ASSINGMENT 1 COMPUTATIONAL SOLID MECHANICS

MARCOS BONIQUET

PART2 (rate dependant models)

Rate dependant+plain strain assumption

d)

Implement “symmetric tension-compression visco-damage model”

To do so, the main thing to take into account is that the damage model will depend on ε_n .
Thus, *rmapdano1* must take this variable in order to deliver it to modelodedano1, which will calculate the strain norm and return it to *rmapdano1*, which will discern load/unload-elastic comparing it to the last internal variable value. This internal variable value defines the elastic domain.

It has also been necessary to add Δ_t at *rmapdano1* to calculate $\mathbb{C}_{alg}$ and also to add Eprop to input variables of *modelodedano1,* which includes α .

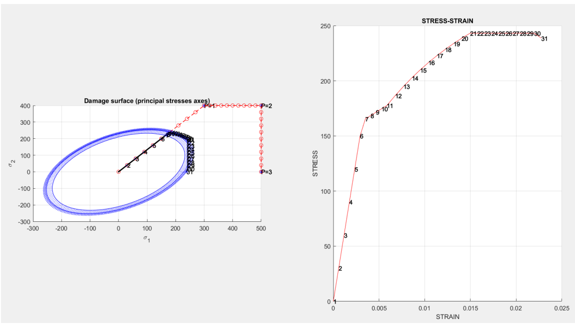
Summarizing, r_{trial} is the strain norm depending on α, now it is a must to calculate damage for either cases, just enable calculation of $\mathbb{C}_{alg.}$

e)

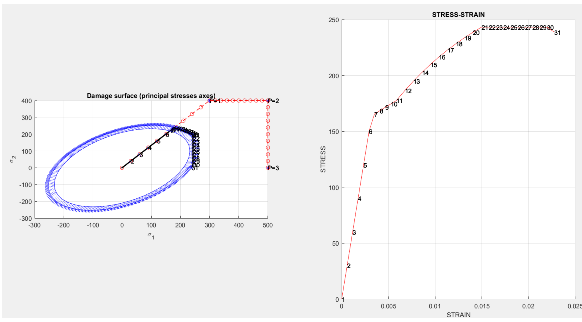
Asses correctness of the implementation

First, we check that at no viscosity (η=0) and strain norm only depending on n+1 strains (α=1) matches the inviscid model. An inviscid model including alfa could have been done (*Backward Euler*).

INVISCID



VISCOUS (η=0) (α=1)



Both are exactly the same.

1. Consider the following cases (linear hardening/softening):

α=0;α=¼,α=½,α=¾,α=1 *(η=0.3)*

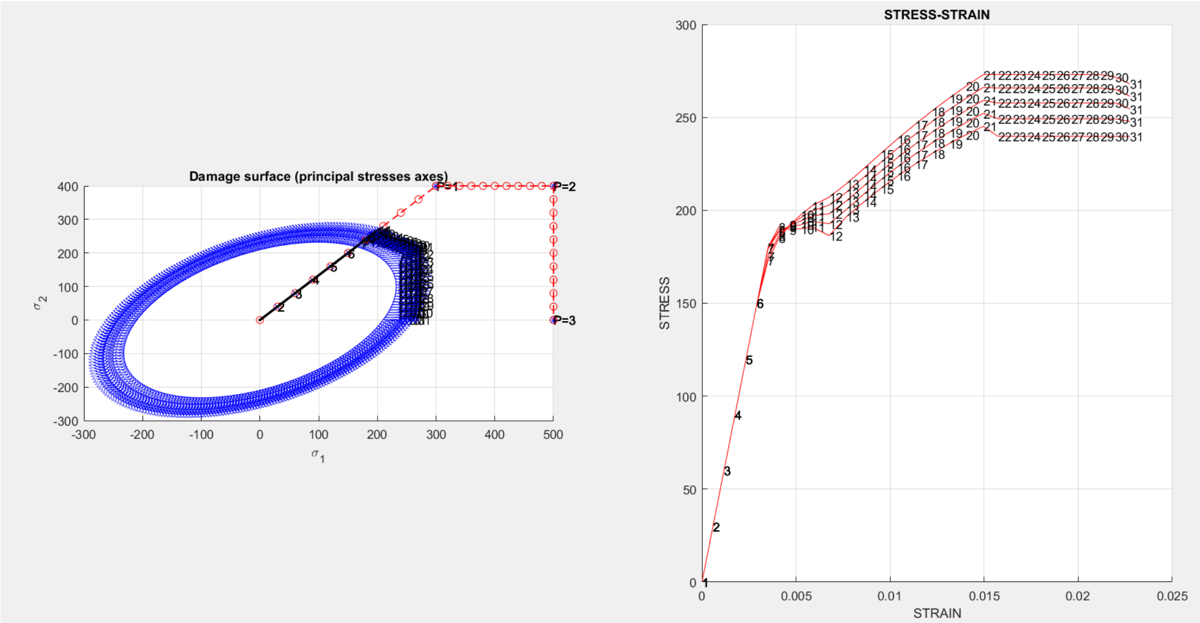
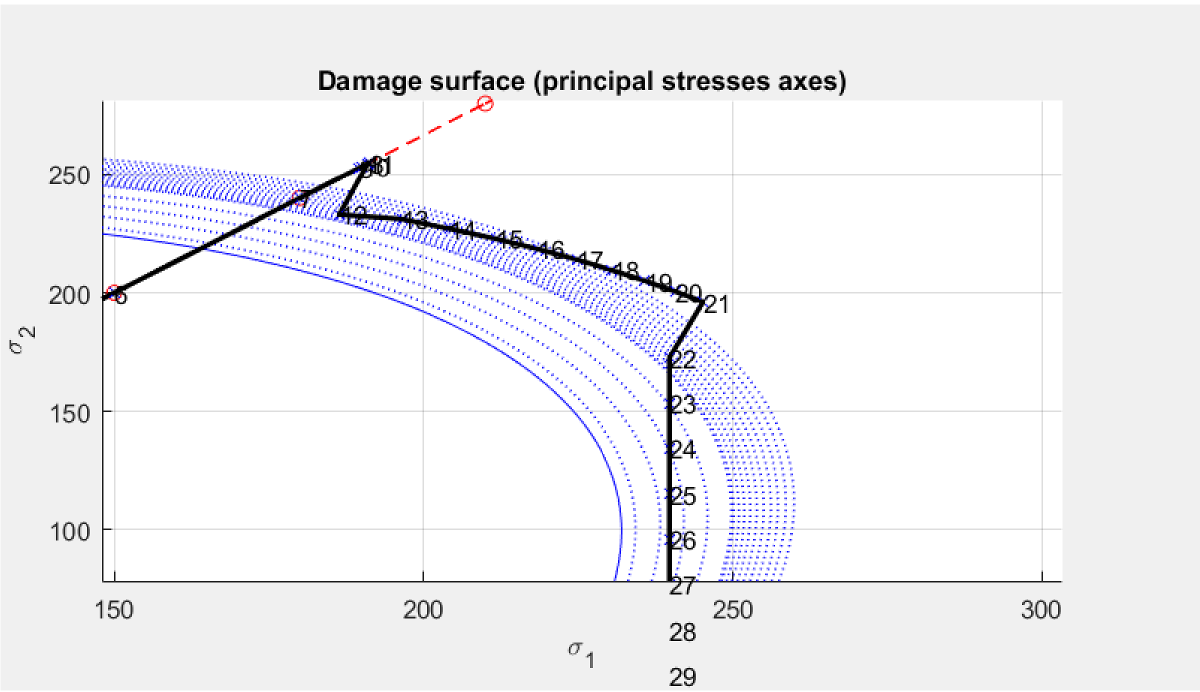


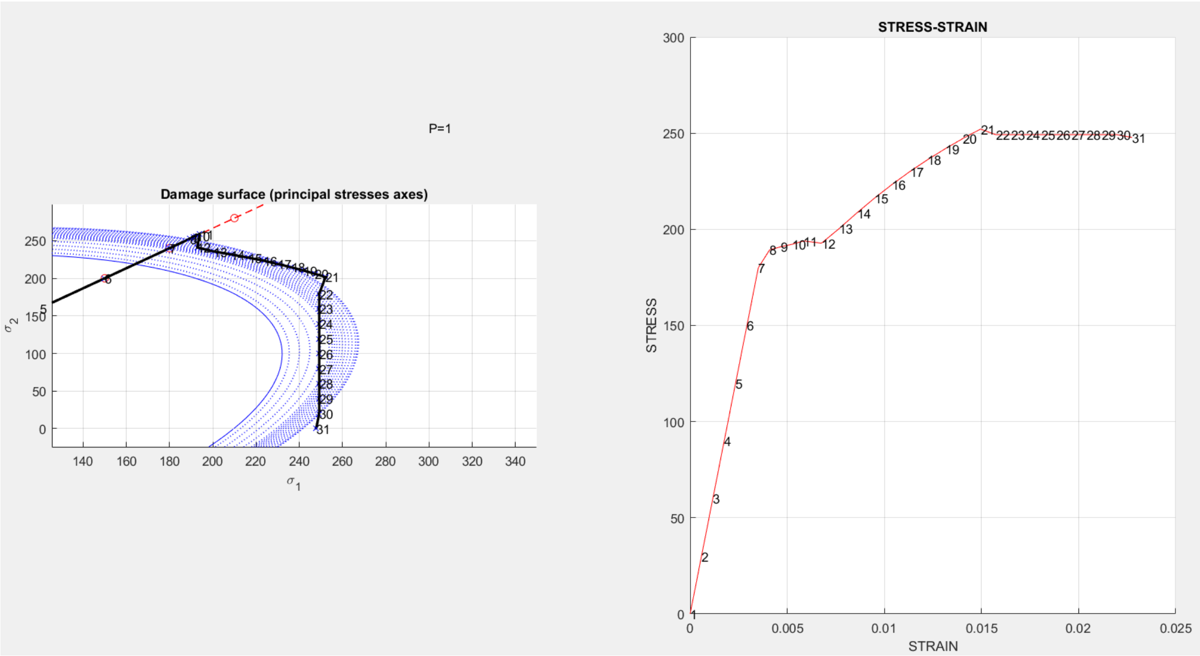
Figure on left: α increases Left to right, and figure on right, down to up

In linear behavior, α doesn't have any effect. However, we ensure stability with **α∊[½,1]** (a^n<1, because a<1). At a load and unload path, an increase of **α translates on an increase on stress for same strain.**

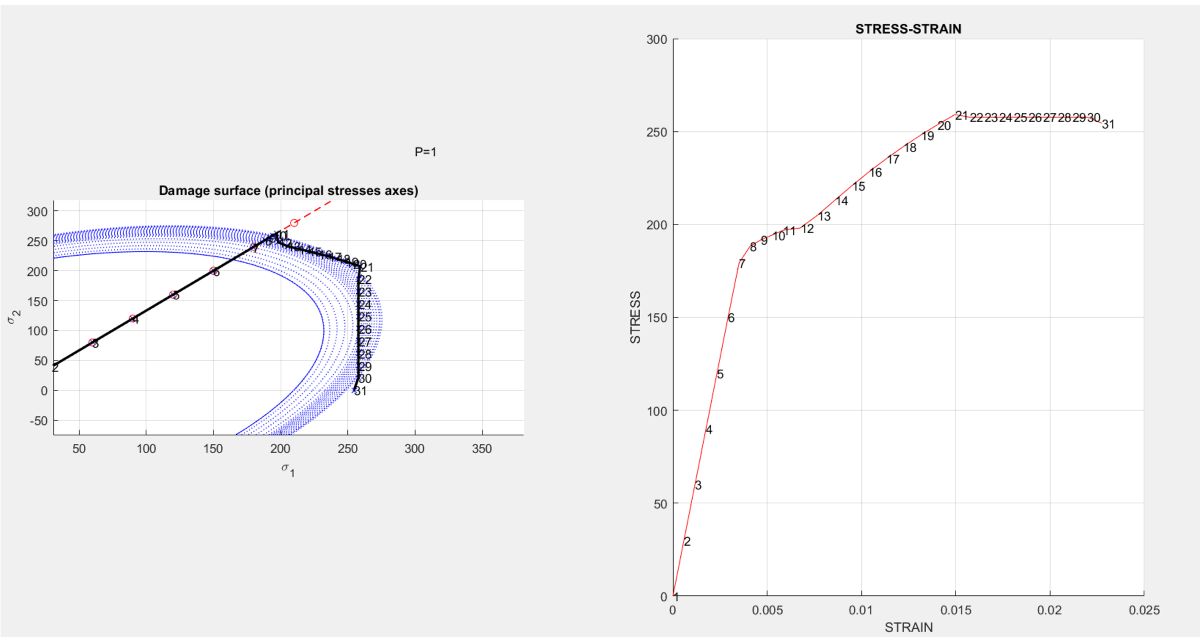
α=0 (non stable)



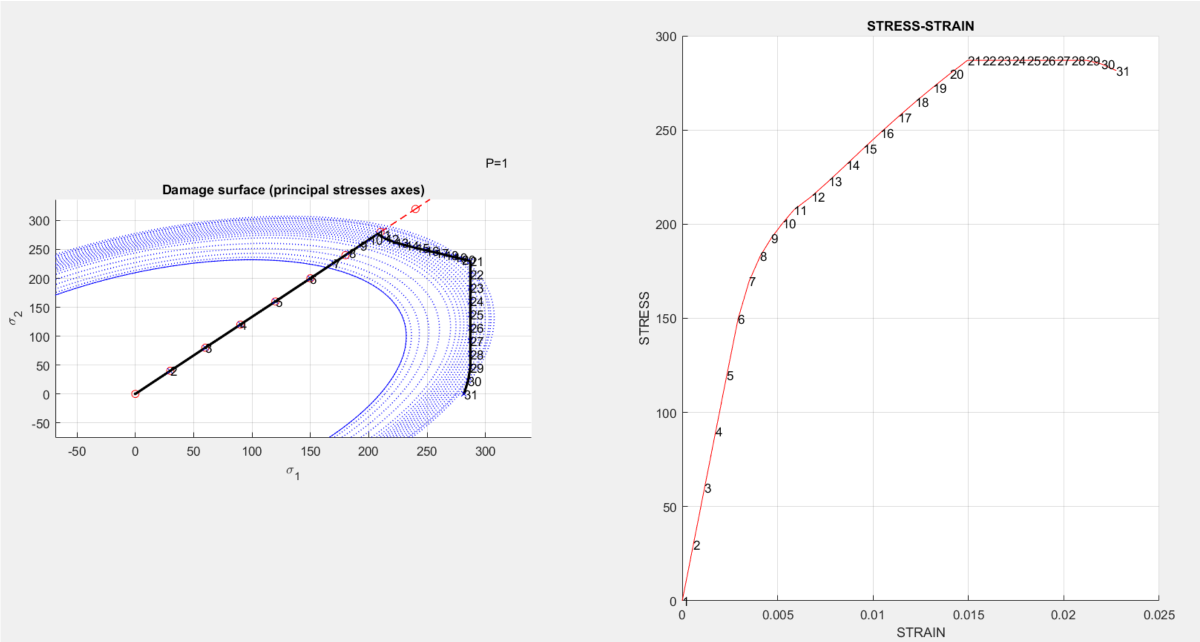
α=¼ (non stable)



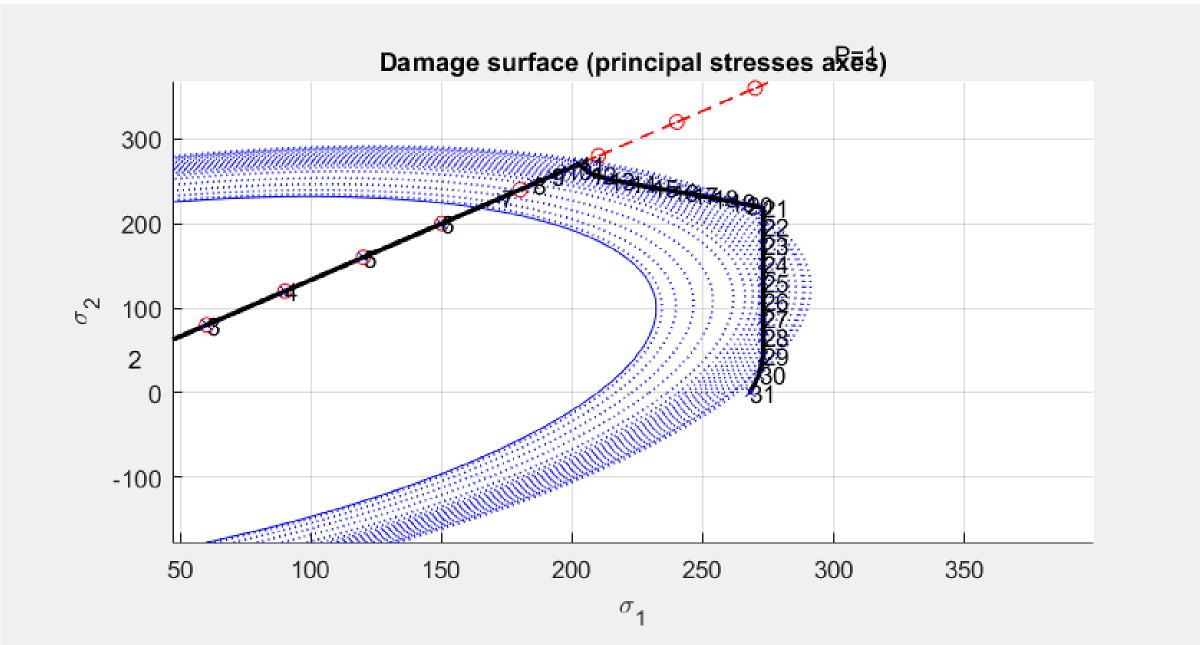
α=1/2 (stable)



α=3/4(stable)



α=1 (stable)



Seems pretty clear that out of range of stability we accumulate enough error that *Modified Elastic Domain* does not match the path, which is impossible.

Different viscosities (and *α=1 -stable-)***:**

η=0,,η=0.1,η=0.2,η=0.3,η=0.4,η=0.5,η=0.6,η=0.7,η=0.8,η=0.9,η=1

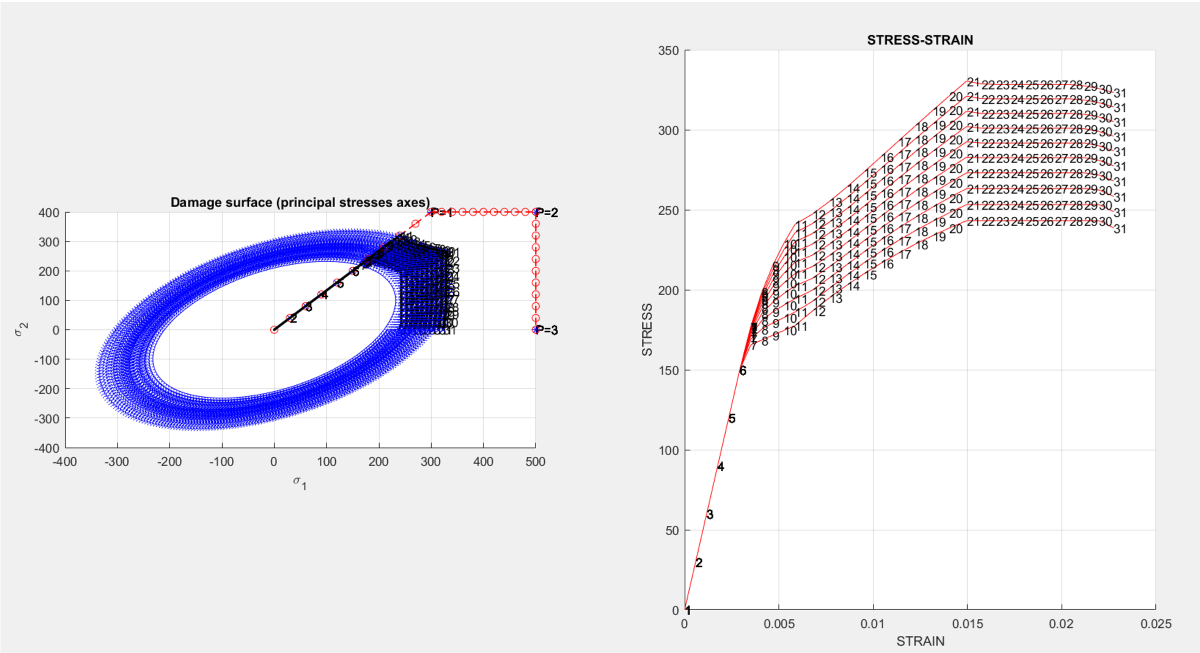
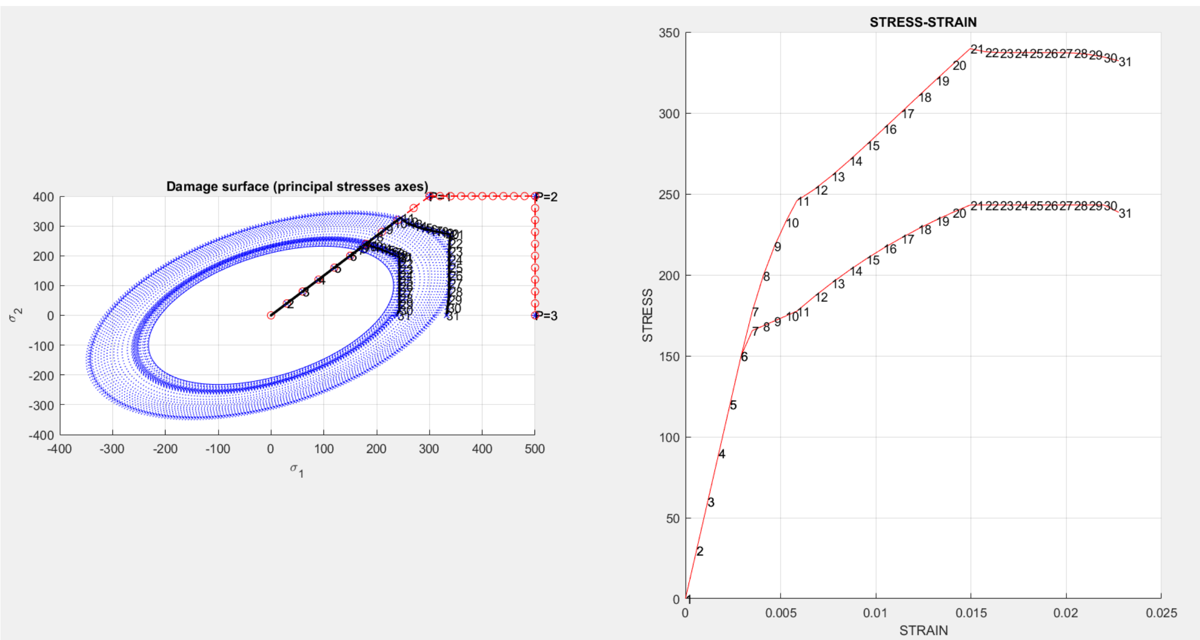


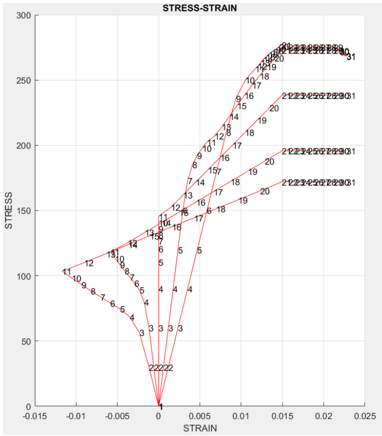
Figure on left: α increases Left to right, and figure on right, down to up

If we study cases η=0 and η=1 we deduce that a bigger viscosity implies a **bigger final elastic domain.** The strain-stress comparison leds to the conclusion that at linear path viscosity has no influence, while on *loading/damage (points 6-21) more viscosity implies that at a same strain the stress is higher. This specially is true at loading (6-11).*



To induce what is the response that strain-stress has by changing έ, we do it indirectly by changing SIGMAP.

α=1 η=0.3



The same stress applied could imply a negative strain if we increase έ.

2.

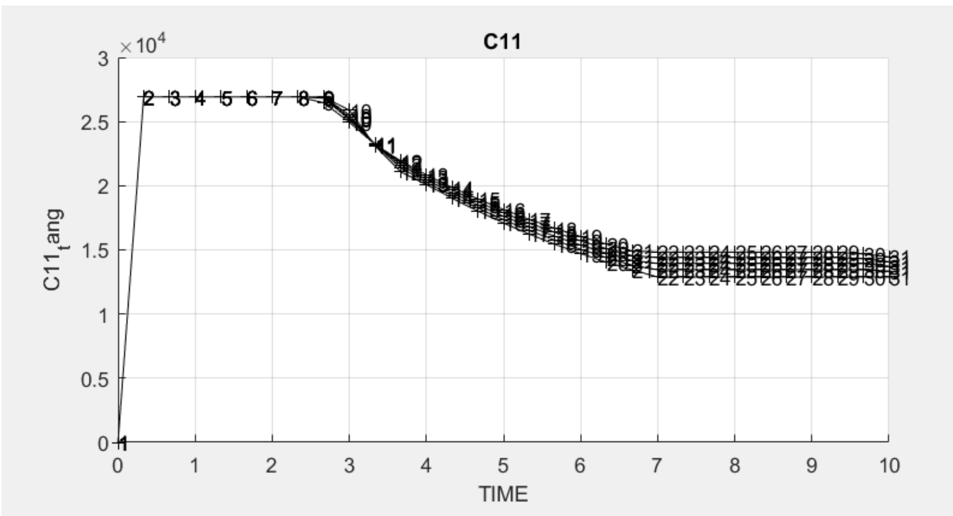
Effects of α values on *C_{11} of C_{tang} and C_{alg} along time:*

We ensured that function *rmapdano1* gives the c_{11} component of *C_{tang} and C_{alg} at any point.*

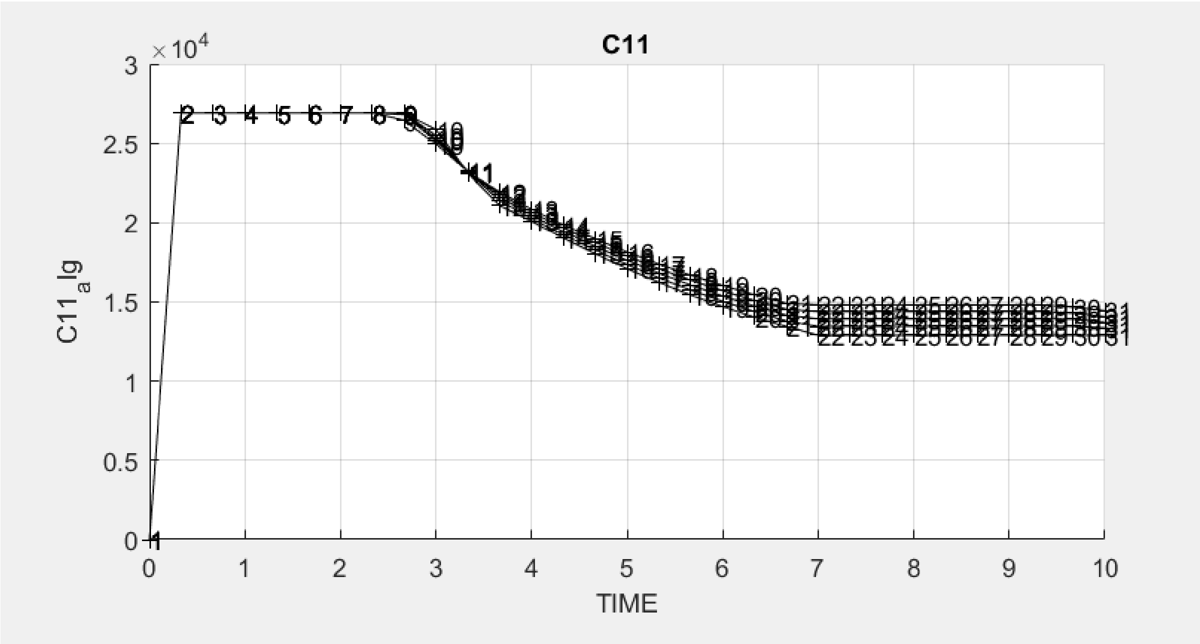
and at *damage_main* function we add:

C_{tang}: vartoplot{i}(4)
C_{alg}: vartoplot{i}(5)

C_{11}^{tang}



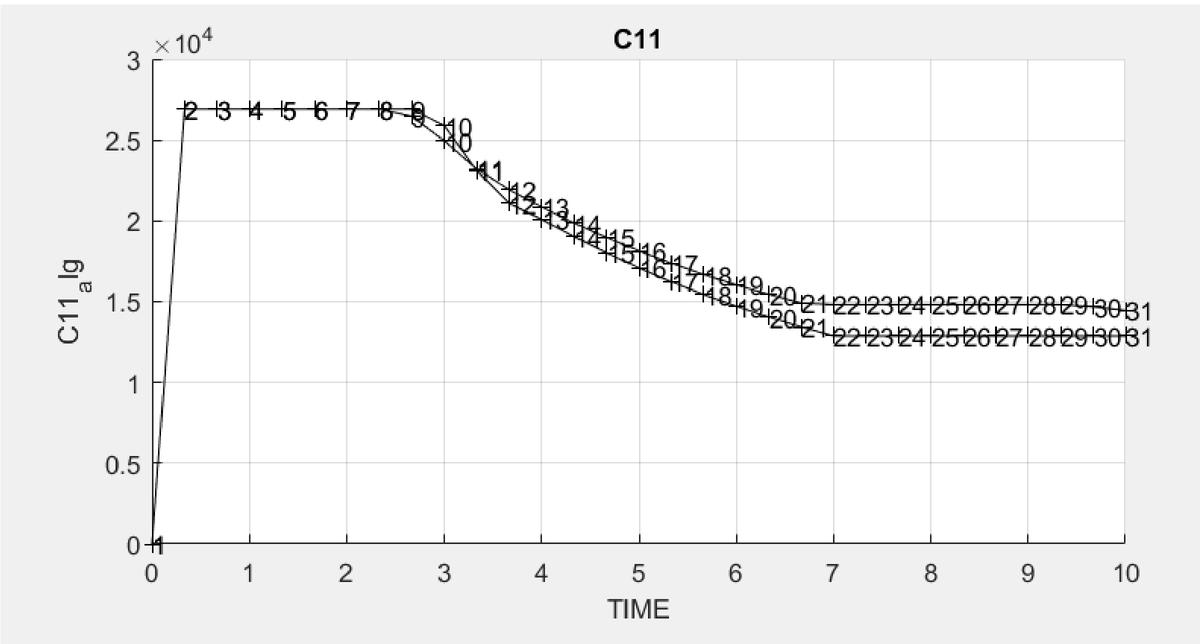
C_{11}^{alg}



For this concrete path, 1-9, which is is elastic, has a linear behaviour with time very fast until it gets constant until damage arrives (node 9). From 9 to 11 we have load, from 11 to 21 we have load (different strain path) and 21 to 31 we have unload, which again stablizies C11.

So, **α** will again not have effect in *elastic domain* but afterachiving damage surface it does.

We have to different behaviours, if we **increase α**, C_{11}^{alg} **reduces** until point 11 and **increases** from this point (change of strain path).



ANEXO 1

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

if Eprop(6)==0

   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
else
    %ONLY SYMMETRIC CASE
```

```
     rtrial_n=sqrt(eps_n*ce*eps_n');
     rtrial_n1= sqrt(eps_n1*ce*eps_n1');
     rtrial=(1-Eprop(8))*rtrial_n+Eprop(8)*rtrial_n1;
    %viscous
end
return
```

```
Not enough input arguments.

Error in Modelos_de_dano1 (line 3)
if Eprop(6)==0
```

Published with MATLAB® R2017a

```
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('C11')
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(2,2,1);
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,:) ;
```

Published with MATLAB® R2017a

```
function [sigma_n1,hvar_n1,aux_var,ce_n1] = rmap_dano1 (eps_n1,hvar_n,Eprop,ce,MDtype,n,ep
s_n,delta_t)

%*************************************************************************************
%*                                         *
%*           Integration Algorithm for a isotropic damage model
%*
%*                                                                                  *
%*            [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n1,hvar_n,Eprop,ce)        *
%*                                                                                  *
%* INPUTS               eps_n1(4)   strain (almansi)    step n+1                    *
%*                                  vector R4    (exx eyy exy ezz)                  *
%*                      hvar_n(6)   internal variables , step n                     *
%*                                  hvar_n(1:4) (empty)                       *
%*                                  hvar_n(5) = r  ; hvar_n(6)=q                    *
%*                      Eprop(:)    Material parameters                             *
%*
%*                      ce(4,4)     Constitutive elastic tensor                     *
%*                                                                                  *
%* OUTPUTS:             sigma_n1(4) Cauchy stress   , step n+1                      *
%*                      hvar_n(6)   Internal variables , step n+1
 *
%*                      aux_var(3)  Auxiliar variables for computing const. tangent tensor
 *
%*************************************************************************************

hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5) ;
%************************************************************************************
%************************************************************************************
%*       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,Eprop,eps_n);
%************************************************************************************
%************************************************************************************
%*   Ver el Estado de Carga                                                         %*
%*   --------->    fload=0 : elastic unload                                         %*
%*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)       %*
fload=0;

if Eprop(6)==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;
%**************************************************************************************
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
else  %%%%%%%%%%%%%%%%%%VISCOUS%%%%%%%%%%%%%%%%%%%%%%%%
if(rtrial > r_n)
    %*   Loading

    etha=Eprop(7);
    alfa=Eprop(8);

    fload=1;
    delta_r=rtrial-r_n;
    r_n1= (etha-delta_t*(1-alfa))*r_n/(etha+alfa*delta_t)+rtrial*delta_t/(etha+alfa*delta_
```

```
t);%%%%
    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
dano_n1   = 1.d0-(q_n1/r_n1);
rtrial_n1= sqrt(eps_n1*ce*eps_n1');
ce_tan=(1-dano_n1)*ce;
ce_alg=ce_tan+alfa*delta_t*(H*r_n1-q_n1)/((etha+alfa*delta_t)* rtrial_n1*(r_n1)^2);

else

    %*     Elastic load/unload
    fload=0;
    r_n1= r_n  ;
    q_n1= q_n  ;
    dano_n1   = 1.d0-(q_n1/r_n1);
    ce_alg=(1-dano_n1)*ce;
    ce_tan=ce_alg;
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)= q_n1 ;
%**************************************************************************
%**************************************************************************
%* Auxiliar variables                                                     %*
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
aux_var(3)=ce_tan(1,1);
aux_var(4)=ce_alg(1,1);

ce_n1=ce_alg;
%%%%%*aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
%**************************************************************************
end
```

```
Not enough input arguments.

Error in rmap_dano1 (line 24)
hvar_n1 = hvar_n;
```

```
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)));
```

Published with MATLAB® R2017a

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 = 'LINEAR' ; %{LINEAR,EXPONENTIAL} %diferente de linear es suficiente
% VISCOUS/INVISCID
% ------------------------
VISCOUS = 'YES' ;
% Viscous coefficient ---- YES or otherssss
% ------------------------
eta = 0.3 ;
% TimeTotal (initial = 0) ----
% ------------------------
TimeTotal = 10 ;
% Integration coefficient ALPHA
% ------------------------
ALPHA_COEFF = 1 ;
% 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='def';
%%%%%%%%%%%%%%%%%%
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)','C11_t
ang','C11_alg'};

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

```
vpy =
```

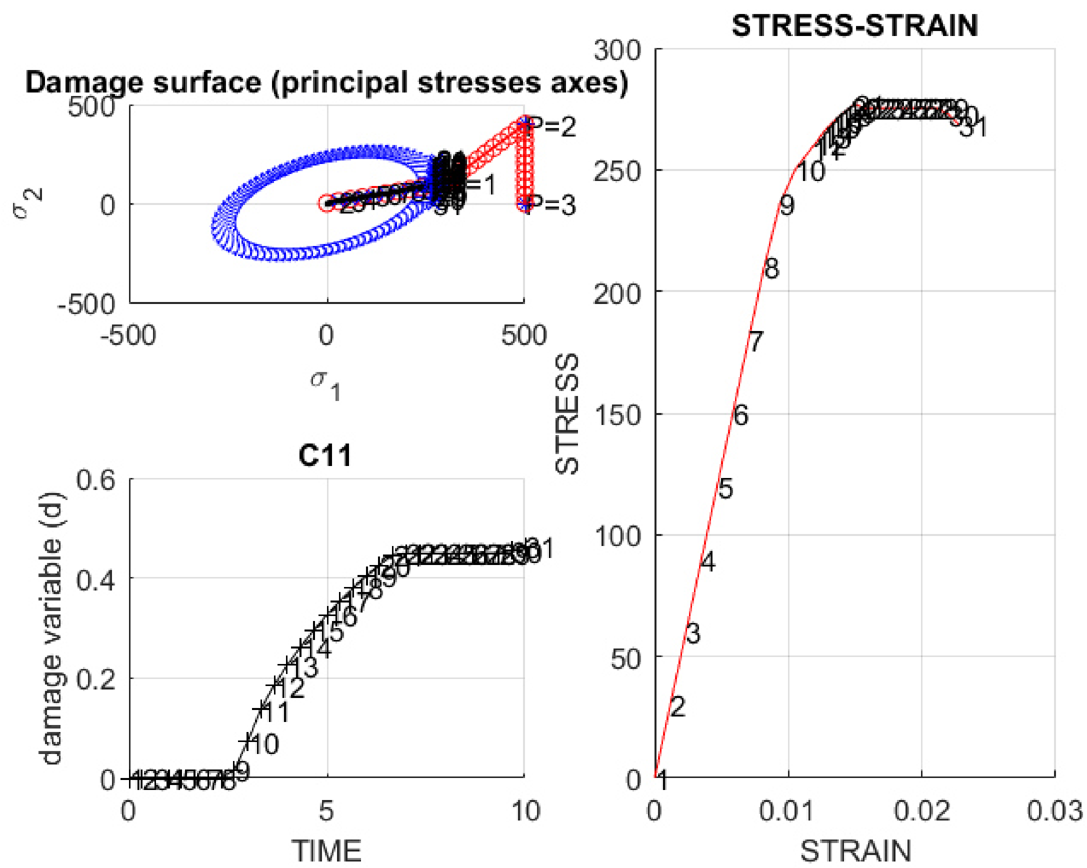
```
'damage variable (d)'
```

Plot Initial Damage Surface and effective stress path

```
 ALFA=1;
 ETA=0.3;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 SIGMAP(1,2)=100;  % por defeto es 400
for i=1:1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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 ALFA]             ;


% 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 REAL PATH
% -------

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



%vpy = 'C11_alg' ;
DATA.sigma_v    = sigma_v ;
DATA.vartoplot  = vartoplot;
DATA.LABELPLOT=LABELPLOT;
DATA.TIMEVECTOR = TIMEVECTOR  ;
DATA.strain = strain_history ;


%%%%%%%%%%plots leabelplot


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);

  ALFA=ALFA+1;%%%%%%%%%%%%%%%%%%%%%%%%%%
  %ETA=ETA+1;
 % SIGMAP(1,2)=SIGMAP(1,2)+1000;
end
```



```
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
   menu({'VISCOUS'},'STOP');
else
   menu({' NON VISCOUS'},'STOP');
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_n=strain(i-1,:);%%%%%%%%%%%! added for viscous
        eps_n1 = strain(i,:) ;
        %*********************************************************************************
*****
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var,ce_n1] = rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n,eps_n
,delta_t);
        % 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)
```

```
        vartoplot{i}(4) = aux_var(3) ; %  C11_TANG VISCOUS
        vartoplot{i}(5) = aux_var(4); %  C11_ALG VISCOUS
    end
end
```

```
Not enough input arguments.

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

Published with MATLAB® R2017a

```
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) ;
```