DAMAGE MODELS

DAVID A. ENCALADA*

*Universidad Politécnica de Cataluña
Campus Norte UPC, 08034 Barcelona, Spain
e-mail: david.encalada@estudiant.upc.edu

Key words: constitutive model, damage, visco-damage

Abstract. This document presents the implementation of rate independent and dependent continuum damage models using a Matlab code. First we run three different stress paths to evaluate the correctness of rate independent models. Then a parametric study of rate dependent model or visco-damage model. The results show a good coherence as expected.

1 INTRODUCTION

Continuum damage constitutive models simulate the behaviour of materials that have loss of stiffness and irreversible degradation. To understand and valid this theory is important to perform computational test simulations.

The primary objective of this study is the development and implementation of the code and algorithm of damage models using Matlab. First we will presents rate independent models and compare some the results between three different stress paths.

The second part of the study involves rate dependent models. We do a parametric study to verified the correctness of the implementation and assess the influence of different parameters as viscosity (η), strain rate and time-integration method.

2 RATE INDEPENDENT MODELS

The code analyses plane strain cases. Symmetric, non-symmetric tension-compression and tension-only damage models are include in the Matlab code.

The material properties employed in the analysis are shown in the table 1. We do not include de units of each parameter because we consider they are consistent between the inputs and the outputs. This parameter will be used in all the problems and in the case that they vary, the value used will be indicated.

In figure 1 shows the initial damage surfaces of the three rate independent models in the stress space. The value of σ_y controls the size of the surface and ν controls the shape. The others parameters control the evolution of the damage surface when the stress tend to be beyond the surface.

To assess the code, three stress paths are used and we observe the evolution of the outputs and results. The stress paths three increments as indicated:

Table 1: Material properties

Parameter	Value
E	20000
ν	0.3
σ_y	250
q_∞	2.5
H	0.5

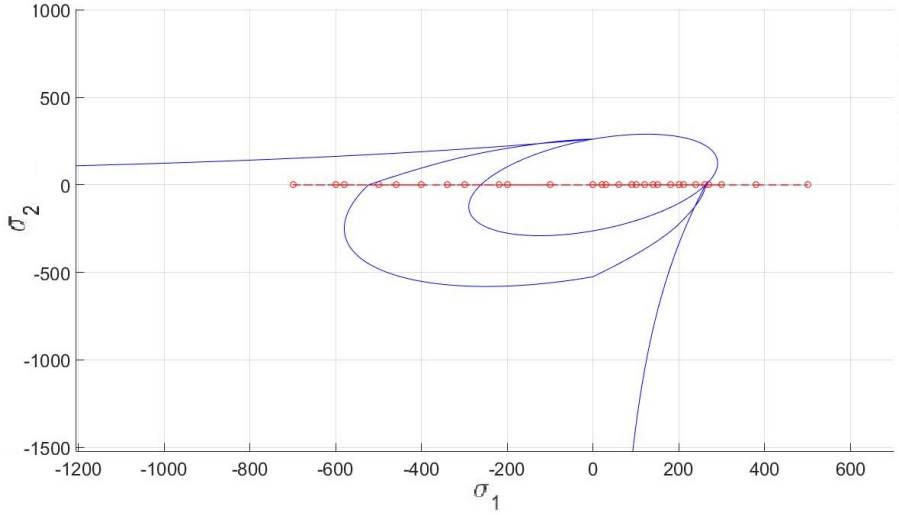


Figure 1: Initial damage surfaces in stress spaces.

1. Uniaxial stress path

$$\Delta\sigma_1^{(1)} = 300 \quad ; \quad \Delta\sigma_2^{(1)} = 0$$
$$\Delta\sigma_1^{(2)} = -1000 \quad ; \quad \Delta\sigma_2^{(2)} = 0$$
$$\Delta\sigma_1^{(3)} = 1500 \quad ; \quad \Delta\sigma_2^{(3)} = 0$$

2. Uniaxial and biaxial stress path

$$\Delta\sigma_1^{(1)} = 300 \quad ; \quad \Delta\sigma_2^{(1)} = 0$$
$$\Delta\sigma_1^{(2)} = -1000 \quad ; \quad \Delta\sigma_2^{(2)} = -1000$$
$$\Delta\sigma_1^{(3)} = 1500 \quad ; \quad \Delta\sigma_2^{(3)} = 1500$$

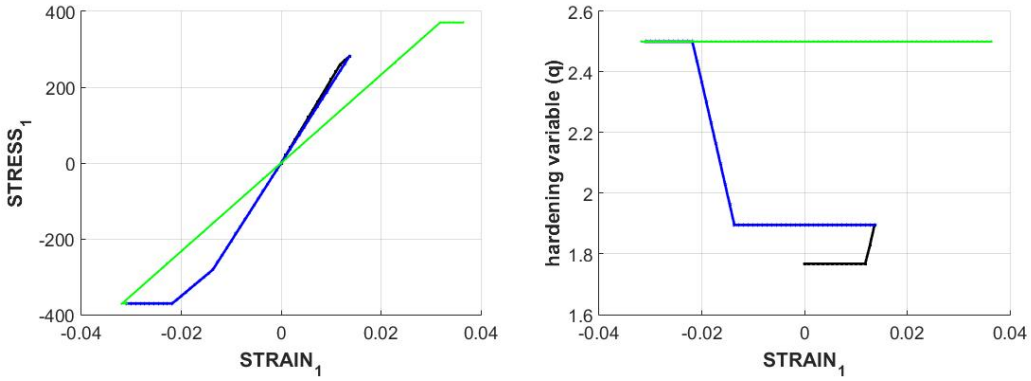
3. Biaxial stress path

$$\Delta\sigma_1^{(1)} = 300 \quad ; \quad \Delta\sigma_2^{(1)} = 300$$
$$\Delta\sigma_1^{(2)} = -1000 \quad ; \quad \Delta\sigma_2^{(2)} = -1000$$
$$\Delta\sigma_1^{(3)} = 1500 \quad ; \quad \Delta\sigma_2^{(3)} = 1500$$

2.1 STRESS PATH 1

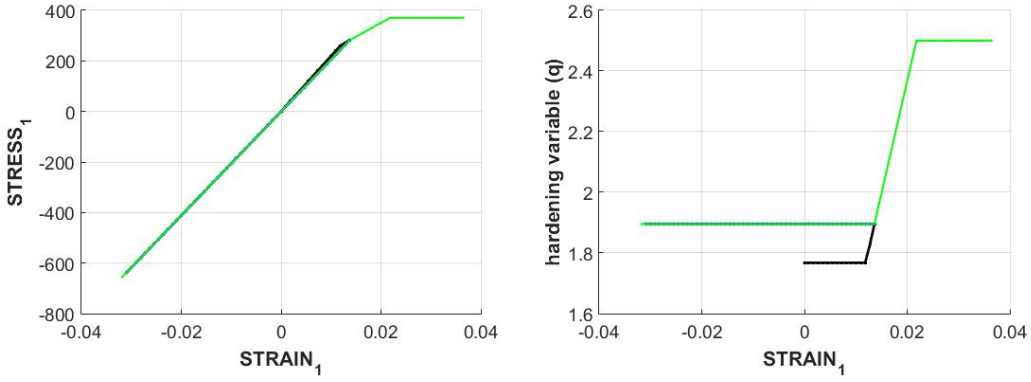
This problem consider uniaxial loading and unloading, for this reason we observe the evolution of stresses in direction 1 against the strain in direction 1 (see figures 2a, 3a and 4a). In figures 2b, 3b and 4b show the evolution of the hardening variable q.

We observe that in all the models the stress σ_1 do not goes further than 400 and q grows until reach the value of 2.5, because the parameter q_∞ limits the expansion of the damage surfaces. We observe that q is steady while the stress is inside the damage surface. For instance, in figure 3b, when σ_1 is in compression the value of q remains constant.



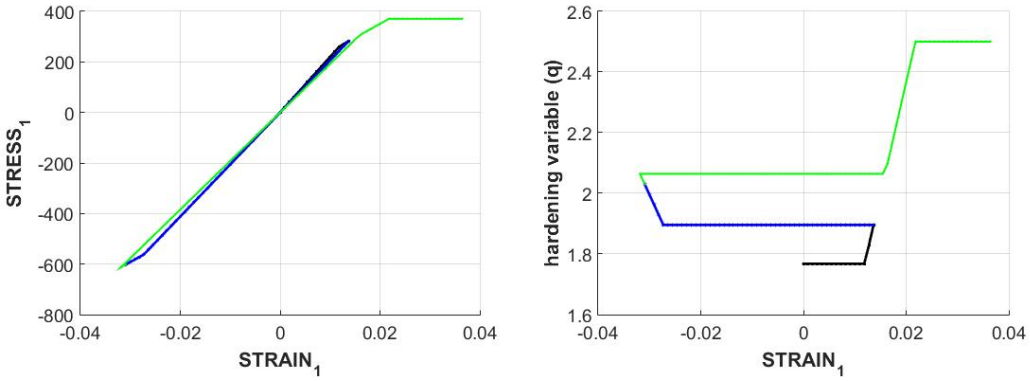
(a) (b)

Figure 2: Symmetric damage model for case 1. (a) Principal major stress evolution. (b)Hardening variable evolution.



(a) (b)

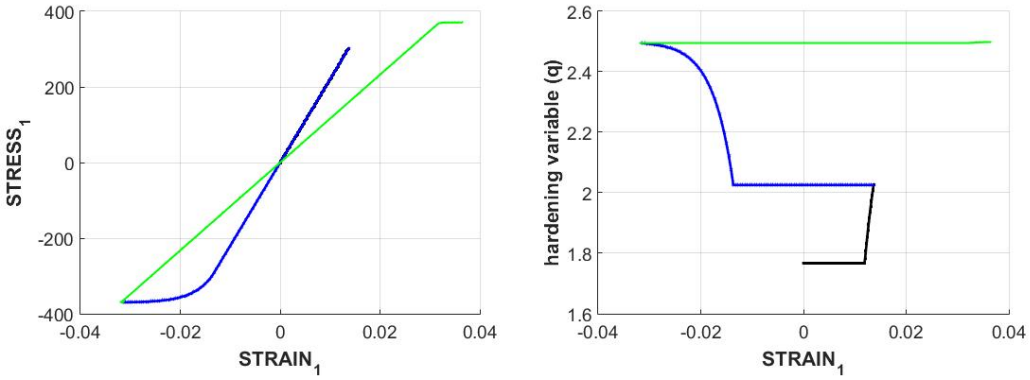
Figure 3: Only-tesion damage model for case 1 (a) Principal major stress evolution. (b)Hardening variable evolution.



(a) (b)

Figure 4: Non-symmetric tension-compression damage model for case 1. (a) Principal major stress evolution. (b)Hardening variable evolution.

Additionally, we compute the exponential hardening with the same parameters, except we added the parameter $A = 3$. The hardening modulus is not constant and his rate of change depends on A. Figure 5 shows the results of symmetric damage model with exponential hardening. We observe that in this model the value of q does not grow linearly and the evolution of σ_1 is not linear.



(a) (b)

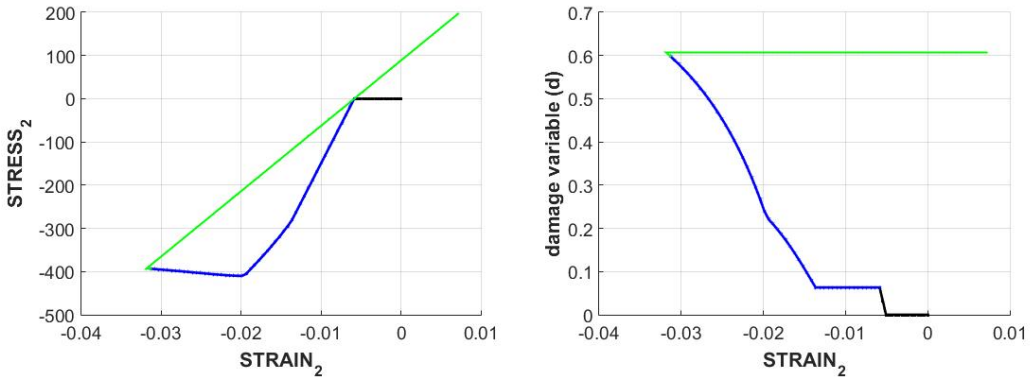
Figure 5: Symmetric damage model with exponential hardening for case 1. (a) Principal major stress evolution. (b)Hardening variable evolution.

2.2 STRESS PATH 2

This stress path considers a first tensile unixial load, afterwards the loading/unloading changes to biaxial. For this case we evaluate the stresses in 2-direction against the strains in 2-direction as shown figures 6b, 7b and 8.

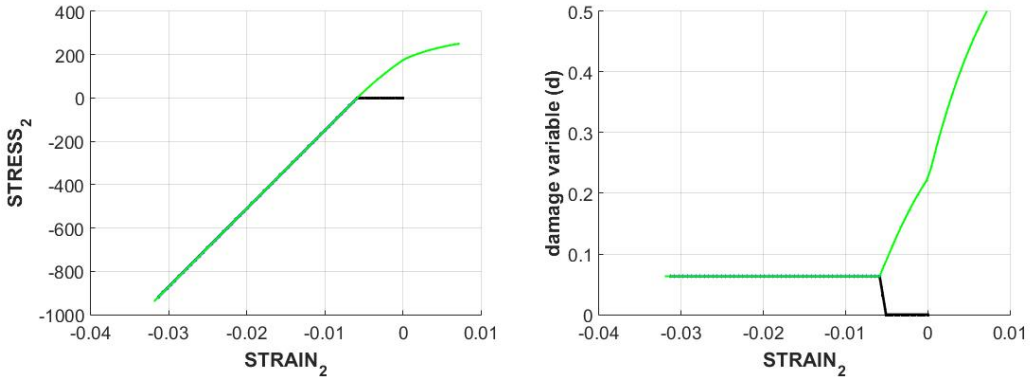
We observe that that in all cases there is a strain ε_2 and the stress σ_2 remains constant. This strain evolution is due to the poisson effect during the uniaxial tensile loading.

In figures 6b, 7b and 8 we plotted the evolution of the damage variable d. The value of d begin at 0 and increases when the stresses tend to be outside the elastic surface. This parameter evolves from 0 to 1. However, in this example as we restrict size of the damage surface with the parameter $q_\infty = 2.5$ and the value of d cannot reach 1.



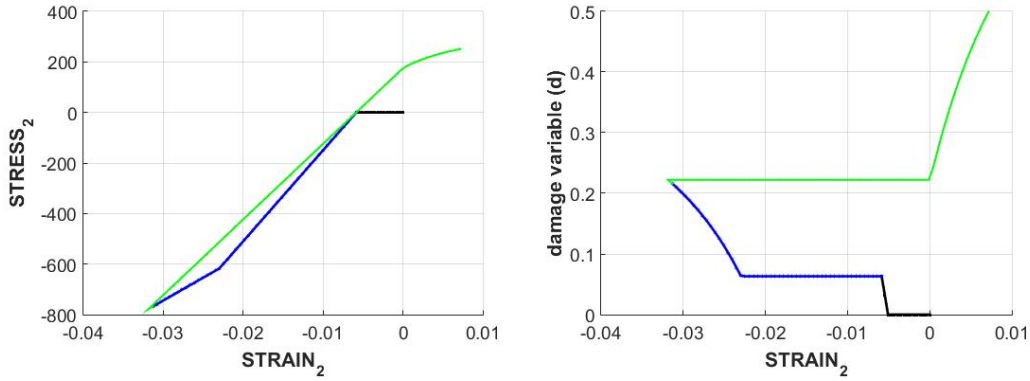
(a) (b)

Figure 6: Symmetric damage model for case 2. (a) Principal major stress evolution. (b)Damage variable evolution.



(a) (b)

Figure 7: Only-tesion damage model for case 2. (a) Principal major stress evolution. (b)Damage variable evolution.

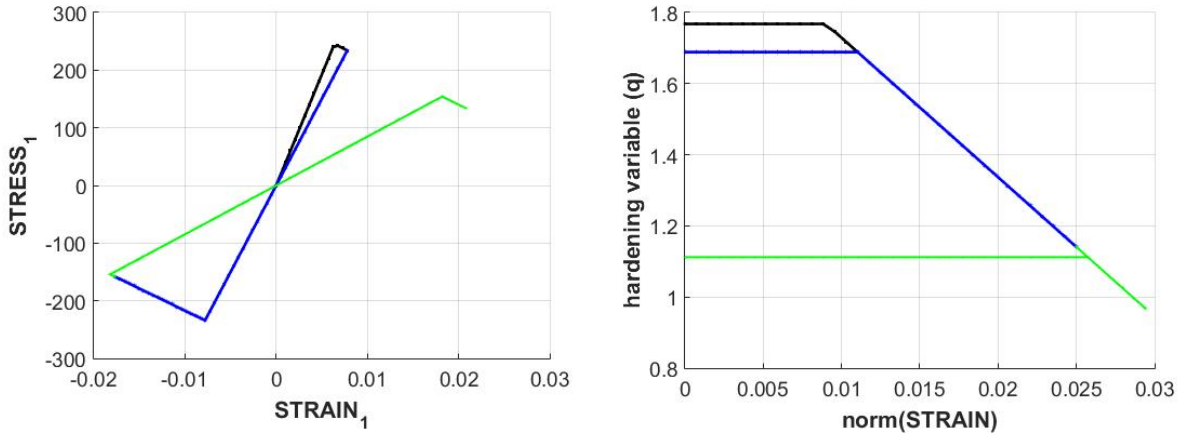


(a) (b)

Figure 8: Non-symmetric tension-compression damage model for case 2 (a) Principal major stress evolution. (b) Damage variable evolution.

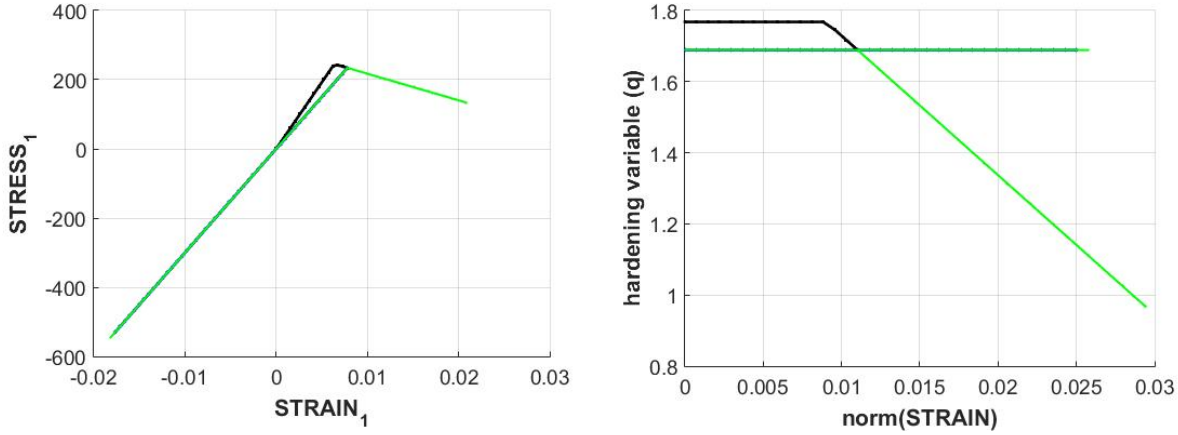
2.3 STRESS PATH 3

In this case we consider softening. To consider softening the value of H has to negative. In this example we change H to -0.2 and q_∞ to 0.5. Figures 9a, 10a and 11a show stress evoltion against strain in 1-direction. The stress decrease after the damage surface is reached due to the the surface size decreases. Other way to observe the softening phenomenon is with that the hardening variable q decreases as shown figures 9b, 10b and 11b.



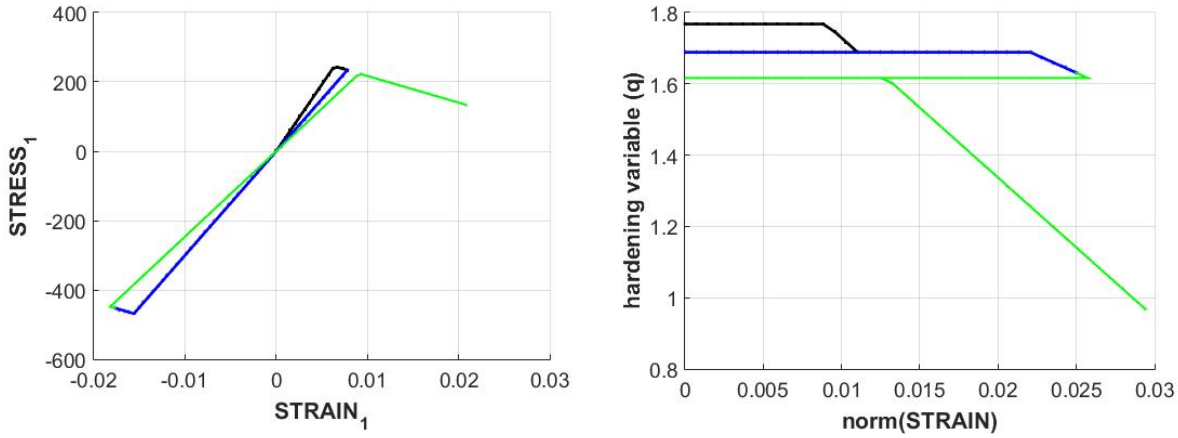
(a) (b)

Figure 9: Symmetric damage model for case 3. (a) Principal major stress evolution. (b)Hardening variable evolution.



(a) (b)

Figure 10: Only-tesion damage model for case 3.(a) Principal major stress evolution. (b)Damage variable evolution.



(a) (b)

Figure 11: Non-symmetric tension-compression damage model for case 3. (a) Principal major stress evolution. (b) Damage variable evolution.

3 RATE DEPENDENT MODELS

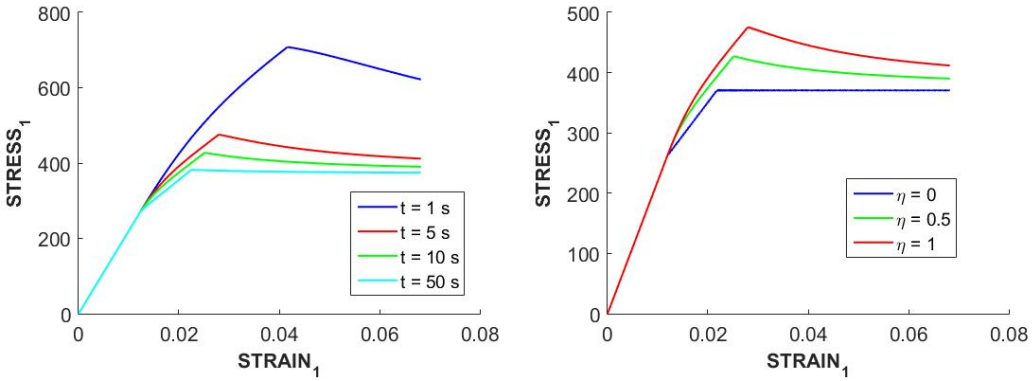
Finally we implement in the Matlab code the integration algorithm for symmetric isotropic visco-damage model. The added a new function called `rmap_dano2.m`. The principal scripts of the code will be attached as an annex.

To do the parametric study we take as base the same parameters of the table 1 and additionally $\eta = 0.5$ and time of 10 s. The stress path to evaluate this model is uniaxial tensile loading to simplify the analysis.

1. Uniaxial stress path

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

Figure 12 show the obtained stress-strain curves for different loading time (strain rate) and viscosity parameter η. As shows figure 12a, high strain rates higher stress σ_1 are reached. In figure 12b shows that high values of *eta* reach higher values of stresses σ_1.



(a) (b)

Figure 12: Initial damage surfaces in stress spaces.

In figure 13 shows the effect of α on time evolution of C_{11} component of the tangent and algorithmic constitutive operators. With a high number of time steps the solution with different values of α the result tend to be equal and the solution is more accurate. The differences is more visible with a smaller number of time steps as shown figure 13b. The value of C_{11} remains steady in the elastic range, then it increases slightly and falls significantly towards zero.

4 CONCLUSIONS

This work implemented the local constitutive response of continuum damage models. The results show good coherence. The evolution of hardening and damage variables (q and d) behaves as expected. For instance the value of d always grows when the stress state is or tend to be

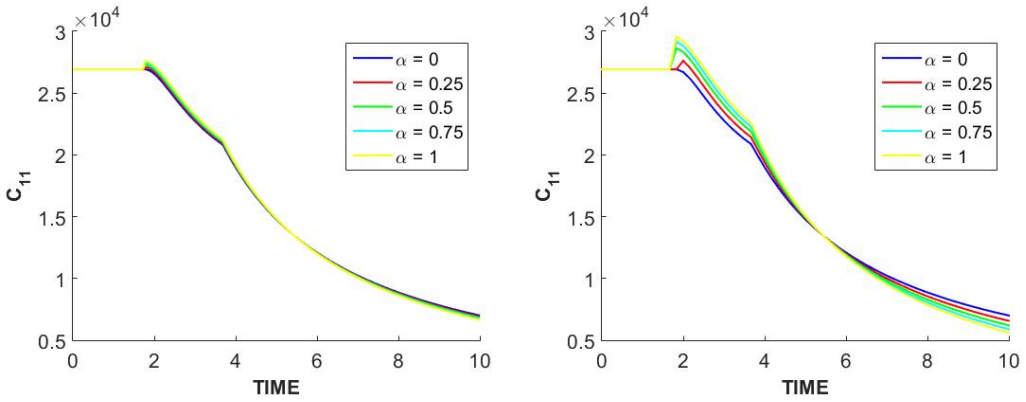


Figure 13: Evolution of C_{11} in time. (a)With 300 time steps. (b) With 30 time steps.

outside the elastic regime. The value of q increases on hardening behaviour and decreases on softening behaviour.

The model with exponential hardening was implemented successfully. This was confirmed with the evolution of hardening variable evolution and consequently the stress-strain curves.

The time steps has influence on the accuracy of rate dependent models.

5 ANNEX

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

                    % ------------------------


current_dir = cd ;
if isunix
    if exist('CALLBACK_main.m') == 2
    else
        addpath([current_dir,'/AUX_SUBROUTINES']);
        % addpath(current_dir);
    end

    if exist('damage_main.m') == 2
    else
        addpath([current_dir]);
        % addpath(current_dir);
    end

    pathdata = [current_dir,'/AUX_SUBROUTINES/WSFILES/'];

else
    if exist('CALLBACK_main.m') == 2
    else
        addpath([current_dir,'\AUX_SUBROUTINES']);
    end
     if exist('damage_main.m') == 2
    else
        addpath([current_dir]);
        % addpath(current_dir);
    end
    % addpath(current_dir);
    pathdata = [current_dir,'\AUX_SUBROUTINES\WSFILES\'];
end

if exist(pathdata,'dir') == 0
    mkdir('\AUX_SUBROUTINES\WSFILES\') ;
end


% ***************
% INPUTS
% ***************
% ----------------------------
% OTHER INPUTS (Graphic Inputs)
```

```
 56 % ----------------------------
 57 Inc      = [0 -0.040 0 0] ;
 58 NameFileExec = 'CALLBACK_main';
 59 NameFileExecP = 'redraw_path';
 60 Position = [0.01 0.9500 0.08 0.030] ;
 61 PTC = {'SYMMETRIC','ONLY TENSION','NON-SYMMETRIC'};
 62 TP = {'PLANE STRESS','PLANE STRAIN','3D'} ;
 63 ce = 0 ;
 64 nnls_s    = 3 ; % Number of load states
 65 mstrain = 4 ;  % Number of components of strain vector
 66 mhist   = 6 ; % Number of componets of historical variables vector
 67 shownumber = 'YES'     ;
 68 axiskind   = 'NON-AUTO'   ;
 69 axislim    = [-500 500 -500 500] ;
 70 ErasePrPlot = 'YES' ;
 71 wplotx      ={'STRAIN_1','STRAIN_2','|STRAIN_1|','|STRAIN_2|','norm↙
(STRAIN)','TIME'};
 72 vpx         = 'STRAIN_1' ;
 73 wploty      ={'STRESS_1','STRESS_2','|STRESS_1|','|STRESS_2|','norm(STRESS)'};
 74 vpy         = 'STRESS_1' ;
 75 splitwind   = 'YES' ;
 76 HARDLIST    = {'LINEAR','EXPONENTIAL'} ;
 77 % Inc      = [0 -0.040 0 0] ;
 78 % ----------------------------------------
 79 %Set of variables (to be stored in DATA)
 80 % ----------------------------------------
 81
 82 %  Workspace name
 83 NameWs = [pathdata,'tmp1_maing.mat'];
 84 %%%%% MODEL INPUTS ( as uicontrols)
 85 wplotx0 = wplotx;wploty0 = wploty;
 86
 87
 88     COMPTA = 0 ;
 89
 90 if exist(NameWs) == 2
 91     try
 92     load(NameWs);
 93 catch
 94     COMPTA = 1 ;
 95 end
 96
 97 else
 98     COMPTA = 1 ;
 99 end
100
101
102 if  COMPTA == 1
103
104     % YOUNG's MODULUS
105     % ---------------
106     YOUNG_M = 20000 ;
107     % Poisson's coefficient
108     % ----------------------
109     POISSON = 0.3 ;
```

```
110     % Hardening/softening modulus
111     % --------------------------
112     HARDSOFT_MOD = -0.1 ;
113     % Yield stress
114     % ------------
115     YIELD_STRESS = 200 ;
116     % Problem type  TP = {'PLANE STRESS','PLANE STRAIN','3D'}
117     % ------------
118     ntype_c = 'PLANE STRAIN' ;
119
120     % Number of increments of each load state
121     % ---------------------------------------
122     istep1 = 5 ;
123     istep2 = 6 ;
124     istep3 = 5 ;
125     % Ratio compression strength / tension strength
126     % --------------------------------------------
127     n = 3 ;
128     % Model    PTC = {'SYMMETRIC','TRACTION','NON-SYMMETRIC'} ;
129     % -----------------------------------------------------
130     MDtype_c = 'SYMMETRIC' ;
131     try
132         save(NameWs) ;
133     catch
134         error('PATTERN PATH INCORRECT: Make sure that the currect directory↙
contains file main.m')
135     end
136     % SOFTENING/HARDENING TYPE
137     % ------------------------
138     HARDTYPE = 'LINEAR' ; %{LINEAR,EXPONENTIAL}
139     % VISCOUS/INVISCID
140     % ------------------------
141     VISCOUS = 'NO' ;
142     % Viscous coefficient ----
143     % ------------------------
144     eta = 0.3 ;
145     % TimeTotal (initial = 0) ----
146     % ------------------------
147     TimeTotal = 10 ;
148     % Integration coefficient ALPHA
149     % ------------------------
150     ALPHA_COEFF = 0.5 ;
151         % Limit for q
152     % ------------
153     q_inf = 2.5;
154 end
155     q_inf = 2.5;
156     A=0.5;
157
158
159 VARIABLES = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','ntype_c', ...
160     'nnls_s','istep1','istep2','istep3','n','MDtype_c','mstrain', ...
161↙
'mhist','shownumber','axiskind','axislim','ErasePrPlot','vpx','vpy','splitwind','pathd↙
ata', ...
```

```
'HARDTYPE','VISCOUS','eta','TimeTotal','ALPHA_COEFF','wplotx','wploty','q_inf','A'} ;

% **********
% UICONTROLS
% **********

clf;figure(1);clf;
hold on
grid on
xlabel('\sigma_1','FontSize',24,'FontWeight','bold');
ylabel('\sigma_2','FontSize',24,'FontWeight','bold');
set(gca,'FontSize',18)
%-----------------------------------------------
%  Edit boxes  (-->VARIABLES_LEG), at the LEFT
%-----------------------------------------------
VARIABLES_LEG = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','q_inf','A'};
VARIABLES_TEXT = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','q_inf','A'};
Inc_tv     = 0.002  ;
Inc_vt     = 0.002 ;
PositionT0 = [0.01 0.925 0.079 0.015 ] ;
PositionV0 = [0.01 0.9 0.079 0.023 ] ;
inclt = [ 0 PositionT0(4) + PositionV0(4) + Inc_tv + Inc_vt 0 0] ;
inclv = [0 PositionT0(4) + PositionV0(4) + Inc_vt + Inc_vt 0 0] ;
for ileg = 1:length(VARIABLES_LEG) ;
    var_i = VARIABLES_LEG{ileg} ;     text_i = VARIABLES_TEXT{ileg} ;
    if ~isempty(num2str(eval(var_i))); var_inum = num2str(eval(var_i));  else;
var_inum = eval(var_i) ;   end
    PositionT = PositionT0 -  (ileg-1)*inclt;
    STRE = ['htext = uicontrol
(''Style'',''text'',''Units'',''normalized'',''Position'',
PositionT'',''FontWeight'',''Bold'',''FontSize'',9,''string'',''',text_i,''');'] ;
    eval(STRE);
    PositionV = PositionV0 -  (ileg-1)*inclv;
    STRE =['hp',num2str(ileg),' = uicontrol(''Style'',''edit'',''String'',
var_inum,''Units'',''normalized'',''Position'',PositionV'',''FontSize'',9,''Tag'',''',
var_i,''', ''Callback'',NameFileExec);'];
    eval(STRE);
end

% %%%%%%%%%%%%%%%%%%%%%%
% 2) MDtype_c
% %%%%%%%%%%%%%%%%%%%%%%
[ok MDtype] = FndStrInCell(PTC,MDtype_c) ;
fff =uicontrol('Style', 'text', 'String', 'Damage model',...
    'Units','normalized','Position', [0.768 0.974 0.18 0.02],'FontSize',
10,'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String', PTC,...
    'Units','normalized','Position', [0.768 0.917 0.18 0.057],'Callback',
NameFileExec,...
    'Tag','MDtype','Value',MDtype);

% %%%%%%%%%%%%%%%%%%%%%%
% 3) PROBLEM TYPE --> TP
% %%%%%%%%%%%%%%%%%%%%%%
```

```
209 [ok ntype] = FndStrInCell(TP,ntype_c) ;
210 fff =uicontrol('Style', 'text', 'String', 'Problem type',...
211     'Units','normalized','Position', [0.568 0.974 0.18 0.02],'FontSize',↙
10,'FontWeight','Bold');
212 fff =uicontrol('Style', 'popupmenu', 'String', TP,...
213     'Units','normalized','Position', [0.568 0.917 0.18 0.057],'Callback',↙
NameFileExec,...
214     'Tag','ntype_c','Value',ntype);
215 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216 % 4) Ratio compression/traction strength
217 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
218 fff =uicontrol('Style', 'text', 'String', {'ONLY FOR ','NON-↙
SYMMETRIC','----------'},...
219     'Units','normalized','Position',[0.01 0.435 0.08 0.050],'FontSize',↙
9,'FontWeight','Bold');
220 fff =uicontrol('Style', 'text', 'String', 'ratio comp/trac',...
221     'Units','normalized','Position',[0.01 0.42 0.08↙
0.02],'FontWeight','Bold','FontSize',9);
222 fff =uicontrol('Style', 'edit', 'String', num2str(n),...
223     'Units','normalized','Position',[0.01 0.4 0.08 0.02],'FontSize',↙
9,'tag','n',...
224     'Callback',NameFileExec);
225 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
226 % 5) INCREMENTS FOR EACH LOAD STATE
227 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
228 fff =uicontrol('Style', 'text', 'String', {'INCREMENTS','----------'},...
229     'Units','normalized','Position',[0.913 0.893 0.08 0.030],'FontSize',↙
9,'FontWeight','Bold');
230 VARIABLES_LEG2 = {'istep1','istep2','istep3'};
231 Position = [0.913 0.893 0.08 0.025] ;
232 Inc      = [0 -0.030 0 0] ;
233
234 for ileg = 1:length(VARIABLES_LEG2)
235     var_i = VARIABLES_LEG2{ileg} ;
236     var_inum = num2str(eval(var_i));
237     Position = Position + Inc ;
238     STRE = ['htext = uicontrol↙
(''Style'',''text'',''Units'',''normalized'',''Position'',Position,''FontSize'',↙
9,''string'',''',var_i,''');'] ;
239     eval(STRE);
240     Position = Position + Inc ;
241     STRE =['hp',num2str(ileg),' = uicontrol(''Style'',''edit'',''String'',↙
var_inum,''Units'',''normalized'',''Position'',Position,''FontSize'',9,''Tag'',''',↙
var_i,''', ''Callback'',NameFileExecP);'];
242     eval(STRE);
243 end
244
245 % ------------------------------------------------
246 % Pushbottoms
247 % ------------------------------------------------
248 hpushsel1 = uicontrol('Style','pushbutton',...
249     'String',{'SELECT LOAD PATH'},'Units','normalized','Position',[0.006 0.119↙
0.08 0.0336],'Callback','select_path', ...
250     'tag','select_load');
251
```

```
252 hpushsel1 = uicontrol('Style','pushbutton',...
253     'String',{'COMPUTE'},'Units','normalized','Position',[0.006 0.065 0.08↙
0.044],'Callback','compute_load', ...
254     'tag','c');
255
256 hpushsel1 = uicontrol('Style','pushbutton',...
257     'String',{'REFRESH'},'Units','normalized','Position',[0.006 0.022 0.08↙
0.03687],'Callback','refresh_main', ...
258     'tag','c');
259
260 hpushsel1 = uicontrol('Style','pushbutton',...
261     'String',{'OPTIONS'},'Units','normalized','Position',[0.1 0.94 0.08↙
0.05],'Callback','showoptions', ...
262     'tag','showoptions2');
263
264
265 % %%%%%%%%%%%%%%%%%%%%%
266 % *) PLOT X
267 % %%%%%%%%%%%%%%%%%%%%%
268 [ok nvx] = FndStrInCell(wplotx,vpx) ;
269 fff =uicontrol('Style', 'text', 'String', 'VAR X',...
270     'Units','normalized','Position', [0.2 0.974 0.14 0.02],'FontSize',↙
10,'FontWeight','Bold');
271 fff =uicontrol('Style', 'popupmenu', 'String', wplotx,...
272     'Units','normalized','Position', [0.2 0.917 0.14↙
0.057],'Callback','plotcurves',...
273     'Tag','xplotc','Value',nvx);
274 % %%%%%%%%%%%%%%%%%%%%%
275 % *) PLOT Y
276 % %%%%%%%%%%%%%%%%%%%%%
277 [ok nvy] = FndStrInCell(wploty,vpy) ;
278 fff =uicontrol('Style', 'text', 'String', 'VAR Y',...
279     'Units','normalized','Position', [0.36 0.974 0.14 0.02],'FontSize',↙
10,'FontWeight','Bold');
280 fff =uicontrol('Style', 'popupmenu', 'String', wploty,...
281     'Units','normalized','Position', [0.36 0.917 0.14↙
0.057],'Callback','plotcurves',...
282     'Tag','yplotc','Value',nvy);
283
284 %%%% HARDTYPE
285 [ok ntype] = FndStrInCell(HARDLIST,HARDTYPE) ;
286 fff =uicontrol('Style', 'text', 'String', 'HARD/SOFT EV.',...
287     'Units','normalized','Position',[0.01 0.665 0.079 0.015] ,'FontSize',↙
9,'FontWeight','Bold');
288 fff =uicontrol('Style', 'popupmenu', 'String', HARDLIST,...
289     'Units','normalized','Position',[0.01 0.64 0.077 0.023] ,'Callback',↙
NameFileExec,...
290     'Tag','HARDTYPE_tag','Value',ntype,'FontSize',8);
291
292 %%%% VISCOUS
293 [ok ntype] = FndStrInCell({'YES','NO'},VISCOUS) ;
294 fff =uicontrol('Style', 'text', 'String', 'VISCOUS MODEL',...
295     'Units','normalized','Position',[0.01 0.615 0.079 0.015] ,'FontSize',↙
9,'FontWeight','Bold');
296 fff =uicontrol('Style', 'popupmenu', 'String',{'YES','NO'} ,...
```

```
297     'Units','normalized','Position',[0.01 0.59 0.077 0.023] ,'Callback',↙
NameFileExec,...
298     'Tag','VISCOUS_tag','Value',ntype,'FontSize',8);
299
300 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
301
302 %-----------------------------------------------
303 %  Edit boxes  OTHER PARAMETERS
304 %-----------------------------------------------
305 VARIABLES_LEG3 = {'eta','TimeTotal','ALPHA_COEFF'};
306 VARIABLES_TEXT = {'viscous coeff.','TIME INT.(s)','ALPHA coeff.'};
307 Inc_tv     = 0.004  ;
308 Inc_vt     = 0.004 ;
309 PositionT0 = [0.913 0.66 0.08 0.025  ] ; %0.01 0.61 0.079 0.015  0.913 0.893 0.08↙
0.025
310 PositionV0 = [0.913 0.645 0.08 0.025  ] ; %0.01 0.865 0.079 0.023
311 inclt = [ 0 PositionT0(4) + PositionV0(4) + Inc_tv + Inc_vt 0 0] ;
312 inclv = [0 PositionT0(4) + PositionV0(4) + Inc_vt + Inc_vt 0 0] ;
313 for ileg = 1:length(VARIABLES_LEG3) ;
314     var_i = VARIABLES_LEG3{ileg} ;     text_i = VARIABLES_TEXT{ileg} ;
315     if ~isempty(num2str(eval(var_i))); var_inum = num2str(eval(var_i));  else;↙
var_inum = eval(var_i) ;   end
316     PositionT = PositionT0 -  (ileg-1)*inclt;
317     STRE = ['htext = uicontrol↙
(''Style'',''text'',''Units'',''normalized'',''Position'',↙
PositionT'',''FontWeight'',''Bold'',''FontSize'',9,''string'',''',text_i,''');'] ;
318     eval(STRE);
319     PositionV = PositionV0 -  (ileg-1)*inclv;
320     STRE =['hp',num2str(ileg),' = uicontrol(''Style'',''edit'',''String'',↙
var_inum,''Units'',''normalized'',''Position'',PositionV'',''FontSize'',9,''Tag'',''',↙
var_i,''', ''Callback'',NameFileExec);'];
321     eval(STRE);
322 end
323
324
325
326 %**********************************
327 % Storing all variables in DATA
328 %**********************************
329 for ivar = 1:length(VARIABLES) ;
330     num_var = VARIABLES{ivar};
331     eval(['DATA.VAR.',num_var,' = ',num_var,';']);
332 end
333 DATA.VARIABLES_LEG = VARIABLES_LEG ;
334 DATA.VARIABLES_LEG2 = VARIABLES_LEG2 ;
335 DATA.VARIABLES_LEG3 = VARIABLES_LEG3 ;
336 DATA.NameWs        = NameWs        ;
337 DATA.wplotx0 = wplotx0 ;
338 DATA.wploty0 = wploty0 ;
339
340 %---------------------------------
341 % Attach DATA to the current figure
342 %---------------------------------
343 guidata(gcf,DATA);
344 CALLBACK_main    ;
```

```
345
346
347 %profile report
348
```

```
  1 function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,↙
strain,MDtype,n,TimeTotal)
  2 global hplotSURF
  3↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%%%%%%%%
  4 % CONTINUUM DAMAGE MODEL
  5 % ----------------------
  6 % Given the almansi strain evolution ("strain(totalstep,mstrain)") and a set of
  7 % parameters and properties, it returns the evolution of the cauchy stress and↙
other  variables
  8 % that are listed below.
  9 %
 10 % INPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 11 % ------------------------------------------------------------------
 12 % Eprop(1) = Young's modulus  (E)
 13 % Eprop(2) = Poisson's coefficient (nu)
 14 % Eprop(3) = Hardening(+)/Softening(-) modulus (H)
 15 % Eprop(4) = Yield stress (sigma_y)
 16 % Eprop(5) = Type of Hardening/Softening law  (hard_type)
 17 %            0 --> LINEAR
 18 %            1 --> Exponential
 19 % Eprop(6) = Rate behavior (viscpr)
 20 %            0 --> Rate-independent (inviscid)
 21 %            1 --> Rate-dependent   (viscous)
 22 %
 23 % Eprop(7) = Viscosity coefficient (eta)  (dummy if inviscid)
 24 % Eprop(8) = ALPHA coefficient (for time integration), (ALPHA)
 25 %             0<=ALPHA<=1 , ALPHA = 1.0 --> Implicit
 26 %                           ALPHA = 0.0 --> Explicit
 27 %            (dummy if inviscid)
 28 %
 29 % ntype    = PROBLEM TYPE
 30 %            1 : plane stress
 31 %            2 : plane strain
 32 %            3 : 3D
 33 %
 34 % istep = steps for each load state (istep1,istep2,istep3)
 35 %
 36 % strain(i,j) = j-th component of the linearized strain vector at the i-th
 37 %               step, i = 1:totalstep+1
 38 %
 39 % MDtype      = Damage surface criterion %
 40 %            1 : SYMMETRIC
 41 %            2 : ONLY-TENSION
 42 %            3 : NON-SYMMETRIC
 43 %
 44 %
 45 % n          = Ratio compression/tension strength (dummy if MDtype is different↙
from 3)
 46 %
 47 % TimeTotal  = Interval length
 48 %
 49 %  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 50 %  ------------------------------------------------------------------
```

```
 51 %  1) sigma_v{itime}(icomp,jcomp)  --> Component (icomp,jcomp) of the cauchy
 52 %                                    stress tensor at step "itime"
 53 %                                    REMARK: sigma_v is a type of
 54 %                                    variable called "cell array".
 55 %
 56 %
 57 %  2) vartoplot{itime}              --> Cell array containing variables one wishes↙
to plot
 58 %                                    ---------------------------------------
 59 %   vartoplot{itime}(1) =   Hardening variable (q)
 60 %   vartoplot{itime}(2) =   Internal variable (r)%
 61
 62 %
 63 %  3) LABELPLOT{ivar}              --> Cell array with the label string for
 64 %                                     variables of "varplot"
 65 %
 66 %          LABELPLOT{1} => 'hardening variable (q)'
 67 %          LABELPLOT{2} => 'internal variable'
 68 %
 69 %
 70 %  4) TIME VECTOR  - >
 71↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%%%%%%%%
 72
 73 % SET LABEL OF "vartoplot" variables  (it may be defined also outside this↙
function)
 74 % ----------------------------------
 75  LABELPLOT = {'hardening variable (q)','internal variable'};
 76  LABELPLOT{3} = 'damage variable (d)';
 77  LABELPLOT{4} = 'C11';
 78
 79 E      = Eprop(1) ; nu = Eprop(2) ;
 80 viscpr = Eprop(6) ;
 81 sigma_u = Eprop(4);
 82
 83
 84
 85 if ntype == 1
 86     menu('PLANE STRESS has not been implemented yet','STOP');
 87     error('OPTION NOT AVAILABLE')
 88     mstrain = 4    ;
 89     mhist   = 6    ;
 90 elseif ntype == 3
 91     menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
 92     error('OPTION NOT AVAILABLE')
 93 else
 94     mstrain = 4    ;
 95     mhist   = 6    ;
 96 end
 97
 98 totalstep = sum(istep) ;
 99
100
101 % 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)
vartoplot{i}(4) = ce(1,1)  ; %  C11 (d)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %*********************************************************************
        % INVISCID
        % *********************************************************************

if viscpr == 0
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)
            vartoplot{i}(4) = aux_var(4)  ; %  C11 (d)

        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %**********************************************************************
            % VISCOUS
            % ***************

    else
        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"
            % ------------------------
            timestep=TIMEVECTOR(i)-TIMEVECTOR(i-1);
            eps_n = strain(i-1,:) ;
            eps_n1 = strain(i,:) ;
            %***********************************************************************************
            %*      DAMAGE MODEL
            % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            [sigma_n1,hvar_n,aux_var] = rmap_dano2(eps_n,eps_n1,hvar_n,Eprop,ce, MDtype,n,timestep);
            % 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(4)  ; %  C11 (d)

        end
    end
end

```

```
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);
viscpr = Eprop(6);
eta = Eprop(7);
alpha = Eprop(8);
q_inf = Eprop(9);
A = Eprop(10);

%↙
```

```
40
41
42 %↙
43 %*       initializing                                                %*
44  r0 = sigma_u/sqrt(E);
45  zero_q=1.d-6*r0;
46 % if(r_n<=0.d0)
47 %     r_n=r0;
48 %     q_n=r0;
49 % end
50 %↙
51
52
53 %↙
54 %*       Damage surface↙
%*
55 [rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
56 %↙
57
58
59 %↙
60 %*   Ver el Estado de Carga↙
%*
61 %*   --------->    fload=0 : elastic unload↙
%*
62 %*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)↙
%*
63 fload=0;
64
65 if(rtrial > r_n)
66     %*   Loading
67
68     fload=1;
69     delta_r=rtrial-r_n;
70     r_n1= rtrial  ;
71     if hard_type == 0
72         %  Linear
73         q_n1= min(q_n+ H*delta_r , q_inf);
74     else
75         %  Exponential
76         q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
77         H=A*(q_inf-r0)*exp(A*(1-r_n1/r0))/r0;
78     end
79
80     if(q_n1<zero_q)
81         q_n1=zero_q;
82     end
83     s_n1  =ce*eps_n1';
84     ce_n1=s_n1*s_n1';
```

```
    c11_n1=ce(1,1)*q_n1/r_n1-(q_n1-H*r_n1)/r_n1^3*ce_n1(1,1);

else

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


end
% Damage variable
% ---------------
dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
%  ****************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
%hold on
%plot(sigma_n1(1),sigma_n1(2),'bx')

%↙
*********************************************************************************
*********************************************************************************
%↙
*********************************************************************************
%* Updating historic variables                                              %*
%  hvar_n1(1:4)  = eps_n1p;
hvar_n1(5)= r_n1 ;
hvar_n1(6)= q_n1 ;
%↙
*********************************************************************************
*********************************************************************************
%↙
*********************************************************************************
%* Auxiliar variables↙
%*
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
aux_var(4) = c11_n1;
%↙
*********************************************************************************
```

```
134
135
```

```
function [sigma_n1,hvar_n1,aux_var] = rmap_dano2 (eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n,timestep)

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


hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
hard_type = Eprop(5);
viscpr = Eprop(6);
eta = Eprop(7);
alpha = Eprop(8);
q_inf = Eprop(9);
A = Eprop(10);
```

```
 38
 39 %↙
*********************************************************************************
 40
 41
 42 %↙
*********************************************************************************
 43 %*       initializing                                                    %*
 44  r0 = sigma_u/sqrt(E);
 45  zero_q=1.d-6*r0;
 46 % if(r_n<=0.d0)
 47 %     r_n=r0;
 48 %     q_n=r0;
 49 % end
 50 %↙
*********************************************************************************
 51
 52
 53 %↙
*********************************************************************************
 54 %*       Damage surface↙
%*
 55 [rtrial] = (1-alpha)*Modelos_de_dano1 (MDtype,ce,eps_n,n)+alpha*Modelos_de_dano1↙
(MDtype,ce,eps_n1,n);
 56 [rtrialexp] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
 57 [rtrialimp] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
 58 %↙
*********************************************************************************
 59
 60
 61 %↙
*********************************************************************************
 62 %*   Ver el Estado de Carga↙
%*
 63 %*   --------->    fload=0 : elastic unload↙
%*
 64 %*   --------->    fload=1 : damage (compute algorithmic constitutive tensor)↙
%*
 65 fload=0;
 66
 67 if(rtrial > r_n)
 68     %*   Loading
 69
 70     fload=1;
 71     r_n1= (eta-timestep*(1-alpha))*r_n/(eta+alpha*timestep)+timestep*rtrial/↙
(eta+alpha*timestep)  ;
 72     delta_r=r_n1-r_n;
 73     if hard_type == 0
 74         %  Linear
 75         q_n1= min(q_n+ H*delta_r , q_inf);
 76     else
 77         %  Exponential
 78         q_n1= q_inf-(q_inf-r0)*exp(A*(1-r_n1/r0));
 79         H=A*(q_inf-r0)*exp(A*(1-r_n1/r0))/r0;
 80
```

```
 81     end
 82
 83     if(q_n1<zero_q)
 84         q_n1=zero_q;
 85     end
 86     s_n1  =ce*eps_n1';
 87     ce_n1=s_n1*s_n1';
 88     c11_n1=ce(1,1)*q_n1/r_n1-alpha*timestep*(H*r_n1-q_n1)/((eta+alpha*timestep)↙
*rtrial*r_n1^2)*ce_n1(1,1);
 89
 90 else
 91
 92     %*     Elastic load/unload
 93     fload=0;
 94     r_n1= r_n  ;
 95     q_n1= q_n  ;
 96     c11_n1=ce(1,1)*q_n1/r_n1;
 97
 98 end
 99 % Damage variable
100 % ---------------
101 dano_n1   = 1.d0-(q_n1/r_n1);
102 %  Computing stress
103 %  ***************
104 sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
105 %hold on
106 %plot(sigma_n1(1),sigma_n1(2),'bx')
107
108 %↙
************************************************************************************
109
110
111 %↙
************************************************************************************
112 %* Updating historic variables                                          %*
113 %  hvar_n1(1:4)  = eps_n1p;
114 hvar_n1(5)= r_n1 ;
115 hvar_n1(6)= q_n1 ;
116 %↙
************************************************************************************
117
118
119
120
121 %↙
************************************************************************************
122 %* Auxiliar variables↙
%*
123 aux_var(1) = fload;
124 aux_var(2) = q_n1/r_n1;
125 aux_var(3) = (q_n1-H*r_n1)/r_n1^3;
126 aux_var(4) = c11_n1;
127 %↙
************************************************************************************
128
```

```
129
130
131
132
133
134
135
136
137
138
139
```

```
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);
%↙
**************************************************************************


```

```
 37
 38
 39
 40 %↙
*********************************************************************************************
 41 % POLAR COORDINATES
 42 if MDtype==1 %Symmetric
 43     tetha=[0:0.01:2*pi];
 44     %↙
*********************************************************************************************
 45     %* RADIUS
 46     D=size(tetha);                          %*  Range
 47     m1=cos(tetha);                          %*
 48     m2=sin(tetha);                          %*
 49     Contador=D(1,2);                        %*
 50     radio = zeros(1,Contador) ;
 51     s1    = zeros(1,Contador) ;
 52     s2    = zeros(1,Contador) ;
 53
 54     for i=1:Contador
 55         radio(i)= q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i) 0↙
...
 56             nu*(m1(i)+m2(i))]');
 57
 58         s1(i)=radio(i)*m1(i);
 59         s2(i)=radio(i)*m2(i);
 60
 61     end
 62     hplot =plot(s1,s2,tipo_linea);
 63
 64
 65 elseif MDtype==2 %Tensile damage
 66     tetha=[0,0:0.005:2*pi];
 67     %↙
*********************************************************************************************
 68     %* RADIUS
 69     D=size(tetha);                          %*  Range
 70     m1=cos(tetha);                          %*
 71     m2=sin(tetha);                          %*
 72     Contador=D(1,2);                        %*
 73     radio = zeros(1,Contador) ;
 74     s1    = zeros(1,Contador) ;
 75     s2    = zeros(1,Contador) ;
 76     for i=1:Contador
 77         radio(i)= q/sqrt([max(m1(i), 0) max(m2(i),0) 0 max(nu*(m1(i)+m2(i)),0)]↙
*ce_inv*[m1(i) m2(i) 0 ...
 78             nu*(m1(i)+m2(i))]');
 79
 80         s1(i)=radio(i)*m1(i);
 81         s2(i)=radio(i)*m2(i);
 82     end
 83     hplot =plot(s1,s2,tipo_linea);
 84
 85
 86 elseif MDtype==3 %Non-symmetric
```

```
 87     tetha=[0:0.005:2*pi];
 88     %↙
*********************************************************************************
 89     %* RADIUS
 90     D=size(tetha);                          %*  Range
 91     m1=cos(tetha);                          %*
 92     m2=sin(tetha);                          %*
 93     Contador=D(1,2);                        %*
 94     radio = zeros(1,Contador) ;
 95     s1    = zeros(1,Contador) ;
 96     s2    = zeros(1,Contador) ;
 97     for i=1:Contador
 98         t=(max(m1(i),0)+max(m2(i),0)+max(nu*(m1(i)+m2(i)),0))/(abs(m1(i))+abs(m2↙
(i))+abs(nu*(m1(i)+m2(i))));
 99
100         radio(i)= q/(sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) m2(i) 0↙
...
101             nu*(m1(i)+m2(i))]')*(t+(1-t)/n));
102
103         s1(i)=radio(i)*m1(i);
104         s2(i)=radio(i)*m2(i);
105     end
106     hplot =plot(s1,s2,tipo_linea);
107
108
109
110 end
111 %↙
*********************************************************************************
112
113
114
115 %↙
*********************************************************************************
116 return
117
118
119
120
```

```
  1 function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,↙
strain,MDtype,n,TimeTotal)
  2 global hplotSURF
  3↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%%%%%%%%
  4 % CONTINUUM DAMAGE MODEL
  5 % ----------------------
  6 % Given the almansi strain evolution ("strain(totalstep,mstrain)") and a set of
  7 % parameters and properties, it returns the evolution of the cauchy stress and↙
other  variables
  8 % that are listed below.
  9 %
 10 % INPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 11 % ------------------------------------------------------------------
 12 % Eprop(1) = Young's modulus  (E)
 13 % Eprop(2) = Poisson's coefficient (nu)
 14 % Eprop(3) = Hardening(+)/Softening(-) modulus (H)
 15 % Eprop(4) = Yield stress (sigma_y)
 16 % Eprop(5) = Type of Hardening/Softening law  (hard_type)
 17 %            0 --> LINEAR
 18 %            1 --> Exponential
 19 % Eprop(6) = Rate behavior (viscpr)
 20 %            0 --> Rate-independent (inviscid)
 21 %            1 --> Rate-dependent   (viscous)
 22 %
 23 % Eprop(7) = Viscosity coefficient (eta)  (dummy if inviscid)
 24 % Eprop(8) = ALPHA coefficient (for time integration), (ALPHA)
 25 %             0<=ALPHA<=1 , ALPHA = 1.0 --> Implicit
 26 %                           ALPHA = 0.0 --> Explicit
 27 %            (dummy if inviscid)
 28 %
 29 % ntype    = PROBLEM TYPE
 30 %            1 : plane stress
 31 %            2 : plane strain
 32 %            3 : 3D
 33 %
 34 % istep = steps for each load state (istep1,istep2,istep3)
 35 %
 36 % strain(i,j) = j-th component of the linearized strain vector at the i-th
 37 %               step, i = 1:totalstep+1
 38 %
 39 % MDtype      = Damage surface criterion %
 40 %            1 : SYMMETRIC
 41 %            2 : ONLY-TENSION
 42 %            3 : NON-SYMMETRIC
 43 %
 44 %
 45 % n          = Ratio compression/tension strength (dummy if MDtype is different↙
from 3)
 46 %
 47 % TimeTotal  = Interval length
 48 %
 49 %  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 50 %  ------------------------------------------------------------------
```

```
 51 %  1) sigma_v{itime}(icomp,jcomp)  --> Component (icomp,jcomp) of the cauchy
 52 %                                    stress tensor at step "itime"
 53 %                                    REMARK: sigma_v is a type of
 54 %                                    variable called "cell array".
 55 %
 56 %
 57 %  2) vartoplot{itime}              --> Cell array containing variables one wishes↙
to plot
 58 %                                    ---------------------------------------
 59 %   vartoplot{itime}(1) =   Hardening variable (q)
 60 %   vartoplot{itime}(2) =   Internal variable (r)%
 61
 62 %
 63 %  3) LABELPLOT{ivar}              --> Cell array with the label string for
 64 %                                     variables of "varplot"
 65 %
 66 %          LABELPLOT{1} => 'hardening variable (q)'
 67 %          LABELPLOT{2} => 'internal variable'
 68 %
 69 %
 70 %  4) TIME VECTOR  - >
 71↙
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%↙
%%%%%%%%%%%%
 72
 73 % SET LABEL OF "vartoplot" variables  (it may be defined also outside this↙
function)
 74 % ----------------------------------
 75  LABELPLOT = {'hardening variable (q)','internal variable'};
 76  LABELPLOT{3} = 'damage variable (d)';
 77  LABELPLOT{4} = 'C11';
 78
 79 E      = Eprop(1) ; nu = Eprop(2) ;
 80 viscpr = Eprop(6) ;
 81 sigma_u = Eprop(4);
 82
 83
 84
 85 if ntype == 1
 86     menu('PLANE STRESS has not been implemented yet','STOP');
 87     error('OPTION NOT AVAILABLE')
 88     mstrain = 4    ;
 89     mhist   = 6    ;
 90 elseif ntype == 3
 91     menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
 92     error('OPTION NOT AVAILABLE')
 93 else
 94     mstrain = 4    ;
 95     mhist   = 6    ;
 96 end
 97
 98 totalstep = sum(istep) ;
 99
100
101 % 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)
vartoplot{i}(4) = ce(1,1)  ; %  C11 (d)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %*********************************************************************
        % INVISCID
        % *********************************************************************

if viscpr == 0
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)
        vartoplot{i}(4) = aux_var(4)  ; %  C11 (d)

    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %**********************************************************************
        % VISCOUS
        % ***************

else
    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"
        % ------------------------
        timestep=TIMEVECTOR(i)-TIMEVECTOR(i-1);
        eps_n = strain(i-1,:) ;
        eps_n1 = strain(i,:) ;
        %*************************************************************************************
        %*      DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] = rmap_dano2(eps_n,eps_n1,hvar_n,Eprop,ce,MDtype,n,timestep);
        % 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(4)  ; %  C11 (d)

    end
    end
end
```