Universitat Politecnica De Catalunya

Computational Solid Mechanics

Rate Dependent and Independent Models

Prakhar Rastogi

March, 2020

Contents

1 Rate Independent Models

In this section, we are required to plot various graphs for rate independent and plane stress case considering different loading conditions such as linear/exponential loading, hardening/softening and symmetric/tension-only/non-symmetric models. But, all the loading conditions undertaken are for in-viscid case.

1.1 Non-symmetric Tension-Compression Model Vs Tension-only model

The non-symmetric tension-compression and tension-only models are implemented by considering linear loading along load path 1 (Load path are mentioned in section 1.3). For non-symmetric model stresses are calculated as:

$$\tau_\sigma = [\theta + \frac{1-\theta}{n}\sqrt{\sigma : C^{-1} : \sigma}]$$

$$\theta = \frac{\sum_{i\ =1}^{3} < \sigma_i >}{\sum_{i\ =1}^{3} |\sigma_i|}$$

For tension-only model :

$$\tau_\sigma^+ = \sqrt{\sigma^+ : \epsilon}$$

From figure, it can be concluded that damage surface is finite for tensile loading but achieves infinity under compression,i.e., that material is infinitely elastic.

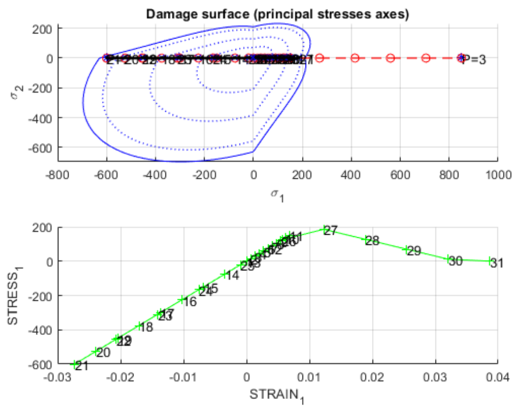

Figure 1: Non-sym. model(H=-0.4)

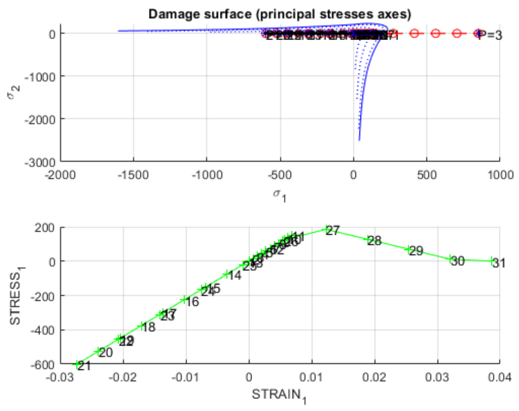

Figure 2: Tension-only model (H=-0.4)

1.2 Linear and Exponential loading

For comparing linear and exponential loading, we have considered both non-symmetric and tension-only models. The matlab file rmap_dano_1.m was modified for non-exponential case.

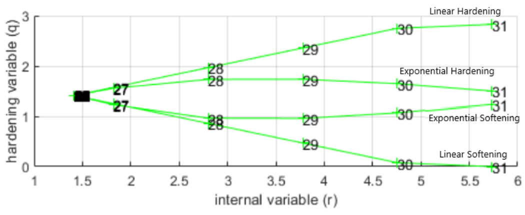

Figure 3: q-r plot (non-symmetric model)

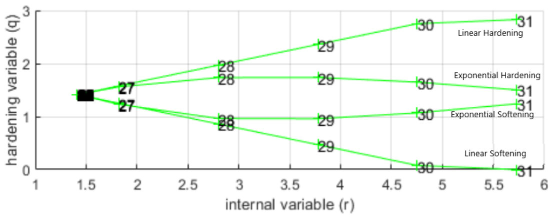

Figure 4: q-r plot (tension-only model)

1.3 Path load and other parameters

The non-symmetric and tension-only damage models will be compared according to the 3 loading paths, as specified in the question. These models will be explained on the basis of exponential hardening. Other parameters are specified below :
$E = 20000$, $\nu = 0.3$, $Y_o = 200$, $Hard/SoftModulus = 0.4$, $\eta(viscosity) = 0.3$, $\alpha = 0.5$

1.3.1 Case1: Uniaxial tensile loading, Uniaxial tensile unloading/compressive loading, Uniaxial compressive unloading/tensile loading

$\Delta\bar{\sigma}_1^{(1)}$=150 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=-700 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=850 MPa
$\Delta\bar{\sigma}_1^{(1)}$=0 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=0 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=0 MPa

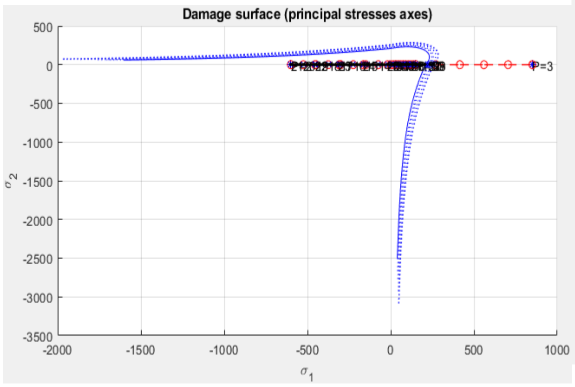

Figure 5: $\sigma_2 - \sigma_1$ plot (tension-only)

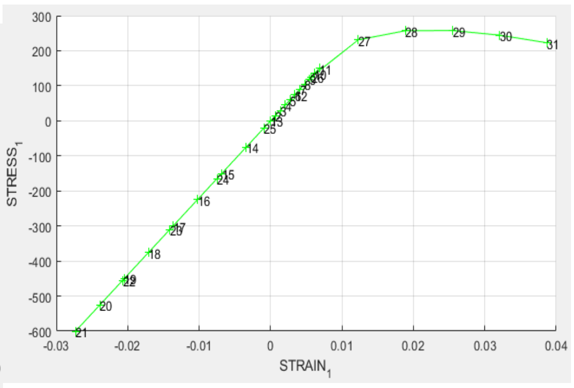

Figure 6: stress-strain plot (tension model)

The blue dotted lines (Fig. 5) are indicative of stress induced exceeding the yield stress during the hardening. It can be interpreted from stress-strain plot (Fig. 6) during the tensile loading path the stress value doesn't exceeds the yield stress value, which means loading is in elastic regime during 1st stage. In the second phase (compression), the stress value exceeds the max. point during elastic loading in opposite direction along the straight line due to specific stress limit during compression, as specified in the solution. If the value have been increased in the second stage, then also the curve wouldn't have crossed elastic region in case of tension-only model.

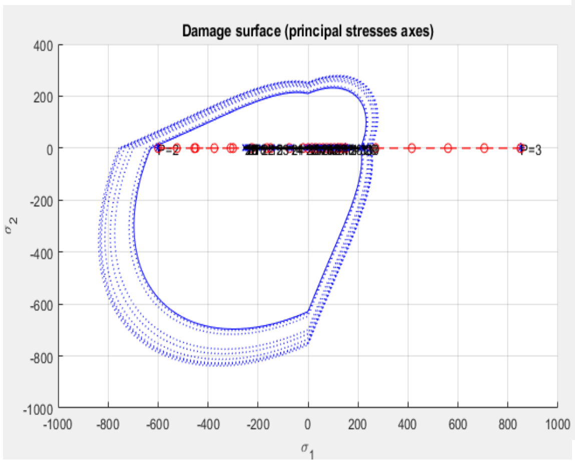

Figure 7: $\sigma_2 - \sigma_1$ plot (non-sym.)

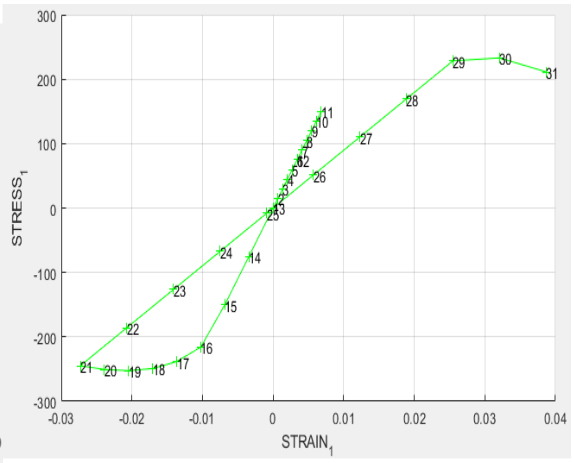

Figure 8: stress-strain plot (non-sym. model)

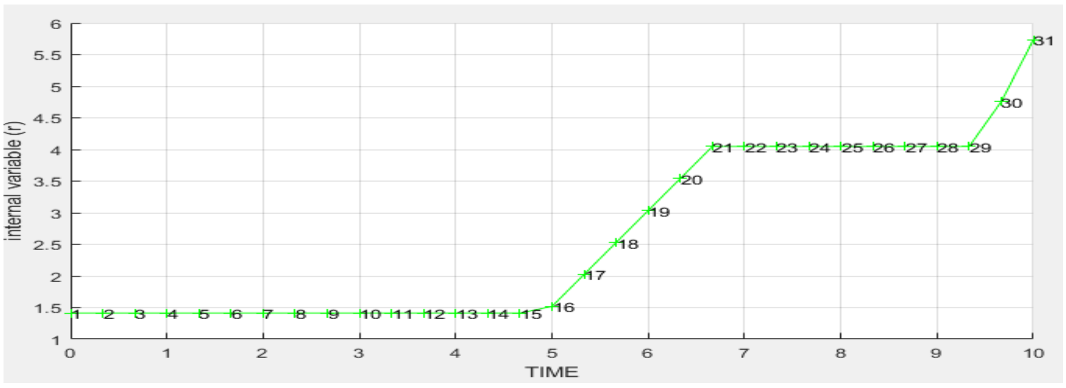

Figure 9: Internal variable (r) Vs time Plot

For non symmetric model (Fig-7), it is observed that during first and second stage of loading the point remains in elastic region. However, unlike tension-only case, the hardening can occurs during compression loading phase if the stress limit during the second stage can be increased. Therefore, partial damage in compression phase can be possible for non-symmetric case (2^{nd} stage).

From (Fig-8), it can be deduced that during 1st stage, loading is in the elastic region. However, there is curvature at the lower part during 2nd stage, indicative of increasing internal variable value (r) with time till the beginning of third stage. At the end, the curve follows a linear path during compression unloading/ tension loading crossing elastic region till point 29 (max. tensile stress value). The value of internal variable (r) increases again w.r.t. time after remaining constant from beginning of 3rd stage till point 29.

1.3.2 Case 2: Uniaxial tensile loading, Biaxial tensile unloading/compressive loading, Biaxial compressive unloading/tensile loading

$\Delta\bar{\sigma}_1^{(1)}$=150 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=-600 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=850 MPa
$\Delta\bar{\sigma}_1^{(1)}$=0 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=-600 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=850 MPa

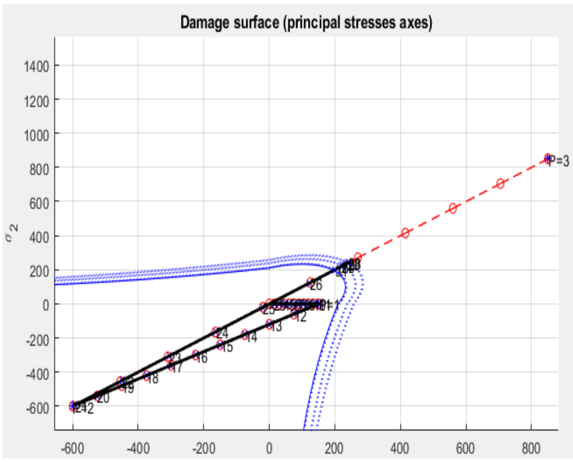

Figure 10: $\sigma_2 - \sigma_1$ plot (tension-only)

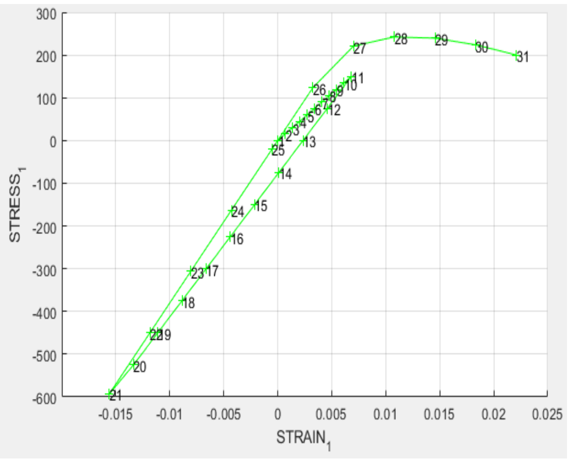

Figure 11: Stress Vs strain (tension-only)

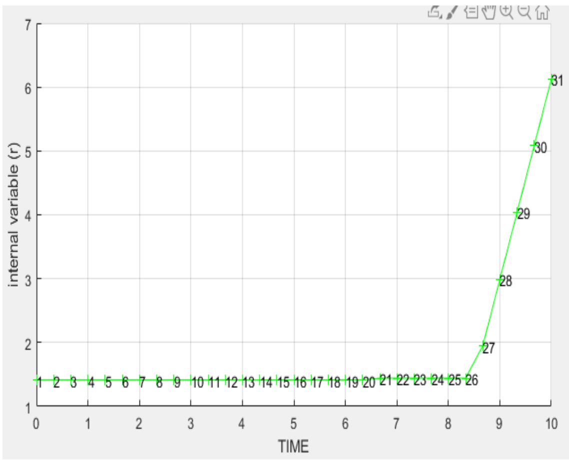

Figure 12: (r) Vs time (tension-only)

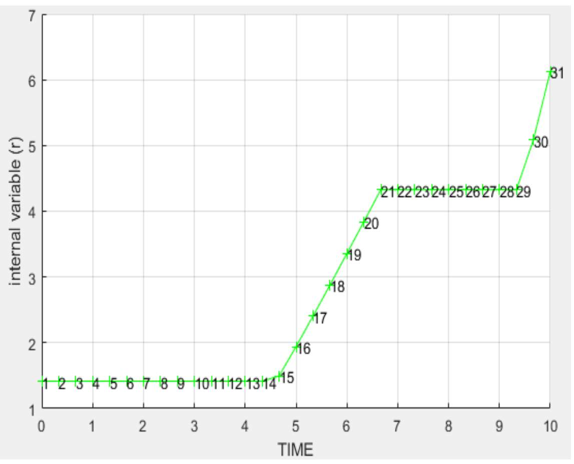

Figure 13: (r) Vs time(non-sym.)

For tension only model, it can be interpreted from the principal axes plot (Fig. 10) that the curve follows a horizontal path similar to case 1 for 1st stage tensile loading. After that, the principal stresses varies linearly in the 2^{nd} and 3^{rd} stage. From Fig. 11, it can be deduced that the curve, unlike Case 1, doesn't trace back the path during 2^{nd} stage tensile unloading, instead proceeds along a different linear path due to biaxial loading. At the 3^{rd} stage, the curve moves along a linear path similar to tension-only damage model passing the origin till pt. 26. Afterwards, the material hardening occurs and internal variable value jumps-up significantly w.r.t time(Fig. 12). Further, the curve follows a exponential path (as shown in Fig. 11(pt. 26)).

For Symmetric-damage model, the curve follows horizontal path ($\sigma_2 = 0$) similarly to tension-only model for 1st stage remaining in the elastic region and surpassing the elastic region during last stage, leading for material hardening and damage surface expansion. However, it can be indicated from Fig. 15, the existence of curvature at lower part of the curve, accounting for increase in the value of r, as shown in Fig.13 (when the curve follows curvature from pt. 15 to 21). Again, the value of r heads up after over-passing the elastic surface. It can be concluded that during the biaxial loading the graph does trace back the path during tensile unloading.

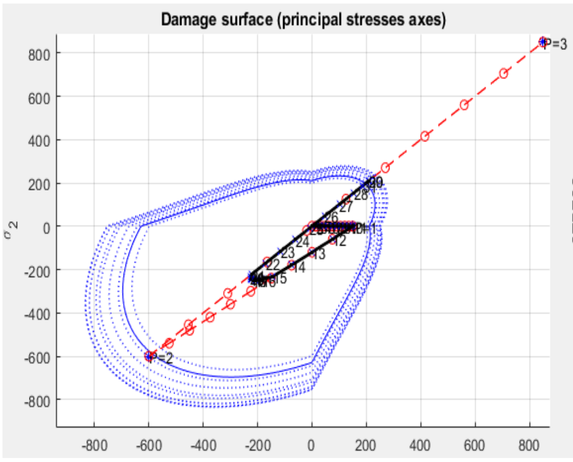

Figure 14: $\sigma_2 - \sigma_1$ plot (non-sym.)

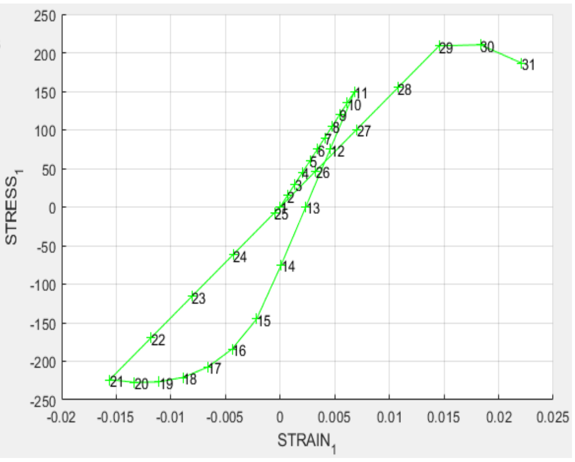

Figure 15: Stress Vs strain (non-sym.)

1.3.3 Case 2: Bi-axial tensile loading, Bi-axial tensile unloading/compressive loading, Bi-axial compressive unloading/tensile loading

$\Delta\bar{\sigma}_1^{(1)}$=300 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=-600 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=850 MPa
$\Delta\bar{\sigma}_1^{(1)}$=300 MPa ; $\Delta\bar{\sigma}_1^{(2)}$=-600 MPa ; $\Delta\bar{\sigma}_1^{(3)}$=850 MPa

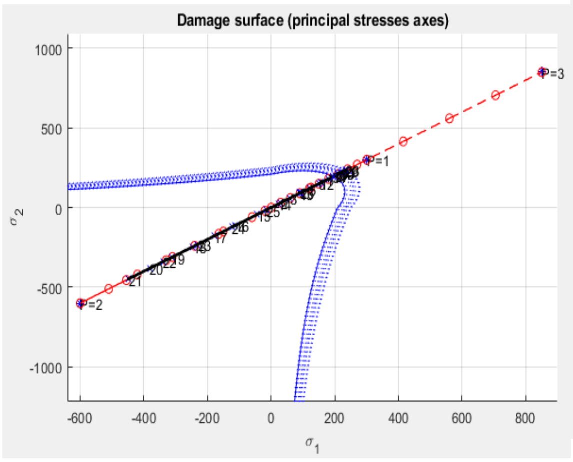

Figure 16: $\sigma_2 - \sigma_1$ plot (tension-only)

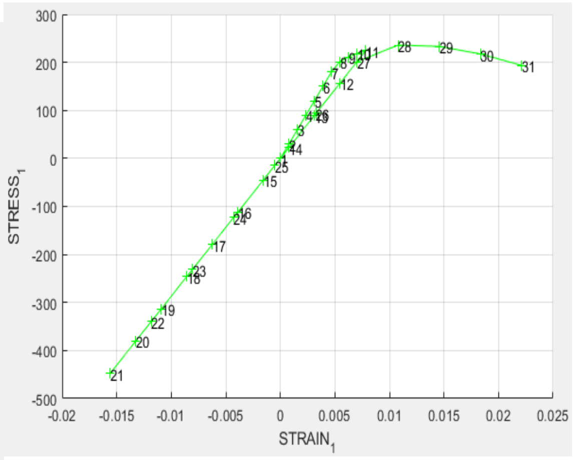

Figure 17: stress-strain plot for tension-only

For both tension and non-symmetric model, there is an increase in the value of internal variable w.r.t time in the 7-9 curve region (biaxial tensile loading region),

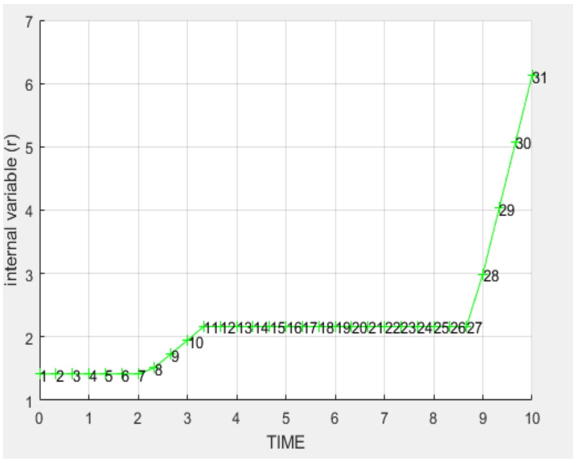
Figure 18: Int. variable-t (tension)

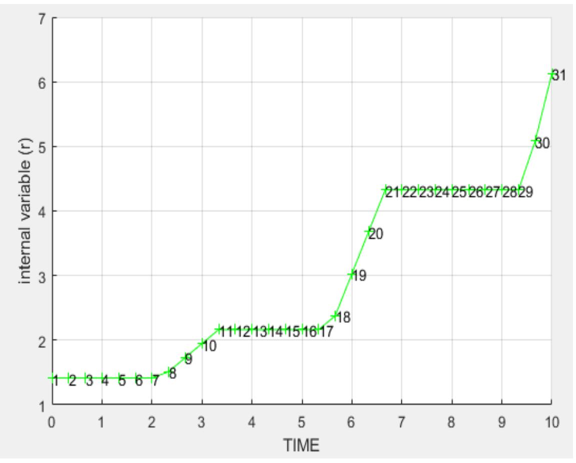
Figure 19: Int. variable(r)-time non-sym.

indicated by curve in the stress-strain graph. Afterwards, the internal variable increases in the similar manner as in case 1 and case 2 loading cases. In the tension-only model, there is a similar stress-strain curve to case1 (uniaxial) except the tensile loading region during the 1st stage. In the non-symmetric model, the graph only differs during the first phase, i.e., the biaxial tension loading.

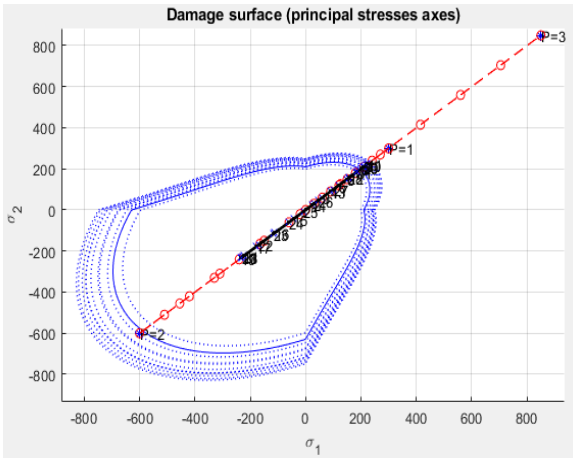

Figure 20: $\sigma_2 - \sigma_1$ plot (non-sym.)

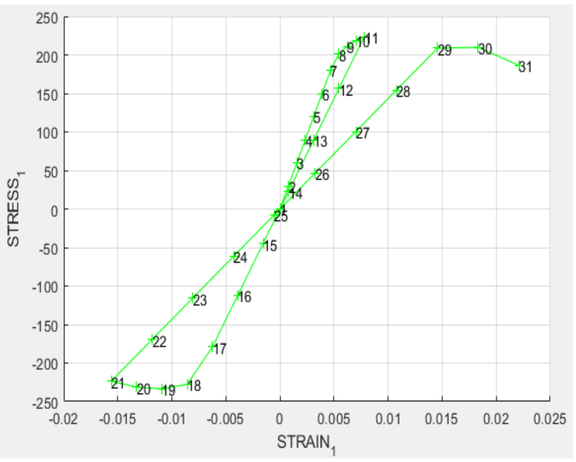
Figure 21: stress-strain (non-sym.)

2 Rate Dependent models

In the rate independent models, we need to implement the modified MATLAB code (plane straincase) for the continuum isotropic visco-damage "symmetric tension compression" model and assess the correctness the implementation for a specific Poisson ratio and linear hardening/softening parameter. The stress/strain values can be outside the elastic domain in viscous case, which signifies that the strain can be constant and stress can vary at the same time. Now, we are taking uniaxial loading for considering the effects of parameter such as viscocity, strain rate, α on the stress-strain curves.

2.1 Effects of viscosity

Black Curve- $\eta = 0.1$ ***Dark blue Curve-*** $\eta = 0.2$ ***Green Curve-*** $\eta = 0.3$ ***Light blue Curve-*** $\eta = 0.4$

The effects of different coefficient of viscosity values are discussed with the aid of graphical representation. 1-8 is the elastic region, therefore, we have a constant value of E for different viscosity. However, after pt. 8 different viscosity values follows paths. As displayed from the graph, higher viscous results in greater stress values.

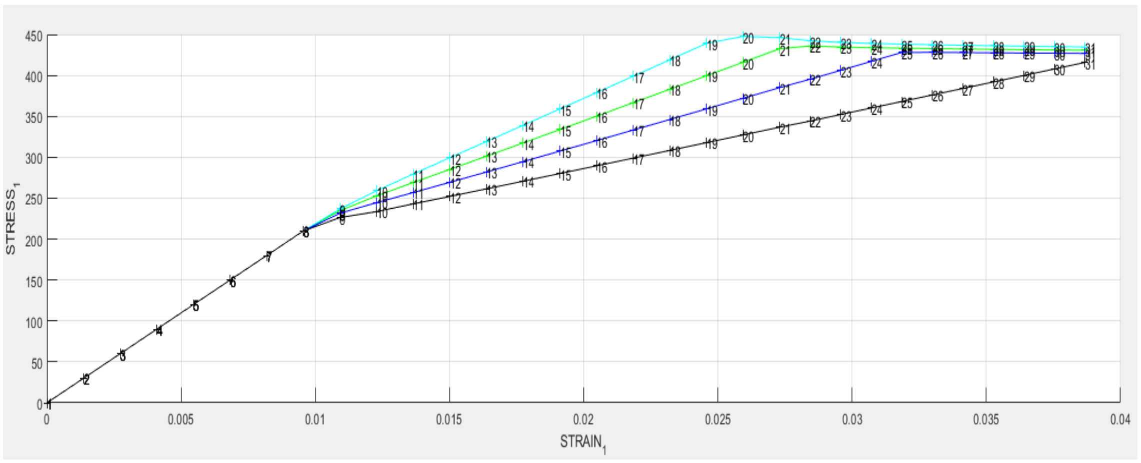

Figure 22: Stress Vs Strain plot for different values for viscosity

2.2 Effects of strain rates

Black Curve- t=2 ***Dark blue Curve- t=5*** ***Green Curve-t=10*** ***Light blue Curve- t=20***

For the consideration of strain rates effects, we have plotted stress-strain graph for different time (total taken for simulation of the results) taking uniaxial loading condition. All the curves traces the same path for the elastic region, however, it can observed that greater strain rates results in higher stress values. This can be explained if we consider 8-12 region for each value of t. For t=2, the increment in stress value is the greatest for same increase in the strain value. After that, the stress values increment follows the order: t=5, t=10 and t=20. It can concluded for less total time of simulation, there are greater strain rates, hence, higher stress values.

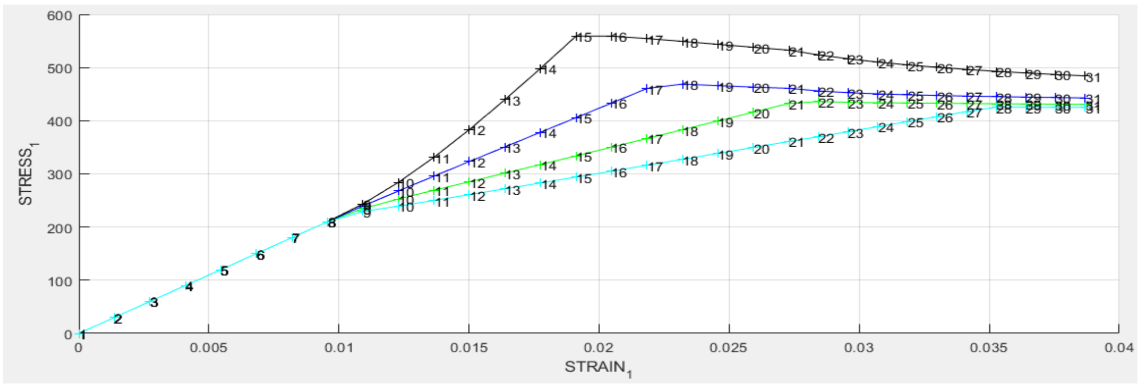

Figure 23: Stress- Strain plot for different strain rates

2.3 Effects of α integration parameters

We are plotting stress-strain plot (at t=20, uniaxial loading) for different values of α. $\alpha = 0$ (explicit Forward-Euler), $\alpha = 0.25$, $\alpha = 0{:}5$ (implicit Crank-Nicholson), $\alpha = 0.75$ (Galerkin scheme), $\alpha = 1.0$ (implicit Barckward-Euler). Of these methods, only crack-Nicholson possess 2^{nd} order accuracy. It can be observed for α=0, the stress is continuously increasing with overwhelming fluctuations, thus, signifying instability condition and erroneous solutions. However, there are no fluctuations for $\alpha = 0.5$ but continuous increment in stress with strain is observed, which violates the ideal stress-strain plot. The other values of α produces constant stress values with increasing strain, which will produce meaning results. In conclusion, if α integration parameter is less than 0.5, it will lead to instability condition. Therefore, the value of alpha should be always, $\alpha >= 0.5$.

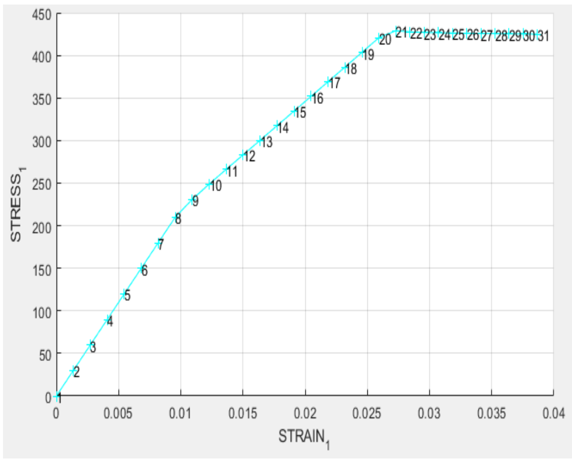

Figure 24: Stress-Strain ($\alpha = 1$)

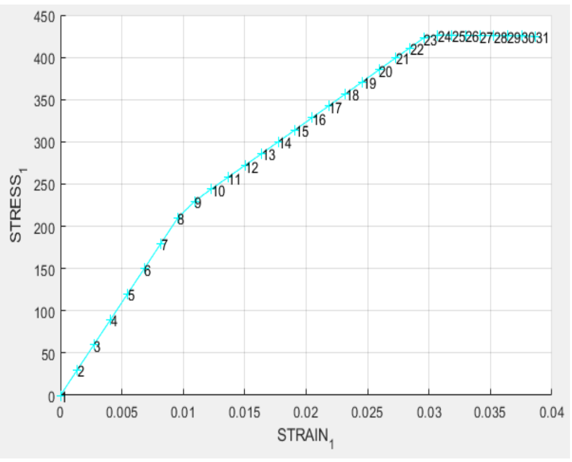

Figure 25: $\sigma - \epsilon$(Galerkin $\alpha = 0.75$)

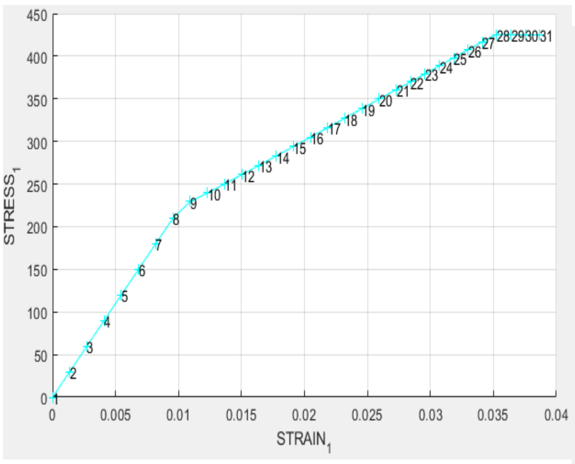

Figure 26: $\sigma - \epsilon$(Crank-Nicolson Scheme $\alpha = 0.75$)

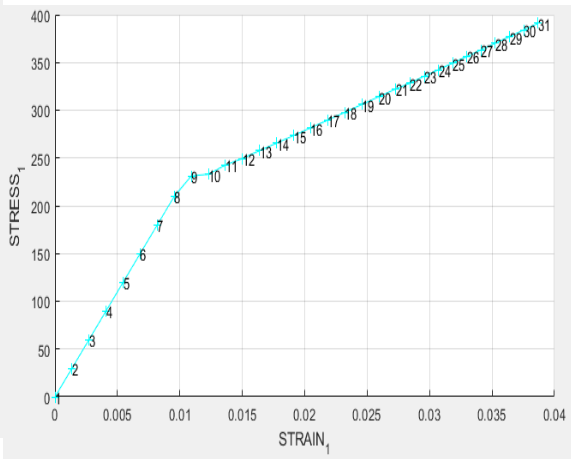

Figure 27: Stress-Strain plot ($\alpha = 0.25$)

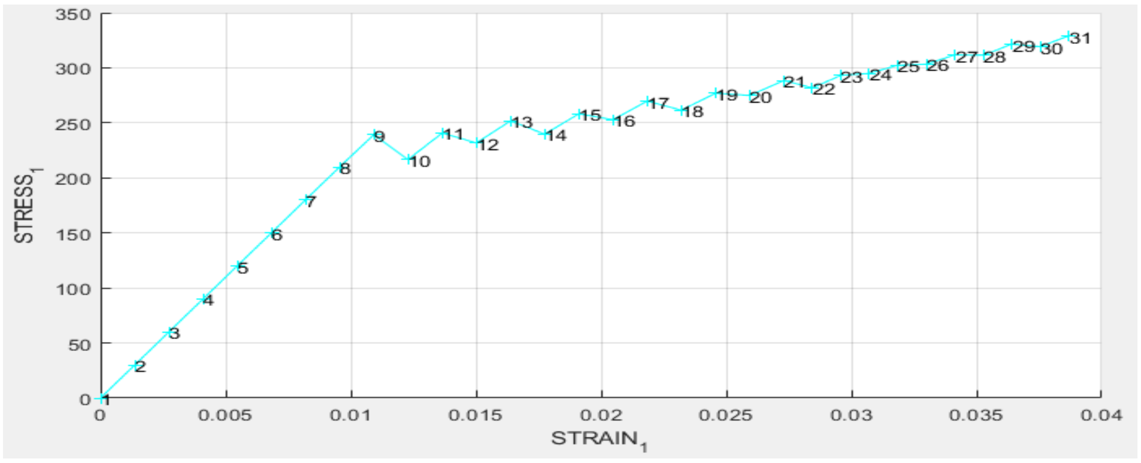

Figure 28: Stress-Strain plot (Explicit Forward-Euler$\alpha = 0.25$)

2.4 Tangent and Algorithmic Constitutive Operators

***Black Curve** ($\alpha = 1$)* ***Dark blue Curve** ($\alpha = 0.5$)* ***Green Curve** ($\alpha = 0$)*
It can be deduced from the graph that for α=0, both C algorithmic and C tangential follows the same path throughout the graph. However, the values of C algorithmic and C tangential changes for $\alpha = 0.5$ after elastic region due to occurrence of damage. Similarly, at α=1, the values of both operators changes after elastic region.

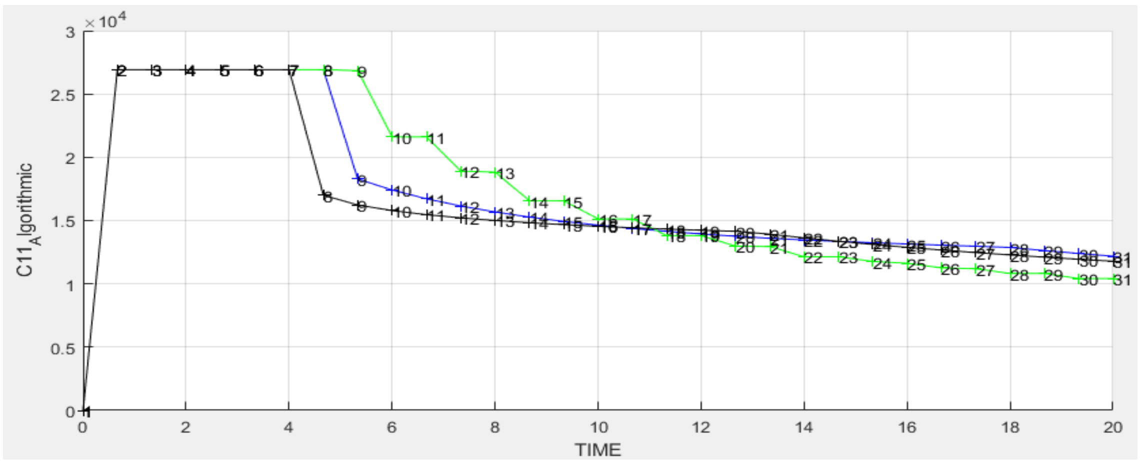

Figure 29: Algorithmic Constitutive Parameter(C-algo) Vs time

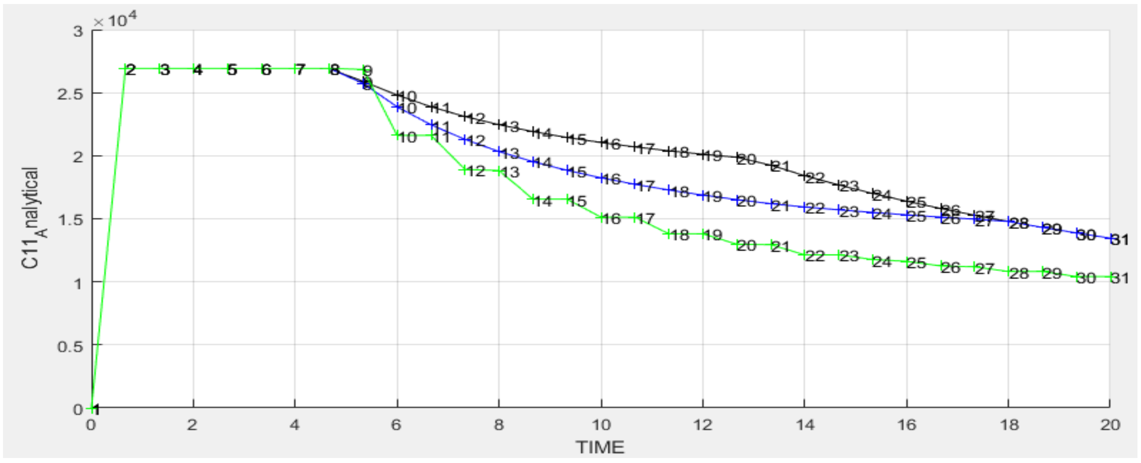

Figure 30: Tangential Constitutive Parameter(C-analytic) Vs time

A Codes

Listing 1: Function Damage Main

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=...
    damage_main(Eprop,ntype,istep,strain,MDtype,n,TimeTotal)
global hplotSURF


% SET LABEL OF "vartoplot" variables  (may be define 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

totalstep = sum(istep) ;


% INITIALIZING GLOBAL CELL ARRAYS
% -------------------------------
sigma_v = cell(totalstep+1,1) ;
TIMEVECTOR = zeros(totalstep+1,1) ;
Δ_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)+ Δ_t(iload) ;
        % Total strain at step "i"
        % −−−−−−−−−−−−−−−−−−−−−−−−
        eps_n1 = strain(i,:) ;
        %*************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%
        if viscpr == 1
            eps_n = strain(i−1,:);
            [sigma_n1,hvar_n,aux_var,C_alg, C_tan] = ...
                rmap_dano_visc(eps_n,eps_n1,Δ_t,hvar_n,Eprop,ce);
        else
            [sigma_n1,hvar_n,aux_var] = ...
                rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n);
        end
        % 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) = C_alg(1,1);
%         C11_Analytical(i) = C_tan(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) = C_alg(1,1) ;
            vartoplot{i}(5) = C_tan(1,1) ;

        end
    end
```

Listing 2: Function rmap dano

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

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;
 inf_q = 2*r0 − zero_q;
% 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

```

```matlab
    fload=1;
    Δ_r=rtrial-r_n;
    r_n1= rtrial  ;
    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*Δ_r;
    else
  % Comment/delete lines below once you have implemented this case
 % ********************************************************
        if H>0 %hardening
            dqdr = H*((inf_q - r_n)/r_n)*exp(H*(1-(r_n1/r_n)));
            q_n1 = q_n + dqdr*Δ_r;
        elseif H<0 %softening
            dqdr = H*((r_n-inf_q)/r_n)*exp(H*(1-(r_n1/r_n)));
            q_n1 = q_n - dqdr*Δ_r;
        end
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1>inf_q)
        q_n1 = inf_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;
%*******************************************************************
```

Listing 3: Function rmap dano visc

```
function [sigma_n1,hvar_n1,aux_var,C_alg,C_tan] = ...
    rmap_dano_visc (eps_n,eps_n1,Δ_t,hvar_n,Eprop,ce)
% Variables
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=2*r0-zero_q; %Define q_infinite
 % Define damage surface
 % Symmetric tension-compresion model
rtrial_n1 = sqrt(eps_n1*ce*eps_n1');
rtrial_n = sqrt(eps_n*ce*eps_n');
% Integration
rtrial_alpha = (1-alpha)*rtrial_n+alpha*rtrial_n1;
if(rtrial_alpha > r_n)
    fload=1;
    %Loading
    Δ_r=rtrial_alpha-r_n;
    r_n1= (eta-Δ_t*(1-alpha))/(eta+alpha*Δ_t)*r_n + ...
        (Δ_t/(eta+alpha*Δ_t))*rtrial_alpha ;
    if hard_type == 0
        % Linear hardening
        q_n1= q_n+ H*Δ_r;
    else
        % Exponential hardening
        dqdr = H*(q_inf - r_n)/r_n*exp(H*(1-r_n1/r_n));
       q_n1 = q_n + dqdr*Δ_r;
    end
    % Restrict value to q_infinite if needed
    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1 > q_inf)
        q_n1=q_inf;
    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';
 % Algorithmic tangent operator
if fload == 1 %(rtrial_alpha > r_n)
     sigma_barra = ce*eps_n1';
     %effective stress 93
     C_alg = (1-dano_n1)*ce +(alpha*Δ_t)/...
         ((eta+alpha*Δ_t)*rtrial_n1)*...
         ((H*r_n1-q_n1)/r_n1^2)*(sigma_barra' * sigma_barra);
     %Only linear case is considered
else
     C_alg = (1-dano_n1)*ce ;
end
 % Analytic tangent operator
C_tan = (1-dano_n1)*ce ;
% Updating historic variables
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;
% Auxiliar variables
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
```

Listing 4: Function dibujar criterio dano1

```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
%*******************************************************************
%*                 PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODE
%*                                                             %*
%*       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------>tensi[U+FFFD][U+FFFD]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
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
    % Comment/delete lines below once you have implemented this case
    % *******************************************************
     limitINF = -pi/2*0.99;
     limitSUP = pi*0.99;
     tetha=[limitINF:0.01:limitSUP];
     % 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)*(m1(i)>0) m2(i)*(m2(i)>0) 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==3
     % Comment/delete lines below once you have implemented this case
     % *******************************************************
      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
          tetha_aux = (m1(i)*(m1(i)>0) + m2(i)*(m2(i)>0))/(abs(m1( i)) ...
              + abs(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))]')/(tetha_aux + ((1 - tetha_aux)/n));
          s1(i)=radio(i)*m1(i);
          s2(i)=radio(i)*m2(i);
      end
      hplot =plot(s1,s2,tipo_linea);

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



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

Listing 5: Function Modelos de dano1

```
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
    rtrial = sqrt(eps_n1.*(eps_n1>0)*ce*eps_n1');




elseif (MDtype==3)   %*Non-symmetric
    s_n1 = ce*eps_n1';
    s1=s_n1(1);
    s2=s_n1(2);
    tetha_aux = (s1*(s1>0) + s2*(s2>0))/(abs(s1)+abs(s2));
    rtrial = (tetha_aux +(1 - tetha_aux)/n)* sqrt(eps_n1*ce*eps_n1');
end
return
```