UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

M.Sc in Computational Mechanics

COMPUTATIONAL SOLID MECHANICS

ASSIGNMENT - 01

Numerical Integration Of Constitutive Damage Models - MATLAB

Shivaprasad Mariswamy

March, 2020

Contents

Abstract

This is an attempt build a robust damage model. Efforts are concerned with incorporating the strain rate independent and dependent phenomenon into a provided MATLAB code. Both the inviscid and viscous damage models are prepared and verified with the given specific parametric values.

1 Introduction

Materials undergo loss of stiffness due to internal pores and crack propagation. But it is too complex to resort to microscopic structure of materials in real time engineering analysis. Damage models are based on "Clausius-Duhem" inequality. They are useful for understanding the limitations of material stiffness over a loading period. In turn these models serve in engineering prediction of initiation, propagation and fracture of materials. A MATLAB integration algorithm is used for the implementation of continuum damage models.
First, you will see the tension only and non-symmetric tension, compression conditions with exponential hardening/softening law. This is an inviscid model and assessed with three different loading paths. Assessment includes the application of the model for three different combinations of uni-axial, bi-axial and compression-tension.
Secondly, continuum isotropic visco-damage "symmetric tension compression" model is described. Assessment is done with specific Poisson's ratio and linear hardening softening Hardening parameters.
Damage models are developed by many researches and used to model different materials due to simpler algorithms and computational effectiveness.

Provided MATLAB code sample consisted of following material parameters;

Young's modulus	2000
Poisson's Ratio	0.3
Yield Strength	250
Hardening/softening Coefficient	0.5

2 Inviscid Model

2.1 Implementation

Tensile only and Non-symmetric damage criteria are incorporated into the provided MATLAB program (symmetric case). Undamaged state of a material is characterized by the initial value of an internal variable (damage threshold). This can be considered as a material parameter, which defines the initial yield i strain space.
Assessment plots show that the current stress state is not crossing the damage surface. Hence, the material shows elastic behaviour during loading and unloading.

2.1.1 Uniaxial-tension-Compression-tension

In the Uniaxial load cases, damage starts when the equivalent strains exceeds the first damage threshold value.

$$\Delta\sigma_1^{(1)} = 200 \ ; \ \Delta\sigma_2^{(1)} = 0$$
$$\Delta\sigma_1^{(2)} = -500 \ ; \ \Delta\sigma_2^{(2)} = 0$$
$$\Delta\sigma_1^{(3)} = 900 \ ; \ \Delta\sigma_2^{(3)} = 0$$

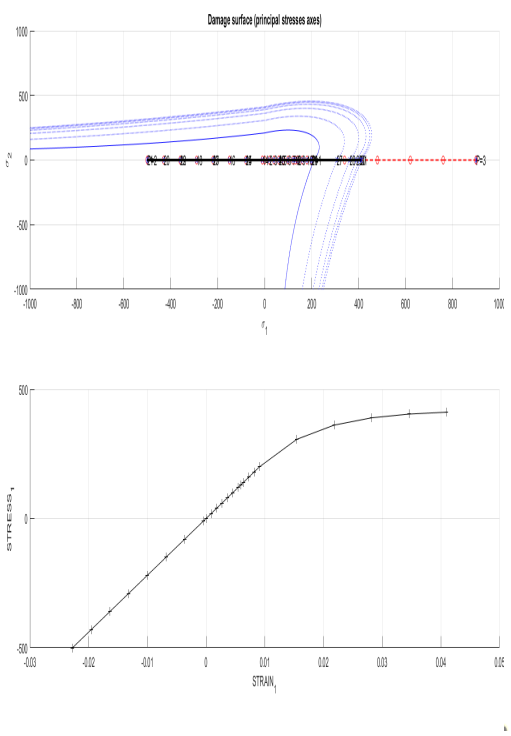

Figure 1: Tensile-compression

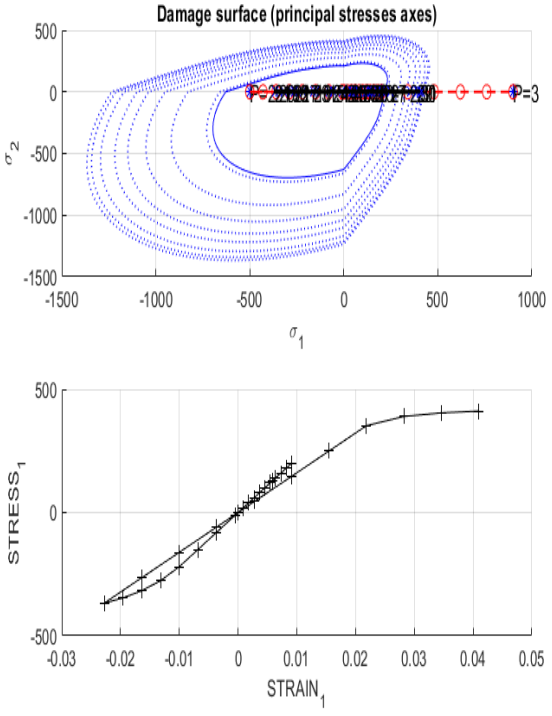

Figure 2: Tensile only

2.1.2 Biaxial tenison-compression-tension

$$\Delta\sigma_1^{(1)} = 200 \; ; \; \Delta\sigma_2^{(1)} = 200$$
$$\Delta\sigma_1^{(2)} = -500 \; ; \; \Delta\sigma_2^{(2)} = -500$$
$$\Delta\sigma_1^{(3)} = 900 \; ; \; \Delta\sigma_2^{(3)} = 900$$

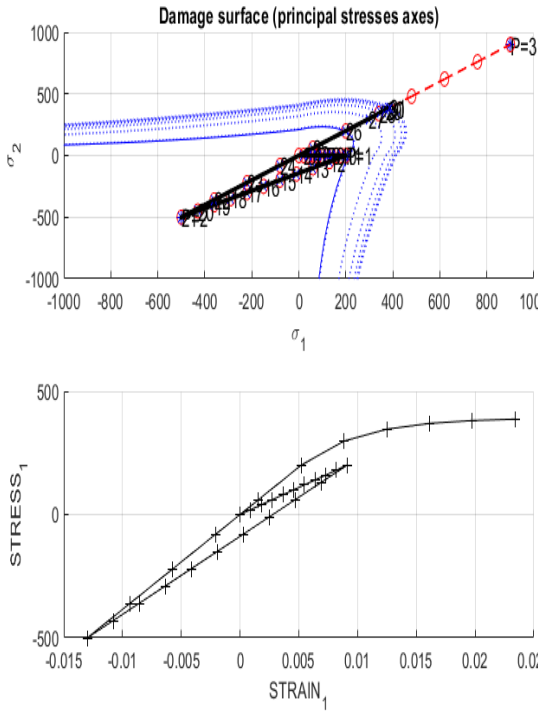

Figure 3: Tensile-compression

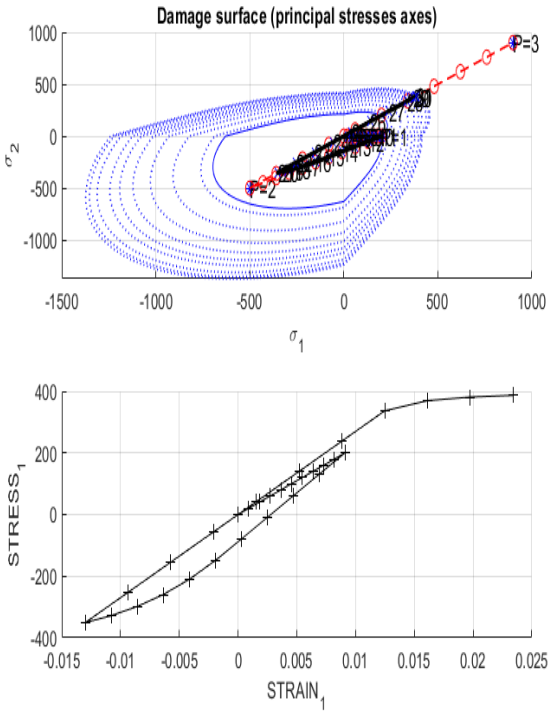

Figure 4: Tensile only

2.1.3 Uniaxial Tension, biaxial compression-tension

$$\Delta\sigma_1^{(1)} = 200 \ ; \ \Delta\sigma_2^{(1)} = 0$$

$$\Delta\sigma_1^{(2)} = -500 \ ; \ \Delta\sigma_2^{(2)} = -500$$

$$\Delta\sigma_1^{(3)} = 900 \ ; \ \Delta\sigma_2^{(3)} = 900$$

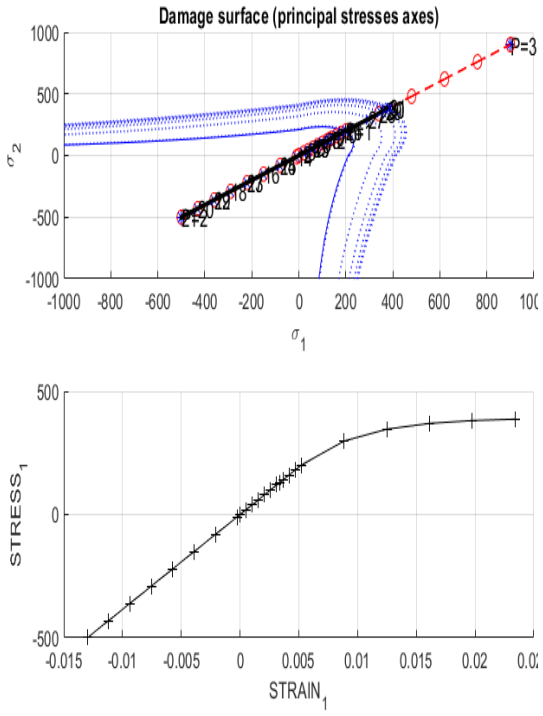

Figure 5: Tensile-compression

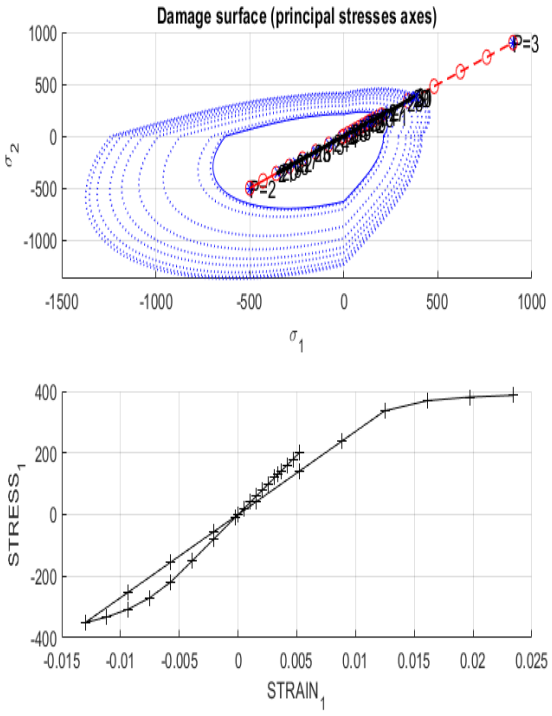

Figure 6: Tensile only

2.2 Discussion

Exponential hardening/softening law is incorporated into the linear model. The damage surface obtained for non-symmetric as well as tensile conditions behave as expected. Elastic regime is well with in the damage surface.

3 Viscous model

Viscous damage model considers the effect of viscosity, η. Uniaxial tensile loading is considered to plot the graphs and analyse the significant parameters.

3.1 Implementation

3.1.1 Specific cases

1. Viscosity values, η

   Viscosity plays a major role in extending the material towards yielding. As the viscosity increases, maximum loading strength of the material keeps reducing. In turn, viscosity induces the yielding behaviour in the material.

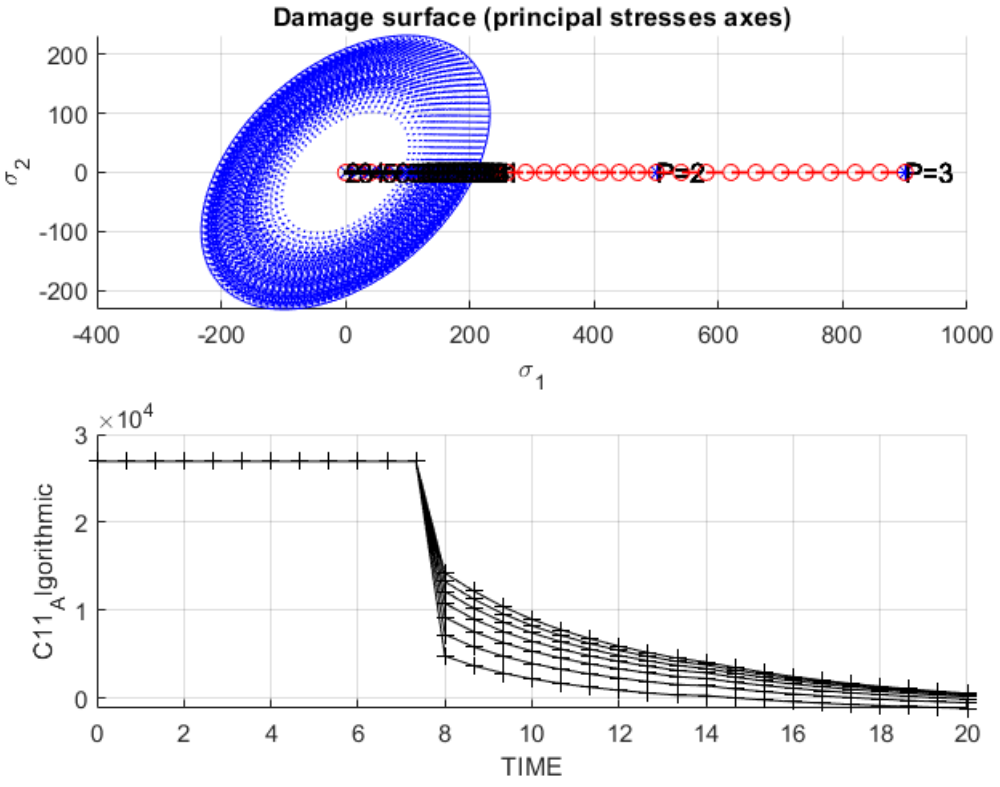

Figure 7: $\sigma - \epsilon$ curves for different values of η

2. Strain rates

   The value of viscosity is kept as a constant value of 0.5 and stress-strain curves are plotted for a various strain rates.

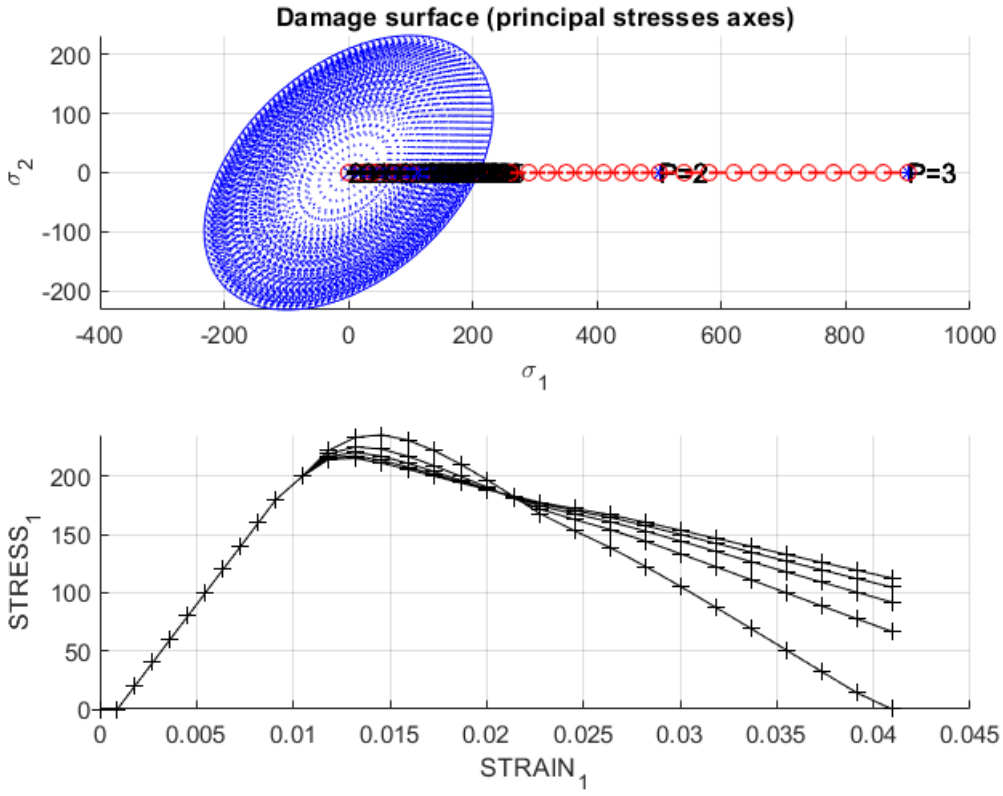

Figure 8: $\sigma - \epsilon$ curves for different values of ϵ

3. values of α

   Variation of C11 values for the values of $\alpha = 0$, 0.25, 0.5, 0.75, 1.0.

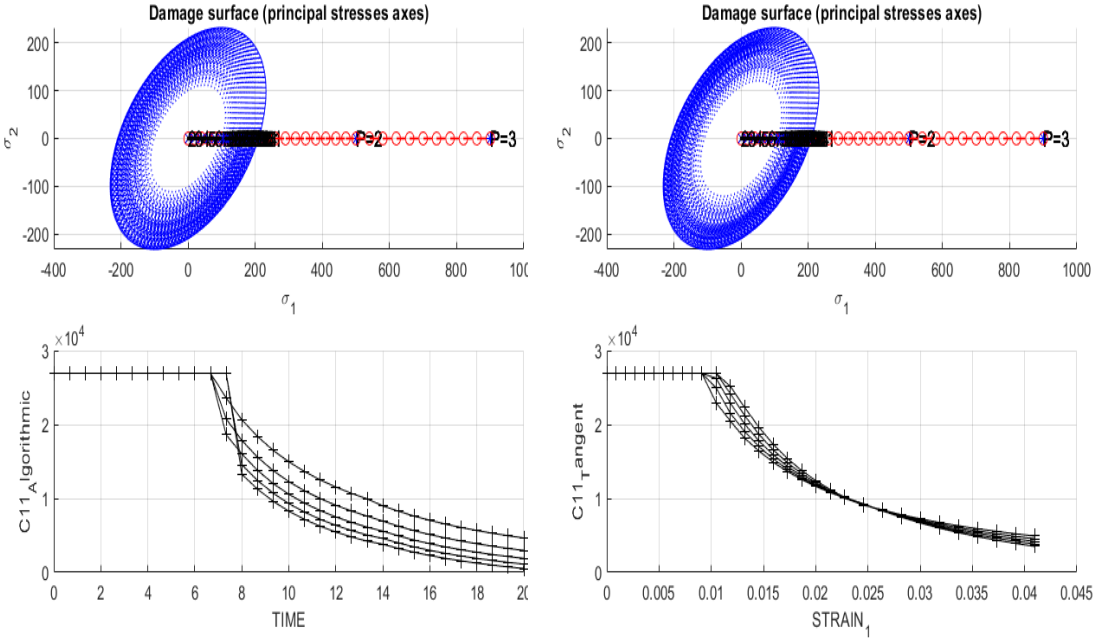

Figure 9: C11 algorithmic values for different values of α

Figure 10: C11 tangent values for different values of α

The above plots 9 and 10 show different tangent and Algorithmic components of constitutive values of C11 for different material viscosity. Algorithmic values reaches the lower most values for higher viscosity. Tangent values for a range of viscosity, will converge during the loading.

3.2 Discussion

The results are in accordance with the theoretical explanation. Hardening or softening modulus values play important role and had to be kept as a significant value in order to simulate the exponential behaviour of the curves.
It should be noted that the inviscid model is retained for a zero viscosity. Hence, it proves the correctness of the viscous damage model.

4 Summary

Overall the implementation had the maximum time effort in the inclusion of the mathematical equations and consistency loops. Following are the major observations;

1. Implementation of a robust damage algorithm requires a thorough knowledge of Constitutive and Governing equations.
2. Hardening/softening parameters have to be significantly kept high in order to get the noticeable effects.
3. This is a non-interactive MATLAB code. Hence, the users are expected to update the values manually inside the code.
4. This damage model is based on material assumption. Hence, would require further updates for other material models
5. Such damage models are capable of predicting the material failure, with out going into the microscopic description of the material. Hence, serve as an aid to Engineering analysis.

MATLAB CODE

File 01 - Non-interactive main file

```
clc
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program for modelling damage model
% (Elemental gauss point level)
% -----------------
% Developed by J.A. Hdez Ortega
% 20-May-2007, Universidad PolitÃ©cnica de CataluÃ±a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%profile on

addpath('AUX_SUBROUTINES')

% ------------------------
% ****************
% INPUTS
% ****************
% YOUNG's MODULUS
% ---------------
YOUNG_M = 20000 ;
% Poisson's coefficient
% ------------------------
POISSON = 0.3 ;
% Hardening/softening modulus
% ---------------------------
HARDSOFT_MOD = 1 ;
% 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         = 3         = 2
% ------------------------------------------------
MDtype = 3;
% Ratio compression strength / tension strength
% ---------------------------------------------
n = 3 ;
% SOFTENING/HARDENING TYPE
% ------------------------
HARDTYPE = 'EXPONENTIAL' ; %{LINEAR,EXPONENTIAL}
% VISCOUS/INVISCID
% ------------------------
VISCOUS = 'NO' ;
% Viscous coefficient ----
% ------------------------
eta = 0.3 ;
% TimeTotal (initial = 0) ----
% ------------------------
```

```
TimeTotal = 10 ; ;
% Integration coefficient ALPHA
% ------------------------
ALPHA_COEFF = 0.5 ;
% Points ---------------------------
% ---------------------------------
nloadstates = 3 ;
SIGMAP = zeros(nloadstates,2) ;
SIGMAP(1,:) =[200 0];
SIGMAP(2,:) =[-500 0];
SIGMAP(3,:) =[900 0];
% Number of time increments for each load state
% ---------------------------------------
istep = 10*ones(1,nloadstates) ;

% VARIABLES TO PLOT
vpx = 'STRAIN_1' ; % AVAILABLE OPTIONS: 'STRAIN_1', 'STRAIN_2'
%                    '|STRAIN_1|', '|STRAIN_2|'
% 'norm(STRAIN)', 'TIME'
vpy = 'STRESS_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)', 'C11_algorithmic','C11_analytical'

%  3) LABELPLOT{ivar}              --> Cell array with the label string for
%                                    variables of "varplot"
%
LABELPLOT = {'hardening variable (q)','internal variable (r)','damage variable (d)'};

%%%%%%%%%%%%%%%%%%%%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,'color',colores(1,:),'Marker',markers{1},'MarkerSi
ze',2);
    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) ;
```

File 02 - Damage main file

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,strain,MDt
ype,n,TimeTotal)
global hplotSURF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% CONTINUUM DAMAGE MODEL
% ----------------------
% Given the almansi strain evolution ("strain(totalstep,mstrain)") and a set of
% parameters and properties, it returns the evolution of the cauchy stress and other
variables
% that are listed below.
%
% INPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% ---------------------------------------------------------------
% Eprop(1) = Young's modulus  (E)
% Eprop(2) = Poisson's coefficient (nu)
% Eprop(3) = Hardening(+)/Softening(-) modulus (H)
% Eprop(4) = Yield stress (sigma_y)
% Eprop(5) = Type of Hardening/Softening law  (hard_type)
%            0 --> LINEAR
%            1 --> Exponential
% Eprop(6) = Rate behavior (viscpr)
%            0 --> Rate-independent (inviscid)
%            1 --> Rate-dependent   (viscous)
%
% Eprop(7) = Viscosity coefficient (eta)  (dummy if inviscid)
% Eprop(8) = ALPHA coefficient (for time integration), (ALPHA)
%             0<=ALPHA<=1 , ALPHA = 1.0 --> Implicit
%                           ALPHA = 0.0 --> Explicit
%            (dummy if inviscid)
%
% ntype    = PROBLEM TYPE
%            1 : plane stress
%            2 : plane strain
%            3 : 3D
%
% istep = steps for each load state (istep1,istep2,istep3)
%
% strain(i,j) = j-th component of the linearized strain vector at the i-th
%               step, i = 1:totalstep+1
%
% MDtype      = Damage surface criterion %
%            1 : SYMMETRIC
%            2 : ONLY-TENSION
```

```
%             3 : NON-SYMMETRIC
%
%
% n          = Ratio compression/tension strength (dummy if MDtype is different from
3)
%
% TimeTotal  = Interval length
%
%  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%  ------------------------------------------------------------------
%  1) sigma_v{itime}(icomp,jcomp)  --> Component (icomp,jcomp) of the cauchy
%                                   stress tensor at step "itime"
%                                   REMARK: sigma_v is a type of
%                                   variable called "cell array".
%
%
%  2) vartoplot{itime}              --> Cell array containing variables one wishes
to plot
%                                    --------------------------------------
%   vartoplot{itime}(1) =   Hardening variable (q)
%   vartoplot{itime}(2) =   Internal variable (r)%

%
%  3) LABELPLOT{ivar}              --> Cell array with the label string for
%                                    variables of "varplot"
%
%          LABELPLOT{1} => 'hardening variable (q)'
%          LABELPLOT{2} => 'internal variable'
%
%
%  4) TIME VECTOR  - >
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

% SET LABEL OF "vartoplot" variables  (it may be defined also outside this function)
% ----------------------------------
 LABELPLOT = {'hardening variable (q)','internal variable'};

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



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

% if viscpr == 1
%     % Comment/delete lines below once you have implemented this case
```

```
%     % ******************************************************
%     menu({'Viscous model has not been implemented yet. '; ...
%         'Modify files "damage_main.m","rmap_dano1" ' ; ...
%         'to include this option'},  ...
%         'STOP');
%     error('OPTION NOT AVAILABLE')
% else
% end


totalstep = sum(istep) ;


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


% Elastic constitutive tensor
% ---------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
eps_n1  = zeros(mstrain,1);
% Historic variables
% hvar_n(1:4) --> empty
% hvar_n(5) = q --> Hardening variable
% hvar_n(6) = r --> Internal variable
hvar_n  = zeros(mhist,1)  ;

% INITIALIZING  (i = 1) !!!!
% ***********i*
i = 1 ;
r0 = sigma_u/sqrt(E);
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_n
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 = 3 ;
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)

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

```
        % ------------------------
        eps_n1 = strain(i,:) ;
        eps_n = strain(i-1,:) ;

%*************************************************************************
******
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n);
%invicid

        %[sigma_n1,hvar_n,aux_var,C11_anl,C11_alg] = rmap_dano2
        %(eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t) %viscous

        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6), 'r:',MDtype,n
);
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1)
;
        end

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

        %C11_algorithmic(i) = C11_alg(1,1)
        %C11_analytical(i) = C11_anl(1,1)



        % 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) = C11_anl(1,1) ; %plotting for C11
        %vartoplot{i}(5) = C11_alg(1,1) ;  %plotting for C11
    end
end
```

File 03 - Damage criterias

```
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----->tensin ltima
%*
%*                      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
    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) ;
%*************************************************************************
******

if MDtype==1

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

        s1(i)=radio(i)*m1(i);
        s2(i)=radio(i)*m2(i);
    end
    hplot =plot(s1,s2,tipo_linea);



elseif MDtype==3
    for i=1:Contador

t=(max(m1(i),0)+max(m2(i),0)+max(nu*(m1(i)+m2(i)),0))/(abs(m1(i))+abs(m2(i))+abs(
nu*(m1(i)+m2(i))));

        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))]')*(t+(1-t)/n));

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

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

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



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

File 04 - Damage models

```
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)  %*Non-symmetric
    s=ce*eps_n1';

t=(max(s(1),0)+max(s(2),0)+max(s(3),0)+max(s(4),0))/(abs(s(1))+abs(s(2))+abs(s(3)
)+abs(s(4)));
    u=(t+(1-t)/n);
    rtrial= sqrt(eps_n1*ce*eps_n1')*u;
```

```
elseif (MDtype==3)  %* Only tension
    rtrial= sqrt(eps_n1*ce*eps_n1');
    rtrial(rtrial < 0) = 0;
end
%*************************************************************************
******
return
```

File 05 - Inviscid case - rmap_dano_1

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

%*************************************************************************
******
%*                                          *
%*           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) ;
%*************************************************************************
*****


%*************************************************************************
*****
%*       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
%*
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
%*************************************************************************
*****


%*************************************************************************
*****
%*   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;
    delta_r=rtrial-r_n;
    r_n1= rtrial  ;
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        % Comment/delete lines below once you have implemented this case
        % ******************************************************
        %H=A*(q_inf-r0)*exp(A*(1-r_n1/r0))/r0;
        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 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')

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


%*************************************************************************
*****
%* 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;
```

File 06 - Viscous case - rmap_dano2

```
function [sigma_n1,hvar_n1,aux_var,C11_an1,C11_alg] =
rmap_dano2(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_dano2
(eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n,delta_t)*
%*
*
%* INPUTS              eps_n1(4)   strain (almansi)    step n+1
*
%*                                 vector R4    (exx eyy exy ezz)
*
%*                     hvar_n(6)   internal variables , step n
*
%*                                 hvar_n(1:4) (empty)
*
%*                                 hvar_n(5) = r  ; hvar_n(6)=q
*
%*                     Eprop(:)    Material parameters
*
%*
%*                     ce(4,4)     Constitutive elastic tensor
*
%*
*
%* OUTPUTS:            sigma_n1(4) Cauchy stress  , step n+1
*
%*                     hvar_n(6)   Internal variables , step n+1
*
%*                     aux_var(3)  Auxiliar variables for computing const. tangent
tensor  *
%*************************************************************************
*******


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


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


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

```
*****
%*       initializing                                                    %*
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
 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
%*
%[rtrial] = (1-alpha)*Modelos_de_dano1 (MDtype,ce,eps_n,n)+alpha*Modelos_de_dano1
(MDtype,ce,eps_n1,n);
[rtrialexp] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
[rtrialimp] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
%*************************************************************************
*****


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

rtrial = (1-alpha)*rtrialexp + alpha*rtrialimp;
if(rtrial > r_n)
    %*   Loading

    fload=1;
    r_n1=
(eta-delta_t*(1-alpha))*r_n/(eta+alpha*delta_t)+delta_t*rtrial/(eta+alpha*delta_t
)  ;
    delta_r=r_n1-r_n;
    if hard_type == 0
        %  Linear
        q_n1= min(q_n+ H*delta_r , q_inf);
    else
        %  Exponential
        q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
        H=A*(q_inf-r0)*exp(A*(1-r_n1/r0))/r0;

    end

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

```
    end
    s_n1  =ce*eps_n1';
    ce_n1=s_n1*s_n1';

c11_n1=ce(1,1)*q_n1/r_n1-alpha*delta_t*(H*r_n1-q_n1)/((eta+alpha*delta_t)*rtrial*
r_n1^2)*ce_n1(1,1);

else

    %*     Elastic load/unload
    fload=0;
    r_n1= r_n  ;
    q_n1= q_n  ;
    c11_n1=ce(1,1)*q_n1/r_n1;

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')

C11_anl = (1.d0-dano_n1)*ce;
C11_alg = c11_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) = c11_n1;
%*********************************************************************************
*****
```