ASSINGMENT 1 COMPUTATIONAL SOLID MECHANICS

MARCOS BONIQUET

PART1 (rate dependant models)

Rate independent+plain strain assumption

a)

1.“NON symmetric tension-compression damage”

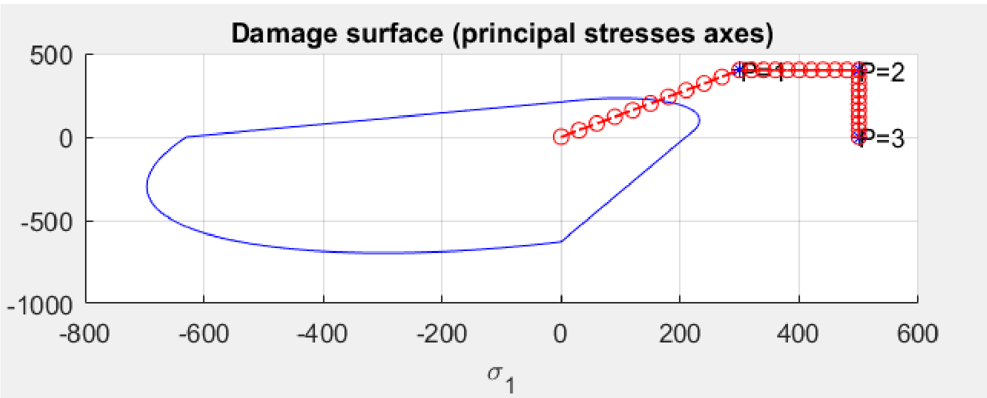
This implies changing *MDtype=3* in *main_no interactive*.

Next is changing the damage surface in *dibujar_criterio_dano1*.

Now the surface is q=stress norm= $\tau_\sigma = \left[\theta + \frac{1-\theta}{n}\right]\sqrt{\boldsymbol{\sigma} : \mathbb{C}^{-1} : \boldsymbol{\sigma}}$, where n defined in main function (n=3) and tetha depending on main stresses. $\theta = \frac{\sum_1^3 \langle \sigma_i \rangle}{\sum_1^3 |\sigma_i|} = \frac{\sum_1^3 \langle \bar{\sigma}_i \rangle}{\sum_1^3 |\bar{\sigma}_i|}$

This theta depends on McAuley and absolute value of main stresses, which can be divided in 4 ranges, for each of the quadrants.

For plotting we use the “$\boldsymbol{\sigma}_1$=r$\boldsymbol{\theta}$cos$\boldsymbol{\theta}$ & $\boldsymbol{\sigma}_1$=r($\boldsymbol{\theta}$)cos($\boldsymbol{\theta}$) polar coordinates. (radius depending on $\boldsymbol{\theta}$ is the key) and vector “s1” and “s2” sum include point son first and third quadrant. Lines on second and fourth quadrant are just made joining to points attached.



For calculating the stresses must be done on *modelos_de_dano1*, *Tangent constitutive operator* $\mathbb{C}$, which is considered constant and given by $\boldsymbol{\nu}$ and **E.**

Quadrant 1 $\boldsymbol{\sigma}_1>0$; $\boldsymbol{\sigma}_2>0$; $\boldsymbol{\theta}_1=1$
Quadrant 2 $\boldsymbol{\sigma}_1>0$; $\boldsymbol{\sigma}_2<0$; $\boldsymbol{\theta}_2=\boldsymbol{\sigma}_1/(\boldsymbol{\sigma}_1+\boldsymbol{\sigma}_2)$
Quadrant 3 $\boldsymbol{\sigma}_1<0$; $\boldsymbol{\sigma}_2<0$; $\boldsymbol{\theta}_3=0$
Quadrant 4 $\boldsymbol{\sigma}_1<0$; $\boldsymbol{\sigma}_2>0$; $\boldsymbol{\theta}_4=\boldsymbol{\sigma}_2/(\boldsymbol{\sigma}_1+\boldsymbol{\sigma}_2)$

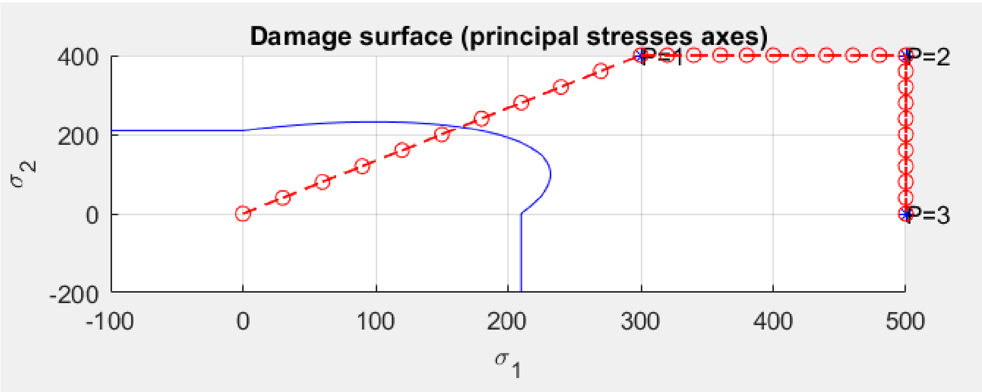
we define non-symmetric factor as $F=(\boldsymbol{\theta}+(1-\boldsymbol{\theta})/n)$.

Quadrant 1 $\boldsymbol{\theta}_1=1$; F=1
Quadrant 2 $\boldsymbol{\theta}_2=\boldsymbol{\sigma}_1/(\boldsymbol{\sigma}_1+\boldsymbol{\sigma}_2)=\cos\boldsymbol{\theta}/(\cos\boldsymbol{\theta}+\sin\boldsymbol{\theta})$; $F=(\boldsymbol{\theta}_2+(1-\boldsymbol{\theta}_2)/n)$
Quadrant3 $\boldsymbol{\theta}_3=0$;
Quadrant 4 $\boldsymbol{\theta}_4=\boldsymbol{\sigma}_2/(\boldsymbol{\sigma}_1+\boldsymbol{\sigma}_2)=\sin\boldsymbol{\theta}/(\cos\boldsymbol{\theta}+\sin\boldsymbol{\theta})$;$F=(\boldsymbol{\theta}_2+(1-\boldsymbol{\theta}_4)/n)$

Strain norm will be same as symmetric case, divided by this factor, which depends on main stresses.

2."Tensile-only" damage model

Plotting is quite trivial, just taking profit of first quadrant and plotting two straight lines. This must be done at *dibujar_criterio_dano1*.



Again, for calculating the stresses must be done on *modelos_de_dano1*, *Tangent constitutive operator* ℂ, which is considered constant and given by ***ν*** and **E.**

3d quadrant

We assume Strain norm =0 for other quadrants than the first one.

b)

Implement linear and exponential hardening/softening (H<0,H>0)
<u>Linear</u>

Linear is already implemented. To implement exponential will be necessary to modify *rmap_dano1*, which calculates the stress path depending on hardening function assumption.

Only loading variant will take into account the hardening function.

It's remarkable that of H<0, the elastic domain diminishes, and if it's H>0 it increases (e.g. metal cases). In this last case the damage leds to a more resistance a posterior stress.

<u>Exponential</u>

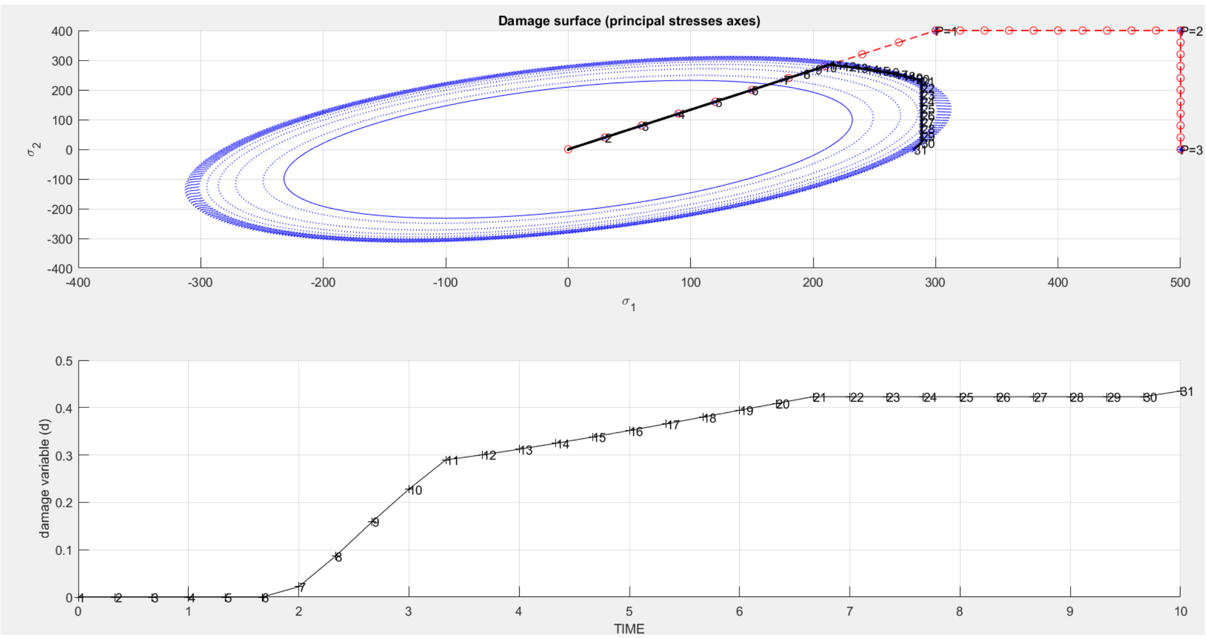
when:
$q_{inf}<r_0$-->elastic domain diminishes
$q_{inf}>r_0$, it increases.
$q_{inf}=r_0$ Elastic domain remains the same

it's been detected that when it $q_{inf}>r_0$, "A" *takes an important role* in correct behaviour of simulation.
Hypothesis: It has to be taken into account that *aux_var(6)*=q_{n+1}/r_{n+1} has to be bigger than q_n/r_n in order to damage never decrease.

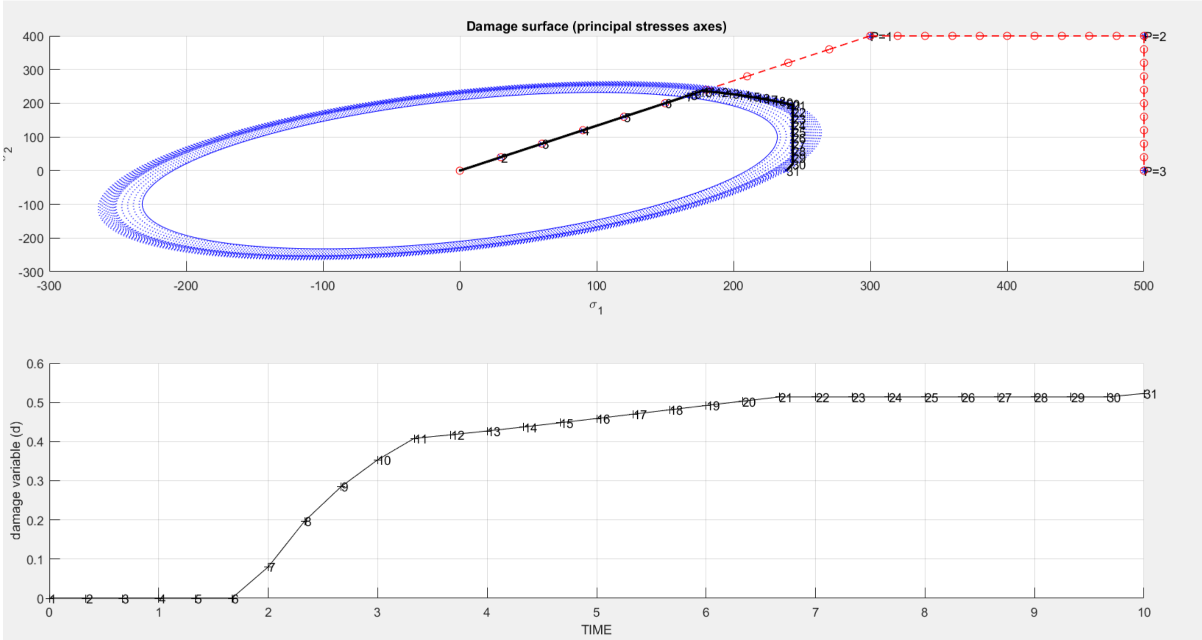
With q_{inf}=2 and A=2:

<u>Examples on 4 combinations:</u>

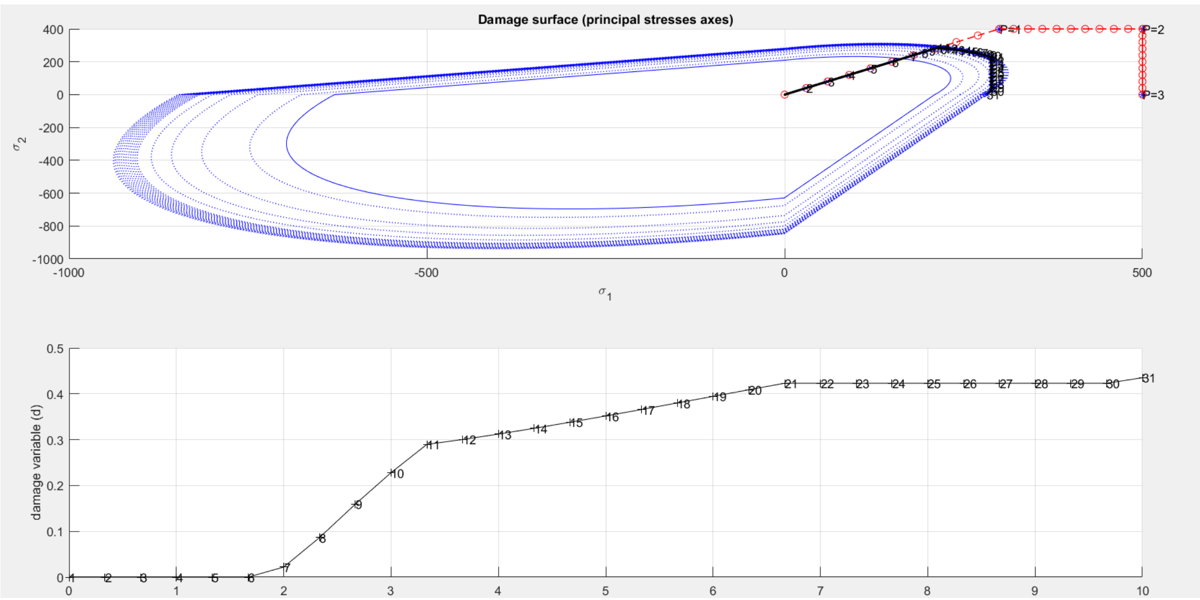
Exponential symmetric case



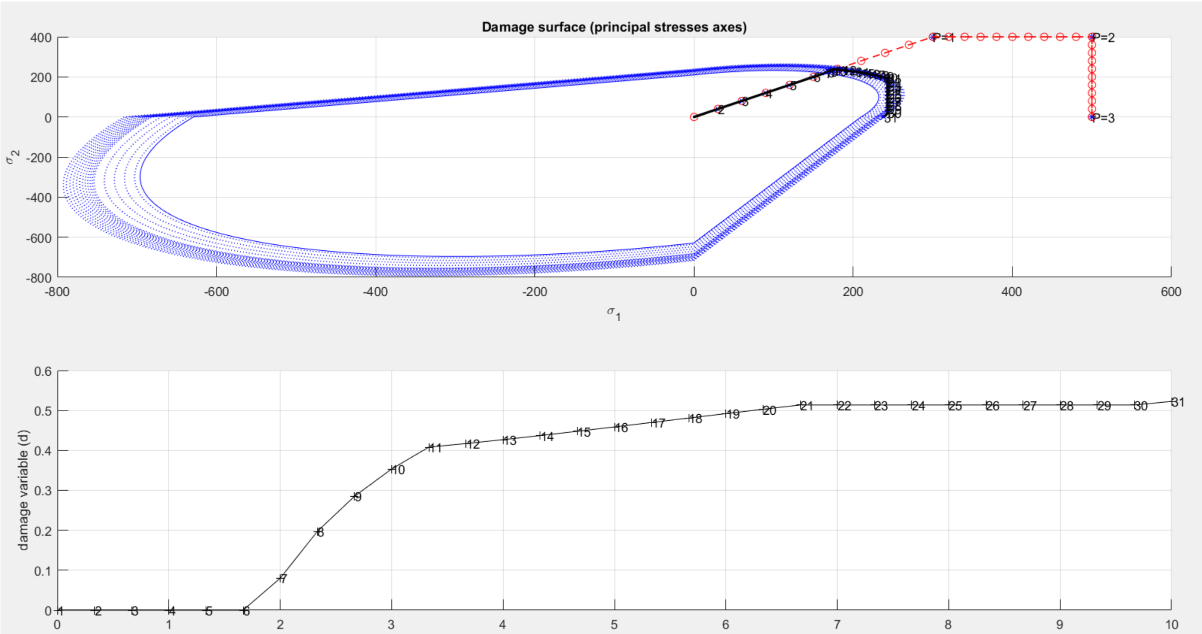
Linear symmetric



Non sym exponential



Non symmetric linear



Both cases, **more damage in linear,** though elastic domain does increases in exponential vs linear.

c)

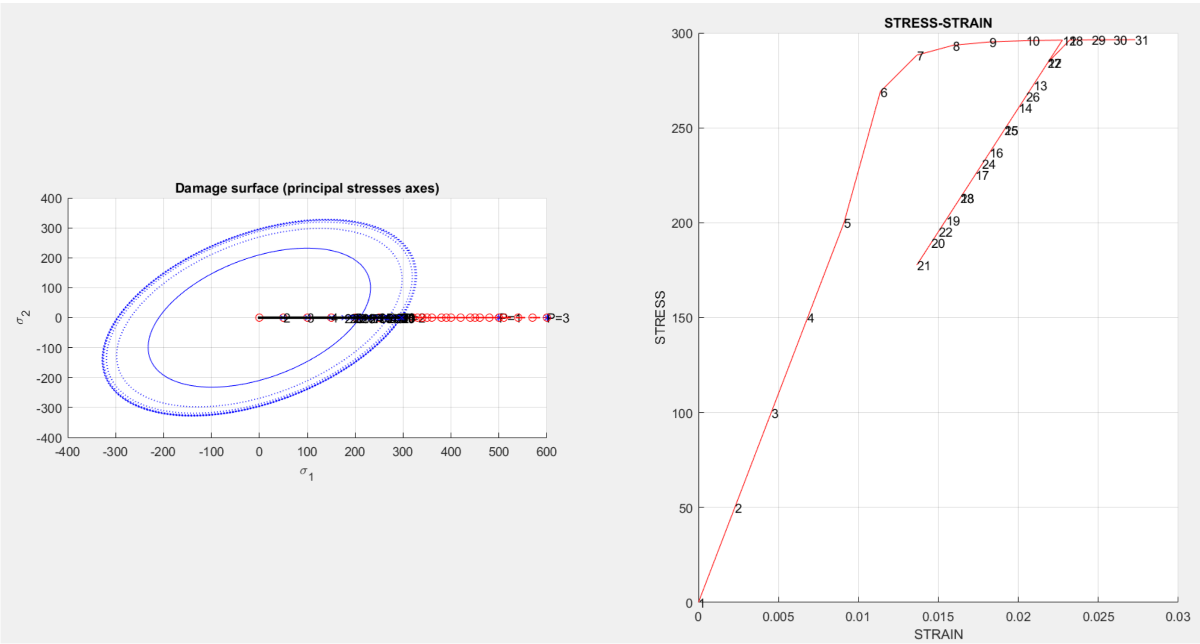
For each case obtain stress space and stress strain

Considering *EXPONENTIAL HARDENING/SOFTENING* in order to linear/non linear behaviour to be more obvious.

1. $\Delta\sigma_1^{(1)}=\alpha$; $\Delta\sigma_2^{(1)}=0$
   $\Delta\sigma_1^{(2)}=-\beta$; $\Delta\sigma_2^{(2)}=0$
   $\Delta\sigma_1^{(3)}=\gamma$; $\Delta\sigma_3^{(3)}=0$

   SIGMAP=(α 0, α-β 0, α-β+γ 0)

Symmetric



First path 1-11
1 to 5   Elastic domain. No damage. Linear behaviour. REACHES damage surface.
5 to 11 Load. Damage. Damage Surface increases.

First path 11-21
11 to 21 Unload. No damage. Elastic behaviour (E_d)

First path 21-31
21-27   Elastic domain. No damage. Linear behaviour. REACHES damage surface.
27 to 31 Load. Damage Surface (already increased) increases more. reaches at σ_2 =284 vs σ_2=200 of point 5.

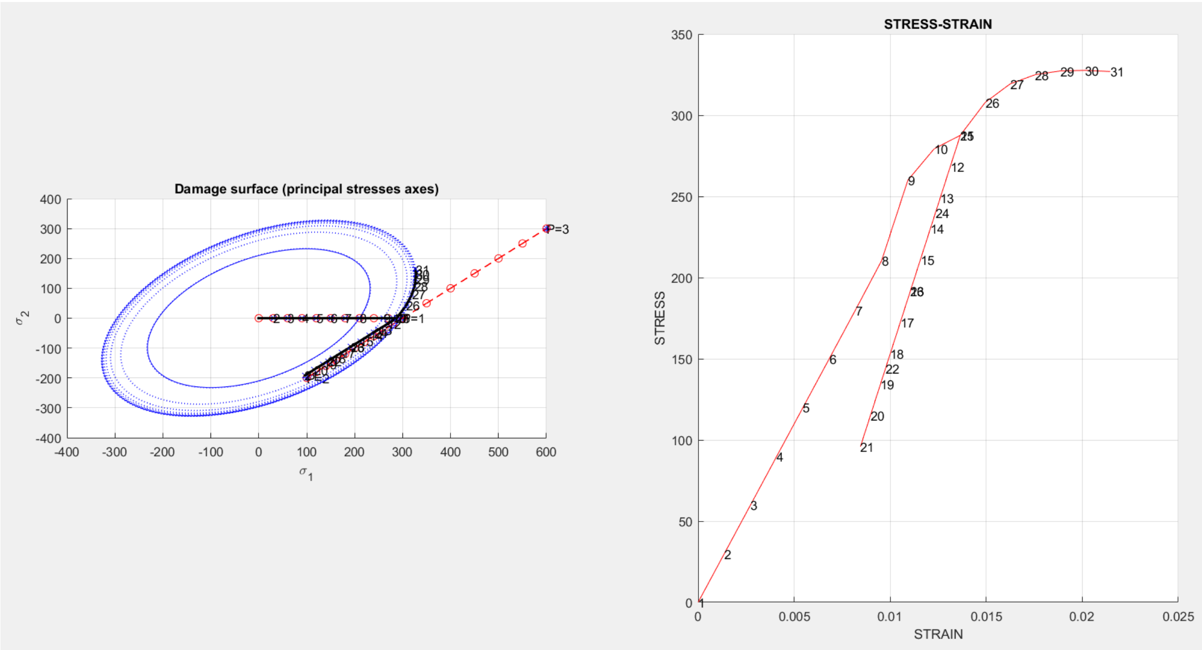
- *Non Symmetric tension-compression: Same*

- *tension-only case:Same*

2. $\Delta\boldsymbol{\sigma}_1^{(1)}$=α ; $\Delta\boldsymbol{\sigma}_2^{(1)}$=0
$\Delta\boldsymbol{\sigma}_1^{(2)}$=-$\beta$; $\Delta\boldsymbol{\sigma}_2^{(2)}$=-$\beta$
$\Delta\boldsymbol{\sigma}_1^{(3)}$=$\gamma$; $\Delta\boldsymbol{\sigma}_3^{(3)}$=$\gamma$

SIGMAP=(α 0, α-β -β, α-β+γ -β+γ)

Symmetric

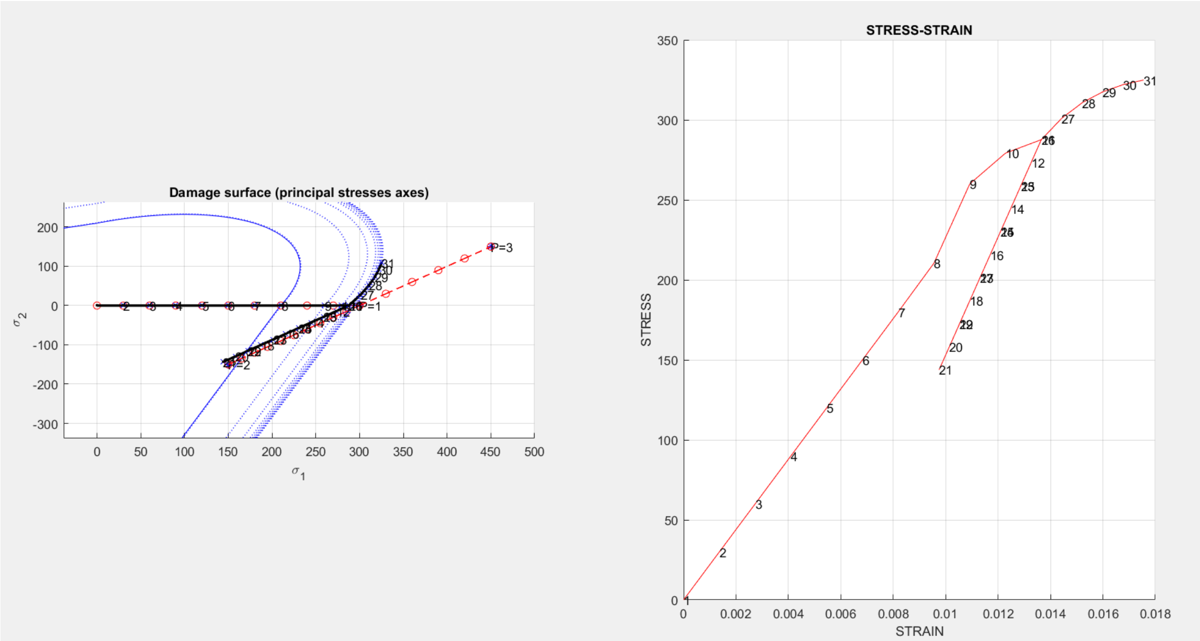


<u>*First path 1-11*</u>
1 to 8 Elastic domain. No damage. Linear behaviour. REACHES damage surface.
8 to 11 Load. Damage. Damage Surface increases.

<u>*First path 11-21*</u>
11 to 21 Unload. No damage. Elastic behaviour (E_d)

<u>*First path 21-31*</u>
21-25 Elastic domain. No damage. Linear behaviour. REACHES damage surface.
25 to 31 Load. Damage Surface (already increased) increases more.

Non Symmetric tension-compression



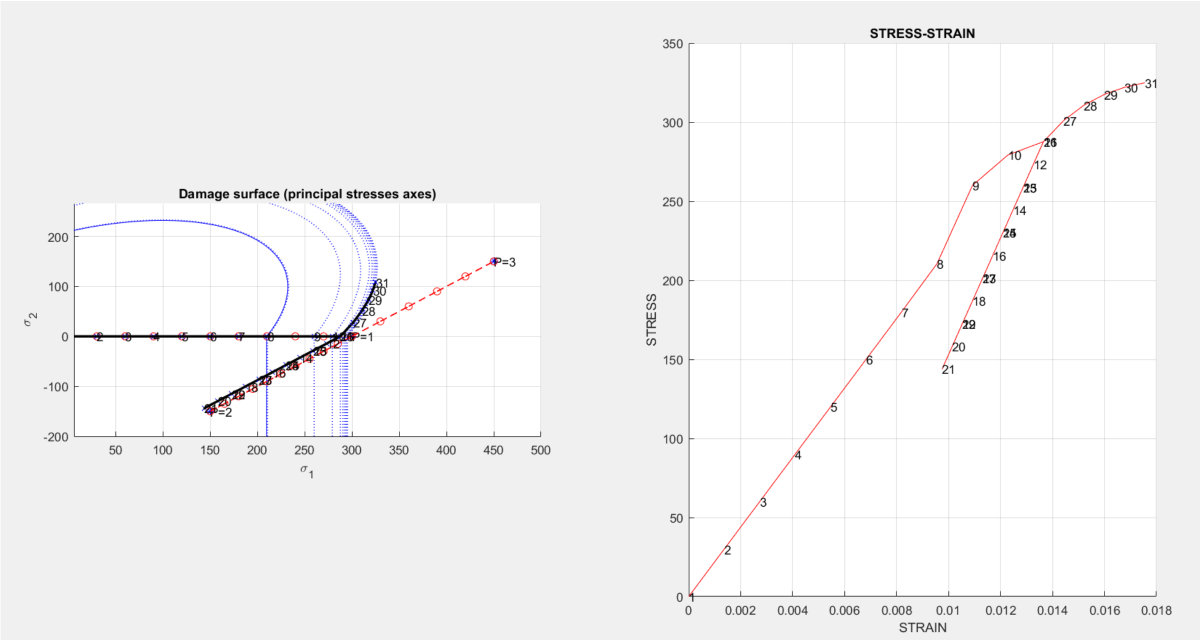
Very similar to symmetric case:

<u>First path 1-11</u>
1 to 8 Elastic domain. No damage. Linear behaviour. REACHES damage surface.
8 to 11 Load. Damage. Damage Surface increases.

<u>First path 11-21</u>
11to 21 Unload. No damage. Elastic behaviour (E_d)

<u>First path 21-31</u>
21-26   Elastic domain. No damage. Linear behaviour. REACHES damage surface.
26 to 31 Load. Damage Surface (already increased) increases more.   ***Reaches surface at σ_2 =287 vs σ_2=158 of point 26.***

Tension-only case



First path 1-11

1 to 8 Elastic domain. No damage. Linear behaviour. REACHES damage surface.

8 to 11 Load. Damage. Damage Surface increases.

First path 11-21

11 to 21 Unload. No damage. Elastic behaviour (E_d)

First path 21-31

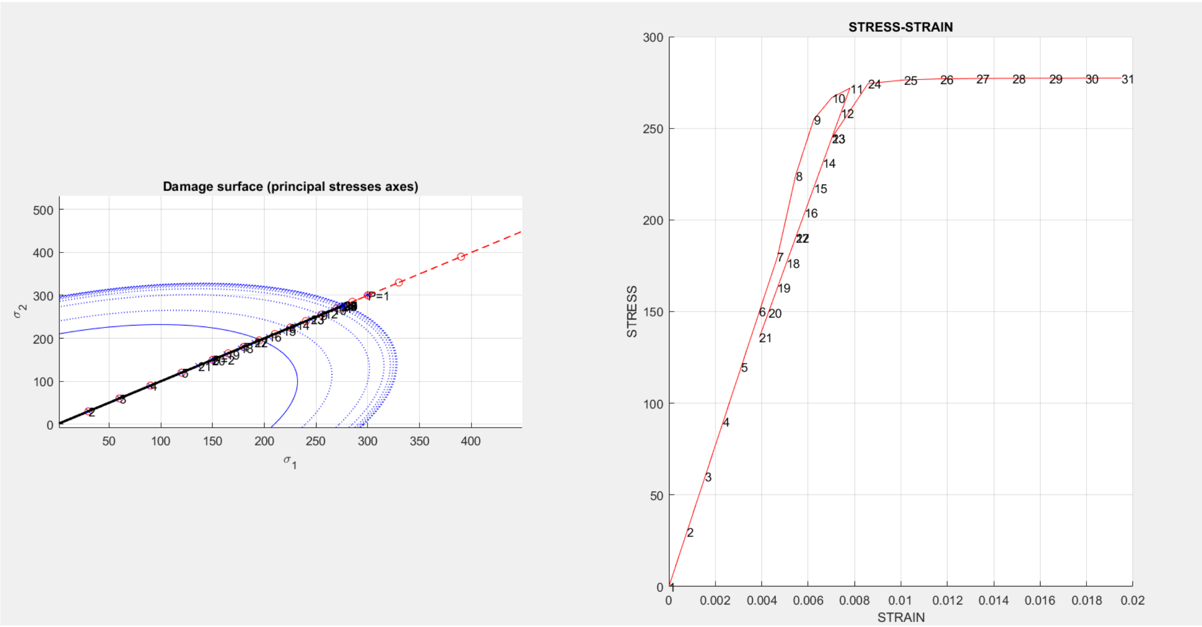
21-26   Elastic domain. No damage. Linear behaviour. REACHES damage surface.

26 to 31 Load. Damage Surface (already increased) increases more.   Reaches surface at $\boldsymbol{\sigma}_2$ *=288 vs* $\boldsymbol{\sigma}_2$*=210 of point 19-20.*

3. $\Delta\boldsymbol{\sigma}_1^{(1)}=\alpha$; $\Delta\boldsymbol{\sigma}_2^{(1)}=\alpha$
$\Delta\boldsymbol{\sigma}_1^{(2)}=-\beta$; $\Delta\boldsymbol{\sigma}_2^{(2)}=-\beta$
$\Delta\boldsymbol{\sigma}_1^{(3)}=\gamma$; $\Delta\boldsymbol{\sigma}_3^{(3)}=\gamma$

SIGMAP=(α α, $\alpha-\beta$ $\alpha-\beta$, $\alpha-\beta+\gamma$ $\alpha-\beta+\gamma$)

Symmetric



<u>*First path 1-11*</u>
1 to 7 Elastic domain. No damage. Linear behaviour. REACHES damage surface.
7 to 11 Load. Damage. Damage Surface increases.

<u>*First path 11-21*</u>
11 to 21 Unload. No damage. Elastic behaviour (E_d)

<u>*First path 21-31*</u>
21-22 Elastic domain. No damage. Linear behaviour. REACHES damage surface.
22 to 31 Load. Damage Surface (already increased) increases more.

- *Non Symmetric tension-compression: Same*
- *tension-only case:Same*

ANEXO 1

```
function [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n)

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

    elseif (MDtype==2)  %* Only tension
   sigma=ce*eps_n1';
   sigma1=sigma(1);
   sigma2=sigma(2);

   if (sigma1>0)&&(sigma2>0)

   rtrial= sqrt(eps_n1*ce*eps_n1')

   elseif (sigma1<0)&&(sigma2<0)

   rtrial=0;

   elseif ((sigma1<0)&&(sigma2>0)) %!!!!
    rtrial=0;
   elseif ((sigma1>0)&&(sigma2<0)) %!!!!
     rtrial=0;
   end

    elseif (MDtype==3)  %*Non-symmetric

   sigma=ce*eps_n1';
   sigma1=sigma(1);
   sigma2=sigma(2);


   if (sigma1>0)&&(sigma2>0)

   rtrial= sqrt(eps_n1*ce*eps_n1')

   elseif (sigma1<0)&&(sigma2<0)

   rtrial= sqrt(eps_n1*ce*eps_n1')*n;

   elseif ((sigma1<0)&&(sigma2>0))

       tetha=sigma2/(sigma1+sigma2);
       F=tetha+(1-tetha)/n;
       rtrial= sqrt(eps_n1*ce*eps_n1')/F;

   elseif ((sigma1>0)&&(sigma2<0))%%%%%%%aqui!!!!!!!

       tetha=sigma1/(sigma1+sigma2);
       F=tetha+(1-tetha)/n;
       rtrial= sqrt(eps_n1*ce*eps_n1')/F;

   end
   end

return
```

```
Not enough input arguments.
```

```
function plotcurvesNEW(DATA,vpx,vpy,LABELPLOT,vartoplot)
% Plot stress vs strain (callback function)
% ------------------------------------------





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%55
% PLOTTING
ncolores = 3 ;
colores =  ColoresMatrix(ncolores);
markers = MarkerMatrix(ncolores) ;


subplot(2,2,3)
title('DAMAGE')
hold on
grid on
xlabel(vpx);
ylabel(vpy);


switch  vpx
    case 'STRAIN_1'
        strx = 'X(i) = DATA.strain(i,1);' ;
        %strx = 'X(i) = max(DATA.strain(i,1),DATA.strain(i,2));' ;
    case 'STRAIN_2'
        strx = 'X(i) = DATA.strain(i,2);' ;
        %strx = 'X(i) = min(DATA.strain(i,1),DATA.strain(i,2));' ;
    case '|STRAIN_1|'
        strx = 'X(i) = abs(DATA.strain(i,1));' ;
        %strx = 'X(i) = abs(max(DATA.strain(i,1),DATA.strain(i,2)));' ;
    case '|STRAIN_2|'
        strx = 'X(i) = abs(DATA.strain(i,2));' ;
        %strx = 'X(i) = abs(min(DATA.strain(i,1),DATA.strain(i,2)));' ;
    case  'norm(STRAIN)'
        strx = 'X(i) =sqrt((DATA.strain(i,1))^2 + (DATA.strain(i,2))^2)) ;';
    case  'TIME'
        strx = 'X(i) =DATA.TIMEVECTOR(i) ;';
    otherwise
        for iplot = 1:length(LABELPLOT)
            switch vpx
                case LABELPLOT{iplot}
                    strx =  ['X(i) = vartoplot{i}(',num2str(iplot),') ;'];
            end
        end
end


X = 0 ;
for i = 1:size(DATA.strain,1)
    eval(strx) ;
end

% DATA Y
% ------

switch  vpy
```

```
    case 'STRESS_1'
        stry = 'Y(i) = DATA.sigma_v{i}(1,1);' ;
        %stry = 'Y(i) = max(DATA.sigma_v{i}(1,1),DATA.sigma_v{i}(2,2));' ;
    case 'STRESS_2'
        stry = 'Y(i) = DATA.sigma_v{i}(2,2);' ;
        %stry = 'Y(i) = min(DATA.sigma_v{i}(1,1),DATA.sigma_v{i}(2,2));' ;
    case '|STRESS_1|'
        %stry = 'Y(i) = abs(max(DATA.sigma_v{i}(1,1),DATA.sigma_v{i}(2,2)));' ;
        stry = 'Y(i) = abs(DATA.sigma_v{i}(1,1));' ;
    case '|STRESS_2|'
        %stry = 'Y(i) = abs(min(DATA.sigma_v{i}(1,1),DATA.sigma_v{i}(2,2)));' ;
        stry = 'Y(i) = abs(DATA.sigma_v{i}(2,2));' ;
    case  'norm(STRESS)'
        stry = 'Y(i) = sqrt((DATA.sigma_v{i}(1,1))^2+(DATA.sigma_v{i}(2,2))^2);' ;
    case  'DAMAGE VAR.'
        stry = 'Y(i) = sqrt((DATA.sigma_v{i}(1,1))^2+(DATA.sigma_v{i}(2,2))^2);' ;

    otherwise



        for iplot = 1:length(LABELPLOT)
            switch vpy
                case LABELPLOT{iplot}
                    stry =  ['Y(i) = vartoplot{i}(',num2str(iplot),') ;'];
            end
        end

end

 Y = 0 ;
for i = 1:length(DATA.sigma_v)
    try
        eval(stry);

    end
end



plot(X,Y,'Marker',markers{1},'Color',colores(1,:));

for i=1:length(X)
    text(X(i),Y(i),num2str(i));
end
```

```
Not enough input arguments.

Error in plotcurvesNEW (line 20)
xlabel(vpx);
```

```
function strain =PlotIniSurf(YOUNG_M,POISSON,YIELD_STRESS,SIGMAP,ntype,MDtype,n,istep)


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

Eprop=[YOUNG_M POISSON 0 YIELD_STRESS];
sigma_u =YIELD_STRESS ;
E = YOUNG_M ;
nu = POISSON ;

%*************************************************************************************
%*       Evaluar el tensor constitutivo elï¿½stico (Matriz de Hooke)                      %*
%*       Llamado de Rutina tensor_elastico1                                            %*
[ce] = tensor_elastico1 (Eprop, ntype);
%*************************************************************************************


%*************************************************************************************
%*       Dibujo de la superficie de daï¿½o                                                %*
%*       Llamado de Rutina dibujar criterio_citerio_daï¿½o1                               %*
figure(1);
set(1,'Name','ANALYSIS OF A DAMAGE MODEL (GAUSS POINT LEVEL)')
hold on;
%dbstop('122')
subplot(1,2,1);
title('Damage surface (principal stresses axes)')
xlabel('\sigma_{1}')
ylabel('\sigma_{2}')
hold on;
grid on;
pbaspect([2 1 1]); %%escala ejes es igual
q=sigma_u/sqrt(E);

hplot = dibujar_criterio_dano1(ce, nu, q , 'b-', MDtype,n );


%%%%%
if  ntype == 2
    SIGMAP = [0 0;SIGMAP] ;
    mstrain = 4 ;
    hplotquiver = [] ;
    STRAIN = zeros(size(SIGMAP,1),4);
    for   iloc = 1:size(SIGMAP,1)-1

        SSS =SIGMAP(iloc,:);
        sigma_bef=[SSS(1) SSS(2) 0  nu*(SSS(1)+SSS(2))];

        SSS =SIGMAP(iloc+1,:);
        sigma_0=[SSS(1) SSS(2) 0  nu*(SSS(1)+SSS(2))];


        %     hplotquiver(end+1) = plot([sigma_bef(1) sigma_0(1)],[sigma_bef(2) sigma_0(2)
]) ;

        plot( sigma_0(1), sigma_0(2),'b*')
        text( sigma_0(1), sigma_0(2),['P=',num2str(iloc)]);

        strain_di =(inv(ce)*sigma_0')';
        STRAIN(iloc+1,:) = strain_di ;
```

```
    end
end

% PLOTTING (PATH)
% ********
% Divide SIGMAP{end} - SIGMAP{end-1} in istep1 steps

hplotp = [];
[ hplotp hplotl]=plotpathNI(SIGMAP,istep);

[strain] = calstrain_NI(istep,STRAIN) ;
```

```
Not enough input arguments.

Error in PlotIniSurf (line 6)
Eprop=[YOUNG_M POISSON 0 YIELD_STRESS];
```

Published with MATLAB® R2017a

```
function [hplotp, hplotl]=plotpathNI(SIGMAP,istep)
% See select_path
% It plots stress path

% Plot iloc-th stretch
% --------------------
PNT = SIGMAP(1,:) ;
hplotp = plot(PNT(1),PNT(2),'ro');
hplotl = [] ;
for iloc = 1:size(SIGMAP,1)-1
    INCSIGMA = SIGMAP(iloc+1,:)-SIGMAP(iloc,:) ;
    for i = 1:istep(iloc)
        PNTb = PNT ;
        % PNT = PNT+INCSIGMA* ;
        PNT = PNT+INCSIGMA/(istep(iloc));
        LINE = [PNTb ; PNT]  ;
        hplotp(end+1) = plot(PNT(1) ,PNT(2),'ro');
        hplotl(end+1) = plot(LINE(:,1) ,LINE(:,2),'r','LineWidth',1,'LineStyle','--');
    end
end
```

```
Not enough input arguments.

Error in plotpathNI (line 7)
PNT = SIGMAP(1,:) ;
```

```matlab
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;
% 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;  %remember that H=0,1, wich is >0, thus hardening
    else
        %exponential!!!!!!!
        A=6; %if A=0; Elastic domain remains the same
        qinf=2;  %qinf<0-->elastic domain diminishes, qinf>r0, it increases.

        q_n1=qinf-(qinf-q_n)*exp(A*(1-rtrial/r_n));

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

```
function [ce] = tensor_elastico1 (Eprop, ntype)
%*************************************************************************
%*       Elastic constitutive tensor                                    %*
%*************************************************************************


%*************************************************************************
%
%*                  G -------->  Shear modulus                          %*
%*                  K -------->  Bulk modulus                           %*
G=Eprop(1)/(2*(1+Eprop(2)));
K=Eprop(1)/(3*(1-2*Eprop(2)));
%*************************************************************************


%*************************************************************************
if(ntype==1)                           % Plane stress
elseif(ntype==2)                       % Plane strain
        ce    = zeros(4,4);            % Init.
        C1=K+(4.0D0/3.0D0)*G;
        C2=K-(2.0D0/3.0D0)*G;
        ce(1,1)=C1;
        ce(2,2)=C1;
        ce(4,4)=C1;
        ce(1,2)=C2;
        ce(1,4)=C2;
        ce(2,4)=C2;
        ce(2,1)=C2;
        ce(4,1)=C2;
        ce(4,2)=C2;
        ce(3,3)=G;
elseif(ntype==4)                       % Tres Dimensiones
end
%*************************************************************************

return
```

```
Not enough input arguments.

Error in tensor_elastico1 (line 11)
G=Eprop(1)/(2*(1+Eprop(2)));
```

Contents


```
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

% ------------------------
% ****************
% INPUTS
% ****************

% YOUNG's MODULUS
% ---------------
YOUNG_M = 20000 ;
% Poisson's coefficient
% -----------------------
POISSON = 0.3 ;
% Hardening/softening modulus
% ---------------------------
HARDSOFT_MOD = 0.1 ;
% 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         = 2         = 3
% ---------------------------------------------------
MDtype =1;
% Ratio compression strength / tension strength
% ---------------------------------------------
n = 3 ;
% SOFTENING/HARDENING TYPE
% ------------------------
HARDTYPE = 'EXP' ; %{LINEAR,EXPONENTIAL} %diferente de linear es suficiente
% 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,:) =[300 400];
SIGMAP(2,:) =[500 400];
SIGMAP(3,:) =[500 0];
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Su=YIELD_STRESS;
alfa=[300 150 600];

%%%%%%%%%%%%%%%%%%
%tipo de caso del problema c
                      num='3';
%%%%%%%%%%%%%%%%%%
switch  num
    case 'def'
    SIGMAP(1,:) =[300 400];
    SIGMAP(2,:) =[500 400];
    SIGMAP(3,:) =[500 0];
    case '1'
    SIGMAP(1,:) =[alfa(1) 0];
    SIGMAP(2,:) =[alfa(1)-alfa(2) 0];
    SIGMAP(3,:) =[alfa(1)-alfa(2)+alfa(3) 0];
    case '2'
    SIGMAP(1,:) =[alfa(1) 0];
    SIGMAP(2,:) =[alfa(1)-alfa(2) (-alfa(2))];
    SIGMAP(3,:) =[alfa(1)-alfa(2)+alfa(3) (-alfa(2)+alfa(3))];
    otherwise
    SIGMAP(1,:) =[alfa(1) alfa(1)];
    SIGMAP(2,:) =[alfa(1)-alfa(2) alfa(1)-alfa(2)];
    SIGMAP(3,:) =[alfa(1)-alfa(2)+alfa(3) alfa(1)-alfa(2)+alfa(3)];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Number of time increments for each load state
% ---------------------------------------
istep = 10*ones(1,nloadstates) ;

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

%  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 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
vpy =
```

```
    'damage variable (d)'
```

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_ei
g(2,2)],'LineWidth',2,'color',colores(1,:),'Marker',markers{1},'MarkerSize',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) ;

subplot(1,2,2)


%%%%%%%%%%%%%%%%%%%%%%%%%%

set(1,'Name','ANALYSIS OF A DAMAGE MODEL (GAUSS POINT LEVEL)')
hold on;

title('STRESS-STRAIN')
xlabel('STRAIN')
ylabel('STRESS')
hold on;
grid on;
%%%%%%%%%

strain11=strain_history(:,1);
stress11=zeros(nloadstates*istep(1)+1,1);

num=zeros(1,nloadstates*istep(1)+1);

for i=1:31
stress11(i)=sigma_v{i}(1,1);
num(i)=i;
end
size(strain11);
size(stress11);

plot(strain11,stress11,'r');
```

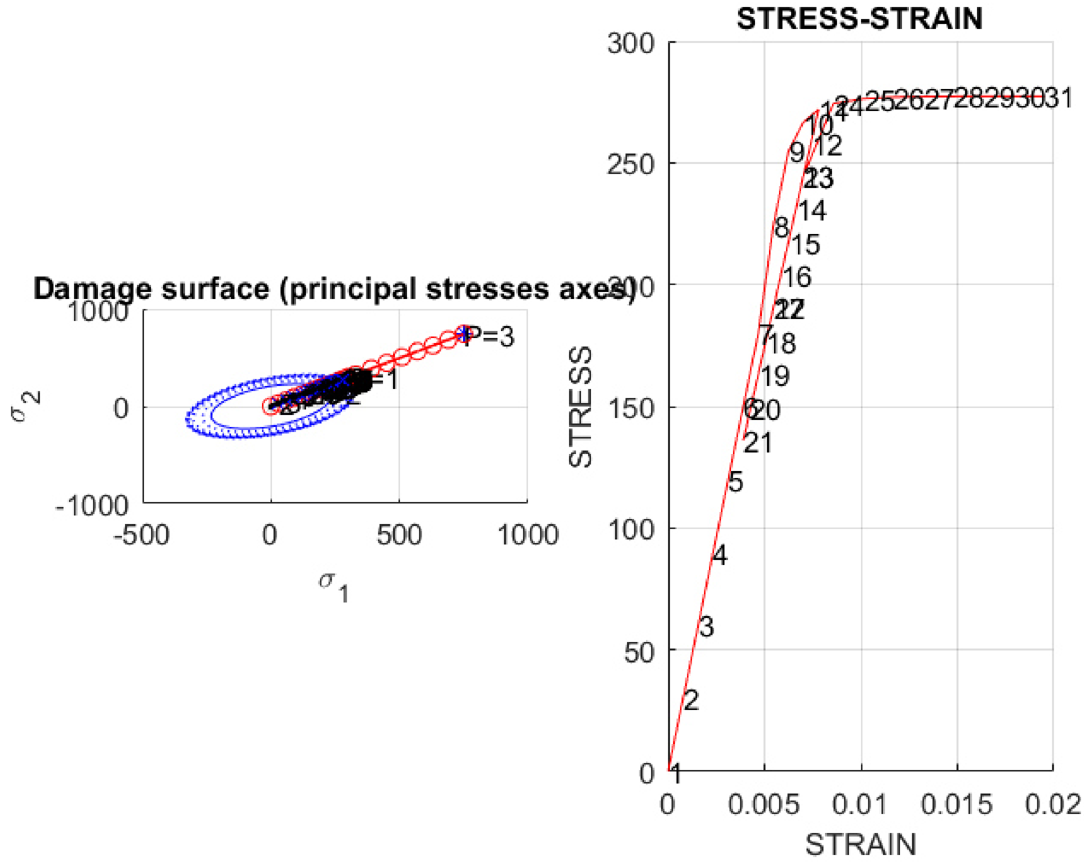
```
ncolores = 3 ;
colores =  ColoresMatrix(ncolores);


for j=1:nloadstates*istep(1)+1
text(strain11(j),stress11(j),['',num2str(num(j))]);
end

sig=26;
stress11(sig)
```

```
ans =

  276.9497
```



```
function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)

%*        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);

if MDtype==1
    tetha=[0:0.01:2*pi];

    D=size(tetha);
    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:pi/2];

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

    radio = zeros(1,Contador+2) ;
    s1    = zeros(1,Contador+2) ;
    s2    = zeros(1,Contador+2) ;

    for i=1:Contador
        radio(i+1)= 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+1)=radio(i+1)*m1(i);
        s2(i+1)=radio(i+1)*m2(i);

    end
    s1(1)= s1(2);
    s2(1)=-200;
    s1(Contador+2)=-100;
    s2(Contador+2)=s1(2);
```

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

elseif MDtype==3

    tetha1=[0:0.01:pi/2];
    tetha2=[pi:0.01:3*pi/2];

    m1=[cos(tetha1) cos(tetha2)];
    m2=[sin(tetha1) sin(tetha2)];

    D=size(m1);
    Contador=D(1,2);

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

    for i=1:Contador/2
        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

    for i=Contador/2+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)*n;
        s2(i)=radio(i)*m2(i)*n;
    end

    %Close surface
    s1(Contador+1)=s1(1);
    hplot =plot(s1,s2,tipo_linea);

end

return
```

```
Not enough input arguments.

Error in dibujar_criterio_dano1 (line 4)
ce_inv=inv(ce);
```

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

        % 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)
    end
end
```

```
Not enough input arguments.

Error in damage_main (line 77)
E      = Eprop(1) ; nu = Eprop(2) ;
```

```
function strain = calstrain_NI(istep,STRAIN)
% See select_path
mstrain = size(STRAIN,2) ;
strain = zeros(sum(istep)+1,mstrain) ;
acum = 0 ;
PNT = STRAIN(1,:) ;
for iloc = 1:length(istep)
    INCSTRAIN = STRAIN(iloc+1,:)-STRAIN(iloc,:);
    for i = 1:istep(iloc)
        acum = acum + 1;
        PNTb = PNT ;
       % PNT = PNT+INCSTRAIN ;
       PNT = PNT + INCSTRAIN/istep(iloc);
        strain(acum+1,:) = PNT ;
    end

end
```

```
Not enough input arguments.

Error in calstrain_NI (line 3)
mstrain = size(STRAIN,2) ;
```

Published with MATLAB® R2017a