Master Erasmus Mundus on Computational Mechanics

COMPUTATIONAL SOLID MECHANICS

ASSIGNMENT # 1

ADITYA MANGAONKAR

Content

1. Rate Independent Models

1.1 Implementing MATLAB code 1

1.2 Implement the following cases for each of those models: 2
Linear and exponential hardening/softening

1.3 Assess the correctness of the implementation 3

1.3.1 Case 1

1.3.2 Case 2

1.3.3 Case 3

2. Rate Dependent Model 9

2.1 Implement in the supplied MATLAB code the integration algorithm 9

2.2 Assess the correctness of the implementation: consider the following cases 9

2.2.1 Case 1

2.2.2 Case 2

2.2.3 Case 3

2.3 Variation of C11 component of tangent and algorithmic operator with alpha(α) 13

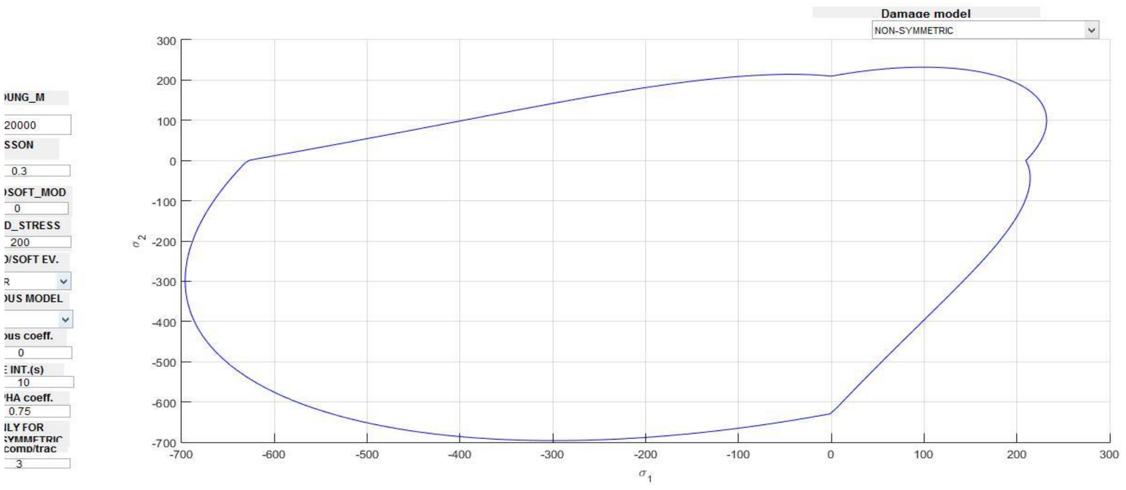
ANNEX

1.PART I (rate independent models):

1.1) Implement in the supplied MATLAB code the integration algorithms (rate independent and plane strain case) for:
1. The continuum isotropic damage “non-symmetric tension-compression damage” model.
2. The “tension-only” damage model.

Ans.
Non-symmetric tension-compression model (FIG.1) used for materials whose tension domain changes with respect to compression like concrete

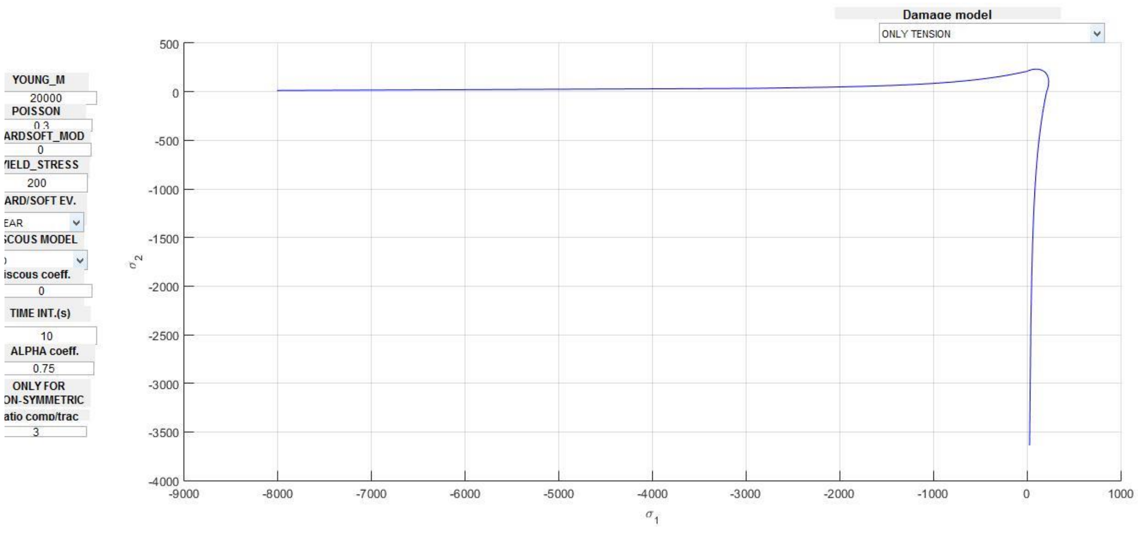
Tension only damage model (fig.2) the material which fails only under tension is considered under this model.



Damage model: NON-SYMMETRIC

)UNG_M 20000
SSON 0.3
)SOFT_MOD 0
D_STRESS 200
)/SOFT EV. R
)US MODEL
›us coeff. 0
E INT.(s) 10
'HA coeff. 0.75
ILY FOR SYMMETRIC comp/trac 3

Figure 1 NON- SYMMETRIC DAMAGE MODEL



Damage model: ONLY TENSION

YOUNG_M 20000
POISSON 0.3
ARDSOFT_MOD 0
/IELD_STRESS 200
ARD/SOFT EV. EAR
;COUS MODEL)
iscous coeff. 0
TIME INT.(s) 10
ALPHA coeff. 0.75
ONLY FOR ON-SYMMETRIC atio comp/trac 3

Figure 2 TENSION ONLY DAMAGE MODEL

1.2) Implement the following cases for each of those models:
1. linear and exponential hardening/softening (H<0 and H>0)
Ans.

The code has been implemented and the following results have been fetched from the code for both linear and exponential hardening

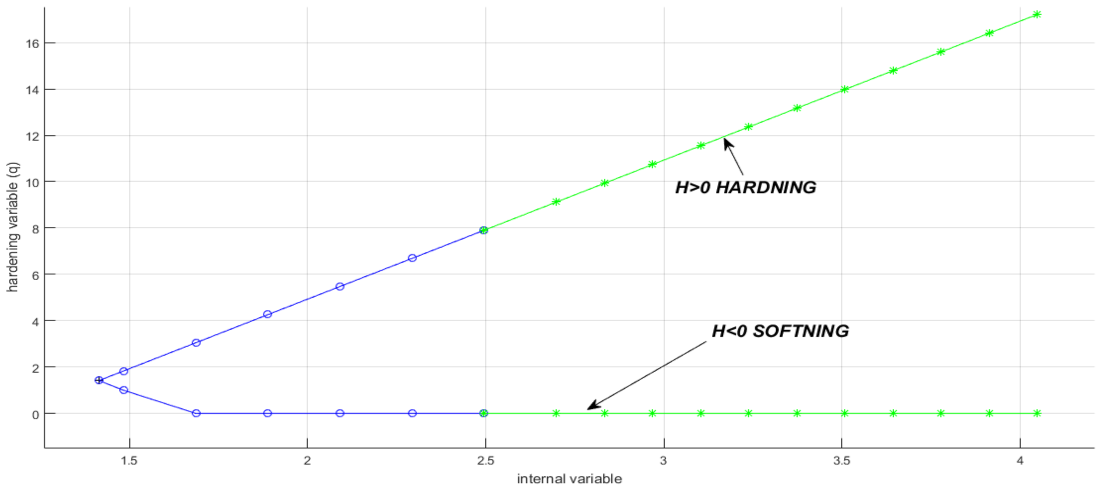


Figure 3 linear hardening and softening

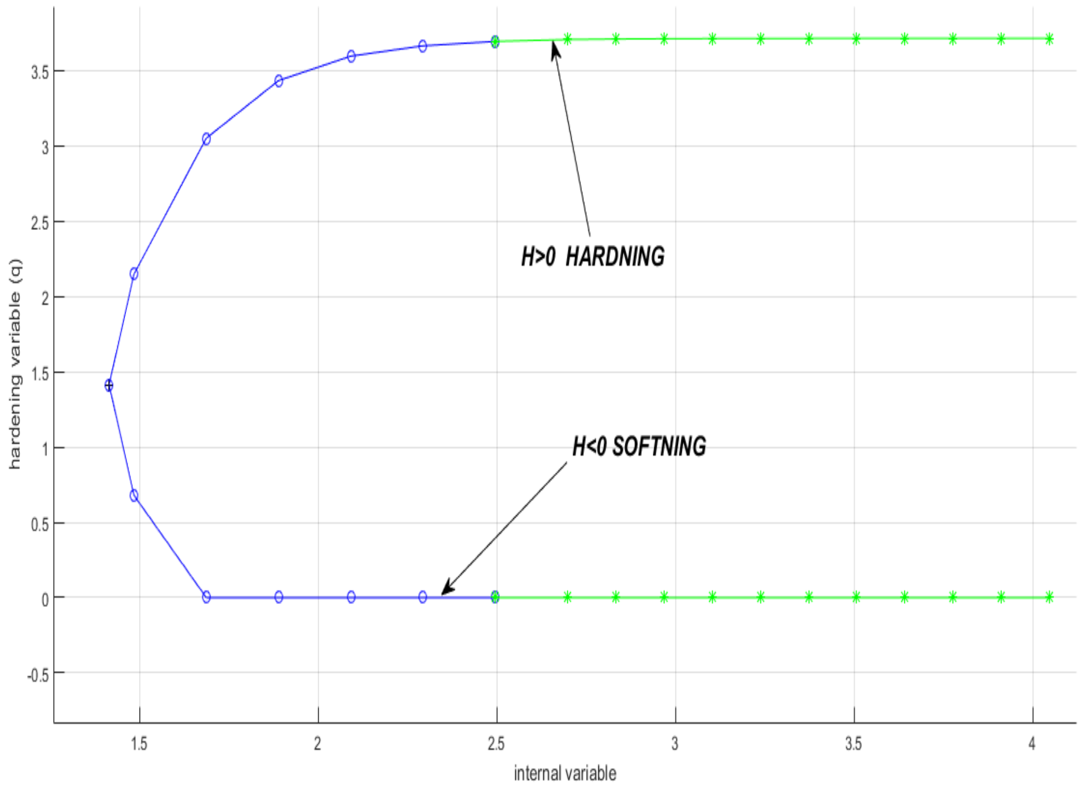


Figure 4 exponential hardening and softening

1.3) Assess the correctness of the implementation: for each of the models in section
Ans.

The outputs are taken for each case with H =(6) and (-6) with exponential and linear case

1.3.1) (i) uniaxial tensile loading {***Δσ1(1) = 300; Δσ2(1) = 0*** }

(ii) uniaxial tensile unloading/compressive loading ***{Δσ1(2) = −700; Δσ2(2) = 0}***

(iii) uniaxial compressive unloading/ tensile loading ***{Δσ1(3) = 400; Δσ2(3) = 0}***

Exponential case H=6

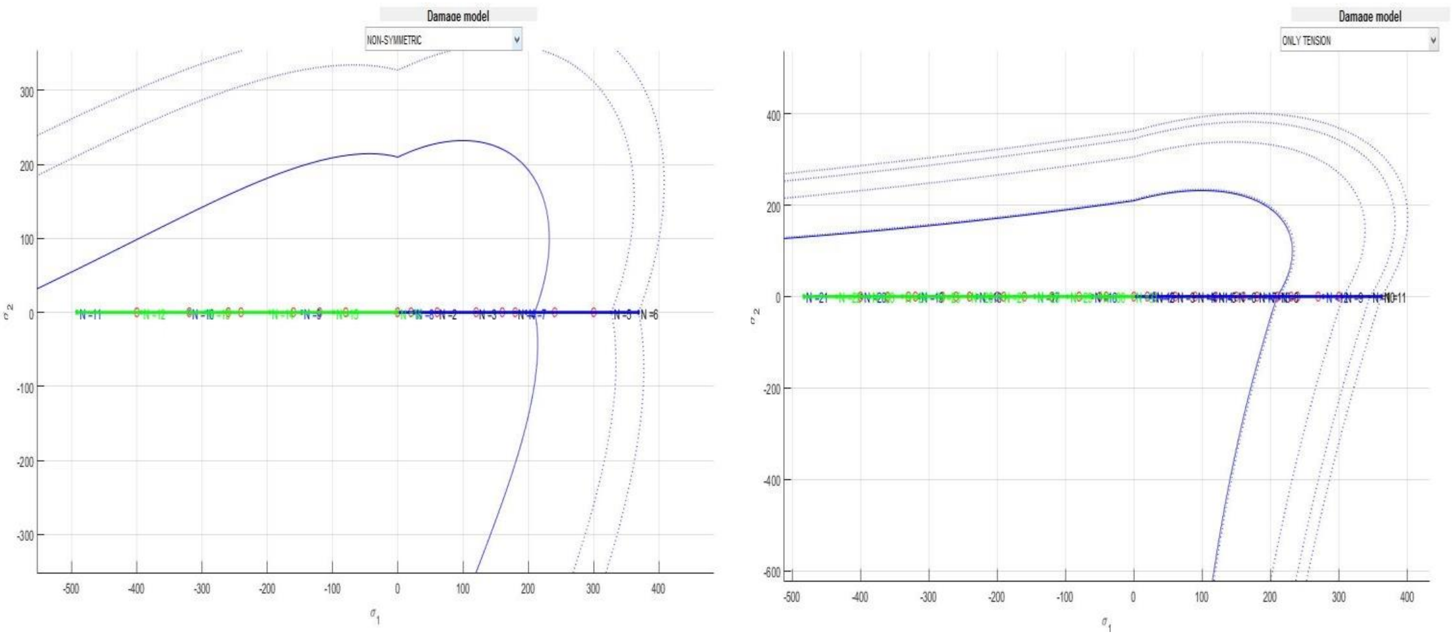


Figure 5 Exponential case Non- symmetric and tension only model behaviour

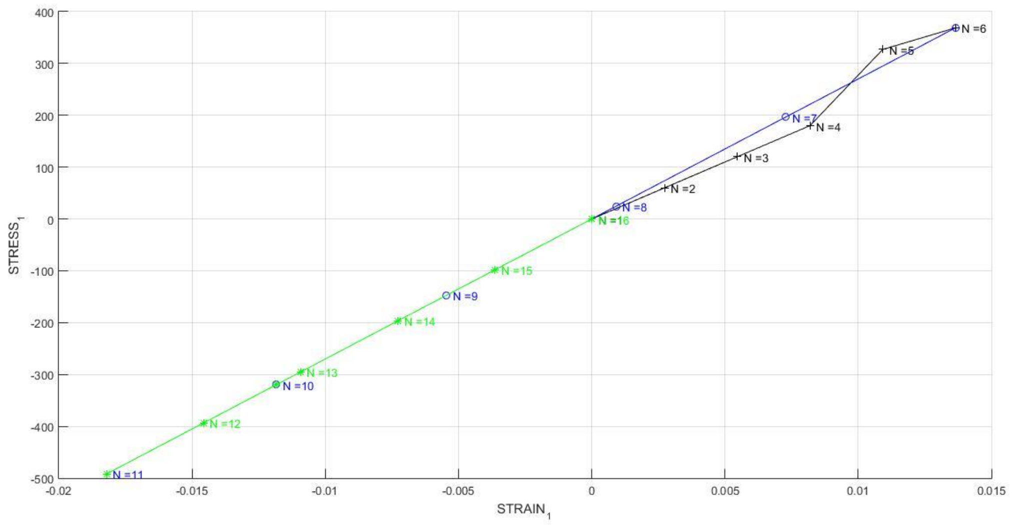


Figure 6 Stress Strain -Exponential Case Non- symmetric and tension only model behaviour

For Same H the stress stain curve gives similar behaviour. The blue region shows uniaxial tensile loading region then the uniaxial tensile unloading happens which is shown by blue region after which material goes into uniaxial compressive unloading region .the stress strain curve for both the cases are same so we can say that the material will fail only in case of tension.

The material is evaluated for two cases exponential hardening and linear softning

Linear case H=-6

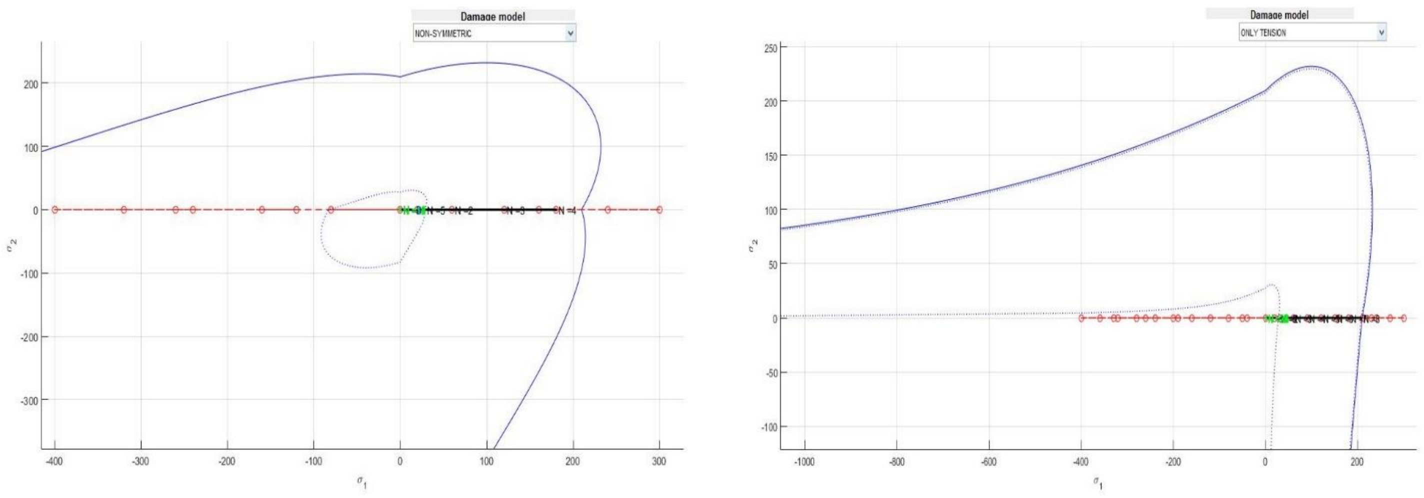


Figure 7 linear case Non- symmetric and tension only model behaviour

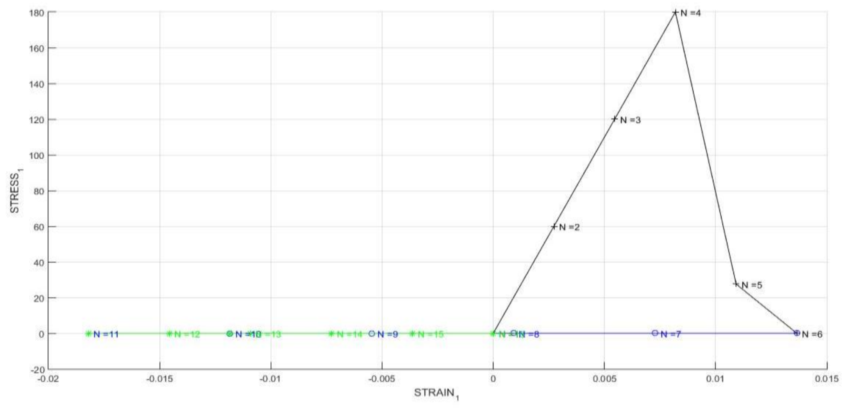


Figure 8 Stress Strain linear case Non- symmetric and tension only model behaviour

1.3.2) (i) uniaxial tensile loading {***Δσ1(1) = 300; Δσ2(1) = 0*** ***}***

(ii) biaxial tensile unloading/compressive loading ***{Δσ1(2) = −700; Δσ2(2) = 0}***

(iii) biaxial compressive unloading/tensile loading ***{Δσ1(3) = 400; Δσ2(3) = 0}***

Ans

The given following fig. shows the behaviour of described material of Exponential softening H= -6 and linear hardening case H=6

The green line shows uniaxial tensile loading the comes biaxial tensile unloading with blue line and in the end, it shows biaxial compressive unloading with black line.

Exponential case H=-6

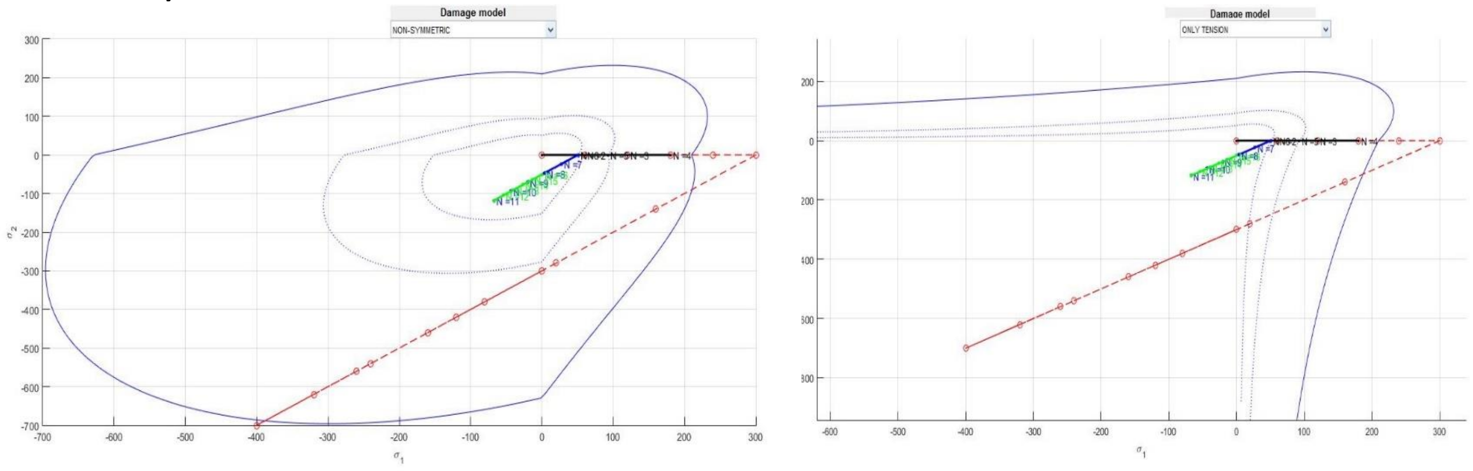


Figure 9 Exponential case Non- symmetric and tension only model behaviour

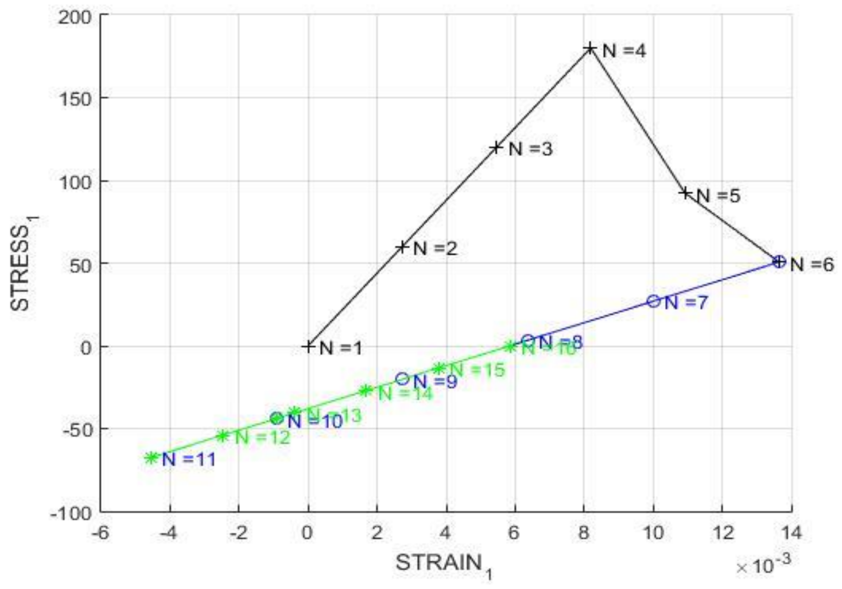


Figure 10 stress strain Exponential Case Non- symmetric and tension only model behaviour

Linear case H=-6

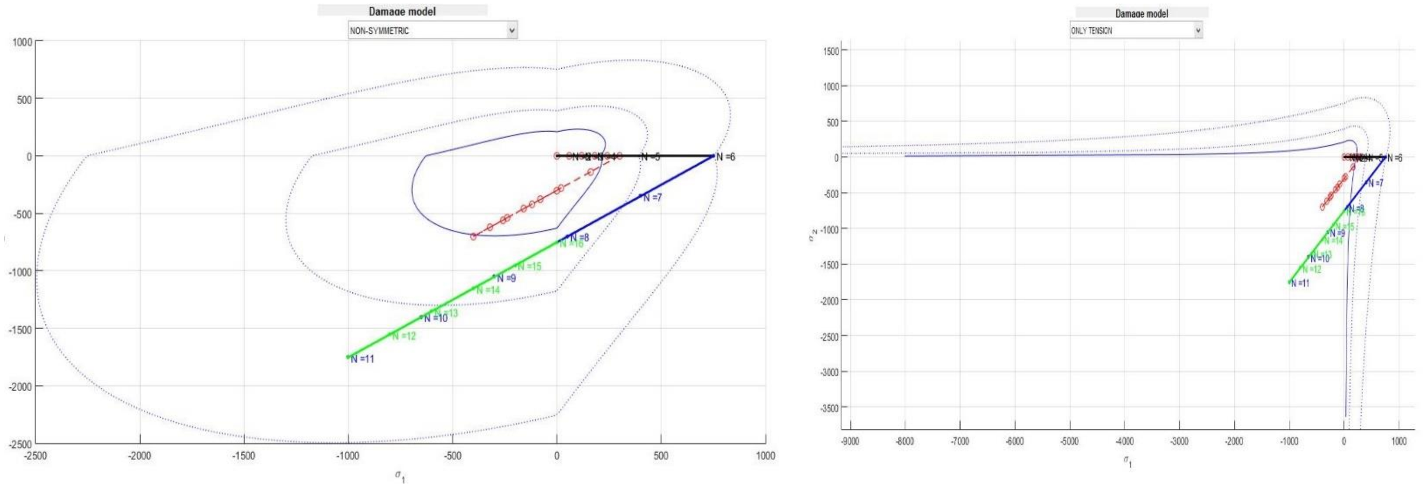


Figure 11 Linear case Non- symmetric and tension only model behaviour

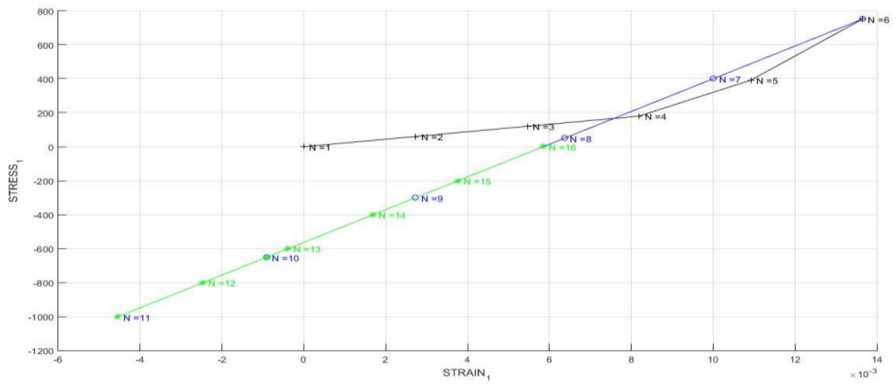


Figure 12 stress strain Linear case Non- symmetric and tension only model behaviour

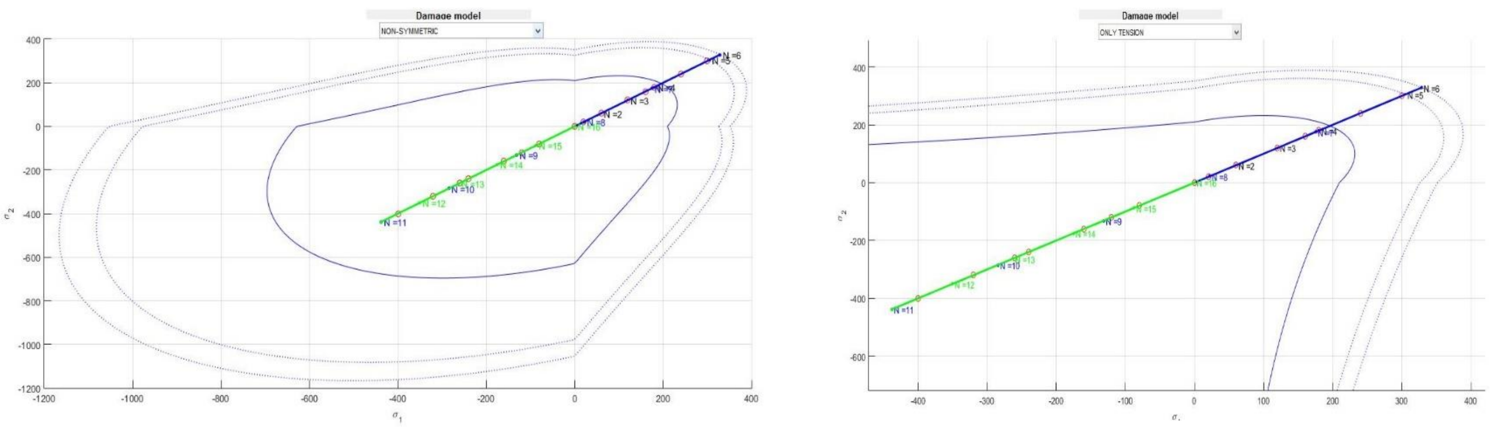
1.3.3) (i) biaxial tensile loading {***Δσ1(1) = 300; Δσ2(1) = 300}***

(ii) biaxial tensile unloading/compressive loading ***{Δσ1(2) = −700; Δσ2(2) = -700}***

(iii)biaxial compressive unloading/tensile loading ***{Δσ1(3) = 400; Δσ2(3) = 400}***

For the loading conditions, exponential harming and linear softening case is chosen.
The behaviour in both the cases shows that, the green line which is biaxial tensile loading is followed by biaxial tensile unloading by blue line and at last tensile loading is shown with black line

Exponential case H=6


Damage model: NON-SYMMETRIC

Damage model: ONLY TENSION

Figure 13 Exponential case Non- symmetric and tension only model behaviour

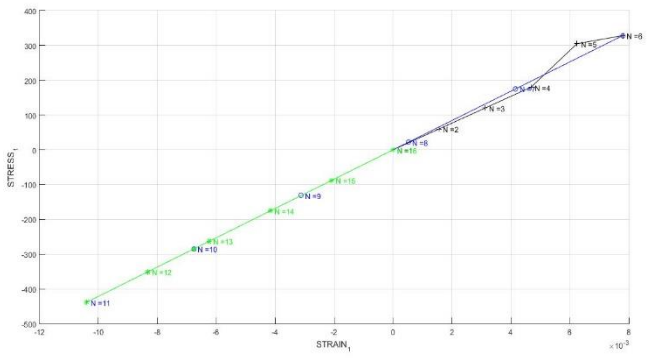
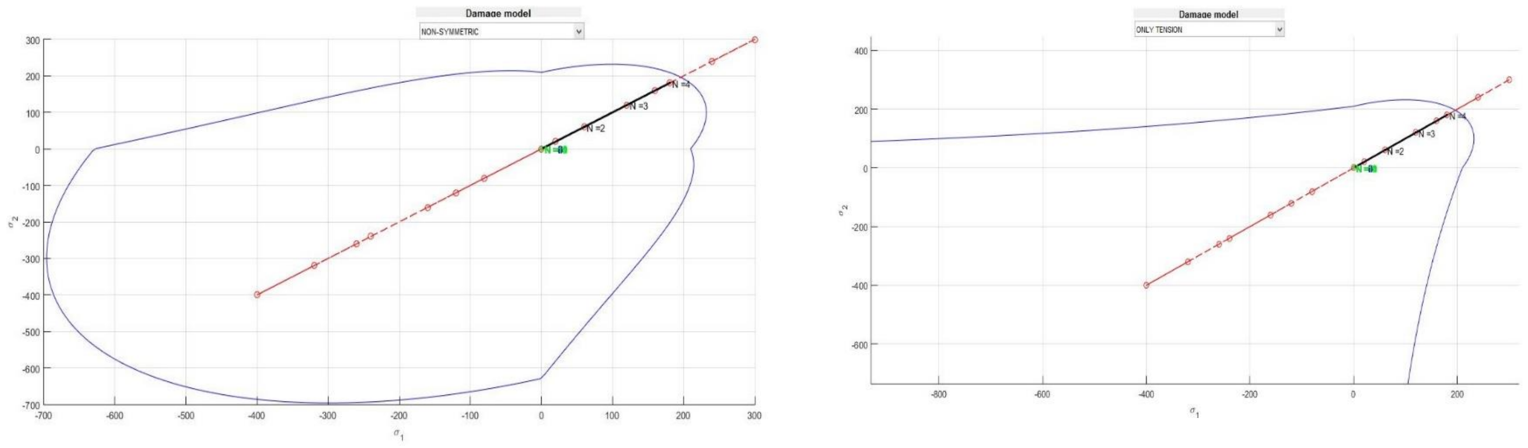


Figure 14 Stress strain Exponential case Non- symmetric and tension only model behaviour

Linear case H=-6


Damage model: NON-SYMMETRIC

Damage model: ONLY TENSION

Figure 15 Linear case Non- symmetric and tension only model behaviour

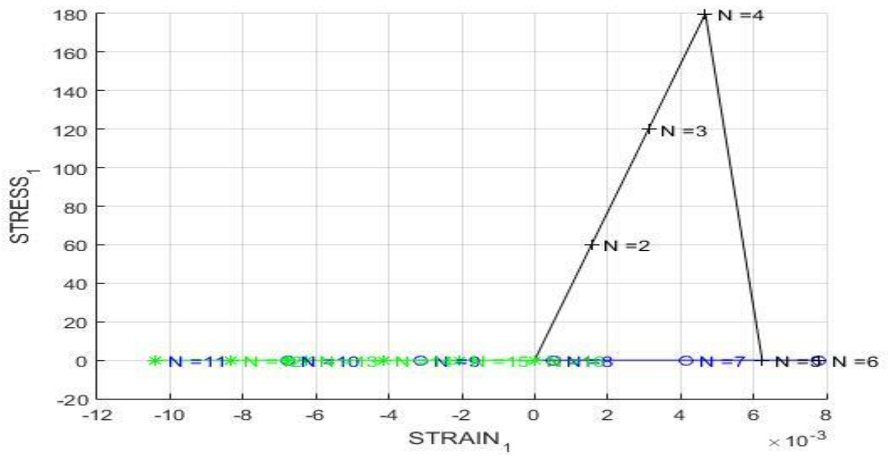


Figure 16 Stress Strain Linear case Non- symmetric and tension only model behaviour

2)PART II (rate dependent models):

2.1) Implement in the supplied MATLAB code the integration algorithm (plane strain case) for the continuum isotropic visco-damage “symmetric tension compression” model.
2.2) Assess the correctness of the implementation: consider the following cases (for a specific given Poisson ratio and linear hardening/softening parameter):
2.2.1)- Different viscosity parameters η.
2.2.2)- Different strain rate ε , values.
2.2.3)- Different α values: α = 0, α =1/ 4, α =1/ 2, α = 3/ 4 and α =1 (for the α time-integration method)
Obtain results displaying:

1) The effects of the previous values on the obtained stress-strain curves in appropriate loading paths.

Ans.
- Different viscosity parameters η used for this model are, 1 ,5,10, 25,50,100

The output which we have got is ,clear in stress strain curve we can say that till the particular elastic limit all the curves goes same , but after certain strain their behaviour starts varying. As the viscosity increases, the the curve goes up and up, so we can say that,as the viscosity increases energy absorption capacity of the body increases

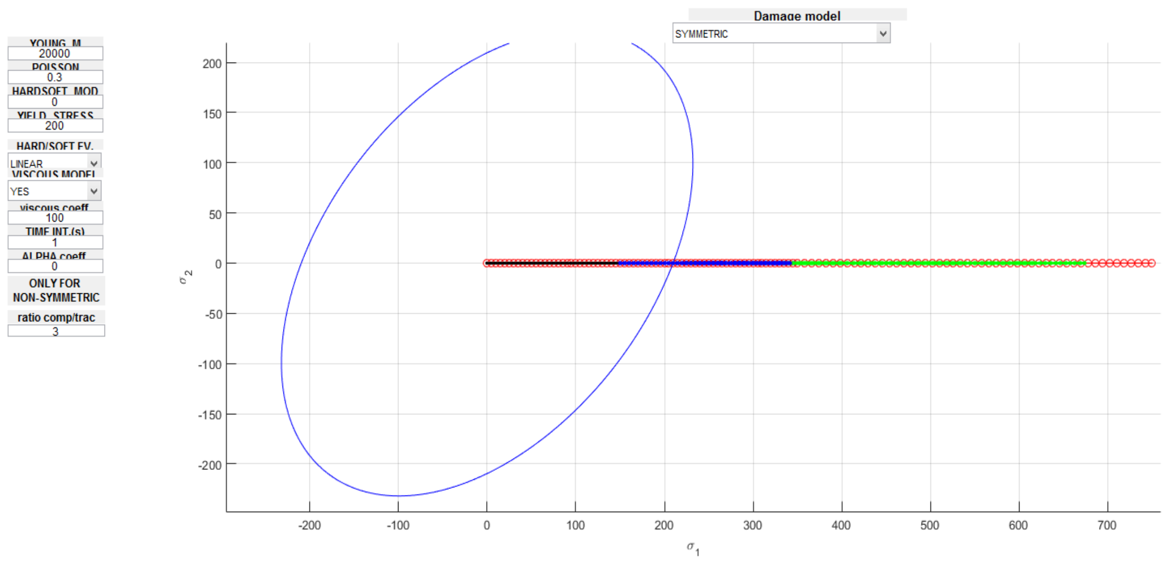

YOUNG_M: 20000
POISSON: 0.3
HARDSOFT_MOD: 0
YIELD_STRESS: 200
HARD/SOFT EV.: LINEAR
VISCOUS MODEL: YES
viscous coeff: 100
TIME INT.(s): 1
ALPHA coeff: 0
ONLY FOR NON-SYMMETRIC
ratio comp/trac: 3
Damage model: SYMMETRIC

Figure 17 Loading unloading curve for change in η

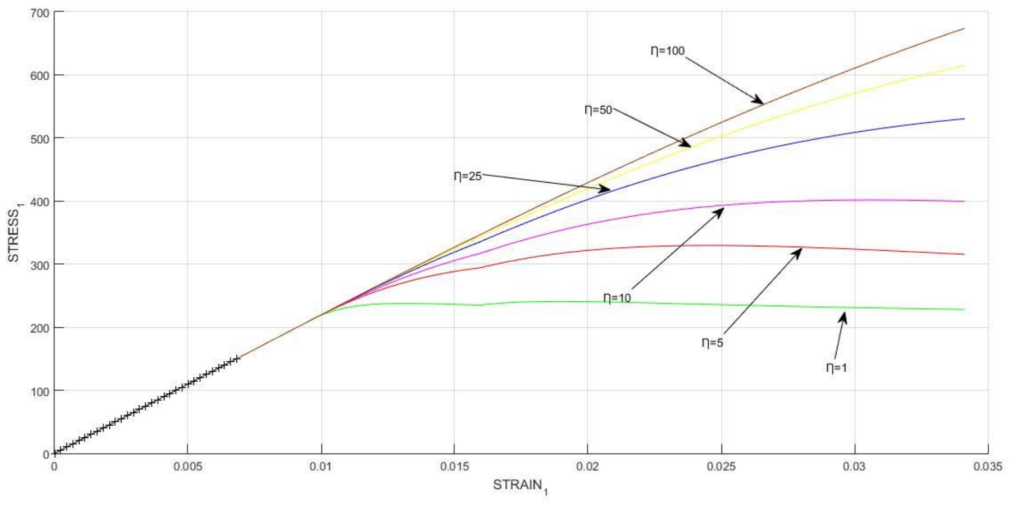


Figure 18 Stress Strain curve for change in η

- *Different strain rate ε , values*

The output which we have got is ,clear in stress strain curve we can say that till the particular elastic limit all the curves goes same , but after certain strain their behaviour starts varying. As the viscosity increases, the curve goes up and up, so we can say that, as the viscosity increases energy absorption capacity of the body increases

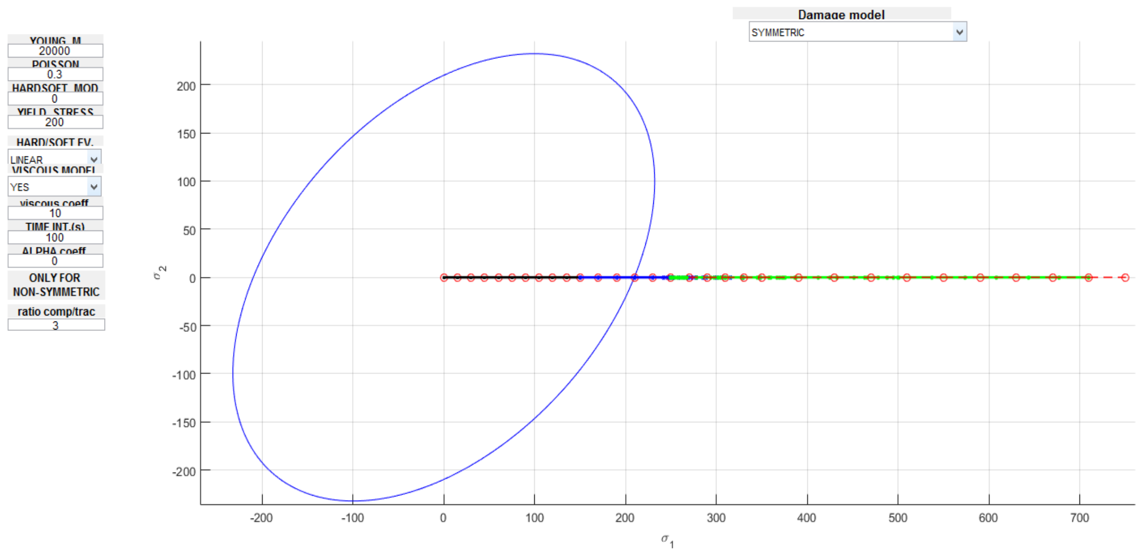

YOUNG_M
20000
POISSON
0.3
HARDSOFT_MOD
0
YIELD_STRESS
200
HARD/SOFT EV.
LINEAR
VISCOUS MODEL
YES
viscous coeff
10
TIME INT.(s)
100
ALPHA coeff
0
ONLY FOR NON-SYMMETRIC
ratio comp/trac
3
Damage model
SYMMETRIC

Figure 19 Loading unloading behaviour for variation in ε̇ value

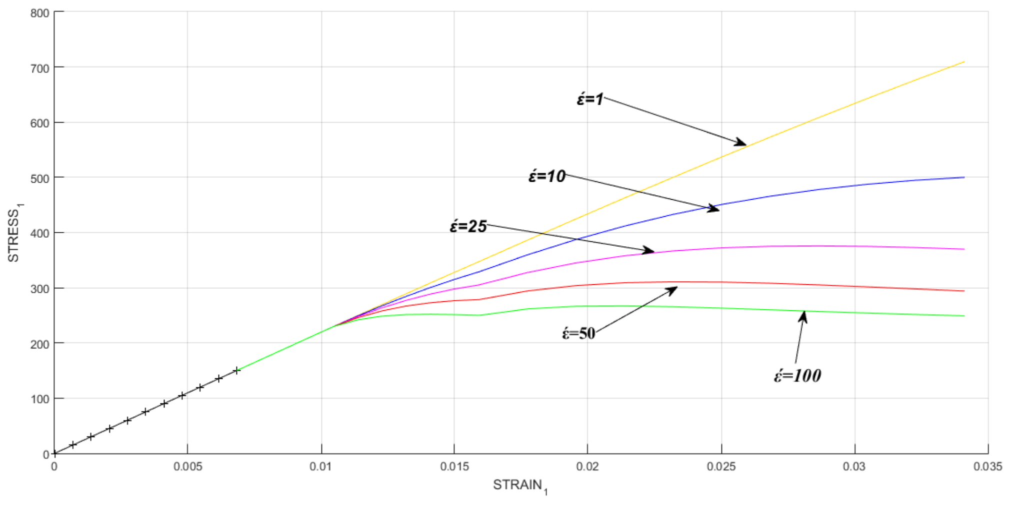
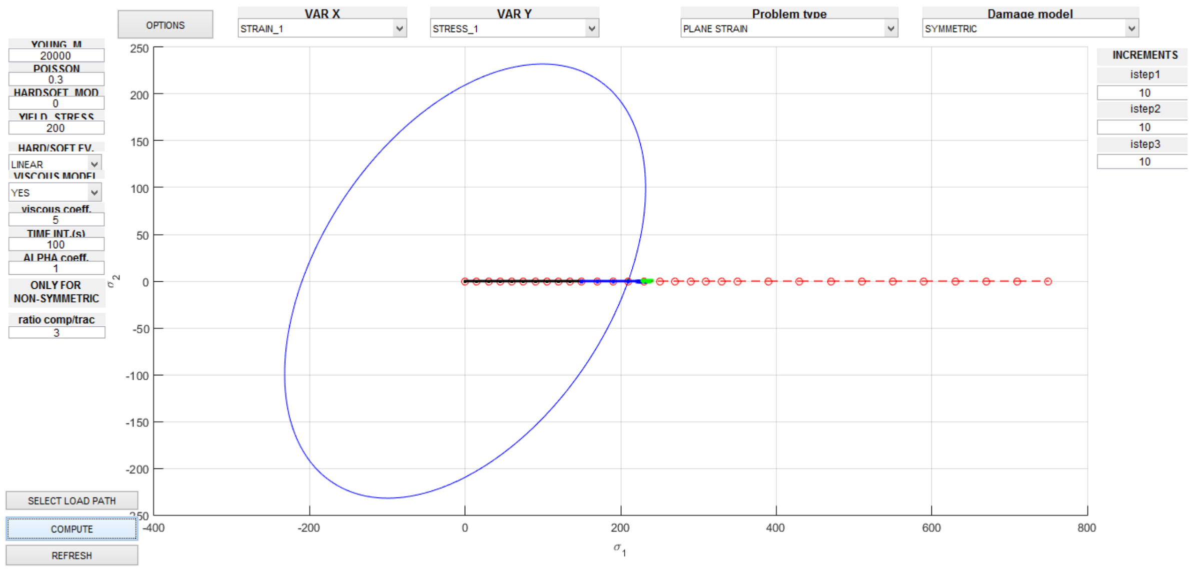


Figure 20 Stress Strain curve for variation in ɛ́ value

- *Different α values: α = 0, α =1/ 4, α =1/ 2,α = 3/ 4 and α =1*

We can see in stress strain curve that , up till certain limit, the the stress strain curve is same for all alfa values but after certain extend it starts varying . alfa is the multiplying factor in the solving of numerical integration .it gives more accurate result at 0.2 because the equation is of second order. On the other hand as it goes from 0.5 to 0 it starts oscillating on the other hand in the region of 0.5 to 1 it gives more stable results



VAR X: STRAIN_1
VAR Y: STRESS_1
Problem type: PLANE STRAIN
Damage model: SYMMETRIC

YOUNG_M: 20000
POISSON: 0.3
HARDSOFT_MOD: 0
YIELD_STRESS: 200
HARD/SOFT EV.: LINEAR
VISCOUS MODEL: YES
viscous coeff.: 5
TIME INT.(s): 100
ALPHA coeff.: 1
ONLY FOR NON-SYMMETRIC
ratio comp/trac: 3

INCREMENTS
istep1: 10
istep2: 10
istep3: 10

Figure 21 Loading unloading Curve for variation of alfa value

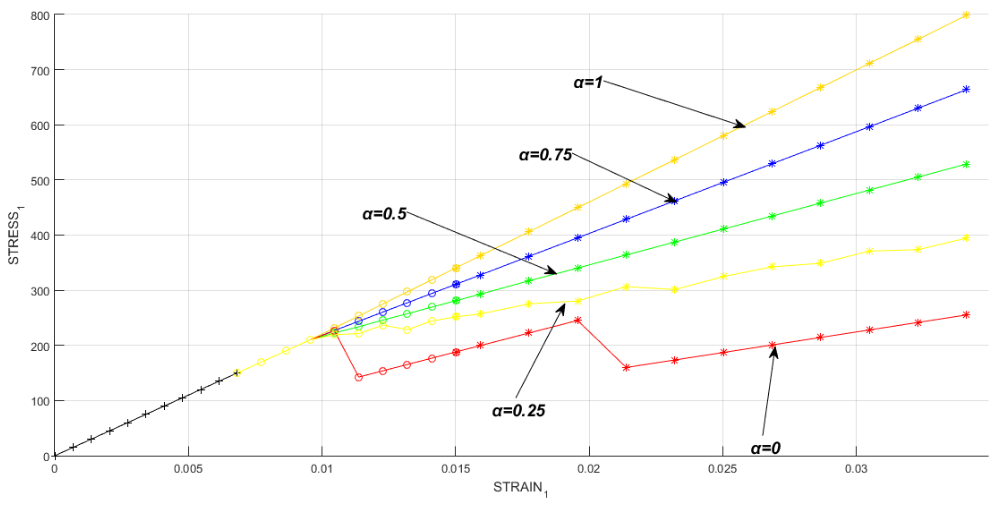
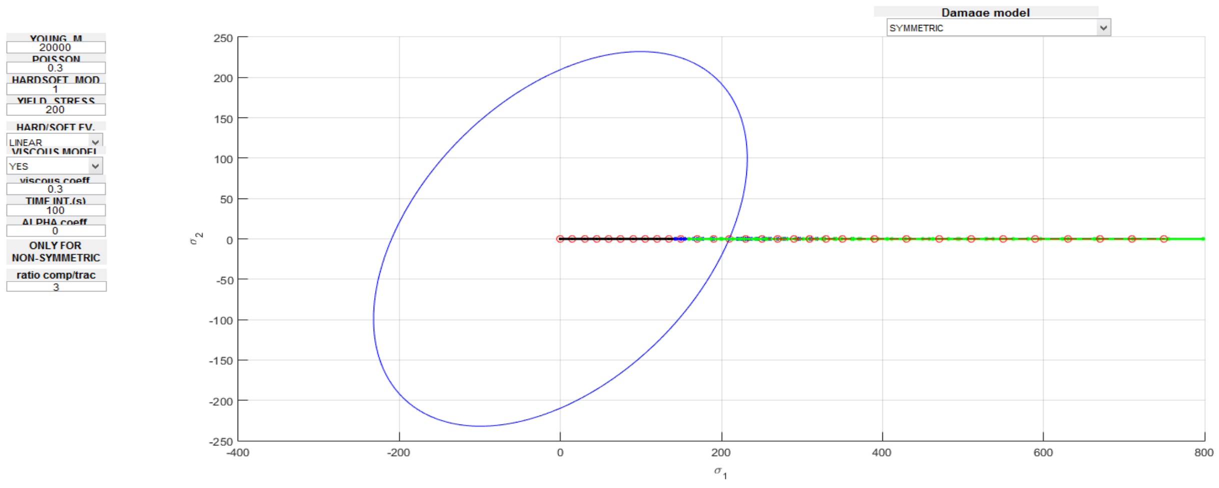


Figure 22 Stress Strain Curve for variation of alfa value

2.3) The effects of the α values, on the evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators

Ans.

α values change on algorithmic C_{11}



YOUNG_M	20000
POISSON	0.3
HARDSOFT_MOD	1
YIELD_STRESS	200
HARD/SOFT EV.	LINEAR
VISCOUS MODEL	YES
viscous coeff	0.3
TIME INT.(s)	100
ALPHA coeff	0
ONLY FOR NON-SYMMETRIC	
ratio comp/trac	3
Damage model	SYMMETRIC

Figure 23 Loading Unloading curve for variation of α

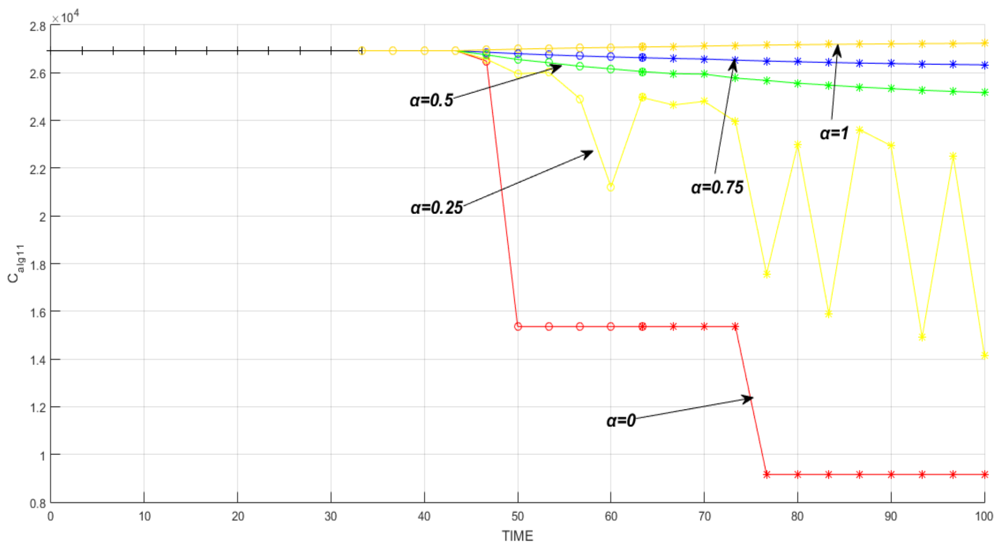


Figure 24 Algebraic C11 Vs time curve for variation of α

We can conclude from both of these stress strain curve that as α is the key factor for gaining the output. A is the multiplication factor while integrating by Fourier series and as the crank Nicolson equation is of second order it gives maximum accurate results at the value of α=0.5 on the other hand it oscilates drastically in the region of 0 to 0.5 and 0.5 to 1 it is approximately stable.As well as logarithmic and algorithmic curve matches at 0 , which proves that the code works properly *α values change on logarithmic C_{11}*

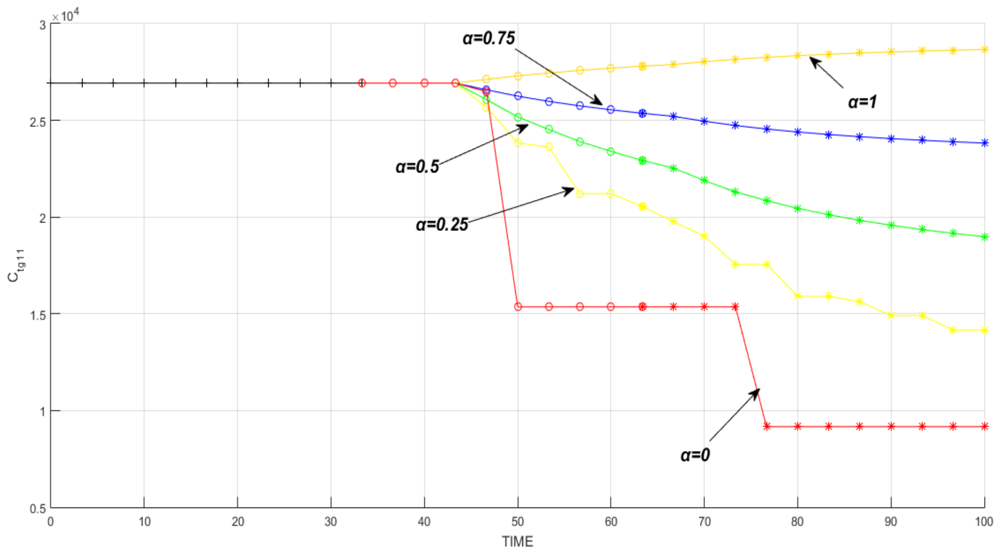


Figure 25 Logarithmic C11 Vs time curve for variation of α

ANNEX

dibujar_criterio_dano1

```
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----->tensiï¿½n
ï¿½ltima        %*
%*                      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
 tetha=[0:0.01:2*pi];
    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
        A = m1(i)*(m1(i)>0);
        B = m2(i)*(m2(i)>0);

        radio(i)= q/sqrt([A B 0 nu*(A+B)]*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
      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) ;
    a_N=0;
    a_D=0;
    for i=1:Contador
        A = m1(i)*(m1(i)>0);
        B = m2(i)*(m2(i)>0);
        a_N =A+B;
        alpha_D =abs(m1(i))+abs(m2(i));
        a = a_N/alpha_D;
        radio(i)= q/((a+(1-a)/n)*(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);

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



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

Damage_main

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,istep]=damage_main(Eprop,ntype,iste
p,strain,MDtype,n,TimeTotal)
global hplotSURF
LABELPLOT = {'hardening variable (q)','internal variable','damage
variable(d)','C_a_l_g_1_1','C_t_g_1_1'}

E = Eprop(1);
nu = Eprop(2);
viscpr = Eprop(6);
sigma_u = Eprop(4);
eta = Eprop(7);
A = Eprop(8);

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) ;
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
hvar_n(7) = 0; % New!!! added 16/03/2018 damage at t = 0
hvar_n(8) = ce(1,1); % C_alg_11
hvar_n(9) = ce(1,1); % C_t_11

eps_n1 = strain(i,:);
```

```
sigma_n1 =ce*eps_n1'; % Elastic (is effective sigma)
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0
sigma_n1(4)];

nplot = 5 ; % New
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) = hvar_n(7) ; %  Damage variable (d)
vartoplot{i}(4) = hvar_n(8); % Component 11 Constitutive alg matrix for
viscous
vartoplot{i}(5) = hvar_n(9); % Component 11 Constitutive tang matrix for
viscous

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

%***************************************************************************
************
        %*      DAMAGE MODEL
        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,viscpr,delta_t);
        % PLOTTING DAMAGE SURFACE
        if viscpr == 0
            if(aux_var(1)>0)
                hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6),
'r:',MDtype,n );
                set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1);
            elseif (aux_var(1)<=0)
            end
        else
        end

        % ------
        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;



        vartoplot{i}(1) = hvar_n(6);
        vartoplot{i}(2) = hvar_n(5);
        vartoplot{i}(3) = hvar_n(7);
        vartoplot{i}(4) = hvar_n(8);
        vartoplot{i}(5) = hvar_n(9);
    end
end
```

rmap_dano1

```
function
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR,istep]=damage_main(Eprop,ntype,iste
p,strain,MDtype,n,TimeTotal)
global hplotSURF
LABELPLOT = {'hardening variable (q)','internal variable','damage
variable(d)','C_a_l_g_1_1','C_t_g_1_1'}

E = Eprop(1);
nu = Eprop(2);
viscpr = Eprop(6);
sigma_u = Eprop(4);
eta = Eprop(7);
A = Eprop(8);

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) ;
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; % nq
hvar_n(7) = 0; % New!!! added 16/03/2018 damage at t = 0
hvar_n(8) = ce(1,1); % C_alg_11
hvar_n(9) = ce(1,1); % C_t_11

eps_n1 = strain(i,:);
```

```
sigma_n1 =ce*eps_n1'; % Elastic (is effective sigma)
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0
sigma_n1(4)];

nplot = 5 ; % New
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) = hvar_n(7) ; %  Damage variable (d)
vartoplot{i}(4) = hvar_n(8); % Component 11 Constitutive alg matrix for
viscous
vartoplot{i}(5) = hvar_n(9); % Component 11 Constitutive tang matrix for
viscous

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

%**************************************************************************
************
        %*      DAMAGE MODEL
        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] =
rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,viscpr,delta_t);
        % PLOTTING DAMAGE SURFACE
        if viscpr == 0
            if(aux_var(1)>0)
                hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6),
'r:',MDtype,n );
                set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1);
            elseif (aux_var(1)<=0)
            end
        else
        end

        % ------
        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;



        vartoplot{i}(1) = hvar_n(6);
        vartoplot{i}(2) = hvar_n(5);
        vartoplot{i}(3) = hvar_n(7);
        vartoplot{i}(4) = hvar_n(8);
        vartoplot{i}(5) = hvar_n(9);
    end
end

function [sigma_n1,hvar_n1,aux_var] = rmap_dano1
(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n,viscpr,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);
nq     = 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);
a = Eprop(8);
%***********************************************************************
***********

%***********************************************************************
***********
%*        initializing                                                %*
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0; %(1x10-6*ro)
% if(r_n<=0.d0)
%     r_n=r0;
%     nq=r0;
% end
%***********************************************************************
***********
```

```
%*************************************************************************
***********
%*       Damage surface
%*
[rtrial_prev] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
[rtrial] = Modelos_de_dano1(MDtype,ce,eps_n1,n);
rtrial_n_a = rtrial_prev*(1-a)+rtrial*a;

fload=0;

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

        fload=1;
        delta_r=rtrial-r_n;
        r_n1= rtrial;
        if hard_type == 0

            nq1= nq+ H*delta_r;
        elseif hard_type == 1

            infinit_q = r0 + (r0-zero_q);
            if H > 0
                H_n1 = H*((infinit_q-r0)/r0)*exp(H*(1-rtrial_n_a/r0));
%calculation of tangent hard modulus
            else
                H_n1 = H*((infinit_q-r0)/r0)*1/(exp(H*(1-
rtrial_n_a/r0)));%calculation of tangent soft modulus
            end
                nq1=nq+H_n1*(delta_r);
        end
        if(nq1<zero_q)
           nq1=zero_q;
        end
    else

        fload=0;
        r_n1= r_n  ;
        nq1= nq  ;

    end
else % Viscous Modelo
    if (rtrial_n_a > r_n)
        % loading
        fload=1;
        delta_r=rtrial_n_a-r_n;
        % computation of r at the step n+1
        r_n1 = (eta - delta_t*(1-a))/(eta + a*delta_t)*r_n + (delta_t/(eta
+ a*delta_t))*rtrial_n_a;

        if hard_type == 0
            %  Linear
            H_n1 = H;
            nq1= nq+ H_n1*delta_r;
        else
            %Hardening/Softening exponential law
            infinit_q = r0 + (r0-zero_q); %calulation of infinit_qinity
            if H > 0
                H_n1 = H*((infinit_q-r0)/r0)*exp(H*(1-rtrial_n_a/r0));
%calculation of tangent hard modulus
            else
```

```
                H_n1 = H*((infinit_q-r0)/r0)*1/(exp(H*(1-
rtrial_n_a/r0)));%calculation of tangent soft modulus
            end
            nq1 = nq + H_n1*delta_r; %calculation of q(n+1)
        end
        if(nq1<zero_q)
           nq1=zero_q;
        end
    else
        % elastic load/unload
        fload=0;
        r_n1= r_n;
        nq1= nq;
    end
end
% Damage variable
% ---------------
dano_n1 = 1-(nq1/r_n1);

%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';

% calculation of the Ce_tang_n1
if viscpr == 1
    if rtrial_n_a > r_n
        %Algorithm Constitutive Tangent Matrix
        Ce_alg_n1 = (1.d0-dano_n1)*ce+((a*delta_t)/(eta+a*delta_t))*...
            (1/rtrial_n_a)*((H_n1*r_n1-
nq1)/(r_n1^2))*((ce*eps_n1')'*(ce*eps_n1'));
        C_alg = Ce_alg_n1(1,1);
        %Constitutive Tangent Matrix
        Ce_tan_n1=(1.d0-dano_n1)*ce;
        C_tan = Ce_tan_n1(1,1);

    else rtrial_n_a <= r_n
        %Algorithm Constitutive Tangent Matrix
        Ce_alg_n1 = (1.d0-dano_n1)*ce;
        C_alg = Ce_alg_n1(1,1);
        %Constitutive Tangent Matrix
        Ce_tan_n1 = Ce_alg_n1;
        C_tan = C_alg;
    end

end
%  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)= nq1 ;
hvar_n1(7)=dano_n1;
if viscpr == 1
    hvar_n1(8)= C_alg;
    hvar_n1(9)= C_tan;
end
%***************************************************************************
***********


%***************************************************************************
***********
%* Auxiliar variables
%*
aux_var(1) = fload;
aux_var(2) = nq1/r_n1;
%*aux_var(3) = (nq1-H*r_n1)/r_n1^3;
%***************************************************************************
***********
```

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
sigma_e = ce*eps_n1';
    for i=1:length(sigma_e)
        sigma_e_plus(i) = sigma_e(i)*(sigma_e(i)>0);
    end
 rtrial= sqrt(sigma_e_plus*eps_n1');

elseif MDtype==3  %*Non-symmetric
    thita_N=0;
    thita_D=0;
    sigma_e = ce*eps_n1';
    for i=1:2 %length(sigma_e)
        sigma_e_plus(i) = sigma_e(i)*(sigma_e(i)>0);
        thita_N = thita_N + sigma_e_plus(i);
        thita_D = thita_D + abs(sigma_e(i));
    end
    thita = thita_N/thita_D;
    rtrial= (thita+(1-thita)/n)*sqrt(eps_n1*ce*eps_n1');
end
%*************************************************************************
***********
return
```