Technical University of Catalonia

NUMERICAL INTEGRATION OF CONSTITUTIVE DAMAGE MODELS USING MATLAB

Joaquín A. Hernández Ortega

Barcelona, March 2011

Table of Contents

Nomenclature

$\boldsymbol{c_e}$	Elasticity tensor
E^e	Young's modulus
H	Hardening/softening modulus.
r_n	Internal variable at time t_n.
ν	Poisson's coefficient
q_∞	Lower limit for q
q_n	Hardening/softening variable at time t_n.
$\tilde{\boldsymbol{\sigma}}$	Cauchy Effective stress tensor
σ_u	Ultimate strength.
η	Viscosity coefficient.

Chapter 1

Using the program

1.1 Description

The primary objective of this *Matlab*'s program is to aid the student in understanding the algorithmic structure underlying the numerical integration of continuum damage constitutive models. We concentrate exclusively on the *local* constitutive response - as distinct from the overall structural response, which would be the focus of a standard finite element program - and, thus, the *local* strain path (at a point) is prescribed by the user by means of a user-friendly graphical interface, which has been programmed using Matlab's *uicontrol* objects. This graphical interface, in turn, allows users to modify **input data** and visualize **output information** in an easy and intuitive manner, so that concepts such as damage surface, elastic domain, Kuhn-Tucker loading/unloading conditions, etc. can be readily grasped and assimilated. The program is invoked from the command line by simply typing *main* - make sure first that the current directory is set to the folder that contains the source files. A Figure window containing the abovementioned graphical user interface will pop up (see Fig.(1.1)).

1.2 Input data

1.2.1 Material parameters

- **Young's modulus** (E^e).
- **Poisson's ratio** (ν).
- **Hardening/softening modulus** (H).

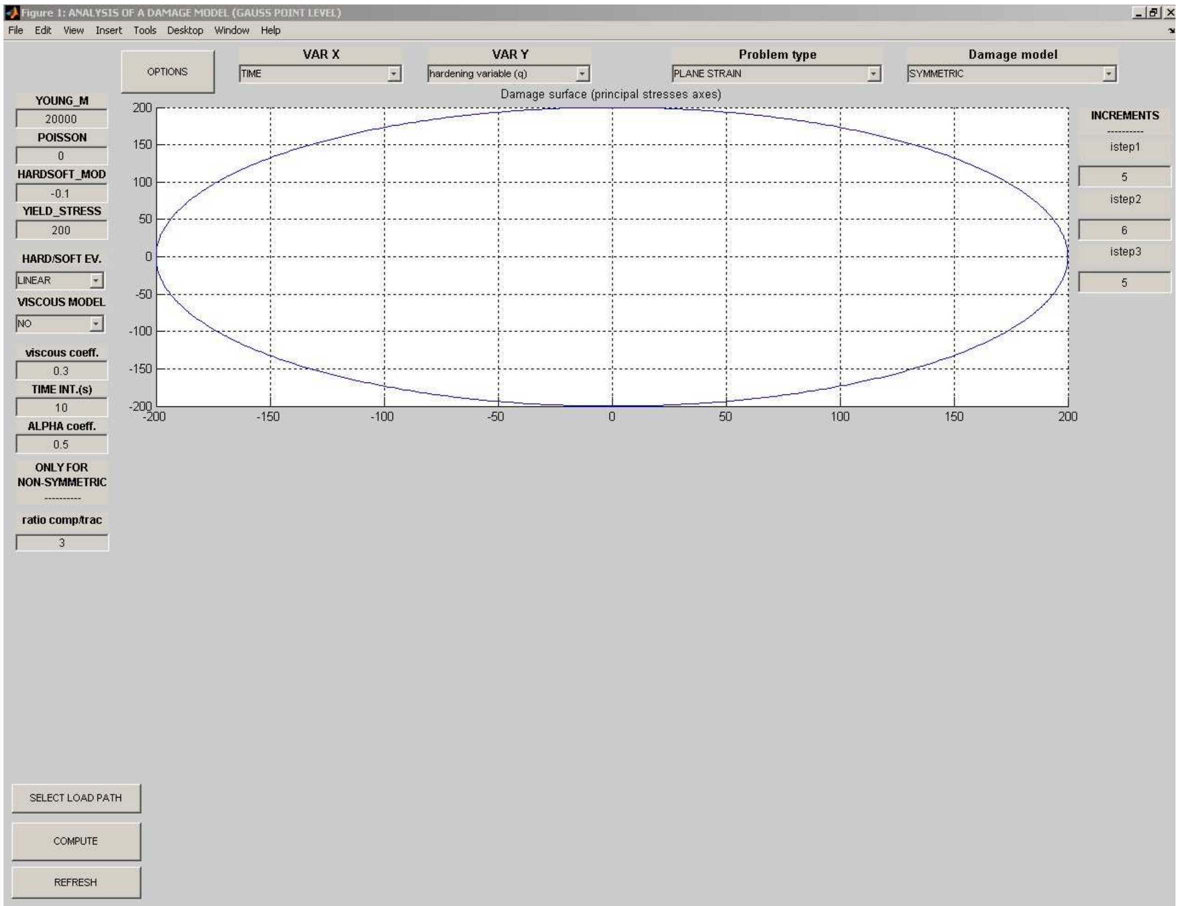

Figure 1.1 *Graphical interface.*

- Type of hardening/softening law:

  Linear law. This option is already implemented. The programmed expression for the update the hardening/softening variable reads:

$$q_{n+1} = max\left(q_n + H(r_{n+1} - r_n), q_\infty\right). \tag{1.2.1}$$

  where q stands for the hardening/softening variable, and r denotes the internal variable. The initial value for r is set to $\sigma_u/\sqrt{E^e}$, and $q_0 = r_0$. Note that variable q is enforced to be greater than a certain value $q_\infty = 10^{-6} q_0$ (named *q_zero* in the code).

  Exponential law. This option is not implemented by default, so if you attempt to launch an analysis with this option, a window will pop up warning you of this fact and indicating the file (*rmap_dano1*) that has to be modified to incorporate this type of constitutive law.

- **Viscous/inviscid case.** Only the inviscid case is available. Thus, similarly to the situation described above, if you try to run the program after selecting this option, a warning message will inform you of the impossibility of proceeding with the analysis.
- **Viscosity coefficient** (η). It only comes into play when the option *VISCOUS MODEL=YES* is selected.
- **Compression/traction ratio**. It is used only when the chosen damage criterion is the non-symmetric one.

1.2.2 Damage criteria

Only the so-called **"symmetric" damage criterion** is implemented, although allowance (see function *Modelos_de_dano1*) has been made for two other criteria: the only tension criterion and then non-symmetric damage model. The program furnishes an output facility that enables the user to visualize both the path traced by the stress state, and the evolution of the elastic domain determined by the corresponding damage criterion. Thus, modifications of the algorithmic part of the code (function *Modelos_de_dano1*) defining a new damage criterion should be accompanied by consistent changes in the function that plots the damage surface in the space of principal stresses for each value of the state function q (*dibujar_criterio_dano1*) In this respect, we recommend to use polar coordinates, as done in the symmetric model case (see listing below) in coding the plotting of the corresponding elliptical surface.

```
    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);
```

Listing 1.1 *Portion of code that plots the symmetric elliptical-shaped damage surface, using polar coordinates. Note that stress tensors are manipulated in vector (Voigt) notation. Besides, due to the plane strain constraint, the fourth component can be expressed simply in terms of the other two components.*

The damage surface plot is automatically redrawn each time the input parameters involved in its definition -in this case only σ_u and ν- are modified.

1.2.2.1 Explicit expression for the symmetric model

The damage surface corresponding to the "symmetric damage model" is defined as

$$\boldsymbol{\sigma} : \boldsymbol{C}^{-1} : \boldsymbol{\sigma} = q^2 \tag{1.2.2}$$

For plotting this damage surface on the plane of principal stresses, it is convenient to first obtain an explicit expression for the inverse of the elasticity tensor:

$$\boldsymbol{C}^{-1} = \frac{1}{9\kappa}\boldsymbol{I}_{\mathrm{vol}} + \frac{1}{2\mu}\boldsymbol{I}_{\mathrm{dev}} \tag{1.2.3}$$

Substitution of the above expression into Eq.(1.2.2) yields

$$\frac{1}{9\kappa}\mathrm{tr}\,\boldsymbol{\sigma}^2 + \frac{1}{2\mu}\|dev\,\boldsymbol{\sigma}\|^2 = q^2. \tag{1.2.4}$$

Since the stress tensor is given in terms of principal stresses, we can write

$$\boldsymbol{\sigma} = \mathrm{diag}\,(\sigma_1, \sigma_2, \nu(\sigma_1 + \sigma_2)) \tag{1.2.5}$$

Inserting Eq.(1.2.5) into Eq.(1.2.4), and after some algebra, we arrive at the following quadratic function of σ_1 and σ_2:

$$\begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix}\begin{bmatrix}1-\nu & -\nu \\ -\nu & 1-\nu\end{bmatrix}\begin{bmatrix}\sigma_1 \\ \sigma_2\end{bmatrix} = \frac{E}{1+\nu}q^2 \tag{1.2.6}$$

The above expression represents the equation of an ellipse in the σ_1-σ_2 plane. The lengths of the major and minor axes of this ellipse can be determined by studying the eigenvalues and eigenvectors of the matrix appearing in the left-hand side of Eq.(1.2.6):

$$\begin{bmatrix}\sigma_1 & \sigma_2\end{bmatrix} \mathbf{T}^{\mathbf{T}} \begin{bmatrix}1 & 0 \\ 0 & 1-2\nu\end{bmatrix} \mathbf{T} \begin{bmatrix}\sigma_1 \\ \sigma_2\end{bmatrix} = \frac{\mathbf{E}}{\mathbf{1}+\nu}\mathbf{q^2} \tag{1.2.7}$$

where

$$\mathbf{T} = \frac{\sqrt{\mathbf{2}}}{\mathbf{2}} \begin{bmatrix}-1 & 1 \\ -1 & -1\end{bmatrix} \tag{1.2.8}$$

is a matrix representing a rotation of 45 degrees. Consequently, we can legitimately write

$$\begin{bmatrix}\sigma_1' & \sigma_2'\end{bmatrix} \begin{bmatrix}\frac{(1+\nu)}{q^2E} & 0 \\ 0 & \frac{(1+\nu)}{q^2E}(1-2\nu)\end{bmatrix}' \begin{bmatrix}\sigma_1' \\ \sigma_2'\end{bmatrix} = 1 \tag{1.2.9}$$

It follows, thus, that the representation of the symmetric tension-compression damage surface in the place of principal stresses is an ellipse rotated 45 degrees and with mayor axes of lengths

$$a = \frac{q\sqrt{E}}{\sqrt{1+\nu}}\frac{1}{\sqrt{1-2\nu}} \tag{1.2.10}$$

and

$$b = \frac{q\sqrt{E}}{\sqrt{1+\nu}}, \tag{1.2.11}$$

respectively. At the first time step, $q_0 = \frac{\sigma_u}{\sqrt{E}}$. Thus:

$$a = \frac{\sigma_u}{\sqrt{1+\nu}}\frac{1}{\sqrt{1-2\nu}} \tag{1.2.12}$$

and

$$b = \frac{\sigma_u}{\sqrt{1+\nu}}, \tag{1.2.13}$$

Observations

Polar coordinates representation. Starting point:

$$\boldsymbol{\sigma} : \boldsymbol{C}^{-1} : \boldsymbol{\sigma} = q^2 \tag{1.2.14}$$

Now we want it in polar coordinates:

$$\sigma_1 = r(\theta)cos(\theta) \tag{1.2.15}$$

$$\sigma_2 = r(\theta)sin(\theta) \tag{1.2.16}$$

Therefore

$$r^2 \overbrace{\begin{bmatrix} cos(\theta) \\ sin(\theta) \\ 0 \\ \nu(sin(\theta) + cos(\theta)) \end{bmatrix}^T \boldsymbol{C}^{-1} \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ 0 \\ \nu(sin(\theta) + cos(\theta)) \end{bmatrix}}^{\tau^2(\theta)} = q^2 \tag{1.2.17}$$

Therefore:

$$r = \frac{q}{\tau(\theta)} \tag{1.2.18}$$

1.2.3 Selection of the strain path

The strain history is determined indirectly by selecting three points - $\tilde{\boldsymbol{\sigma}}_1$, $\tilde{\boldsymbol{\sigma}}_2$ and $\tilde{\boldsymbol{\sigma}}_3$ - in the space of principal stresses (see Fig.(1.1)). To select these points, just press the button located in the lower left-hand part of the figure and identified by the label "SELECT LOAD PATH". This "push button" is an *uicontrol object*, in Matlab's terminology, that, in being pressed, invokes a function named *select_path* . This function displayed a menu window (Fig.(1.2)) that prompts the user to choose between graphic selection (i.e., by simply clicking on the graph containing the plot of the damage surface, on the top half of the figure window) or selection by relative coordinates. By connecting the introduced points, we get the "effective" stress path, i.e., the path that would have traced the stress state had the material been purely elastic. Accordingly, to obtain the corresponding strain quantities, we simply multiply the stress points $\tilde{\boldsymbol{\sigma}}_1$, $\tilde{\boldsymbol{\sigma}}_2$ and $\tilde{\boldsymbol{\sigma}}_3$ by the inverse of the elasticity tensor $\boldsymbol{c_e}$.

1.2.4 Time integration parameters

- **Final time** (T).
- α **constant in generalized midpoint rule** (not used in the inviscid case).

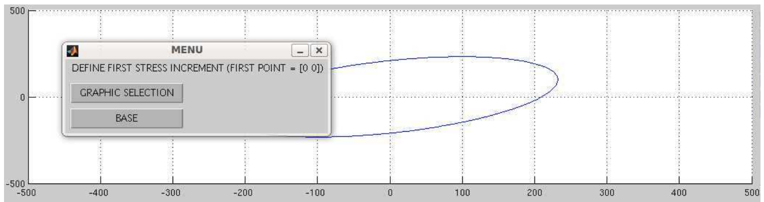

Figure 1.2 *Selection of the points defining the stress path can be done either graphically or by introducing the* relative *coordinates (with respect the previously introduced point) of each stress point.*

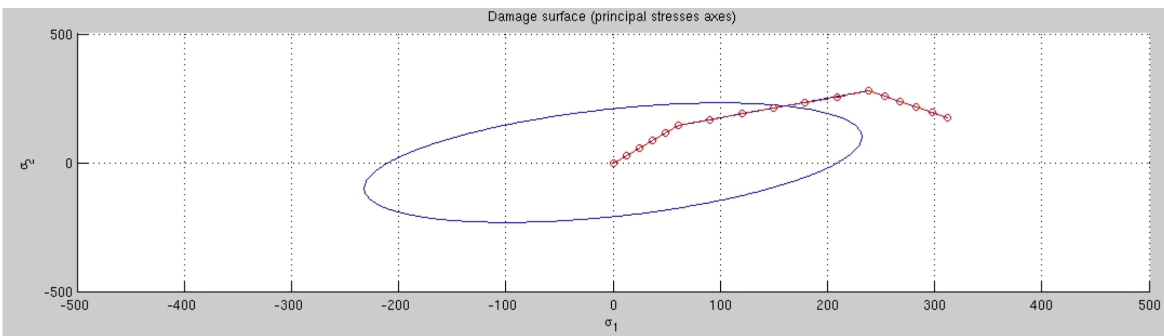

Figure 1.3 *Prescribed "effective" stress trajectory and initial damage surface*

.

- **Number of time steps** (right-hand side on the Figure window). Each of the three straight portions comprising the prescribed effective stress path is divided into a number of steps determined by these integer values (see function *calstrain*).

1.2.5 How to add new input parameters

To define new input parameters, not contemplated in the default graphic environment, function *main* and other subsidiary functions should be conveniently modified to include new *uicontrol* objects, attach the new data to the active figure (using the intrinsic *Matlab's* function *guidata*) and pass the information to function *damage_main* , which is the one that contains the integration algorithm. For users not sufficiently steeped in the intricacies of Matlab's graphic interface design[1], this task may result,

[1]Users unfamiliar with this powerful Matlab's tool are advised to consult Marchand and Holland (2003).

understandably, unduly complicated. An alternative and less sophisticated route to introduce new input parameters is to simply define the new data in the preamble of file *damage_main* , by expanding the vector of material properties **Eprop** (see listing below).

```
%%%% NEW INPUT PARAMETERS (to avoid the nuisance of modifying graphic
    user interface)
%Eprop{end+1} = NEW_DATA_1
%Eprop{end+1} = NEW_DATA_2
%Eprop{end+1} = NEW_DATA_3
%.
%.
%.
%%%%
```

Listing 1.2 *Preamble of file damage_main. New input parameters can be included by simply expanding vector Eprop.*

1.3 Output facilities

To execute the function that carries out the integration of the damage constitutive equations (*damage_main*), just press the "COMPUTE" button, located closed to the lower left-hand corner of the figure windows. Output results can be visualized in two different graphs, which can be either displayed in the same figure window, or in separate windows (see section 1.3.3).

1.3.1 Computed stress path (in principal stresses plane)

In a first graph, see top half of figure 1.4, the stresses computed at each time step are represented in the plane of maximum principal stresses σ_1- σ_2 ($\sigma_1 > \sigma_2$), together with the successive damage surfaces. This plot provides a great deal of insight into the phenomenological behavior of the material since, depending on the location of the stress state in relation with the prevailing damage surfaces, one can ascertain at each step whether the material deforms elastically (if the computed stress path lies entirely within the elastic domain) or in an inelastic manner. Material hardening (or softening) will be reflected in an expansion (or contraction) of the elastic domain in stress space.

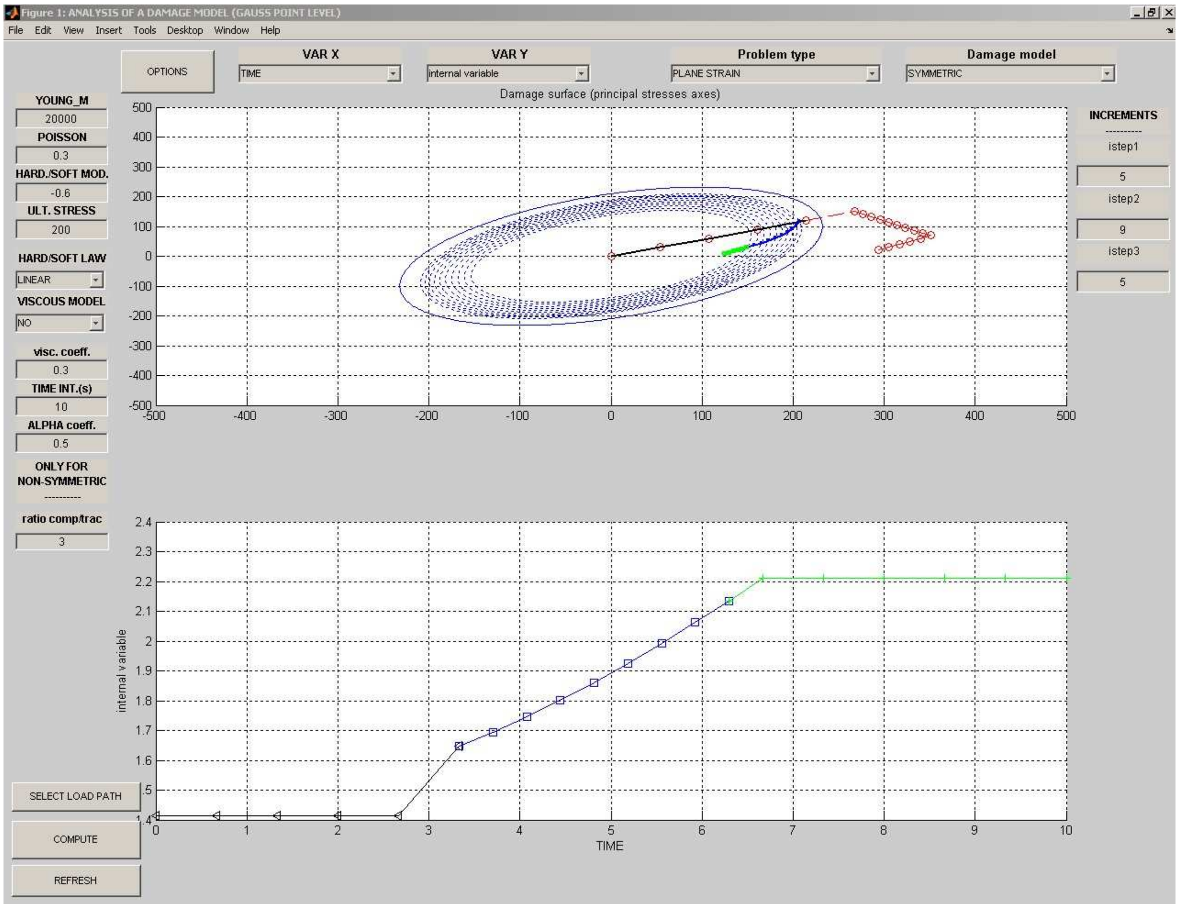

Figure 1.4 *Output results.*

1.3.2 $X - Y$ graph

In conjunction with the stress space representation mentioned above, the program permits to represent standard 2-D plots (bottom half of figure 1.4). Quantities to be plotted can be selected in the pop-up menus situated on the upper part of the figure. Available options are, for the x-axis, principal strains (e_1, e_2, $|e_1|$, $|e_2|$, $\sqrt{e_1^e + e_2^2}$), time vector, hardening/softening variable q and internal variable r; for the y-axis: principal stresses (σ_1, σ_2, $|\sigma_1|$, $|\sigma_2|$, $\sqrt{\sigma_1^2 + \sigma_2^2}$), time, hardening/softening variable q and internal variable r.

1.3.2.1 How to post-process other variables

This list of scalar variables that can be post-processed can be easily extended by the user. To illustrate how to do this, let us suppose that we want to include the damage variable, defined as $d = 1 - q/r$. First, we have to define the label that will identify the variable in the pop-up menu. The chosen label is stored in cell array **LABELPLOT**, at the beginning of file *damage_main* :

```
% SET LABEL OF "vartoplot" variables
% ----------------------------------------
LABELPLOT = {'hardening variable (q)','internal variable'};
LABELPLOT{3} = 'damage variable (d)'  % <<<<< -- NEW LINE
%LABELPLOT{4} = Define new label
% .
% .
```

Listing 1.3 *How to post-process additional variables. Definition of the label associated with damage variable (function damage_main).*

Then, the variable itself is computed and stored in the $i-th$ component (associated to the $i-th$ time step) of cell array **vartoplot** :

```
  % 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)
  % NEW ONE
        vartoplot{i}(3) = 1-hvar_n(6)/hvar_n(5) ; % Damage variable
  %vartoplot{i}(4) = DefineNewVariable
```

Listing 1.4 *How to post-process additional variables. Definition of variable itself (at the end of function **damage_main**, within the time step loop)*

Press the “COMPUTE” buttom to make changes effective: the new item “damage variable (d)” will appear in both “VAR-X” and “VAR-Y” pop-up menus.

1.3.3 Post-process options

Some post-process details can be customized by selecting the “OPTIONS” button, located in the upper left-hand side of the figure windows. In the following, we describe briefly these options. Other changes in post-process options, such as properties (color, width, font style ...) of points, lines, legends and other graphic objects, can be carried out either by modifying the arguments of the plotting functions *plotcurves* and *dibujar_criterio_dano1* or by using *Matlab*'s Property Editor.

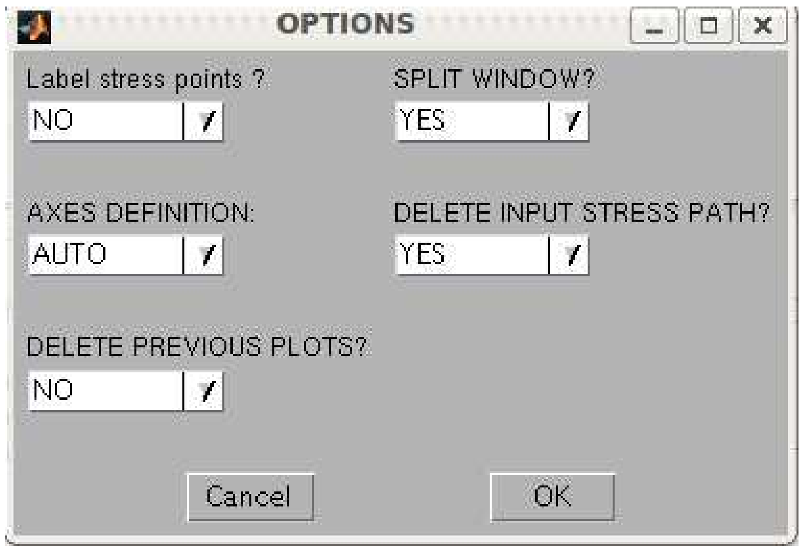

Figure 1.5 *Post-process options.*

- **Label stress points**. If set to "YES", then represented points are displayed with an integer-label that indicates the corresponding step number.
- **Axes definition**. It refers to axes scaling. By default, scaling mode is set to "auto". "Non-auto" mode corresponds to user defined axis, whereas "Current mode" means that the program will freeze the scaling at the current limits. Current limits, in turn, can be modified using "zoom" buttons.
- **Delete previous plots.** If set to "YES", old plots are kept while drawing new results. Otherwise, they will be automatically deleted.
- **Split window.** Stress path plot and XY graph can be shown in separate windows by selecting option "NO".
- **Delete input stress path**. If option "YES" is selected, the "effective" stress path is deleted from the current windows upon pressing the "COMPUTE" button.

Appendix A

Listing of functions

A.1 FUNCTION *CALLBACK_main.m*

(External link to the source file)

```
function CALLBACK_main
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Callback ---> See main.m
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ************************
%1) Extract DATA from gcf
% ************************
DATA = guidata(1);

%2) Defining variables (local names)
%    like alfa_03 = getfield(DATA.VAR.alfa_03)
%-----------------------------------------------
VARIABLES = fieldnames((DATA.VAR))  ;
for ivar = 1:length(VARIABLES)
    STRE = [VARIABLES{ivar},' = getfield(DATA.VAR,VARIABLES{ivar});' ];
    eval(STRE)  ;
end
% Name = Data.Name
% ---------------------
fn = fieldnames(DATA);
for i = 1:length(fn)  ;
    STR = [fn{i},' = getfield(DATA,fn{i});'];
    eval(STR)  ;
end
```

```
%4) Read inputs from graphic (uicontrols)
%   ----------------------------------------
% A) editboxes (VARIABLES_LEG), at the left
% ******************************************************
fhandle = guihandles(gcf) ; % --> Tag
VARIABLES_LEG = DATA.VARIABLES_LEG ;
for ivar = 1:length(VARIABLES_LEG)
    hread = getfield(fhandle,VARIABLES_LEG{ivar});
    STRE = [VARIABLES_LEG{ivar},' = str2num(get(hread,''String''));'];
    eval(STRE);
end
% %%%%%%%%%%%%%%%%%%%%
% 2) MDtype_c
% %%%%%%%%%%%%%%%%%%%%
hread = getfield(fhandle,'MDtype');
String = get(hread,'String')        ;
MDtype  = get(hread,'Value')          ;
MDtype_c = String{MDtype} ;
% %%%%%%%%%%%%%%%%%%%%
% 3) ntype_c
% %%%%%%%%%%%%%%%%%%%%
hread = getfield(fhandle,'ntype_c');
String = get(hread,'String')        ;
ntype  = get(hread,'Value')          ;
ntype_c = String{ntype} ;
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4) Ratio compression/traction strength
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hread = getfield(fhandle,'n');
String = get(hread,'String')        ;
n = str2num(String) ;
% %%%%%%%%%%%%%%%%%%%%
% *) HARDTYPE
% %%%%%%%%%%%%%%%%%%%%
hread = getfield(fhandle,'HARDTYPE_tag');
String = get(hread,'String')        ;
nnn  = get(hread,'Value')          ;
HARDTYPE = String{nnn} ;
% %%%%%%%%%%%%%%%%%%%%
% *) VISCOUS
% %%%%%%%%%%%%%%%%%%%%
hread = getfield(fhandle,'VISCOUS_tag');
String = get(hread,'String')        ;
nnn  = get(hread,'Value')          ;
VISCOUS = String{nnn} ;
```

```
%     ----------------------------------------
%  A) editboxes (VARIABLES_LEG3), at the left
% ****************************************************
fhandle = guihandles(gcf) ; % --> Tag
VARIABLES_LEG3 = DATA.VARIABLES_LEG3 ;
for ivar = 1:length(VARIABLES_LEG3)
    hread = getfield(fhandle,VARIABLES_LEG3{ivar});
    STRE = [VARIABLES_LEG3{ivar},' = str2num(get(hread,''String''));'];
    eval(STRE);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  COMPUTING AND PLOTING                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    ***************************************************************************************

%*        Inicializaciï¿½n de variables y puestas en ceros
%%*
if ntype == 1
    menu('PLANE STRESS has not been yet implemented','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3-DIMENSIONAL PROBLEM has not been yet implemented','STOP');
    error('OPTION NOT AVAILABLE')
else
    mstrain = 4     ;
    mhist   = 6     ;
end


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

Eprop=[YOUNG_M POISSON HARDSOFT_MOD 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')
if strcmp(splitwind,'YES')
    subplot(2,1,1);
    title('Damage surface (principal stresses axes)')

end
hold on;
grid on;
q=sigma_u/sqrt(E);
switch ErasePrPlot
    case 'YES'
        if isfield(DATA,'hplot')
            if ishandle(DATA.hplot) & DATA.hplot ~= 0
                delete(DATA.hplot)
            end
        end
end
hplot = dibujar_criterio_dano1(ce, nu, q , 'b-',MDtype ,n );
DATA.hplot = hplot ;

%*****************************
% Recomputing strains
%*****************************
if isfield(DATA,'SIGMAP')
    [SIGMAP,STRAIN,strain]=recompstr(SIGMAP,nnls_s,nu,ce,STRAIN,mstrain,
       istep1, istep2, istep3);
    DATA.SIGMAP = SIGMAP ;
```

```
    DATA.STRAIN = STRAIN ;
    DATA.strain = strain ;
end


hold on
switch axiskind
    case 'NON-AUTO'
        axis(axislim);
    otherwise
        axis auto
        %axis equal
end





%*********************************
% Storing all variables in DATA
%*********************************
for ivar = 1:length(VARIABLES) ;
    num_var = VARIABLES{ivar};
    eval(['DATA.VAR.',num_var,' = ',num_var,';']);
end

% New ones ...
DATA.VAR.ntype = ntype;
DATA.VAR.MDtype = MDtype;


guidata(gcf,DATA)
try
    save(DATA.NameWs);

    % save(DATA.NameWs,'-append');
    %save(DATA.NameWs,'DATA','YOUNG_M','POISSON','HARDSOFT_MOD','
        YIELD_STRESS','ntype_c','istep1','istep2','istep3',...
    %     'n','MDtype_c','NameWs','HARDTYPE','VISCOUS','eta','TimeTotal','
        ALPHA_COEFF');
catch
    if isunix
        % if exist('CALLBACK_main.m')
        % addpath([cd,'/AUX_SUBROUTINES']);
        % addpath(cd);
```

```
        pathdata = [cd,'/WSFILES/'];
        % end
    else
        %addpath([cd,'\AUX_SUBROUTINES']);
        pathdata = [cd,'\WSFILES\'];
    end

    rmdir(pathdata,'s');
    mkdir(pathdata) ;
    current_dir = cd ;
    cd(current_dir); run([cd,'/main.m'])

    %error('ERROR IN READING STORED DATA. RUN IT AGAIN')

end


%{'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','ntype_c','istep1','
    istep2','istep3',...
%      'n','MDtype_c','NameWs','HARDTYPE','VISCOUS','eta','TimeTotal','
    ALPHA_COEFF'}
%

%
%
%      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'}
%      % ------------
%      ntype_c = 'PLANE STRAIN' ;
%
%      % Number of increments of each load state
%      % ---------------------------------------
%      istep1 = 5 ;
%      istep2 = 6 ;
%      istep3 = 5 ;
%      % Ratio compression strength / tension strength
```

```
%       % ------------------------------------------------------------
%       n = 3 ;
%       % Model      PTC = { 'SYMMETRIC' , 'TRACTION' , 'NON-SYMMETRIC'}  ;
%       % --------------------------------------------------------------------
%       MDtype_c = 'SYMMETRIC' ;
%       try
%           save(NameWs) ;
%       catch
%           error('PATTERN PATH INCORRECT: Make sure that the currect
    directory contains file main.m')
%       end
%       % SOFTENING/HARDENING TYPE
%       % ----------------------------
%       HARDTYPE = 'LINEAR' ; %{LINEAR,EXPONENTIAL}
%       % VISCOUS/INVISCID
%       % ----------------------------
%       VISCOUS = 'NO' ;
%       % Viscous coefficient ----
%       % ----------------------------
%       eta = 0.3 ;
%       % TimeTotal (initial = 0) ----
%       % ----------------------------
%       TimeTotal = 10 ; ;
```

Listing A.1 *FUNCTION CALLBACK_main (Listed up to line 254)*

A.2 FUNCTION *Modelos_de_dano1.m*

(External link to the source file)

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

%*              Defining damage criterion surface
                                                    %*
%*


    %*
%*
%*                              MDtype=  1          :  SYMMETRIC
                                    %*
%*                              MDtype=  2          :  ONLY TENSION
                                 %*
```

```
%*                                    MDtype=   3          :  NON-SYMMETRIC
                                       %*
%*

    %*
%*

    %*
%*  OUTPUT:

    %*
%*                                    rtrial
                                                                     %*
%
    ***************************************************************************************




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

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

elseif  (MDtype==2)   %*  Only  tension


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

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

return
```

Listing A.2 *FUNCTION Modelos_de_dano1 (Listed up to line 28)*

A.3 FUNCTION *calstrain.m*

(External link to the source file)

```
function  strain = calstrain(istep,mstrain,STRAIN)
% See  select_path
```

```
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
```

Listing A.3 *FUNCTION calstrain (Listed up to line 17)*

A.4 FUNCTION *compute_load.m*

(External link to the source file)

```
function compute_load
global hplotSURF
%profile on
% See main.m
% Callback function for computing sigma = f(strain)

% ***********************
%1) Extract DATA from gcf
% ***********************
%dbstop('11')
DATA = guidata(gcf);
try
load(DATA.NameWs);
catch
    error('MAKE SURE CURRENT DIRECTORY CONTAINS main.m')
end

% For plotting
% ************
ncolores = 3 ;
colores = ColoresMatrix(ncolores);
markers = MarkerMatrix(ncolores) ;
```

```
if strcmp(splitwind,'YES')
    subplot(2,1,1);
end
hold on

%2) Defining variables (local names)
%    like alfa_03 = getfield(DATA.VAR.alfa_03)
%--------------------------------------------------
VARIABLES = fieldnames((DATA.VAR)) ;
for ivar = 1:length(VARIABLES)
    STRE = [VARIABLES{ivar},' = getfield(DATA.VAR,VARIABLES{ivar});' ];
    eval(STRE) ;
end
% Name = Data.Name
% ------------------
fn = fieldnames(DATA);
for i = 1:length(fn) ;
    STR = [fn{i},' = getfield(DATA,fn{i});'];
    eval(STR) ;
end


% PLOTTING (PATH)
% ********
% Divide SIGMAP{end} - SIGMAP{end-1} in istep1 steps
istep = [istep1 istep2 istep3] ;
try
[ hplotp hplotl]=plotpath(SIGMAP,hplotp,nnls_s,istep,hplotl);
catch
    error('ERROR: Select load path ')
end

DATA.hplotp = hplotp ;
DATA.hplotl = hplotl ;
% --------------
% INITIALIZING
% --------------
% For storing cauchy stress and others
% % **********************************
% sigma_v = cell(sum(istep)+1,1) ;
% hvar_n_v = cell(sum(istep)+1,1) ;
```

```
LISTH = {'hplots','hplotLABN','hplotSURF','hplotLLL','hplotquiver'};
strcom = {};
for ilist = 1:length(LISTH)
    hplotlocal = LISTH{ilist} ;
    switch ErasePrPlot
        case 'YES'
            strl = ['for ih = 1:length(',hplotlocal,');',[' if ishandle('
                ,hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;delete(',
                hplotlocal,'(ih))  ;        end;'],'      end'] ;
        otherwise
            strl = '' ;
    end
    eval([ 'if isfield(DATA,','''',hplotlocal,''')  ; ',hplotlocal,' =
        DATA.',hplotlocal,';',strl,'; end ;']);
    %eval(['for ih = 1:length(',hplotlocal,');',[' if ishandle(',
        hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;delete(',hplotlocal,'(
        ih))  ;        end;'],'      end']);
    eval([hplotlocal,'= 0;'])
end


hplots = 0 ;
hplotLABN = 0 ;
hplotSURF = 0 ;
hplotLLL  = 0 ;
%%%%%%%%%%%%

switch axiskind
    case 'NON-AUTO'
        axis(axislim);
    otherwise
        axis auto
end


% VARIABLES = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','ntype_c
    ', ...
%       'nnls_s','istep1','istep2','istep3','n','MDtype_c','mstrain', ...
%       'mhist','shownumber','axiskind','axislim','ErasePrPlot','vpx','vpy
    ','splitwind','pathdata', ...
%       'HARDTYPE','VISCOUS','eta','TimeTotal','alpha'} ;

%% Changing names
% ---------------
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





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

%delta_t = TimeTotal./istep/3 ;

Eprop   = [E nu HARDSOFT_MOD sigma_u hard_type viscpr eta ALPHA_COEFF]
              ;


% ---------------
% DAMAGE MODEL
% ---------------
[sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype,istep,
    strain,MDtype,n,TimeTotal);


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



%
% [ce]     = tensor_elastico1 (Eprop, ntype);
% eps_n1   = zeros(mstrain,1);
% hvar_n   = zeros(mhist,1);
```

```
% for i = 1:istep1+istep2++istep3+1
%
%      % 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
%      % ***************
%      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 ;
%      hvar_n_v{i} =  hvar_n ;
%      %aux_var_v{i} = aux_var ;
% end
%
%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% PLOTTING
% ————————

% PLOTTING
% —————————

% LABEL
% ——————
if strcmp(shownumber, 'YES')
    % strt = ['''\leftarrow N ='',','num2str(i)'];
        strt = [''' N ='',','num2str(i)'];
    string_1 = ['hplotLABN(end+1) = text(sigma_v{i}(1,1),sigma_v{i}(2,2)
        ,[',strt,'],''Color'',colores(1,:));']  ;
    string_2 = ['hplotLABN(end+1) = text(sigma_v{i}(1,1),sigma_v{i}(2,2)
        ,[',strt,'],''Color'',colores(2,:));']  ;
    string_3 = ['hplotLABN(end+1) = text(sigma_v{i}(1,1),sigma_v{i}(2,2)
        ,[',strt,'],''Color'',colores(3,:));']  ;
else
    string_1 = '' ;        string_2 = '' ;        string_3 = '' ;
end

for i = 2:istep1+1
    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_eig(2,2)],'LineWidth',2,'color',colores
       (1,:),'Marker',markers{1},'MarkerSize',2);
    eval(string_1);
    % SURFACES
    % ——————

end
for i = istep1+2:istep1+istep2+1
    stress_eig = (sigma_v{i}) ;
    tstress_eig = (sigma_v{i-1}) ;
    hplotLLL(end+1) = plot([tstress_eig(1,1) stress_eig(1,1) ],[
       tstress_eig(2,2) stress_eig(2,2)],'LineWidth',2,'color',colores
       (2,:),'Marker',markers{2},'MarkerSize',2);
    eval(string_2);

end
for i = istep1+istep2+2:istep1+istep2+istep3+1
    stress_eig = (sigma_v{i}) ;
    tstress_eig = (sigma_v{i-1}) ;
```

```
    hplotLLL(end+1) = plot([tstress_eig(1,1) stress_eig(1,1) ],[
        tstress_eig(2,2) stress_eig(2,2)],'LineWidth',2,'color',colores
        (3,:),'Marker',markers{3},'MarkerSize',2);
    eval(string_3);

end






% % SURFACES
% % ------
% if(aux_var(1)>0)
%     hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n(6), 'r:',
    MDtype,n );
%     set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1);
% end



DATA.sigma_v      = sigma_v      ;
DATA.vartoplot    = vartoplot    ;
DATA.LABELPLOT    = LABELPLOT    ;
DATA.TIMEVECTOR = TIMEVECTOR     ;

% Modify wplotx/y
% ----------------
wplotx = cat(2,wplotx0,LABELPLOT);
wploty = cat(2,wploty0,LABELPLOT);




for ilist = 1:length(LISTH)
    hplotlocal = LISTH{ilist} ;
    eval(['DATA.',hplotlocal,' = ',hplotlocal,';']) ;
end




%**********************************
% Storing all variables in DATA
%**********************************
```

```
for ivar = 1:length(VARIABLES) ;
    num_var = VARIABLES{ivar};
    eval(['DATA.VAR.',num_var,' = ',num_var,';']);
end









guidata(gcf,DATA)
save(DATA.NameWs);

%save(DATA.NameWs,'-append');
%save(DATA.NameWs,'DATA','YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS
    ','ntype_c','istep1','istep2','istep3',...
%      'n','MDtype_c','NameWs','HARDTYPE','VISCOUS','eta','TimeTotal','
    ALPHA_COEFF');

plotcurves ;

%profile report
```

Listing A.4 *FUNCTION compute_load (Listed up to line 278)*

A.5 FUNCTION *damage_main.m*

(External link to the source file)

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,ntype
    ,istep,strain,MDtype,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 historic variables
%  (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) = x 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  matlab
%                                        vLABELPLOTariable 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
% ——————————————————————————
LABELPLOT = {'hardening variable (q)','internal variable','NEW_VARIABLE'
    };

E       = Eprop(1)  ; nu = Eprop(2)  ;
viscpr = Eprop(6)  ;
sigma_u = Eprop(4);



if ntype == 1
    menu('PLANE STRESS has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
elseif ntype == 3
    menu('3-DIMENSIONAL PROBLEM has not been implemented yet','STOP');
    error('OPTION NOT AVAILABLE')
```

```
else
    mstrain = 4      ;
    mhist   = 6      ;
end

if viscpr == 1
    % Comment/delete lines below once you have implemented this case
    % *******************************************************
    menu({'Viscous model has not been implemented yet. '; ...
        'Modify files "damage_main.m","rmap_dano1" ' ; ...
        'to include this option'}, ...
        'STOP');
    error('OPTION NOT AVAILABLE')
else
end


totalstep = sum(istep) ;


% INITIALIZING GLOBAL CELL ARRAYS
% ----------------------------------
sigma_v = cell(totalstep+1,1) ;
TIMEVECTOR = zeros(totalstep+1,1) ;
delta_t = TimeTotal./istep/3 ;


% 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 =sqrt(1-nu*nu)*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)];
vartoplot{i}(1) = hvar_n(6) ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n(5) ; % Internal variable (r)


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) = hvar_n(5)^2  ; % Internal variable (r)
    end
```

Listing A.5 *FUNCTION damage_main (Listed up to line 175)*

A.6 FUNCTION *dibujar_criterio_dano1.m*

(External link to the source file)

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

%*                   PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODEL
                                        %*
%*

   %*
%*       function [ce] = tensor_elastico (Eprop, ntype)
                                %*
%*

   %*
%*       INPUTS                                                          %*
%*

   %*
%*                        Eprop(4)     vector de propiedades de material
                        %*
%*                                          Eprop(1)=  E------>modulo de
   Young              %*
%*                                          Eprop(2)=  nu------>modulo de
   Poisson            %*
%*                                          Eprop(3)=  H------>modulo de
   Softening/hard. %*
```

```
%*                                              Eprop(4)=sigma_u———>tensiï¿½n
    ï¿½ltima               %*
%*                          ntype                                       %*
%*                                        ntype=1   plane  stress
                                    %*
%*                                        ntype=2   plane  strain
                                    %*
%*                                        ntype=3   3D
                                              %*
%*                          ce(4,4)       Constitutive  elastic  tensor   (PLANE S.
           )        %*
%*                          ce(6,6)                                       ( 3D)
                         %*
%
    *****************************************************************************************

%
    *****************************************************************************************
%*              Inverse  ce
                                                                                 %*
ce_inv=inv(ce);
c11=ce_inv(1,1);
c22=ce_inv(2,2);
c12=ce_inv(1,2);
c21=c12;
c14=ce_inv(1,4);
c24=ce_inv(2,4);
%
    *****************************************************************************************

%
    *****************************************************************************************
% POLAR COORDINATES
if MDtype==1
```

```
    tetha=[0:0.01:2*pi];
elseif MDtype==2
    % Comment/delete lines below once you have implemented this case
    % *****************************************************
         menu({'Damage surface "ONLY TENSION" has not been implemented
             yet. '; ...
                'Modify files "damage_main.m","rmap_dano1" and "
                   dibujar_criterio_dano1"' ; ...
                'to include this option'}, ...
              'STOP');
         error('OPTION NOT AVAILABLE')

elseif MDtype==3
    % Comment/delete lines below once you have implemented this case
    % *****************************************************
       menu({'Damage surface "NON-SYMMETRIC" has not been implemented yet.
           '; ...
                'Modify files "damage_main.m","rmap_dano1" and "
                   dibujar_criterio_dano1"' ; ...
                'to include this option'}, ...
              'STOP');
         error('OPTION NOT AVAILABLE')

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



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

%* RADIUS
D=size(tetha);                       %*  Range
m1=cos(tetha);                       %*
m2=sin(tetha);                       %*
Contador=D(1,2);                     %*


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

for i=1:Contador
```

```
    %
        ************************************************************************************
    %* DAMAGE MODEL
        %*
    %*        Modelos de Daï¿½o
                                                                                       %*
    %*        MDtype=1 SYMMETRIC
                                                                                    %*
    %*        MDtype=2 TENSILE
                                                                                      %*
    %*        MDtype=3 NON-SYMMETRIC


    if MDtype==1
        %dbstop('86')
        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);

    elseif MDtype==2



    elseif MDtype==3


    end


end
hplot =plot(s1,s2,tipo_linea);
%
    ************************************************************************************
return
```

Listing A.6 *FUNCTION dibujar_criterio_dano1 (Listed up to line 112)*

A.7 FUNCTION *main.m*

(External link to the source file)

```
clc
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program for modelling damage model
% (Elemental gauss point level)
% GRAPHIC INTERFACE
% ---------------------
% Developed by J.Hdez Ortega
% 20-May-2007, TEchnical University of Catalonia
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%profile on

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


current_dir = cd ;
if isunix



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

else

    % addpath(current_dir);
    pathdata = [current_dir,'\WSFILES\'];
end
%
% if exist(pathdata,'dir') == 0
%      mkdir('\AUX_SUBROUTINES\WSFILES\')  ;
% end


% ****************
% INPUTS
% ****************
% ------------------------------------
% OTHER INPUTS (Graphic Inputs)
% ------------------------------------
Inc       = [0 -0.040 0 0] ;
NameFileExec = 'CALLBACK_main';
NameFileExecP = 'redraw_path';
```

```
Position = [0.01  0.9500  0.08  0.030]  ;
PTC = {'SYMMETRIC','ONLY TENSION','NON-SYMMETRIC'};
TP = {'PLANE STRESS','PLANE STRAIN','3D'}  ;
ce = 0  ;
nnls_s     = 3  ; % Number of load states
mstrain = 4  ;   % Number of components of strain vector
mhist    = 6  ; % Number of componets of historical variables vector
shownumber = 'YES'        ;
axiskind    = 'NON-AUTO'     ;
axislim     = [-500 500 -500 500]  ;
ErasePrPlot = 'YES'  ;
wplotx        ={'STRAIN_1','STRAIN_2','|STRAIN_1|','|STRAIN_2|','norm(
    STRAIN)','TIME'};
vpx           = 'STRAIN_1'  ;
wploty        ={'STRESS_1','STRESS_2','|STRESS_1|','|STRESS_2|','norm(
    STRESS)'};
vpy           = 'STRESS_1'  ;
splitwind    = 'YES'  ;
HARDLIST      = {'LINEAR','EXPONENTIAL'}  ;
% Inc         = [0  -0.040  0  0]  ;
% ------------------------------------------------
%Set of variables (to be stored in DATA)
% -------------------------------------------------

%   Workspace name
NameWs = [pathdata,'tmp1_maing.mat'];
%%%%% MODEL INPUTS ( as uicontrols)
wplotx0 = wplotx;wploty0 = wploty;



    COMPTA = 0  ;

if exist(NameWs) == 2
     try
     load(NameWs);
catch
    COMPTA = 1  ;
end

else
    COMPTA = 1  ;
end


if  COMPTA == 1
```

```
% 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'}
% ------------
ntype_c = 'PLANE STRAIN' ;

% Number of increments of each load state
% ---------------------------------------
istep1 = 5 ;
istep2 = 6 ;
istep3 = 5 ;
% Ratio compression strength / tension strength
% ---------------------------------------------
n = 3 ;
% Model     PTC = { 'SYMMETRIC', 'TRACTION', 'NON-SYMMETRIC'} ;
% ------------------------------------------------------
MDtype_c = 'SYMMETRIC' ;
try
    save(NameWs) ;
catch
    error('PATTERN PATH INCORRECT: Make sure that the currect
        directory contains file main.m')
end
% SOFTENING/HARDENING TYPE
% -------------------------
HARDTYPE = 'LINEAR' ; %{LINEAR,EXPONENTIAL}
% VISCOUS/INVISCID
% -------------------------
VISCOUS = 'NO' ;
% Viscous coefficient ----
% -------------------------
eta = 0.3 ;
% TimeTotal (initial = 0) ----
% -------------------------
```

```
    TimeTotal = 10 ; ;
    % Integration coefficient ALPHA
    % ---------------------------
    ALPHA_COEFF = 0.5 ;
end



VARIABLES = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS','ntype_c',
    ...
    'nnls_s','istep1','istep2','istep3','n','MDtype_c','mstrain', ...
    'mhist','shownumber','axiskind','axislim','ErasePrPlot','vpx','vpy','
       splitwind','pathdata', ...
    'HARDTYPE','VISCOUS','eta','TimeTotal','ALPHA_COEFF','wplotx','wploty
       '} ;

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

clf;figure(1);clf;
hold on
grid on
xlabel('\sigma_1');
ylabel('\sigma_2');
%-----------------------------------------------------
% Edit boxes (-->VARIABLES_LEG), at the LEFT
%-----------------------------------------------------
VARIABLES_LEG = {'YOUNG_M','POISSON','HARDSOFT_MOD','YIELD_STRESS'};
VARIABLES_TEXT = {'YOUNG_M','POISSON','HARD./SOFT MOD.','ULT. STRESS'};
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
% %%%%%%%%%%%%%%%%%%%
[ok ntype] = FndStrInCell(TP,ntype_c) ;
fff =uicontrol('Style', 'text', 'String', 'Problem type',...
    'Units','normalized','Position', [0.568 0.974 0.18 0.02],'FontSize'
        ,10,'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String', TP,...
    'Units','normalized','Position', [0.568 0.917 0.18 0.057],'Callback',
        NameFileExec,...
    'Tag','ntype_c','Value',ntype);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4) Ratio compression/traction strength
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fff =uicontrol('Style', 'text', 'String', {'ONLY FOR ','NON-SYMMETRIC','
   ------------'},...
    'Units','normalized','Position',[0.01 0.472 0.08 0.050],'FontSize',9,
        'FontWeight','Bold');
fff =uicontrol('Style', 'text', 'String', 'ratio comp/trac',...
    'Units','normalized','Position',[0.01 0.442 0.08 0.02],'FontWeight','
        Bold','FontSize',9);
fff =uicontrol('Style', 'edit', 'String', num2str(n),...
    'Units','normalized','Position',[0.01 0.418 0.08 0.02],'FontSize',9,'
        tag','n',...
    'Callback',NameFileExec);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% 5) INCREMENTS FOR EACH LOAD STATE
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fff =uicontrol('Style', 'text', 'String', {'INCREMENTS','-----------------'},...
    'Units','normalized','Position',[0.913 0.893 0.08 0.030],'FontSize'
        ,9,'FontWeight','Bold');
VARIABLES_LEG2 = {'istep1','istep2','istep3'};
Position = [0.913 0.893 0.08 0.025] ;
Inc      = [0 -0.030 0 0] ;

for ileg = 1:length(VARIABLES_LEG2)
    var_i = VARIABLES_LEG2{ileg} ;
    var_inum = num2str(eval(var_i));
    Position = Position + Inc ;
    STRE = ['htext = uicontrol(''Style'',''text'',''Units'',''normalized'
        ',''Position'',Position,''FontSize'',9,''string'',''',var_i,''');'
        ] ;
    eval(STRE);
    Position = Position + Inc ;
    STRE =['hp',num2str(ileg),' = uicontrol(''Style'',''edit'',''String''
        ,var_inum,''Units'',''normalized'',''Position'',Position,''
        FontSize'',9,''Tag'',''',var_i,''', ''Callback'',NameFileExecP);'
        ];
    eval(STRE);
end

% ----------------------------------------------------
% Pushbottoms
% ----------------------------------------------------
hpushsel1 = uicontrol('Style','pushbutton',...
    'String',{'SELECT LOAD PATH'},'Units','normalized','Position'
        ,[0.00703125 0.119 0.11015 0.0336],'Callback','select_path', ...
    'tag','select_load');

hpushsel1 = uicontrol('Style','pushbutton',...
    'String',{'COMPUTE'},'Units','normalized','Position',[0.00703125
        0.065 0.11015 0.044],'Callback','compute_load', ...
    'tag','c');

hpushsel1 = uicontrol('Style','pushbutton',...
    'String',{'REFRESH'},'Units','normalized','Position',[0.00703125
        0.022 0.11015 0.03687],'Callback','refresh_main', ...
    'tag','c');

hpushsel1 = uicontrol('Style','pushbutton',...
    'String',{'OPTIONS'},'Units','normalized','Position',[0.1 0.94 0.08
```

```
        0.05],'Callback','showoptions', ...
    'tag','showoptions2');


% %%%%%%%%%%%%%%%%%%%%
% *) PLOT X
% %%%%%%%%%%%%%%%%%%%%
[ok nvx] = FndStrInCell(wplotx,vpx) ;
fff =uicontrol('Style', 'text', 'String', 'VAR X',...
    'Units','normalized','Position', [0.2 0.974 0.14 0.02],'FontSize',10,
        'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String', wplotx,...
    'Units','normalized','Position', [0.2 0.917 0.14 0.057],'Callback','
        plotcurves',...
    'Tag','xplotc','Value',nvx);
% %%%%%%%%%%%%%%%%%%%%
% *) PLOT Y
% %%%%%%%%%%%%%%%%%%%%
[ok nvy] = FndStrInCell(wploty,vpy) ;
fff =uicontrol('Style', 'text', 'String', 'VAR Y',...
    'Units','normalized','Position', [0.36 0.974 0.14 0.02],'FontSize'
        ,10,'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String', wploty,...
    'Units','normalized','Position', [0.36 0.917 0.14 0.057],'Callback','
        plotcurves',...
    'Tag','yplotc','Value',nvy);

%%%% HARDTYPE
[ok ntype] = FndStrInCell(HARDLIST,HARDTYPE) ;
fff =uicontrol('Style', 'text', 'String', 'HARD/SOFT LAW',...
    'Units','normalized','Position',[0.01 0.745 0.079 0.015] ,'FontSize'
        ,9,'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String', HARDLIST,...
    'Units','normalized','Position',[0.01 0.7150 0.077 0.023] ,'Callback'
        ,NameFileExec,...
    'Tag','HARDTYPE_tag','Value',ntype,'FontSize',8);

%%%% VISCOUS
[ok ntype] = FndStrInCell({'YES','NO'},VISCOUS) ;
fff =uicontrol('Style', 'text', 'String', 'VISCOUS MODEL',...
    'Units','normalized','Position',[0.01 0.696 0.079 0.015] ,'FontSize'
        ,9,'FontWeight','Bold');
fff =uicontrol('Style', 'popupmenu', 'String',{'YES','NO'} ,...
    'Units','normalized','Position',[0.01 0.666 0.077 0.023] ,'Callback',
        NameFileExec,...
```

```
    'Tag','VISCOUS_tag','Value',ntype,'FontSize',8);

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

%--------------------------------------------------------
%   Edit  boxes   OTHER PARAMETERS
%--------------------------------------------------------
VARIABLES_LEG3 = {'eta','TimeTotal','ALPHA_COEFF'};
VARIABLES_TEXT = {'visc. coeff.','TIME INT.(s)','ALPHA coeff.'};
Inc_tv      = 0.002   ;
Inc_vt      = 0.002  ;
PositionT0 = [0.01  0.638  0.079  0.015   ]  ;
PositionV0 = [0.01  0.613  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_LEG3)  ;
    var_i = VARIABLES_LEG3{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



%*********************************
% Storing  all  variables  in  DATA
%*********************************
for ivar = 1:length(VARIABLES)  ;
    num_var = VARIABLES{ivar};
    eval(['DATA.VAR.',num_var,' = ',num_var,';']);
end
DATA.VARIABLES_LEG = VARIABLES_LEG  ;
DATA.VARIABLES_LEG2 = VARIABLES_LEG2  ;
DATA.VARIABLES_LEG3 = VARIABLES_LEG3  ;
DATA.NameWs            = NameWs              ;
DATA.wplotx0 = wplotx0  ;
```

```
DATA.wploty0 = wploty0 ;

%---------------------------------------
% Attach DATA to the current figure
%---------------------------------------
guidata(gcf,DATA);
CALLBACK_main ;


%profile report
```

Listing A.7 *FUNCTION main*

A.8 FUNCTION *plotcurves.m*

(External link to the source file)

```
function plotcurves
% Plot stress vs strain (callback function)
% ---------------------------------------------



% ***********************
%1) Extract DATA from gcf
% ***********************
DATA = guidata(gcf);

%2) Defining variables (local names)
%    like alfa_03 = getfield(DATA.VAR.alfa_03)
%------------------------------------------------
VARIABLES = fieldnames((DATA.VAR)) ;
for ivar = 1:length(VARIABLES)
    STRE = [VARIABLES{ivar},' = getfield(DATA.VAR,VARIABLES{ivar});' ];
    eval(STRE) ;
end
% Name = Data.Name
% ------------------
fn = fieldnames(DATA);
for i = 1:length(fn) ;
    STR = [fn{i},' = getfield(DATA,fn{i});'];
    eval(STR) ;
end
fhandle = guihandles(gcf) ; % --> Tag
```

```
% %%%%%%%%%%%%%%%%%%%%
% 2) PLOT X
% %%%%%%%%%%%%%%%%%%%%
hread     = getfield(fhandle,'xplotc');
String    = get(hread,'String')       ;
nvx       = get(hread,'Value')        ;
vpx       = String{nvx}  ;

% %%%%%%%%%%%%%%%%%%%%
% 3) PLOT Y
% %%%%%%%%%%%%%%%%%%%%
hread     = getfield(fhandle,'yplotc');
String    = get(hread,'String')       ;
nvy       = get(hread,'Value')        ;
vpy       = String{nvy}  ;




%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

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


if strcmp(splitwind,'YES')
    subplot(2,1,2)
else
    figure(2)
    set(2,'Name','CURVES')
end
hold on
grid on
xlabel(vpx);
ylabel(vpy);

% wplotx        ={'STRAIN_1','STRAIN_2','|STRAIN_1|','|STRAIN_2|','norm(
    STRAIN)'};
```

```
% warning('JAHO_B')
% load('tmp.mat');

% DATA X
% ———

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) =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);' ;
    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);
        catch
            warning('*  ')
        end
    end

    LISTH = {'hplotgraph','hplotLABN2'};
    strcom = {};
    for ilist = 1:length(LISTH)
        hplotlocal = LISTH{ilist} ;
```

```
        switch ErasePrPlot
            case 'YES'
                strl = ['for ih = 1:length(',hplotlocal,');',[' if
                    ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih)
                    ~=0 ;delete(',hplotlocal,'(ih))   ;        end;'],'
                        end']  ;
            otherwise
                strl = '' ;
        end
        eval([ 'if isfield(DATA,','''',hplotlocal,''') ; ',hplotlocal
            ,' = DATA.',hplotlocal,';',strl,'; end ;']);
        %eval(['for ih = 1:length(',hplotlocal,');',[' if ishandle(',
            hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;delete(',
            hplotlocal,'(ih))   ;        end;'],'       end']);
        eval([hplotlocal,'= 0;'])
    end



    hplotgraph(end+1) = plot(X(1:istep1+1),Y(1:istep1+1),'Marker',
        markers{1},'Color',colores(1,:));
    hplotgraph(end+1) = plot(X(istep1+1:istep1+istep2),Y(istep1+1:
        istep1+istep2),'Marker',markers{2},'Color',colores(2,:));
    hplotgraph(end+1) = plot(X(istep1+istep2:end),Y(istep1+istep2:end
        ),'Marker',markers{3},'Color',colores(3,:));


    % % LABEL
    % % ------
    % switch ErasePrPlot
    %     case 'YES'
    %         if isfield(DATA,'hplotLABN2')
    %             if ishandle(DATA.hplotLABN2) & DATA.hplotLABN2~=0
    %                 delete(DATA.hplotLABN2)
    %             end
    %         end
    %         hplotLABN2 = 0 ;
    %     otherwise
    %         hplotLABN2 = DATA.hplotLABN2 ;
    % end



    if strcmp(shownumber,'YES')
        for i = 1:istep1+1
```

```
            % strt = ['\leftarrow N =',num2str(i)];
              strt = ['  N =',num2str(i)];
             hplotLABN2(end+1) = text(X(i),Y(i),strt,'Color',colores
                 (1,:));
        end
        for i = istep1+2:istep1+istep2+1
          %  strt = ['\leftarrow N =',num2str(i)];
          strt = ['  N =',num2str(i)];
             hplotLABN2(end+1) = text(X(i),Y(i),strt,'Color',colores
                 (2,:));
        end
        for i = istep1+istep2+2:istep1+istep2+istep3+1
          %  strt = ['\leftarrow N =',num2str(i)];
          strt = ['  N =',num2str(i)];
             hplotLABN2(end+1) = text(X(i),Y(i),strt,'Color',colores
                 (3,:));
        end

    end


    DATA.hplotgraph = hplotgraph ;
    DATA.hplotLABN2 =hplotLABN2 ;




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


    %*********************************
    % Storing all variables in DATA
    %*********************************
    for ivar = 1:length(VARIABLES) ;
        num_var = VARIABLES{ivar};
        eval(['DATA.VAR.',num_var,' = ',num_var,';']);
    end


    guidata(1,DATA)
  % save(DATA.NameWs,'-append');
           save(DATA.NameWs);

 %   save(DATA.NameWs, 'DATA');
```

Listing A.8 *FUNCTION plotcurves (Listed up to line 228)*

A.9 FUNCTION *plotpath.m*

(External link to the source file)

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

% Plot iloc-th stretch
% --------------------
PNT = SIGMAP{1} ;
hplotp(end+1) = plot(PNT(1),PNT(2),'ro');
for iloc = 1:nnls_s
    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
```

Listing A.9 *FUNCTION plotpath (Listed up to line 19)*

A.10 FUNCTION *rmap_dano1.m*

(External link to the source file)

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

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

%*                                                              *
%*           Integration Algorithm for a isotropic damage model
```

```
%*
%*

   *
%*              [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps_n1,hvar_n,
   Eprop,ce)          *
%*

   *
%* INPUTS                    eps_n1(4)    strain (almansi)      step n+1
                                *
%*                                        vector R4     (exx eyy exy ezz)
                             *
%*                           hvar_n(6)    internal variables , step n
                                *
%*                                        hvar_n(1:4)  (empty)
                                   *
%*                                        hvar_n(5) = r   ; hvar_n(6)=q
                                *
%*                           Eprop(:)     Material parameters
                                         *
%*
%*                           ce(4,4)      Constitutive elastic tensor
                                *
%*

   *
%* OUTPUTS:                  sigma_n1(4)  Cauchy stress   , step n+1
                                   *
%*                           hvar_n(6)    Internal variables , step n+1
                                     *
%*                           aux_var(3)   Auxiliar variables for computing const
   . tangent tensor  *
%
   ***********************************************************************************



hvar_n1 = hvar_n;
r_n     = hvar_n(5);
q_n     = hvar_n(6);
E       = Eprop(1);
nu      = Eprop(2);
H       = Eprop(3);
sigma_u = Eprop(4);
```

```
hard_type = Eprop(5) ;
%
   ****************************************************************************



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

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



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

%*          Damage surface
                                                                      %*
[rtrial] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
%
   ****************************************************************************



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

%*     Ver el Estado de Carga
                                                                   %*
%*    ----------->      fload=0 : elastic unload
                                                      %*
%*    ----------->      fload=1 : damage (compute algorithmic constitutive
    tensor)             %*
fload=0.D0;

if(rtrial > r_n)
    %*     Loading
```

```
    fload=1.D0;
    delta_r=rtrial-r_n;
    r_n1= rtrial   ;
    if hard_type == 0
        % Linear
        q_n1= q_n+ H*delta_r;
    else
        % Comment/delete lines below once you have implemented this case
        % *******************************************************
        menu({'Hardening/Softening exponential law has not been
            implemented yet. '; ...
             'Modify file "rmap_dano1" ' ; ...
             'to include this option'},  ...
             'STOP');
        error('OPTION NOT AVAILABLE')
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    end


else

    %*      Elastic load/unload
    fload=0.D0;
    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';

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


%
```

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

Listing A.10 *FUNCTION rmap_dano1 (Listed up to line 129)*

A.11 FUNCTION *select_path.m*

(External link to the source file)

```
function selec_path
% Selecting stress path
% ------------------------------
%profile on
% ***********************
%1) Extract DATA from gcf
% ***********************
DATA = guidata(gcf);
load(DATA.NameWs);

if strcmp(splitwind,'YES')
```

```
    subplot(2,1,1);
end
hold on
xlabel('\sigma_1') ;
ylabel('\sigma_2') ;

%2) Defining variables (local names)
%   like alfa_03 = getfield(DATA.VAR.alfa_03)
%-----------------------------------------------
VARIABLES = fieldnames((DATA.VAR)) ;
for ivar = 1:length(VARIABLES)
    STRE = [VARIABLES{ivar},' = getfield(DATA.VAR,VARIABLES{ivar});' ];
    eval(STRE) ;
end


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

[ce]    = tensor_elastico1_d (E,nu, ntype);

switch ntype_c
    case 'PLANE STRAIN'
        % POLINOMIAL PATH
        % ***************
        % (3 steps)
        SIGMAP = cell(1,nnls_s+1) ;
        SIGMAP{1} = zeros(1,4)  ;
        STRAIN = cell(1,nnls_s+1) ;
        STRAIN{1} =  zeros(1,4)  ;

        % ERASE
        % *****
        LISTH = {'hplots','hplotLABN','hplotSURF','hplotLLL','hplotp','
            hplotl','hplotquiver'};
        strcom = {};
        for ilist = 1:length(LISTH)
            hplotlocal = LISTH{ilist} ;
            switch ErasePrPlot
                case 'YES'
                    strl = ['for ih = 1:length(',hplotlocal,');',[' if
                        ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih)
                        ~=0 ;delete(',hplotlocal,'(ih))  ;       end;'],'
```

```
                        end'] ;
            otherwise
                strl = '' ;
        end
        strl = ['for ih = 1:length(',hplotlocal,');',[' if ishandle('
            ,hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;delete(',
            hplotlocal,'(ih))    ;         end;'],'       end'] ;
        eval([  'if isfield(DATA,','''',hplotlocal,''') ;  ',hplotlocal
            ,' = DATA.',hplotlocal,';',strl,';end ;']);
        %           eval(['for ih = 1:length(',hplotlocal,');',[' if
            ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;
            delete(',hplotlocal,'(ih))    ;         end;'],'       end']);
        eval([hplotlocal,'= 0;'])
    end


    %%%  STRETCHES
    %%%%%%%%%%%%%%%
    %          LABELT = {'DEFINE FIRST STRECHT (FIRST POINT = [0  0]) ';
    %                'DEFINE SECOND STRECHT';
    %                'DEFINE THIRD STRECHT'};
    LABELT = {'DEFINE FIRST STRESS INCREMENT (FIRST POINT = [0  0]) ';
        'DEFINE SECOND STRESS INCREMENT';
        'DEFINE  THIRD STRESS INCREMENT'};
    istep = [istep1,istep2,istep3] ;

    aold = [0   0 ] ;


    for iloc = 1:nnls_s
        choice = menu(LABELT{iloc},'GRAPHIC SELECTION','BASE');
        if choice == 1
            [a]=ginput(1);
            SaveAns{1} = num2str(a(1));
            SaveAns{2} = num2str(a(2));
            save([pathdata,'tmpsp',num2str(iloc),'.mat'],'SaveAns');
        else
            [inca1 inca2] = ...
                MenuMake(['STRESS INCREMENT COORDINATES (',num2str(
                    iloc),')'],'on',[pathdata,'tmpsp',num2str(iloc),'.
                    mat'] ,0, ...
                'INCREMENT SIGMA 1 = ',[1  10],'1.0',0,0,[1],0,...
                'INCREMENT SIGMA 2 = ',[1  10],'1.0',0,0,[1],0);
            a(1) = aold(1)+inca1 ; a(2) = aold(2)+inca2 ;
        end
```

```
            % We assume we are in the elastic range, thus sigma_33 =
                poisson*(sigma_11+sigma_22)
            sigma_0=[a(1) a(2) 0  nu*(a(1)+a(2))];
            aold = a ;
            % iloc-th point of the path is stored in SIGMAP ;
            SIGMAP{iloc+1} =  sigma_0 ;
            sigma_bef = SIGMAP{iloc} ;
            stress_incre = sigma_0 - sigma_bef ;
            % Plot stress increment vector
            % **************************
% %              if exist('quiver.m') > 0
%                   if isunix
%                       hplotquiver(end+1) = quiver(sigma_bef(1),sigma_bef
    (2),stress_incre(1),stress_incre(2),0) ;
%                       set(hplotquiver(end),'LineWidth',1)
%                   else
%                       [aaa ] = quiver(sigma_bef(1),sigma_bef(2),
    stress_incre(1),stress_incre(2),0) ;
%                        hplotquiver(end+1:end+2) = aaa ;
%                   end
%
%              else
                hplotquiver(end+1) = plot([sigma_bef(1) sigma_0(1)],[
                    sigma_bef(2) sigma_0(2)]) ;
                %end

            % Strain
            % ------------------
            strain_di =(inv(ce)*sigma_0') ';
            STRAIN{iloc+1} = strain_di ;

        end


        % Plot iloc-th stretch
        % ----------------------
        [ hplotp hplotl]=plotpath(SIGMAP,hplotp,nnls_s,istep,hplotl);

        %%% STRAIN EVOLUTION
        %%%%%%%%%%%%%%%%%%%%
        [strain] = calstrain(istep,mstrain,STRAIN) ;
end


switch axiskind
```

```
    case 'NON-AUTO'
        axis(axislim);
    otherwise
        axis auto
end
% OUTPUTS
% -------
DATA.hplotp = hplotp ;
DATA.hplotl = hplotl ;
DATA.hplotquiver = hplotquiver ;
DATA.strain = strain ;
DATA.LABELT = LABELT ;
DATA.SIGMAP = SIGMAP ;
DATA.STRAIN = STRAIN  ;
%DATA.istep  = istep  ;

guidata(gcf,DATA)


save(DATA.NameWs);

%save(DATA.NameWs,'DATA','-append');
```

Listing A.11 *FUNCTION select_path (Listed up to line 153)*

A.12 FUNCTION *showoptions.m*

(External link to the source file)

```
function showoptions
% Set preferences
% ---------------



% ***********************
%1) Extract DATA from gcf
% ***********************
DATA = guidata(gcf);
load(DATA.NameWs);

%2) Defining variables (local names)
%    like alfa_03 = getfield(DATA.VAR.alfa_03)
%-----------------------------------------------
VARIABLES = fieldnames((DATA.VAR))  ;
```

```
for ivar = 1:length(VARIABLES)
    STRE = [VARIABLES{ivar},' = getfield(DATA.VAR,VARIABLES{ivar});' ];
    eval(STRE) ;
end

%%%
% Check if axiskind change its values
% -----------------------------------------
LISTErasePrPlot = {'YES','NO'} ;
LISTaxiskind = {'AUTO','NON-AUTO','CURRENT'} ;
if exist([pathdata,'showoopt.mat'])

    try
        load([pathdata,'showoopt.mat']) ; % --> SaveAns
        axiskindBEF = LISTaxiskind(SaveAns{2}) ;
        ErasePrPlotBEF = LISTErasePrPlot(SaveAns{3}) ;
        ErasePrPlotBEF_path = LISTErasePrPlot(SaveAns{5}) ;
    catch
        axiskindBEF = 'AUTO' ;
        ErasePrPlotBEF = 'YES' ;
        ErasePrPlotBEF_path = 'YES' ;
    end
else
    axiskindBEF = 'AUTO' ;
    ErasePrPlotBEF = 'YES' ;
    ErasePrPlotBEF_path = 'YES' ;

end


%%%

% warning('JAHO')
% load('/home/joaquin/USO_COMUN_MATLAB/COMMON_FILES/GuillPracMatlab/
    AUX_SUBROUTINES/tmp_miiiiiii.mat')

try
[shownumber,axiskind,ErasePrPlot,splitwind,ErasePrPlot_path] = ...
    MenuMake('OPTIONS','on',[pathdata,'showoopt.mat'] ,0, ...
    'Label stress points ? ',[2 10],{'YES','NO'},2,0,[1],0, ...
    'AXES DEFINITION: ',[2 10],LISTaxiskind,1,0,[1],0, ...
    'DELETE PREVIOUS PLOTS? ',[2 10],LISTErasePrPlot,1,0,[1],0,...
    'SPLIT WINDOW? ',[2 10],{'YES','NO'},1,0,[1],0, ...
    'DELETE INPUT STRESS PATH? ',[2 10],LISTErasePrPlot,1,0,[1],0);
catch
```

```
    ErasePrPlot_path = 'YES';
end

if ~strcmp(ErasePrPlotBEF,ErasePrPlot)
    switch  ErasePrPlot
        case 'YES'
            LISTH = {'hplot','hplotLABN','hplotLABN2','hplotLLL','
                hplotSURF','hplotl','hplotp','hplotgraph','hplotquiver'};
            strcom = {};
            for ilist = 1:length(LISTH)
                hplotlocal = LISTH{ilist} ;
                switch ErasePrPlot
                    case 'YES'
                        strl = ['for ih = 1:length(',hplotlocal,');',['
                            if ishandle(',hplotlocal,'(ih)) & ',hplotlocal
                            ,'(ih) ~=0 ;delete(',hplotlocal,'(ih))    ;
                                      end;'],'       end'] ;
                    otherwise
                        strl = '' ;
                end
                eval([ 'if isfield(DATA,','''',hplotlocal,''') ; ',
                    hplotlocal,' = DATA.',hplotlocal,';',strl,'; end ;']);
                %eval(['for ih = 1:length(',hplotlocal,');',[' if
                    ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0
                    ;delete(',hplotlocal,'(ih))   ;          end;'],'
                    end']);
                eval([hplotlocal,'= 0;'])
            end
    end
end



    switch  ErasePrPlot_path
        case 'YES'
            LISTH = {'hplotp','hplotquiver','hplotl'};
            strcom = {};
            for ilist = 1:length(LISTH)
                hplotlocal = LISTH{ilist} ;
                switch ErasePrPlot_path
                    case 'YES'
                        strl = ['for ih = 1:length(',hplotlocal,');',['
                            if ishandle(',hplotlocal,'(ih)) & ',hplotlocal
                            ,'(ih) ~=0 ;delete(',hplotlocal,'(ih))    ;
                                      end;'],'       end'] ;
```

```
                    otherwise
                        strl = '' ;
                end
                eval([ 'if isfield(DATA,','''',hplotlocal,''') ; ',
                    hplotlocal,' = DATA.',hplotlocal,';',strl,'; end ;']);
                %eval(['for ih = 1:length(',hplotlocal,');',[' if
                    ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0
                    ;delete(',hplotlocal,'(ih)) ;          end;'],'
                    end']);
                eval([hplotlocal,'= 0;'])
            end
    end




if ~strcmp(axiskindBEF,axiskind)
    switch axiskind
        case 'NON-AUTO'
            [X_min,X_max,Y_min,Y_max] = ...
                MenuMake('Axes limits','on',[pathdata,'showoopt1.mat']
                    ,0, ...
                'X_min',[1 12],'-100.0',0,0,{},0,...
                'X_max',[1 12],'40.0',0,0,{},0,...
                'Y_min',[1 12],'0.0',0,0,{},0,...
                'Y_max',[1 12],'80',0,0,{},0);
            axislim = [X_min,X_max,Y_min,Y_max] ;
        case 'CURRENT'
            axislim = axis ;
            SaveAns{1} = axislim(1) ;
            SaveAns{2} = axislim(2) ;
            SaveAns{3} = axislim(3) ;
            SaveAns{4} = axislim(4) ;
            save([pathdata,'showoopt1.mat'],'SaveAns') ;
            load([pathdata,'showoopt.mat'],'SaveAns') ;
            SaveAns{2} = 2 ;
            save([pathdata,'showoopt.mat'],'SaveAns') ;
            axiskind = 'NON-AUTO' ;

    end
end

if strcmp(shownumber,'NO')
    LISTH = {'hplotLABN','hplotLABN2'};
    strcom = {};
```

```
    for ilist = 1:length(LISTH)
        hplotlocal = LISTH{ilist} ;
        switch ErasePrPlot
            case 'YES'
                strl = ['for ih = 1:length(',hplotlocal,');',[' if
                   ishandle(',hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0
                   ;delete(',hplotlocal,'(ih))   ;         end;'],'
                   end'] ;
            otherwise
                strl = '' ;
        end
        eval([ 'if isfield(DATA,'','''',hplotlocal,''') ; ',hplotlocal,' =
            DATA.',hplotlocal,';',strl,'; end ;']);
        %eval(['for ih = 1:length(',hplotlocal,');',[' if ishandle(',
           hplotlocal,'(ih)) & ',hplotlocal,'(ih) ~=0 ;delete(',
           hplotlocal,'(ih))   ;         end;'],'       end']);
        eval([hplotlocal,'= 0;'])
    end

end





%**********************************
% Storing all variables in DATA
%**********************************
for ivar = 1:length(VARIABLES) ;
    num_var = VARIABLES{ivar};
    eval(['DATA.VAR.',num_var,' = ',num_var,';']);
end


guidata(gcf,DATA)
%save(DATA.NameWs,'-append');
save(DATA.NameWs);

CALLBACK_main ;
```

Listing A.12 *FUNCTION showoptions (Listed up to line 166)*

A.13 FUNCTION *tensor_elastico1.m*

(External link to the source file)

```
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
```

Listing A.13 *FUNCTION tensor_elastico1 (Listed up to line 36)*

References

Marchand, P. and Holland, O. (2003). *Graphics and GUIs with MATLAB*. CRC Pr I Llc.