UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Master in Computational Mechanics

Computational Solid Mechanics:
Computational Elasticity

Assignment I

Numerical Integration of Constitutive Damage Models using MATLAB

Submitted By :
Moritz Jokeit

Contents

1 Introduction

The aim of the first assignment in the course "Computational Solid Mechanics" is to grasp the algorithmic structure underlying the numerical integration of continuum damage constitutive models. For educational purposes the focus lies on the local constitutive response of a fixed point in the continuum rather than computing a whole domain in a finite element matter. Furthermore, only plane strain is considered. The students are asked to implement these models in an existing MATLAB code skeleton. The template already provides a working implementation for computing and plotting the inviscid symmetric case utilizing a linear hardening/softening law.

It is worth mentioning that the existing code also offers a graphical user interface that allows the user to interactively set the loading path and to plot different internal variables. However, to avoid a tedious adaption of the GUI, changes are limited to the `main_nointeractive.m` file and its containing functions where all input parameters have to be set inline.

The report follows the assignments' structure which divides the implementation into several steps. In the first part the rate independent model is modified to incorporate a non-symmetric tension-compression damage model as well as tension-only damage model. Additionally, an exponential hardening and softening law is introduced. In a next step the correctness of the implementation is assessed using three different loading and unloading scenarios. The second part focuses on the integration algorithm representing a continuum isotropic visco-damage model for the symmetric tension-compression case. Again, to verify the results a parameter study is conducted.

Before the discussion of the actual implementation all input parameters used for the computations and assessments are introduced in the following section.

Remark: All mentioning of the underlying theory refers to the lecture held by Prof. Huespe and the corresponding lecture notes.

2 Input parameters

As mentioned above this sections gives an overview of the parameters chosen for the computations executed throughout the assessments of the implementations. If not further specified the parameters are defined as follows:

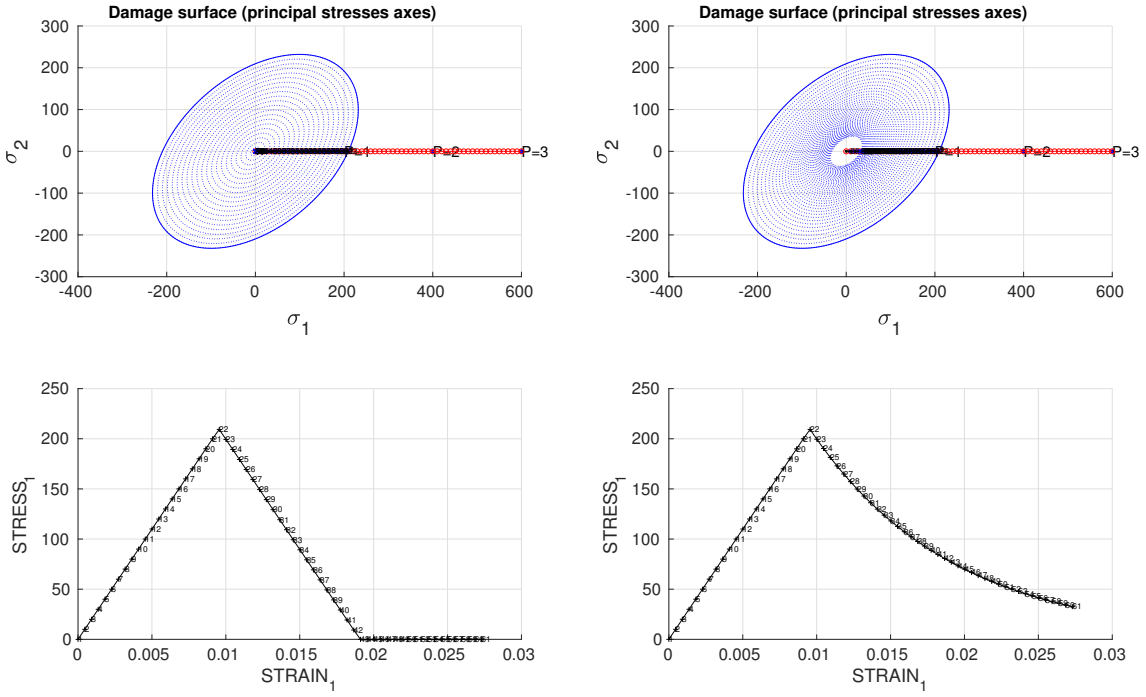
- Young's modulus $E = 20\,000$ [MPa]
- Poisson's ratio $\nu = 0.3$
- Hardening modulus $H = \begin{cases} 1.0 & \text{if } H > 0 \\ -1.0 & \text{if } H < 0 \end{cases}$
- Yield stress $\sigma_y = 200$ [MPa]
- Ratio between compression and tension strength $n = 3$
- Viscosity coefficient $\eta = 0.0$ [MPa/s]
- Total time $t = 10$ [s]
- Integration coefficient $\alpha = 1.0$ (Implicit Backward Euler scheme)
- Number of time increments per load state $n_{t,inc} = 20$

For simplicity reasons all values in the following sections will be given in dimensionless form.

3 Rate-independent damage models

3.1 Exponential hardening/softening law

Before dealing with the implementation of the non-symmetric damage models an exponential hardening respectively softening law is introduced to the function `rmap_dano1.m`. To be specific the term for calculating the hardening variable q is changed from the linear expression $q(r) = r_0 + H(r - r_0)$ to the exponential expression $q(r) = q_\infty - (q_\infty - r_0) * exp(A * (1 - \frac{r}{r_0}))$ with A chosen as $|H|$. In order to see the influence of this exponential term a computation with a symmetric tension-compression model and a negative hardening modulus $H = -1.0$ (softening) was conducted. The differences between the already implemented linear law and the introduced exponential law can be observed in Figure 1. The upper plots where the coordinate axes represent the principal stresses σ_1 and σ_2, show the corresponding damage surfaces as blue ellipses for each time step. The prescribed loading path for an ideal linear isotropic material is denoted in red and the actual path in the stress space is shown in black. In this case uniaxial tensile loading was applied starting from $(\sigma_1^{(0)}, \sigma_2^{(0)}) = (0, 0)$ with stress increments of $\Delta\bar{\sigma}_1^{(1)} = 200$, $\Delta\bar{\sigma}_1^{(2)} = 200$, $\Delta\bar{\sigma}_1^{(3)} = 200$. The plots on the bottom of the figure show the principal stress σ_1 plotted against the principal strain ε_1. Here, the differences between the two models become particularly evident. After approaching the yield stress, the left plot (Figure 1a) shows a linear behaviour until complete damage is reached, whereas on the right (Figure 1b) an exponential softening can be observed.



(a) Linear softening $H = -1.0$

(b) Exponential softening $H = -1.0$

Figure 1: Comparison of linear and exponential softening in a symmetric tension-compression model

Henceforth, all computations in this chapter will be executed using the exponential hard-

ening respectively softening law.

3.2 Tension-only damage model

This section will deal with the implementation of the tensile-only damage model which is handled inside the `Modelos_de_dano1.m` function. With respect to the calculation of $\bar{\sigma}$ two cases have to be distinguished. In case the values of $\bar{\sigma}$ are positive they stay untouched, if they are negative they are set to zero. The new stress tensor (or here, vector in Voigt notation) is called $\bar{\sigma}^+$. Apart from that the calculation of τ_ε stays similar to the symmetric case and takes the form $\tau_\varepsilon = \sqrt{\bar{\sigma}^+ : \varepsilon}$.

3.3 Non-symmetric tension-compression damage model

The implementation of the non-symmetric tension-compression damage model takes also place inside the `Modelos_de_dano1.m` function. The main difference between the symmetric and non-symmetric damage model is the introduction of a factor of the form $[\Theta + \frac{1-\Theta}{n}]$ where $\Theta = \frac{\sum_1^3 \langle\bar{\sigma}\rangle}{\sum_1^3 |\bar{\sigma}|}$ and n stands for the compression-tension strength ratio. $\bar{\sigma}$ can be computed by the double contraction of $\mathbb{C}$ and ε taking the first, second and fourth value of the resulting vector (Voigt notation). By setting all values to zero which are smaller than zero we obtain the McAuley bracket of this vector and by simply applying MATLAB's `abs()` function we get a vector with the absolute values of $\bar{\sigma}$ respectively.

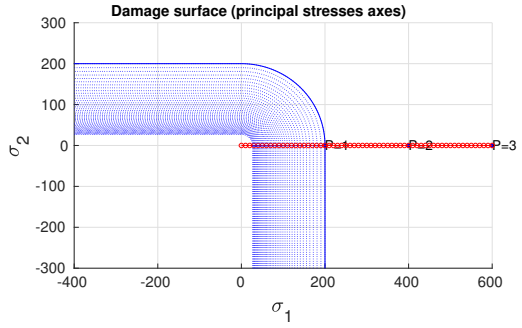
3.4 Validation of the implementation

In both cases the function `dibujar_criterio_dano1.m`, that is responsible for plotting the damage surface, had to be modified accordingly. To check the correct visualization of the damage surface the test case from section 3.1 is investigated again. For the non-symmetric tension-compression damage model shown in Figure 2b we can observe, that the symmetric model (upper plot of Figure 1b) is recovered, when the compression-tension strength ratio n is set to one. In case of the tension-only damage model the Poisson's ratio ν is set to zero which should lead to a horizontal respectively vertical contour lines of the damage surface in areas where one of the principal stresses is negative. The plot in Figure 2a confirms the expected behaviour resulting in an "infinitely" large damage surface.

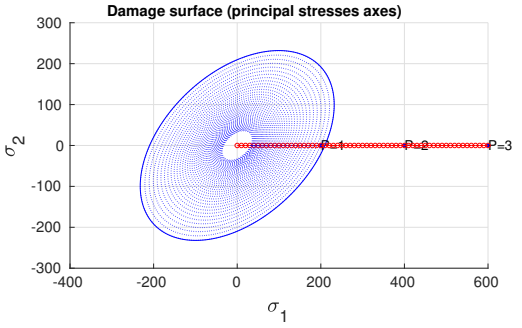
Furthermore, the correct computation of τ_ε should be assessed for both damage models. Therefore, three different loading scenarios will be introduced. As a starting point for all loading paths $(\sigma_1^{(0)}, \sigma_2^{(0)}) = (0, 0)$ is chosen. The stress increments for the three cases read as follows:

1.

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



(a) Tension-only damage model with $\nu = 0.0$ (Poisson's ratio)



(b) Non-symmetric damage model with $n = 1$ (compression-tension strength ratio)

Figure 2: Load case 1: Damage surface (upper) and $\varepsilon_1 - \sigma_1$-plot (lower) for the non-symmetric damage models

2.

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

3.

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

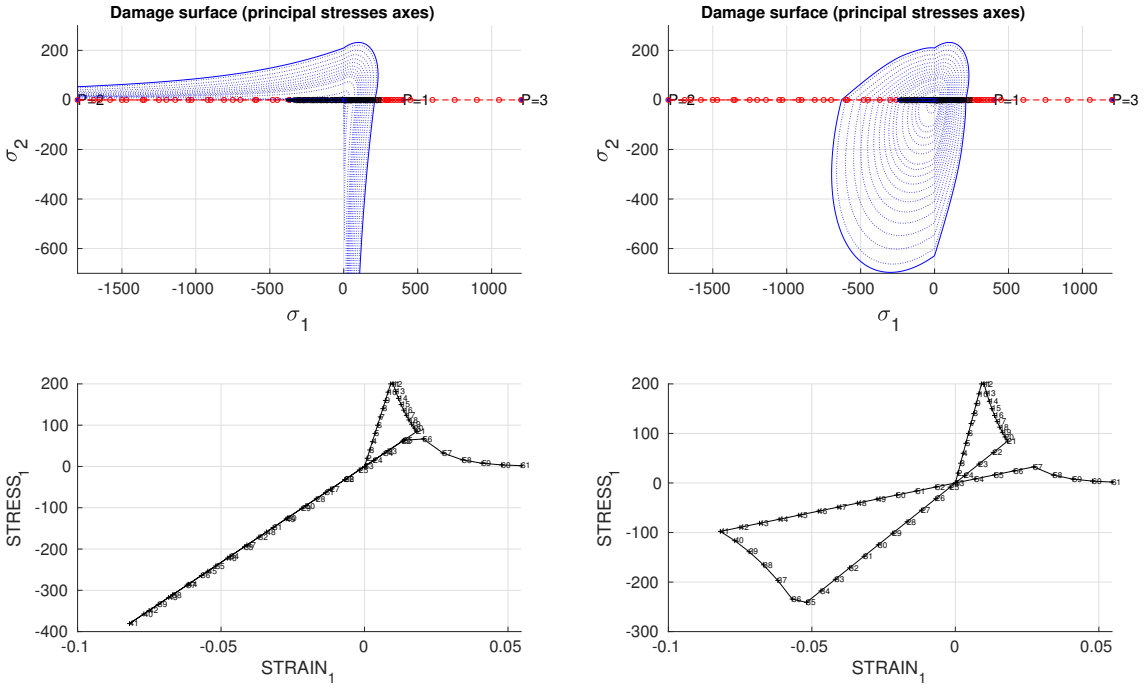
where the parameters are defined as:

$$\begin{aligned}
\alpha &= 400 \\
\beta &= 2200 \\
\gamma &= 3000.
\end{aligned}$$

3.4.1 Loading scenario 1

In Figure 3 the results for the first uniaxial loading scenario are displayed. The upper plots show the damage surface (blue) in the space of principal stresses, the theoretical loading path (red) and the actual stress path (black). For the tension-only model we see the "infinite" damage surface with asymptotic behaviour ($\nu = 0.3$) of the contour lines in the range where only one principal stress is positive. On the other hand a closed damage surface can be observed for the non-symmetric model. However, compared to the symmetric model (see Figure 1) it is expanded by a factor of three ($n = 3$) in the compressive stress regime. The corresponding stress-strain progression for both models is presented in the lower plots.

Now we take a closer look at these $\varepsilon_1 - \sigma_1$-plots to see if tensile and compressive damage is induced according to the chosen theory. As expected both models show the same



(a) Tension-only damage model

(b) Non-symmetric damage model

Figure 3: Load case 1: Damage surface (upper) and $\varepsilon_1 - \sigma_1$-plot (lower) for the non-symmetric damage models

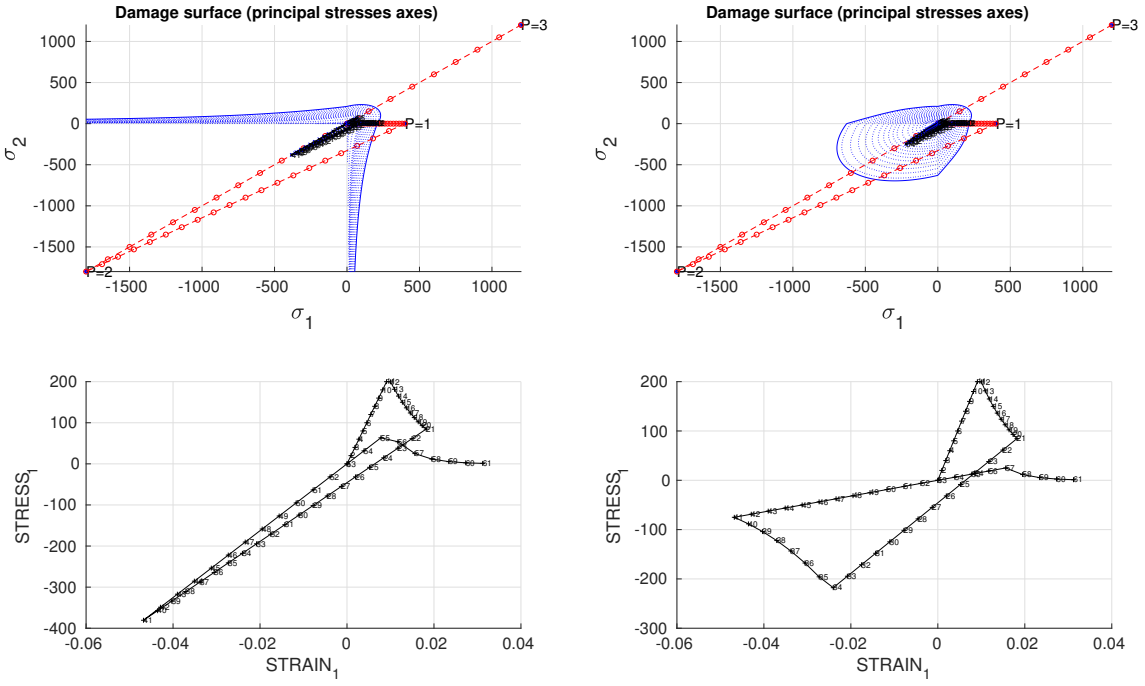
behaviour in the first tensile stress regime. After reaching the yield stress at $\sigma_y = 200$ an exponential softening until $\sigma_y^{(1)} \approx 80$ can be observed followed by linear tensile unloading. Since the tension-only model does not consider a yield stress for compression, no damage is introduced and the resulting stress-strain curve is linear and identical for compressive loading and unloading. When again tensile loading is applied the tension-only model basically continues the softening where it left off before ($\sigma_y^{(1)} \approx 80$). Only a small gap due to the interpolation between the time steps is visible. In case of the non-symmetric model exponential softening starts as the stress approaches a value of $\sigma_y^{(2)} = n * -\sigma_y^{(1)} \approx -240$. Further compressive loading leads to a reduction of the yield stress to $\sigma_y^{(3)} \approx -100$. Hence, during compressive unloading and tensile loading a flatter linear stress-strain curve occurs. This leads to a notable gap in the softening curve in the tensile loading regime because the yield stress is already reached at $\sigma_y^{(3)} = 1/n * -\sigma_y^{(2)} \approx 33$.

It is also worth mentioning that the first load case is a symmetrical loading scenario meaning that the linear $\varepsilon_1 - \sigma_1$-curves always go through the origin of the coordinate system $(\sigma_1, \sigma_2) = (0, 0)$.

3.4.2 Loading scenario 2

To continue the assessment the results of the second loading scenario shown in Figure 4 will be investigated. The first major difference with respect to the preceding load case is the unsymmetrical loading caused by a uniaxial loading followed by biaxial loading during the second stress increment. Again, both models show the same behaviour during the first

tensile loading phase. However, due to the unsymmetrical loading mentioned before the linear stress-strain curve does not cross the starting point during tensile unloading as well as compressive loading inside the linear regime. As before no damage is introduced by tension-only model during the second stress increment and therefore the stress-strain curve remains straight. The third stress increment leads to a symmetrical biaxial compressive unloading and tensile loading which means that the stress-strain curve crosses through the point of origin on the way to the tensile stress regime. It is again linear but does not follow the same path during compressive unloading and linear tensile loading as seen in the first load case. Damage occurs before the reduced stress limit from the first stress increment is reached leaving a significant distortion in the softening curve. For the non-symmetric damage model the same observations can be made with the addition, that damage is introduced during compressive loading. This results in a flatter stress-strain curve and a smaller stress limit during the third stress increment.



(a) Tension-only damage model

(b) Non-symmetric damage model

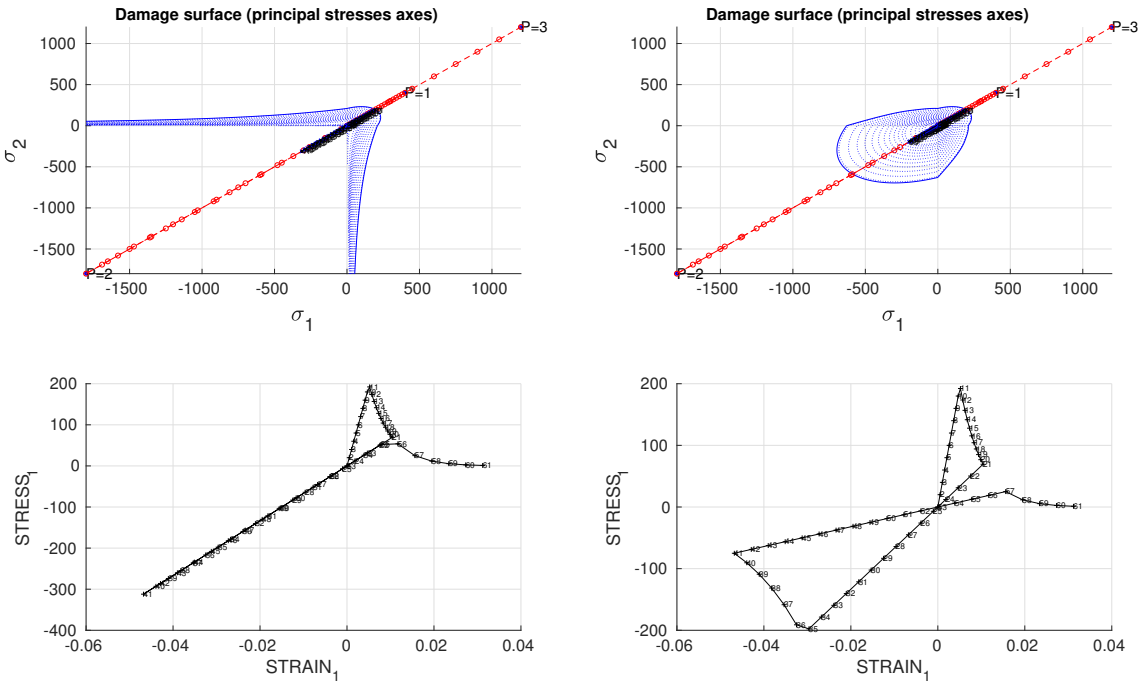
Figure 4: Load case 2: Damage surface (upper) and $\varepsilon_1 - \sigma_1$-plot (lower) for the non-symmetric damage models

One can note that the unsymmetrical loading leads to the fact that the yield stress is reached for smaller strain values when compared with the first example. Also a relationship between the tensile and compressive yield stresses as seen before can not be derived by simply looking at the plots.

3.4.3 Loading scenario 3

The third load case applies biaxial loading, though in a symmetrical manner. Therefore, the results presented in Figure 5 look very similar to the ones of the first load case (Figure 3) with uniaxial symmetrical loading. The absolute stress values obtained at the turning

points are the same. The only difference is that they occur for smaller strains which is reasonable considering that more loading is applied. Due to the symmetry the relationships between the tensile and compressive yield stress as stated for the first loading scenario are still valid.



(a) Tension-only damage model

(b) Non-symmetric damage model

Figure 5: Load case 3: Damage surface (upper) and $\varepsilon_1 - \sigma_1$-plot (lower) for the non-symmetric damage models

To summarize the assessment it can be noted that for all three load cases the presented stress-strain plots show the expected behaviour and therefore validate the correct implementation of both damage models.

4 Rate-dependent models

4.1 Isotropic visco-damage model

By introducing viscosity to the damage model the stress tensor does not only dependent on the strain, but additionally depends on time. Since the time is an independent variable an ordinary first order differential equation for calculating the internal variable r of the damage variable d has to be solved. In general an analytical solution can not always be found, meaning the use of a numerical integration scheme is necessary. For the academic example dealt with in this report, an analytic solution exists which gives us the chance to evaluate the algorithmic solution in the following sections.

4.2 Validation of the implementation

Firstly, the influence of the viscosity coefficient η and the strain rate $\dot{\varepsilon}$ on the strain-stress curve is investigated. Afterwards, different integration schemes are evaluated by comparing the analytical and algorithmic tangent operator for specific values of the integration coefficient α.

4.2.1 Influence of viscosity coefficient η

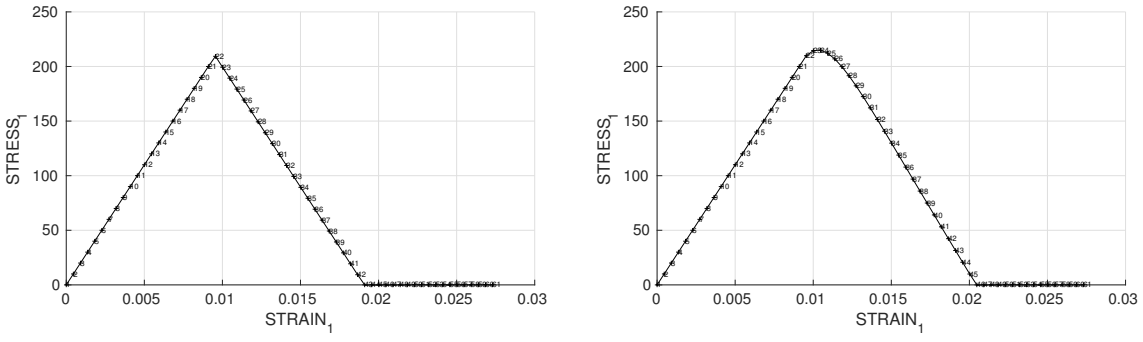
We reproduce the uniaxial tensile loading scenario from Section 3.1 to investigate the influence of a change in the viscosity coefficient η. For $\eta = 0$ (Figure 6a) the non-viscous case from Section 3.1 is retrieved. When viscosity is introduced (see Figure 6b) the linear softening does not start immediately after reaching the prescribed yield stress. Rather hardening and a higher maximum stress in form of a rounded top can be observed. Figures 6c and 6d show that with an increasing η these effects are amplified. To get an idea of the quantitative influence of the viscosity coefficient the maximum stresses reached for different values of η are presented in Table 1.

η	0.0	0.5	1.0	3.0
max σ_1	209	215	222	247

Table 1: Maximum stresses for different values of η

4.2.2 Influence of strain rate $\dot{\varepsilon}$

In order to assess the change of the strain rate the viscosity coefficient is fixed to $\eta = 0.5$. Since the strain rate $\dot{\varepsilon}$ is not a direct input parameter of the code, it has to be manipulated in a different way. From theory we know that the strain rate is indirectly proportional to the loading time. Meaning e.g. by increasing the total loading time t, the strain rate is decreased. Taking Figure 6b with a total time $t = 10$ as a reference, we can see that for a small strain rate (Figure 7a) the viscosity effects intensify. On the other hand the influence of the viscosity diminishes with smaller strain rates as highlighted in Figure 7b. This is in accordance with the fact that the non-viscous case is recovered for strain rates $\dot{\varepsilon} \to 0$.



(a) Viscosity coefficient $\eta = 0.0$

(b) Viscosity coefficient $\eta = 0.5$

(c) Viscosity coefficient $\eta = 1.0$

(d) Viscosity coefficient $\eta = 3.0$

Figure 6: Influence of viscosity coefficient η on the $\varepsilon_1 - \sigma_1$-plot of the symmetric damage model

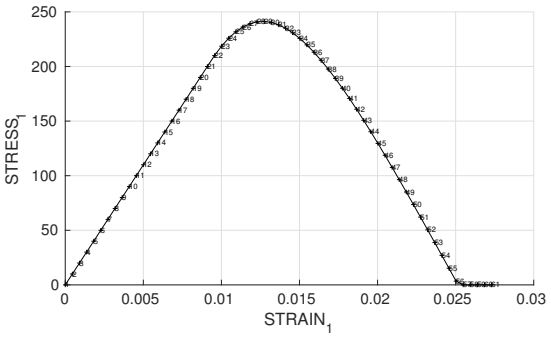
Overall it can be stated, that the strain rate $\dot{\varepsilon}$ has a similar influence on the strain-stress curve as the viscosity parameter η (see Figure 6).

4.2.3 Comparison of analytic and algorithmic tangent operator

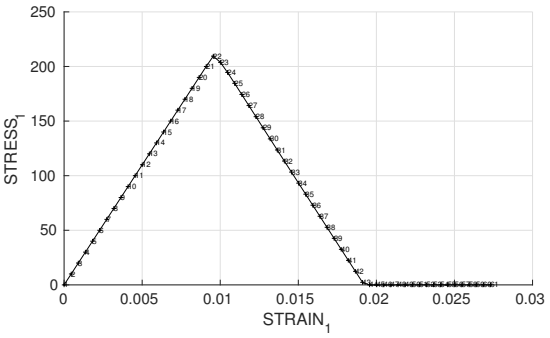
The underlying numerical integration scheme is based on the so-called alpha method. This subsection analyses the influence of the integration coefficient α on the algorithmic tangent operator. For the assessment a uniaxial loading scenario was applied starting from $(\sigma_1^{(0)}, \sigma_2^{(0)}) = (0, 0)$ with stress increments of $\Delta\bar{\sigma}_1^{(1)} = 400$, $\Delta\bar{\sigma}_1^{(2)} = -1000$, $\Delta\bar{\sigma}_1^{(3)} = 1400$.

Before continuing with the comparison the correct implementation of both tangent operators is assessed by testing specific cases where the analytical and algorithmic solution should coincide. As shown later both tangent operator are equal when $\alpha = 0$. Also when the time step size $\Delta t = 0$ the analytic solution is recovered. Since $\Delta t = 0$ can not be properly displayed either the number of time steps has to be increased or the total loading time can be reduced as presented in Figure 8a. Another example is the case where the algorithmic tangent operator coincides with the one of the rate independent model at the inviscid limit. This behaviour can be observed in Figure 8b.

Now we will investigate the change of the integration coefficient α. The graphs in Figure 9 show each the first entry of the analytic and the algorithmic tangent operator plotted against time for five values of α between 0 to 1. Figure 9a confirms the statement above

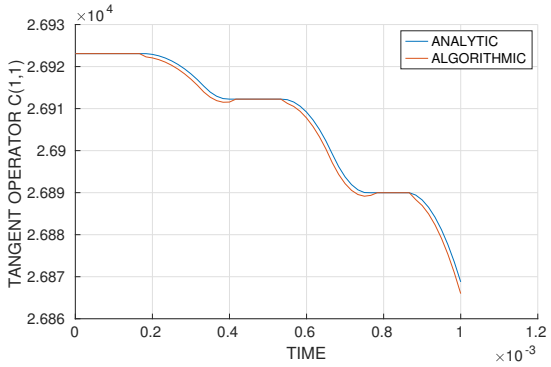


(a) Large strain rate $\dot{\varepsilon}$ for $t = 2$

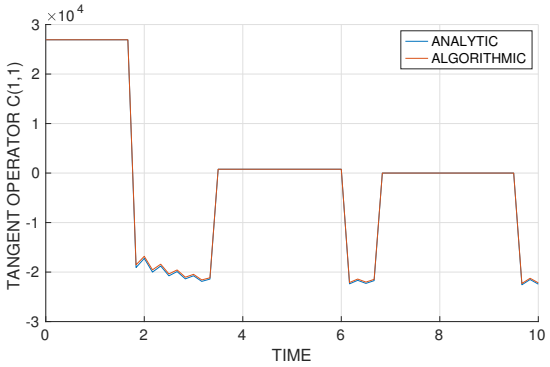


(b) Small strain rate $\dot{\varepsilon}$ for $t = 100$

Figure 7: Influence of viscosity coefficient $\dot{\varepsilon}$ on the $\varepsilon_1 - \sigma_1$-plot of the symmetric damage model



(a) Analytic and algorithmic tangent operator for $\alpha = 0.5$, $\eta = 0.5$ and total time $t = 0.001$



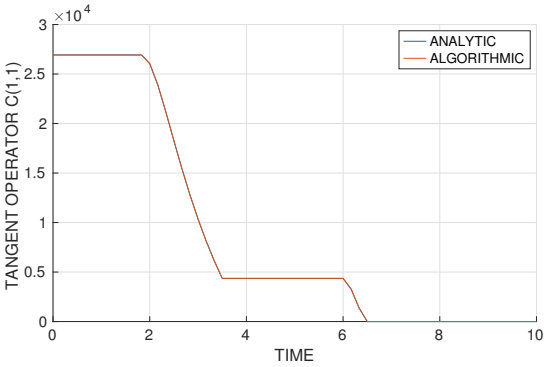
(b) Analytic and algorithmic tangent operator at the inviscid limit for $\alpha = 0.5$, $\eta = 0.001$ and total time $t = 10$

Figure 8: Influence of viscosity coefficient $\dot{\varepsilon}$ on the $\varepsilon_1 - \sigma_1$-plot of the symmetric damage model

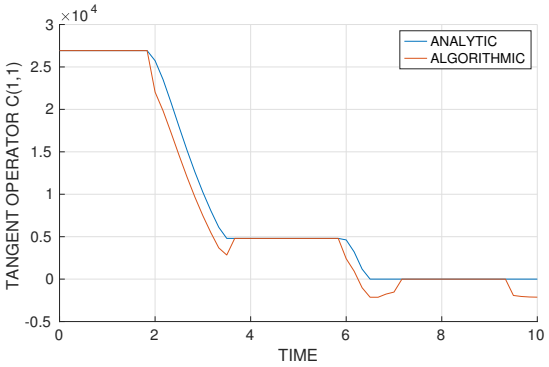
that for $\alpha = 0$ representing forward Euler explicit scheme both tangent operators coincide. Next we take a look at Figure 9b & 9c where the later with a value of $\alpha = 0.5$ represents the Crank-Nicolson integration scheme. As the value of the integration coefficient increases we can observe that during inealstic loading the value of the algorithmic tangent operator decreases faster, but corrects properly when the loading returns to the elastic regime. However, when the material is heavily damaged or even completely destroyed artificial negative values with no physical meaning occur. These phenomena are amplified for even larger α-values as visualized in Figures 9d & 9e. With $\alpha = 1.0$ the implicit forward Euler scheme is used.

Since the strain-stress plots did not show significant differences for the chosen values of α, the discussion was limited to the influence on the algorithmic tangent operator.

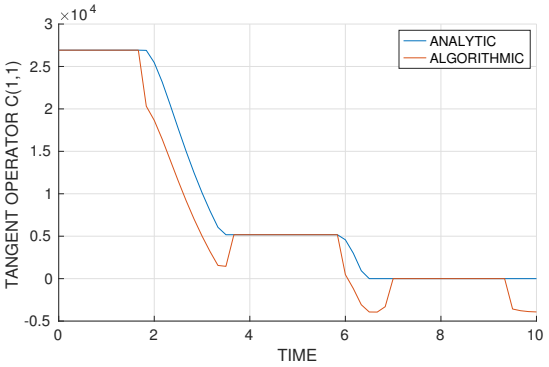
The obtained results show that in practice the choice of the integration parameter can have a great impact on the accuracy of certain output parameters and therefore, should be thoroughly investigated.



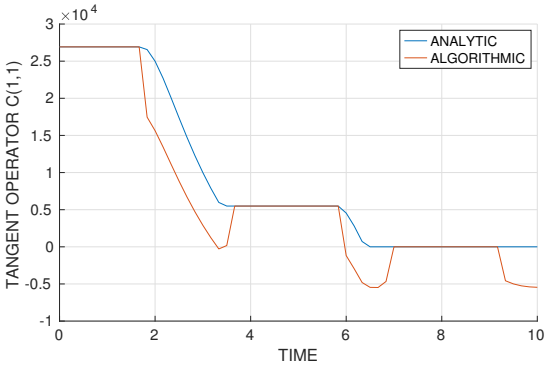
(a) Analytic and algorithmic tangent operator for $\alpha = 0.0$



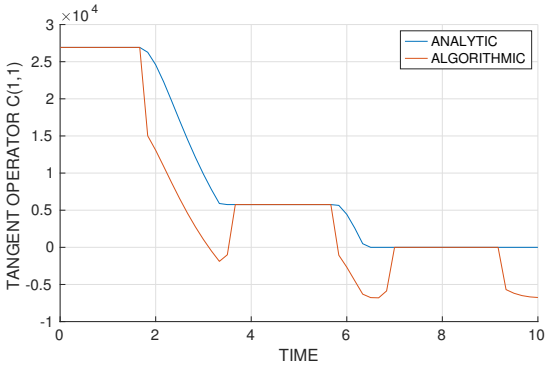
(b) Analytic and algorithmic tangent operator for $\alpha = 0.25$



(c) Analytic and algorithmic tangent operator for $\alpha = 0.5$



(d) Analytic and algorithmic tangent operator for $\alpha = 0.75$



(e) Analytic and algorithmic tangent operator for $\alpha = 1.0$

Figure 9: Influence of integration coefficient α on the algorithmic tangent operator

A Appendix

```
clc
clear all
%close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program for modelling damage model
% (Elemental gauss point level)
% ------------------
% Developed by J.A. Hdez Ortega
% 20-May-2007, Universidad Politecnica de Cataluna
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%profile on

% ----------------------------
% ****************
% INPUTS
% ****************
%addpath('AUX_SUBROUTINES')

% YOUNG's MODULUS
% ---------------
YOUNG_M = 20000 ;
% Poisson's coefficient
% ----------------------
POISSON = 0.3 ;
% Hardening/softening modulus
% ---------------------------
HARDSOFT_MOD = -1.0;
% Yield stress
% ------------
YIELD_STRESS = 200 ;
% Problem type   TP = {'PLANE STRESS','PLANE STRAIN','3D'}
% ------------------------- = 1              =2          =3
% ------------
ntype= 2 ;
% Model     PTC = {'SYMMETRIC','TENSION','NON-SYMMETRIC'} ;
%                       = 1          = 2          = 3
% ---------------------------------------------------------
MDtype =1;
% Ratio compression strength / tension strength
% ---------------------------------------------
n = 3 ;
% SOFTENING/HARDENING TYPE
% ------------------------
HARDTYPE = 'LINEAR' ; %{LINEAR,EXPONENTIAL}
% VISCOUS/INVISCID
% ----------------
VISCOUS = 'YES' ;
% Viscous coefficient -----
% -------------------------
eta = 0.5;
% TimeTotal (initial = 0) -----
% -----------------------------
TimeTotal = 0.001 ;
% Integration coefficient ALPHA
% -----------------------------
ALPHA_COEFF = 0.5;
% Points ----------------------------
% -----------------------------------
nloadstates = 3 ;
SIGMAP = zeros(nloadstates,2) ;
SIGMAP(1,:) =[400 0];
SIGMAP(2,:) =[-600 0];
SIGMAP(3,:) =[800 0];
% Number of time increments for each load state
% ---------------------------------------------
istep = 20*ones(1,nloadstates) ;

% VARIABLES TO PLOT
vpx = 'TIME' ; % AVAILABLE OPTIONS: 'STRAIN_1', 'STRAIN_2'
%                     '|STRAIN_1|', '|STRAIN_2|'
% 'norm(STRAIN)', 'TIME'
%vpy = 'analytic tangent operator (C(1,1))'
vpy = 'algorithmic_tangent_operator_(C(1,1))'% AVAILABLE OPTIONS: 'STRESS_1', 'STRESS_2'
%                     '|STRESS_1|', '|STRESS_2|'
% 'norm(STRESS)', 'TIME', 'DAMAGE VAR.','hardening variable (q)','damage variable (d)'
%  'internal variable (r)', 'analytic tangent operator (C(1,1))', 'algorithmic tangent operator (C(1,1))'

%  3) LABELPLOT{ivar}                --> Cell array with the label string for
%                                        variables of "varplot"
%
LABELPLOT = {'hardening_variable_(q)','internal_variable_(r)','damage_variable_(d)', 'analytic_tangent_operator_(C(1

%%%%%%%%%%%%%%%%%%%%%55 END INPUTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Plot Initial Damage Surface and effective stress path
strain_history = PlotIniSurf(YOUNG_M,POISSON,YIELD_STRESS,SIGMAP,ntype,MDtype,n,istep);



E         = YOUNG_M         ;
```

```
nu       = POISSON       ;
sigma_u = YIELD_STRESS ;


switch  HARDTYPE
    case 'LINEAR'
        hard_type = 0  ;
    otherwise
        hard_type = 1  ;
end
switch  VISCOUS
    case 'YES'
        viscpr = 1      ;
    otherwise
        viscpr = 0      ;
end


Eprop   = [E nu HARDSOFT_MOD sigma_u hard_type viscpr eta ALPHA_COEFF]              ;


% DAMAGE MODEL
% ------------
[sigma_v,vartoplot,LABELPLOT_out,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strain_history,MDtype,n,TimeTotal);


try; LABELPLOT;catch;LABELPLOT = LABELPLOT_out ; end ;



% PLOTTING
% --------

ncolores = 3 ;
colores =  ColoresMatrix(ncolores);
markers = MarkerMatrix(ncolores) ;
hplotLLL = [] ;

for i = 2:length(sigma_v)
    stress_eig  = sigma_v{i} ; %eigs(sigma_v{i}) ;
    tstress_eig = sigma_v{i-1}; %eigs(sigma_v{i-1}) ;
    hplotLLL(end+1) = plot([tstress_eig(1,1) stress_eig(1,1) ],[tstress_eig(2,2) stress_eig(2,2)],'LineWidth',2,'colo
    plot(stress_eig(1,1),stress_eig(2,2),'bx')
    text(stress_eig(1,1),stress_eig(2,2),num2str(i))


    % SURFACES
    % ------

end




% % SURFACES
% % ------
% 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



DATA.sigma_v     = sigma_v      ;
DATA.vartoplot   = vartoplot    ;
DATA.LABELPLOT   = LABELPLOT    ;
DATA.TIMEVECTOR = TIMEVECTOR   ;
DATA.strain = strain_history ;


plotcurvesNEW(DATA,vpx,vpy,LABELPLOT,vartoplot) ;

% fig_name = input('Save figure as ...?: ','s');
% if ~isempty(fig_name)
%     saveas(figure(2),[pwd '/fig/' fig_name],'epsc')
% end
```

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



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

elseif (MDtype==2)  %* Only tension
    s_bar = ce*eps_n1'; % [4x1] vector see Lecture 4 slide 10...
    s_bar(s_bar<0) = 0;  % all negative values set to zero -> s_bar_plus
    rtrial= sqrt(s_bar'*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
    s_bar = ce*eps_n1';
    s_bar = [s_bar(1) s_bar(2) s_bar(4)];
    s_bar_p = s_bar;
    s_bar_p(s_bar_p<0)=0;
    theta =  sum(s_bar_p)/sum(abs(s_bar));
    fact = theta + (1-theta)/n;
    rtrial= fact*sqrt(eps_n1*ce*eps_n1');
end
%*************************************************************************************
return
```

```
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------>tension ultima            %*
%*                       ntype                                                       %*
%*                                ntype=1  plane stress                              %*
%*                                ntype=2  plane strain                              %*
%*                                ntype=3  3D                                        %*
%*                       ce(4,4)    Constitutive elastic tensor  (PLANE S.      )    %*
%*                       ce(6,6)                                 ( 3D)              %*
%*************************************************************************************


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




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


    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
    % TASK 1a
    % Slides 4!!!
    % copy but change radio according to slides
    % put in an if statement for negative values
        tetha=[0:0.01:2*pi];
    %*****************************************************************************************
    %* RADIUS
    D=size(tetha);                          %*  Range
    m1=cos(tetha);                          %*
    m2=sin(tetha);                          %*
    Contador=D(1,2);                        %*
    m1_p=m1;
    m2_p=m2;
    m1_p(m1_p<0) = 0;
    m2_p(m2_p<0) = 0;

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

    for i=1:Contador
        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))]');

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

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


    % Comment/delete lines below once you have implemented this case
    % ******************************************************
    % menu({'Damage surface "ONLY-TENSION" has not been implemented yet. '; ...
    %     'Modify files "Modelos_de_dano1" and "dibujar_criterio_dano1"' ; ...
    %     'to include this option'},  ...
    %     'STOP');
    % error('OPTION NOT AVAILABLE')



elseif MDtype==3
    % TASK 1a
    % copy symmetric but change radio according to slides
    % Slides 4!!!
     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
    sigma = [m1(i) m2(i) nu*(m1(i)+m2(i))];
    s_bar_p = sigma;
    s_bar_p(s_bar_p<0)=0;
    theta =  sum(s_bar_p)/sum(abs(sigma));
    fact = theta + (1-theta)/n;
        radio(i)= q/(fact*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);

    % Comment/delete lines below once you have implemented this case
    % ******************************************************
%     menu({'Damage surface "NON-SYMMETRIC" has not been implemented yet. '; ...
%         'Modify files "Modelos_de_dano1" and "dibujar_criterio_dano1"' ; ...
%         'to include this option'},  ...
%         'STOP');
%     error('OPTION NOT AVAILABLE')

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



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

```
function [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,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) ;
eta     = Eprop(7);
alpha   = Eprop(8);
%*************************************************************************************


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


%*************************************************************************************
%*        Damage surface                                                                %*
[tau_eps_n] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
[tau_eps_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
[rtrial] = (1-alpha)*tau_eps_n+alpha*tau_eps_n1;
%*************************************************************************************

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

if(rtrial > r_n)
    %*    Loading

    fload=1;
    r_n1= ((eta-delta_t*(1-alpha))/(eta+alpha*delta_t))*r_n+(delta_t/(eta+alpha*delta_t))*rtrial
;
    delta_r=r_n1-r_n;
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        % TASK 1b
        % check slides: slidesLecture6.pdf maybe 4/5
        % exp(r?) r current or old time step

        q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0)); %QUESTION 1 r0, q_n???

        % Comment/delete lines below once you have implemented this case
        % ******************************************************
        %menu({'Hardening/Softening exponential law has not been implemented yet. '; ...
        %     'Modify file "rmap_dano1" ' ; ...
        %     'to include this option'},  ...
        %     'STOP');
        %error('OPTION NOT AVAILABLE')
    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.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')
%  Computing consistent tangent operator
%  *************************************
C_ana = (1-dano_n1)*ce;
%C_inv = C_ana - fload*1/tau_eps_n1*(q_n1-H*r_n1)/r_n1^2*(ce*eps_n1'*(ce*eps_n1'));
%Test if C_alg = C_inviscid_limit
C_alg = C_ana - fload*(alpha*delta_t)/((eta+alpha*delta_t)*tau_eps_n1)*(q_n1-H*r_n1)...
/r_n1^2*(ce*eps_n1'*(ce*eps_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;
aux_var(4) = C_ana(1,1);
%aux_var(4) = C_inv(1,1);
aux_var(5) = C_alg(1,1);

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