approaches $2r_0$ and 0 in the hardening and softening cases, respectively.



(a) (b)

Fig. 2.4. – Hardening (—)/softening (—) linear (a) and exponential(b) laws for a non-symmetric material with $n = 2$

In order to asses the correctness of the implementation, the code was additionally tested for several loading states starting at $\sigma_1^{(0)} = \sigma_2^{(0)} = 0$. For the cases to be presented below, it was considered that $\alpha = 300$ Pa, $\beta = 350$ Pa and $\gamma = 400$ Pa.

1. Non-symmetric model ($n = 3$) and exponential softening law ($A = 1$ and $q_\infty = 0$)

$$\begin{cases} \Delta\sigma_1^{(1)} = \alpha \ ; \ \Delta\sigma_2^{(1)} = 0 \\ \Delta\sigma_1^{(2)} = -\beta \ ; \ \Delta\sigma_2^{(2)} = 0 \\ \Delta\sigma_1^{(3)} = \gamma \ ; \ \Delta\sigma_2^{(3)} = 0 \end{cases}$$

2. Tension-only model and linear softening law ($H = -0.8$)

$$\begin{cases} \Delta\sigma_1^{(1)} = \alpha \ ; \ \Delta\sigma_2^{(1)} = 0 \\ \Delta\sigma_1^{(2)} = -\beta \ ; \ \Delta\sigma_2^{(2)} = -\beta \\ \Delta\sigma_1^{(3)} = \gamma \ ; \ \Delta\sigma_2^{(3)} = \gamma \end{cases}$$

3. Non-symmetric model ($n = 3$) and exponential softening law ($A = 1$ and $q_\infty = 0$)

$$\begin{cases} \Delta\sigma_1^{(1)} = \alpha \ ; \ \Delta\sigma_2^{(1)} = \alpha \\ \Delta\sigma_1^{(2)} = -\beta \ ; \ \Delta\sigma_2^{(2)} = -\beta \\ \Delta\sigma_1^{(3)} = \gamma \ ; \ \Delta\sigma_2^{(3)} = \gamma \end{cases}$$

Results of case 1 are plotted in Figure (2.5). It can be noticed that the stress-strain curve passes through point $(0, 0)$ since no plastic damage should be remaining after unloading the sample completely. Moreover, since the model is non-symmetric, during compression the material behaves elastically for the given load states analyzed (yield limit is not reached during the compression loading).

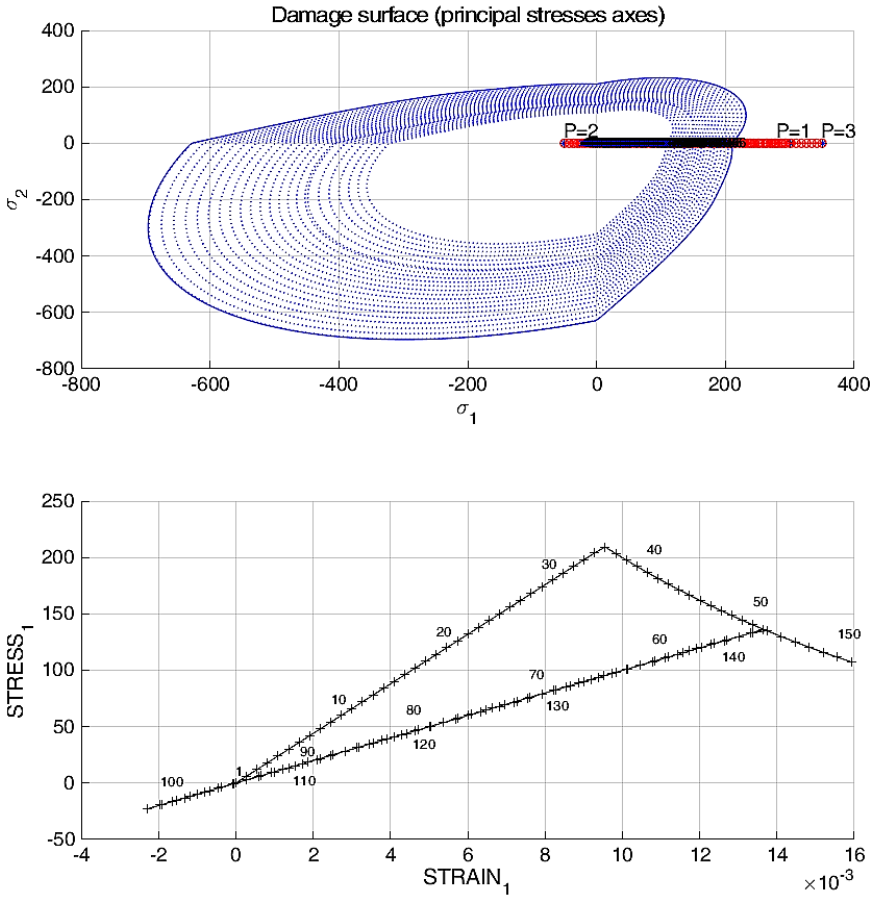


Fig. 2.5. – Damage surface and stress-strain curve for case 1

Figure (2.6) depicts the results computed for case 2. Stress-strain ($\sigma_1 - \varepsilon_1$) and ($\sigma_2 - \varepsilon_2$) curves for this case are shown. It is important to mention that even though for the first load stage σ_2 remains equal to zero, a strain is present due to the effects of the Poisson ratio. This transverse strain also explains why the $\sigma_1 - \varepsilon_1$ curves does not go through $(0, 0)$ since this would be satisfied only if $\sigma_1 = \sigma_2 = 0$ at the same time and not because a plastic deformation is produced as it might be erroneously understood. As expected, the sample behaves as a perfectly elastic material in compression.

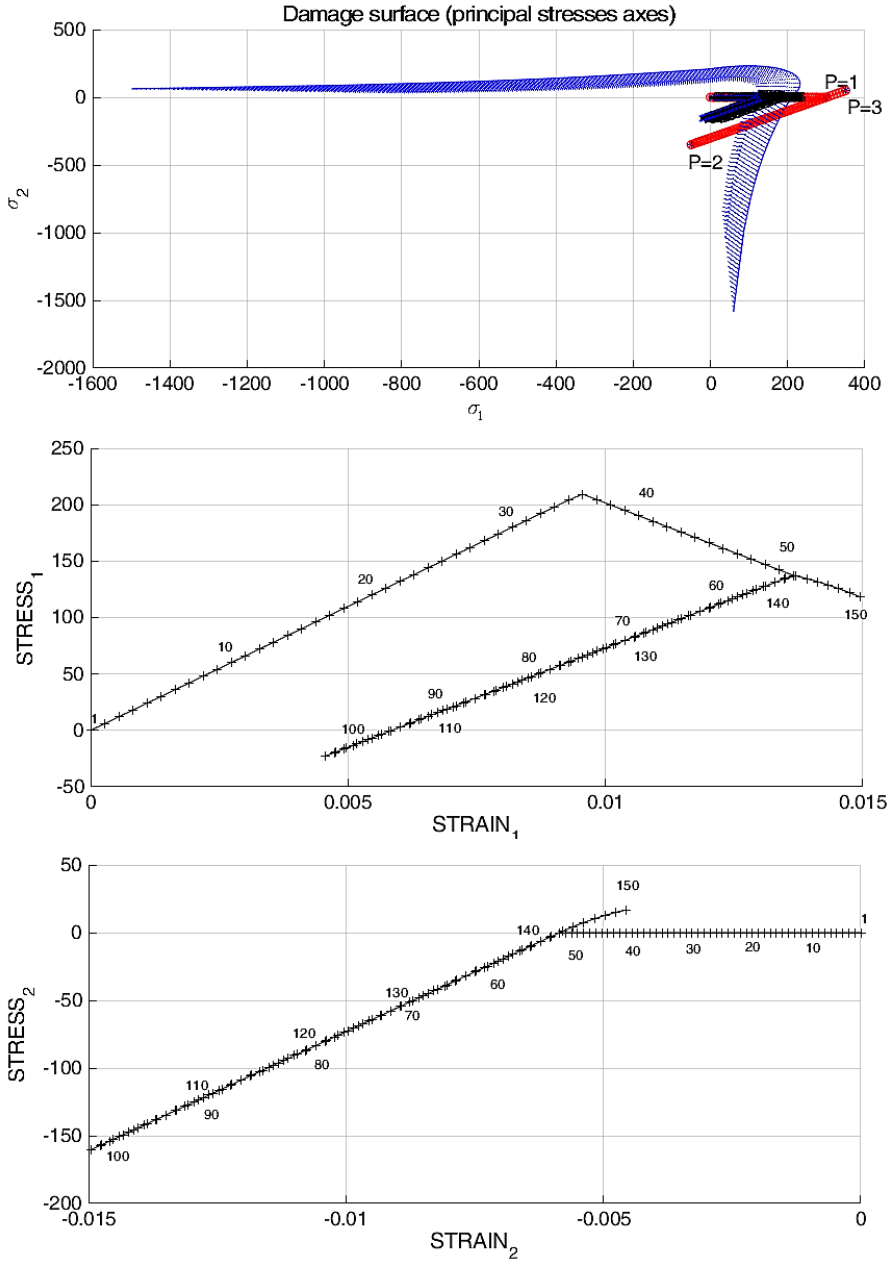


Fig. 2.6. – Damage surface and stress-strain curve for case 2

Figure (2.7) presents the stress surface and $\sigma_1 - \varepsilon_1$ curve for case 3 (curve $\sigma_2 - \varepsilon_2$ is omitted since it perfectly coincides with $\sigma_1 - \varepsilon_1$, due to the orthotropic assumed property of the material and the loading applied). For this case, it is important to point out that the curve does go through point $(0, 0)$ since the condition $\sigma_1 = \sigma_2 = 0$ is satisfied during the unloading/compression stage. Also, as it was observed in the two previous cases, it is worth noticing that during traction re-loading the stress-strain curve goes back to its initial softening curve and continues its behavior prior to unloading.

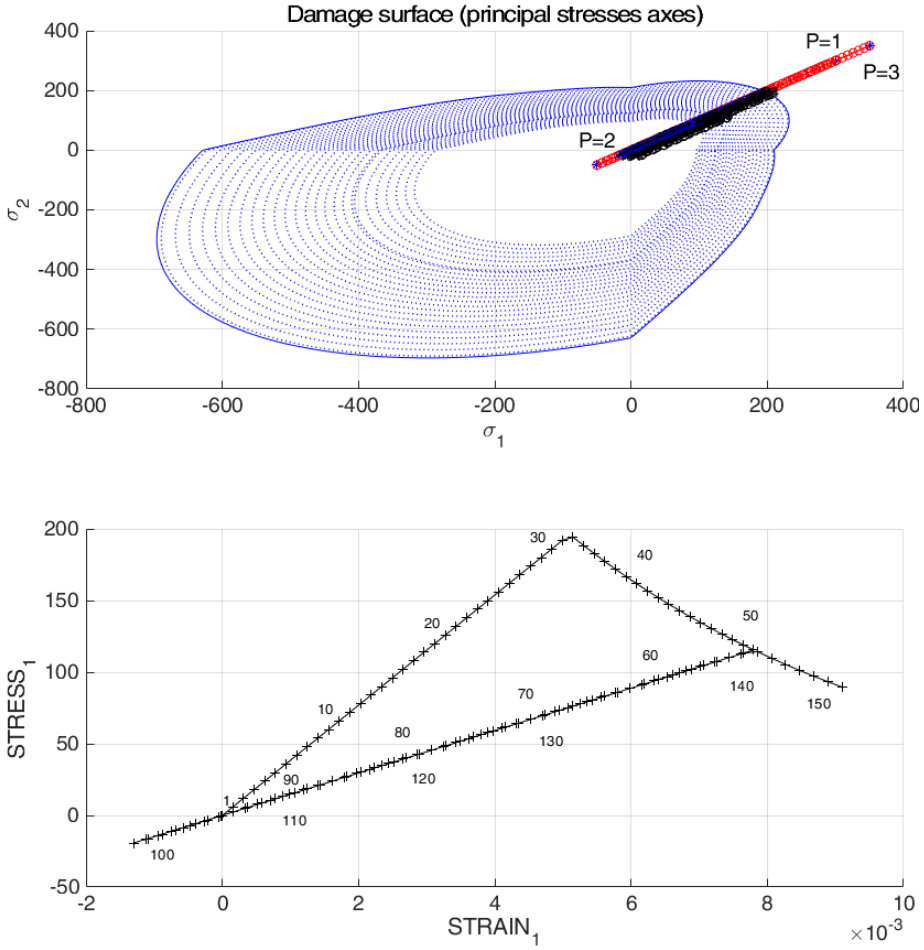


Fig. 2.7. – Damage surface and stress-strain curve for case 3

3. Rate-dependent damage model

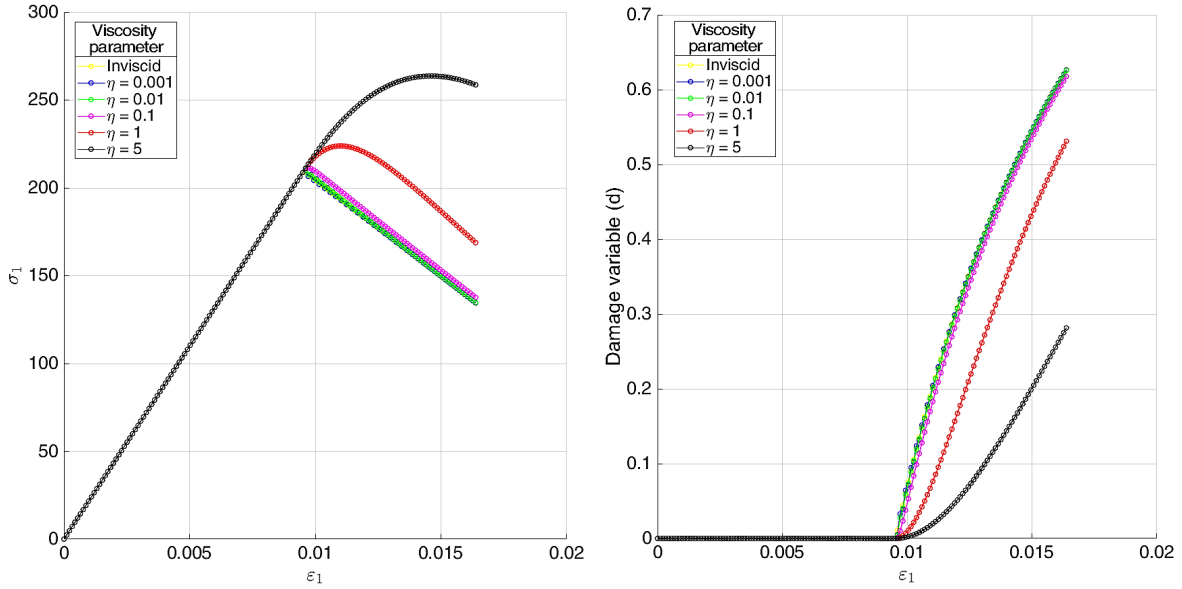
Using the supplied MATLAB code, the alpha-method integration algorithm (plane strain case) was implemented for the isotropic visco-damage model (refer to Appendix C).

For all the tests carried out in this chapter, the following biaxial loading path is considered:

$$\begin{cases} \sigma^{(1)} = [120\ ,\ 120] \\ \sigma^{(2)} = [240\ ,\ 240] \\ \sigma^{(3)} = [360\ ,\ 360] \end{cases}$$

The model analysed is symmetric in tension and compression with linear softening ($H = -0.5$).

Viscoelastic effects on damage were studied, firstly, by varying the value of the viscosity parameter η. For these tests, the total time T equals 10 s and the integration is computed using $\alpha = 0.5$. Stress-strain curves and the evolution of the damage variable are plotted in Figure (3.1) for different viscosity parameters.

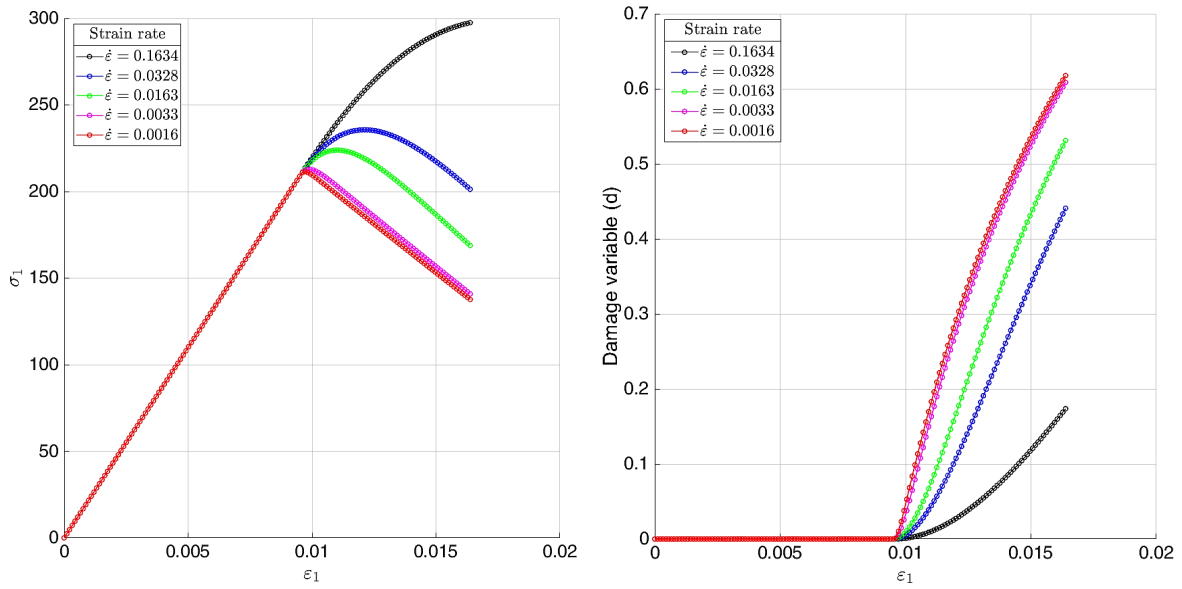


(a) (b)

Fig. 3.1. – Stress-strain curves (a) and damage evolution (b) for different viscosity parameters η

Figure (3.1) shows that an increase in the viscosity parameter causes a rise in the resistance of the material to yield, therefore, originating a delayed yield (softening in the case studied). Moreover, it can be noticed that for a very small viscosity value the rate-independent case is recovered.

To give an illustration of the effect of strain rate on the behavior of a viscous material during damage, the final total time was varied. For these computations, the viscosity parameter η was set to 0.1 and the integration was carried out for $\alpha = 0.5$. Figure (3.2) summarises the stress-strain curve variation and the evolution of the damage variable as the total time T is modified, i.e. the strain rate. For the plotted curves, the corresponding total times are: 10, 5, 1, 0.5, 0.1 seconds.



(a) (b)

Fig. 3.2. – Stress-strain curves (a) and damage evolution (b) for different strain rates $\dot{\varepsilon}$

A significant aspect to understand from Figure (3.2) is that for very low strain rates, the rate-independent case seems to be recovered, which coincides with the case of a small viscosity parameter. In other words, decreasing the viscosity is equivalent to slowing down the process of deformation, hence an equivalence between a limit of zero viscosity and an infinitely slow process is demonstrated [2].

For the final set of tests, the effect of the integration parameter α was analysed. For these computations, the final time T was set to be 10 s and the viscosity parameter η equals 0.1.

Figures (3.3) and (3.4) depict the results of the computations of element C_{11} of the tangent and algorithmic tensorial modulii. Firstly, it becomes evident that, for $\alpha = 0$, $\mathbb{C}^n_{alg} = \mathbb{C}^n_{tan}$. Furthermore, during the elastic part of the loading, both modulii are constant and equal to element C_{11} of the constitutive elastic matrix $\mathbb{C}$, since no damage is present. Finally, it is worth mentioning that the "jump" exhibited in the $\mathbb{C}^{alg}_{11}$ curves is reduced as the number of time steps increases (see Figure (3.5)).

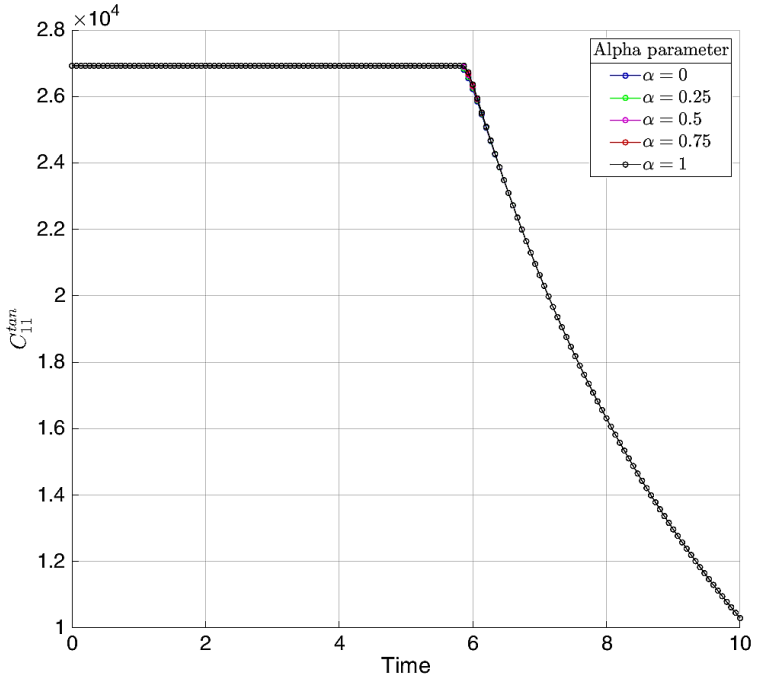


Fig. 3.3. – Evolution of C_{11}^{tan} using different α values ($T = 10$ s and 50 time steps)

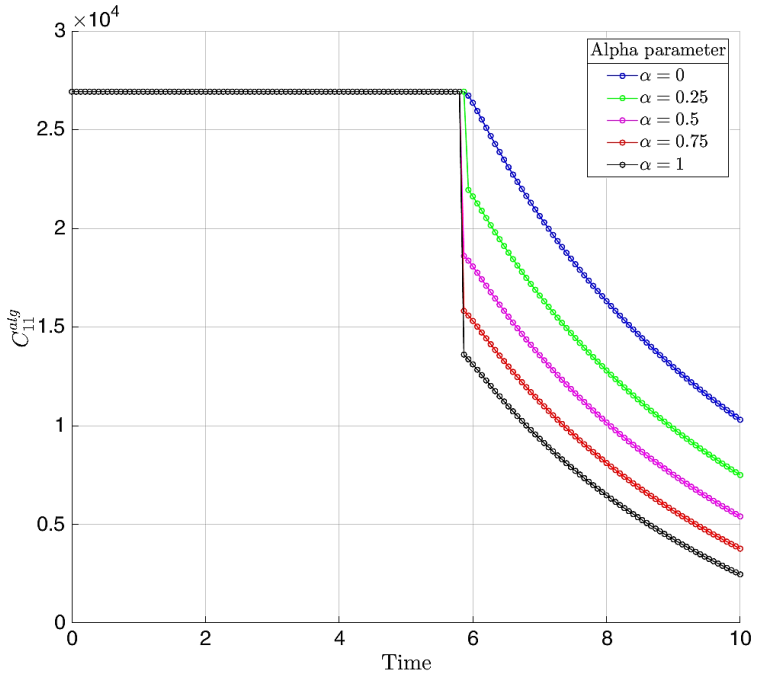


Fig. 3.4. – Evolution of C_{11}^{alg} using different α values ($T = 10$ s and 50 time steps)

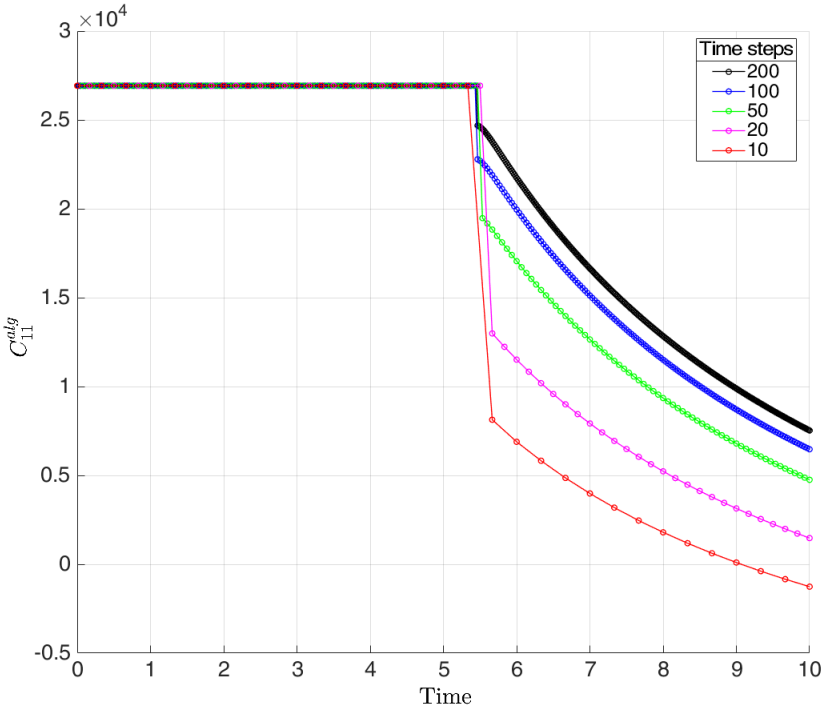


Fig. 3.5. – Variation in the computations of C_{11}^{alg} ($\alpha = 0.5$) using different time discretizations

It is important however not to assume the stability of the computations of $\mathbb{C}_{alg}$ and $\mathbb{C}_{tan}$ in all cases. Theory in [2] suggests that the integration algorithm is conditionally stable for values $\alpha < 0.5$. This stability range can be noticed if the discretization and total time is changed in order to enter into a potentially unstable zone. Figures (3.6) and (3.7) show the results obtained for element C_{11} of $\mathbb{C}_{alg}$ and $\mathbb{C}_{tan}$ analysing the same loading path, but using 20 time steps and $T = 50$ s.

For this particular case, the integration becomes unstable for $\alpha < 0.5$ as oscillations are present. Conversely, for $\alpha \geq 0.5$ the integration is stable and no spurious results are obtained.

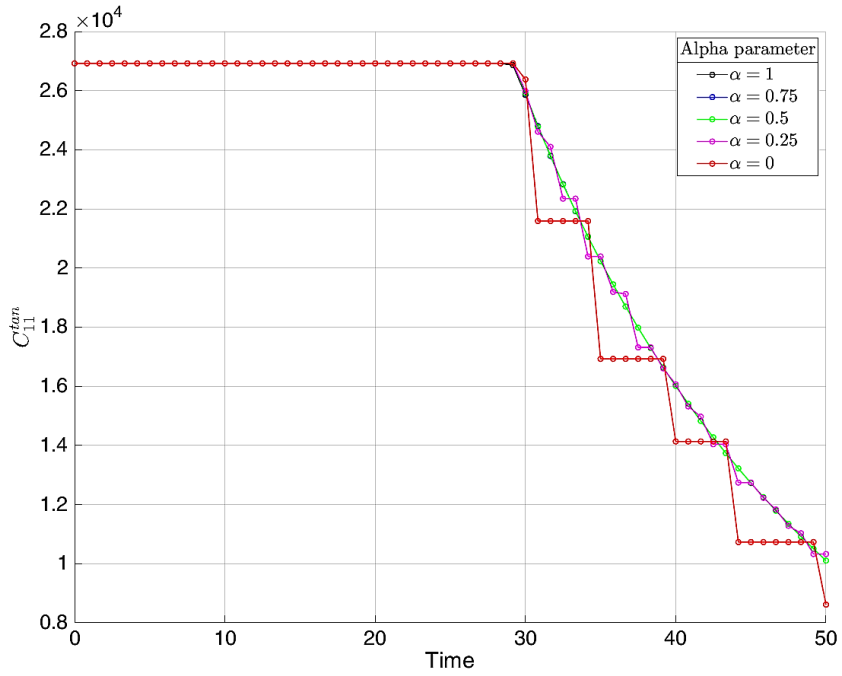


Fig. 3.6. – Evolution of C_{11}^{tan} using different α values ($T = 50$ s and 20 time steps)

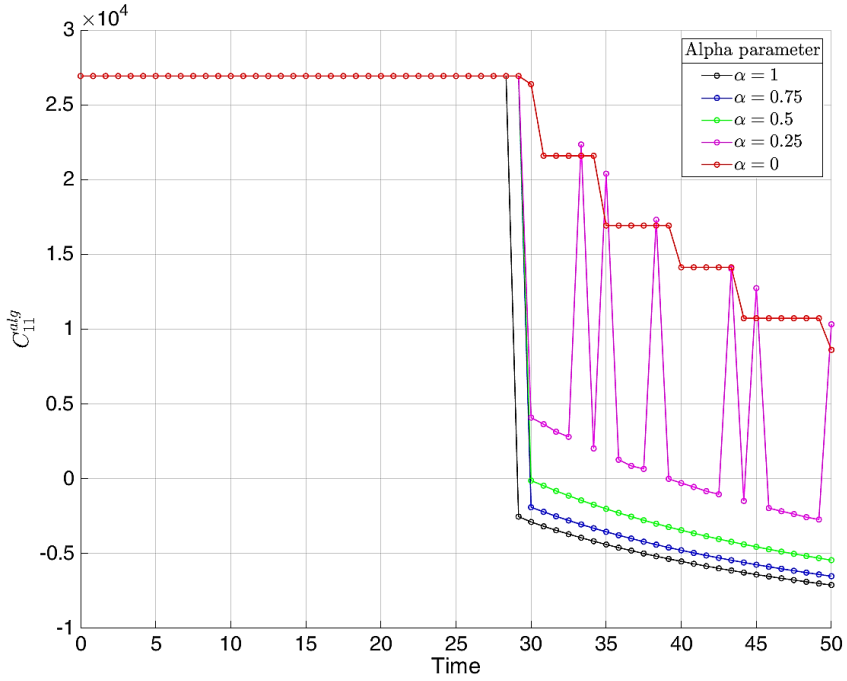


Fig. 3.7. – Evolution of C_{11}^{alg} using different α values ($T = 50$ s and 20 time steps)

Bibliography

[1] Eduardo Chaves. *Notes on continuum mechanics*. Lecture Notes on Numerical Methods in Engineering and Sciences. CINME, Springer, 2013. ISBN: 978-94-007-5985-5 (HB).

[2] Jacob Lubliner. *Plasticity theory*. Courier Corporation, 2008. ISBN: 9780486462905.

Appendix A. Tension-only and non-symmetric model

Listing A.1 – Lines added/modified in subroutine *Modelos_de_dano1.m*

```
if (MDtype==1)       % Symmetric model
rtrial= sqrt(eps_n1*ce*eps_n1');

elseif (MDtype==2)  % Only-tension model
sigma_n1 =ce*eps_n1';
    for j=1:4
            if sigma_n1(j)<0
               sigma_n1(j)=0;
            end
    end
    cei=inv(ce);
    eps_n1m=(cei*sigma_n1)';

    rtrial= sqrt(eps_n1m*ce*eps_n1m');

elseif (MDtype==3)  % Non-symmetric model
    sigma_n1 =ce*eps_n1';
    sigma_vec=sigma_n1;
    sum_eps_abs=sum(abs(sigma_n1));
    sum_eps=0;
    for j=1:4
        if sigma_vec(j)<0
           sigma_vec(j)=0;
        end
        sum_eps=sum_eps+sigma_vec(j);
    end

    if sum_eps_abs ==0
        coeft=0;
    else
    theta_sigma=sum_eps/sum_eps_abs;
    coeft=theta_sigma+(1-theta_sigma)/n;
    end

    rtrial= coeft*sqrt(eps_n1*ce*eps_n1');
end
```

Listing A.2 – Lines added/modified in subroutine *dibujar_criterio_dano1.m*

```
elseif MDtype==2 %Tension-only model
   tetha=[0:0.05:2*pi];% Range of angles
    D=size(tetha);
    m1=cos(tetha);
    m2=sin(tetha);
    Contador=D(1,2);

    %Stress and Macaulay bracket initialization
    radio = zeros(1,Contador) ;
    s1    = zeros(1,Contador) ;
    s2    = zeros(1,Contador) ;
    m1_p    = zeros(1,Contador) ;
    m2_p    = zeros(1,Contador) ;

    %Macaulay bracket
    for i=1:Contador
        m1_p(i)=m1(i);
        m2_p(i)=m2(i);
        if m1(i)<0
            m1_p(i)=0;
        end
        if m2(i)<0
            m2_p(i)=0;
        end

    %Computation of radius
        radio(i)= q/sqrt([m1_p(i) m2_p(i) 0 ...
        nu*(m1_p(i)+m2_p(i))]*ce_inv*[m1(i) m2(i) 0 ...
        nu*(m1(i)+m2(i))]');

    %Computation of the stresses
        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);
    end
    hplot =plot(s1,s2,tipo_linea);

elseif MDtype==3 %Non-symmetric model
s0 = q/sqrt([1 0 0 nu]*ce_inv*[1 0 0 nu]'); %Base stress value
      tetha=[0:0.01:2*pi];% Range of angles

      D=size(tetha);
      m1=cos(tetha);
      m2=sin(tetha);
      Contador=D(1,2);

```

```
46        %Radius and Stress initialization
47        radio = zeros(1,Contador) ;
48        s1    = zeros(1,Contador) ;
49        s2    = zeros(1,Contador) ;
50
51        %Computation of coefficient theta
52          for i=1:Contador
53          sum_stress=0;
54          vec_stress=[m1(i) m2(i) nu*(m1(i)+m2(i))];
55          sum_stress_abs=sum(abs(vec_stress));
56          for j=1:3
57              if vec_stress(j)<0
58                  vec_stress(j)=0;
59              end
60              sum_stress=sum_stress+vec_stress(j);
61          end
62
63          theta_stress=sum_stress/sum_stress_abs;
64          coeft=theta_stress+(1-theta_stress)/n;
65
66        %Computation of the radius
67          radio(i)= q/(coeft*(sqrt([m1(i) m2(i) 0 ...
68          nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i) 0 ...
69              nu*(m1(i)+m2(i))]')));
70
71        %Computation of the stresses
72          s1(i)=radio(i)*m1(i);
73          s2(i)=radio(i)*m2(i);
74  end
75       hplot =plot(s1,s2,tipo_linea);
76  end
```

Appendix B.Linear and exponential hardening

Listing B.1 – Lines added/modified in subroutine *rmap_dano1.m*

```
H = Eprop(3); % Parameter for linear law (user-defined)
q_inf=0.0005; % Parameter for exponential law (user-defined)
A=1; % Parameter for exponential law (user defined)
if hard_type == 0   % Linear law
        q_n1= q_n+ H*delta_r;
        else         %Exponential law
        q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
end
```

Appendix C.Rate-dependent model

Listing C.1 – Lines added/modified in subroutine *rmap_dano1.m*

```
fload=0;
if viscpr == 1 %viscous case

% Strain norm at time n
[tau_eps_n] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
% Strain norm at time n+1
[tau_eps_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
%Srain norm at time n+alpha
[tau_eps_na] = (1-alpha)*tau_eps_n+alpha*tau_eps_n1;

if tau_eps_na > r_n  %loading
    fload=1;
    r_n1 = (eta-t*(1-alpha))/(eta+alpha*t)*r_n+...
    t/(eta+alpha*t)*tau_eps_na;
    delta_r=r_n1-r_n;
        if hard_type == 0
        q_n1= q_n+ H*delta_r;
        else
        q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
        end
   if(q_n1<zero_q)
        q_n1=zero_q;
   end
else %elastic/unloading
   r_n1= r_n ;
   q_n1= q_n ;
end

else %inviscid case
%strain norm at n+1
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);

fload=0;
if(rtrial > r_n) %loading
    fload=1;
    delta_r=rtrial-r_n;
    r_n1= rtrial;
    if hard_type == 0
        q_n1= q_n+ H*delta_r;
    else
        q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
    end
```

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

else %Elastic unloading/elastic loading
    fload=0;
    r_n1= r_n  ;
    q_n1= q_n  ;
end
end

% Damage variable
dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
%* Updating historic variables
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;

%Auxiliar variables for C_tan and C_alg modulii

if viscpr ==1 %viscous case

C_alg = (1.d0-dano_n1)*ce - fload*Eprop(8)*...
t/(Eprop(7)+Eprop(8)*t)*...
1/tau_eps_n1*(q_n1-H*r_n1)/r_n1^2*(sigma_n1)*...
(sigma_n1')/((1.d0-dano_n1)^2);

C_tan = (1.d0-dano_n1)*ce;

else %inviscid case
C_alg = (1.d0-dano_n1)*ce - fload*Eprop(8)*...
t/(Eprop(7)+Eprop(8)*t)*1/r_n1*...
((q_n1-H*r_n1)/r_n1^2)*(sigma_n1)*...
(sigma_n1')/((1.d0-dano_n1)^2);

C_tan = (1.d0-dano_n1)*ce -fload*...
((q_n1-H*r_n1)/r_n1^3)*(sigma_n1)*...
(sigma_n1')/((1.d0-dano_n1)^2);

end

aux_var(2) = C_tan(1,1);
aux_var(3) = C_alg(1,1);
```

Listing C.2 – Lines added/modified in subroutine *damage_main.m*

```
eps_n1 = strain(i,:) ;    %Input of strain at time n+1
eps_n = strain(i-1,:) ;   %Input of strain at time n

%Changes in the variables of function rmap_dano1
[sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t(iload));

%Addition of new variables to be plotted
vartoplot{i}(4) = aux_var(2); %Tangent constitutive tensor C11
vartoplot{i}(5) = aux_var(3); %Algorithmic tangent tensor C11
```