UPC

COMPUTATIONAL SOLIDS MECHANICS

Assignment I - Program for modeling damage model

Alba Navarro Casanova

08/04/2016

INDEX

1.Introduction

In the world of solid mechanics are used generally continuous damage models. These models are helpful for numerical simulations when the material has micro fractures. The aim of this work is correctly implement algorithms and perform numerical integration of constitutive models continuous damage as well.

It is also important to demonstrate the accuracy of the code. The purpose of the code for a material with mechanical properties is calculated for plane stress problem strains, damage variables, and as a post process effective stresses and the tangent constitutive tensor.

Program for modelling damage model

The main objective of this assignment is to create a program that allows us to calculate the damage on a solid. For this they have provided us a code that we have to complete. The default program allows to calculate the level of damage to the non-viscous symmetric case.

What we are asked is:

a) Implement in the supplied MATLAB code the integration algorithms (rate independent and plane strain case) for:

1. The continuum isotropic damage "non-symmetric tension-compression damage" model.

2. The "tension-only" damage model.

b) Implement the following cases for each of those models:

1. linear and exponential hardening/softening (H<0 and H>0)

c) Assess the correctness of the implementation: for each of the models in section

a) obtain the path at the stress space and the stress-strain curve, corresponding to appropriate loading paths starting at the point $\sigma_1^{(0)} = 0$; $\sigma_2^{(0)} = 0$ and described by three-segment paths in the strain space $\Delta\varepsilon^{(1)} \rightarrow \Delta\varepsilon^{(2)} \rightarrow \Delta\varepsilon^{(3)}$ .They are defined, in terms of their corresponding effective stress increments $\Delta\bar{\sigma}^{(1)} = \boldsymbol{C} : \Delta\varepsilon^{(1)} \rightarrow \Delta\bar{\sigma}^{(2)} = \boldsymbol{C} : \Delta\varepsilon^{(2)} \rightarrow \Delta\bar{\sigma}^{(3)} = \boldsymbol{C} : \Delta\varepsilon^{(3)}$, as:

1.

$\Delta\bar{\sigma}_1^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0$ (uniaxial tensile loading)
$\Delta\bar{\sigma}_1^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = 0$ (uniaxial tensile unloading/compressive loading)
$\Delta\bar{\sigma}_1^{(3)} = \gamma \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = 0$ (uniaxial compressive unloading/ tensile loading)

2.

$\Delta\bar{\sigma}_1^{(1)} = \alpha \quad ; \quad \Delta\bar{\sigma}_2^{(1)} = 0$ (uniaxial tensile loading)
$\Delta\bar{\sigma}_1^{(2)} = -\beta \quad ; \quad \Delta\bar{\sigma}_2^{(2)} = -\beta$ (biaxial tensile unloading/compressive loading)
$\Delta\bar{\sigma}_1^{(3)} = \gamma \quad ; \quad \Delta\bar{\sigma}_2^{(3)} = \gamma$ (biaxial compressive unloading/tensile loading)

3.

$\Delta\bar{\sigma}_1^{(1)} = \alpha \;\; ; \;\; \Delta\bar{\sigma}_2^{(1)} = \alpha$ (biaxial tensile loading)
$\Delta\bar{\sigma}_1^{(2)} = -\beta \;\; ; \;\; \Delta\bar{\sigma}_2^{(2)} = -\beta$ (biaxial tensile unloading/compressive loading)
$\Delta\bar{\sigma}_1^{(3)} = \gamma \; ; \;\; \Delta\bar{\sigma}_2^{(3)} = \gamma$ (biaxial compressive unloading/tensile loading)

d) Implement in the supplied MATLAB code the integration algorithm (plane strain case) for the continuum isotropic visco-damage "symmetric tensioncompression" model.

e) Assess the correctness of the implementation: consider the following cases (for a specific given Poisson ratio and linear hardening/softening parameter):

- Different viscosity parameters η.
- Different strain rate, $\dot{\varepsilon}$, values.
- Different α values: 0.25, 0.5, 0.75, and $\alpha = 1$ (for the α time-integration method)

Obtain results displaying:

1) The effects of the previous values on the obtained stress-strain curves in appropriate loading paths.
2) The effects of the α values, on the evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators.

2.Models for the elastic domain

In this section we will consider three models for the elastic domain (we have the symmetric case, traction only, and not symmetrical).

Now we will explain each one of the types

-*symmetric model* of the three cases is the simplest. Has the same yield stress limits for compressive and tensile σ1 and σ2 the two principal stresses. In our case σu = 200 and υ = Poisson coefficient 0.3. Poisson's ratio increases the ellipse. The yield stress ellipsoid radii controls the domain.

-*reaction-damage-only model* is useful if the materials are not pure compression injuries.

-*Not symmetrical push-pull model*: for materials in which the damage occurs in pure tension for older states that the tensile strength.

3.Rate independent model

The independent rate model considers the state of stress (strain) is not dependent on the speed and load can be calculated regardless of the time. For pure load condition changes they are on the surface domain, but for the inviscid model, states stress must be within the elastic domain to meet the conditions Karush-Kuhn-Tucker.

For pure load state stress leads to surface damage, this behavior depends on the material. It is interesting to use a linear or an exponential model. This damage process can be a

hardening or softening. The linear / softening hardening law it was already implemented in the code, so the next section is to assess the accuracy of the exponential law.

It is noted that for hardening the surface is expanding, while for softening is being circumvented, but in both cases the state of tension is always in the elastic domain.

4. Rate dependent models

We use this for those materials which shows different behaviour when the load is applied in a rate of velocities, that could be modelled as a rate dependent model that takes into account the viscous effects.

It is interesting to check that it is possible to recall the rate independent results with the rate dependent code when the viscous effects are small enough. The new parameters for the viscous model are the viscosity parameter η, and the T time of the load application.

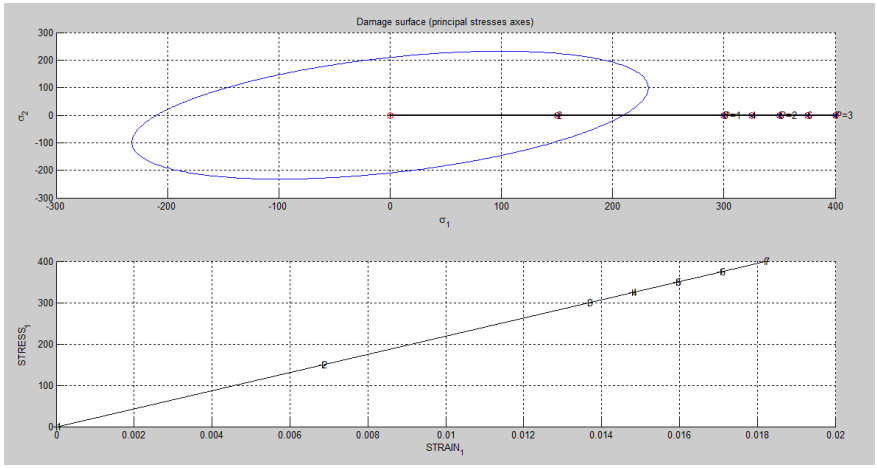
However, two more parameters: ΔT which discretizes in time (ΔT is the number of steps for each load state, thus larger values means that there is more discretization), and α which defines the integration type, both variables are needed to perform the numerical integration of the constitutive equation.

It's important to take care about the influence of ΔT and α on the results, in order to not introduce numerical errors or instabilities on the computations. The following sections describe the effects of these parameters are shown.

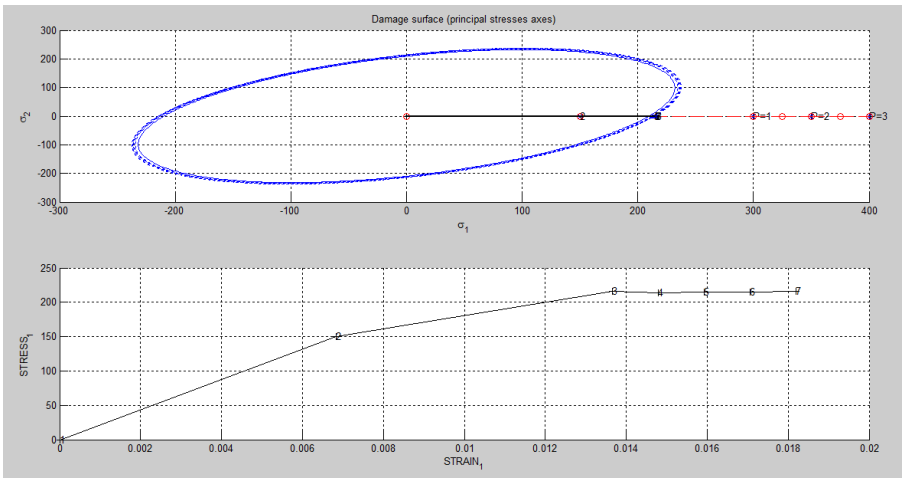
5. Figures

In the images that follow then shown as the solution it varies when we change the values of the different parameters. Let's check the results for different values of the parameter alpha

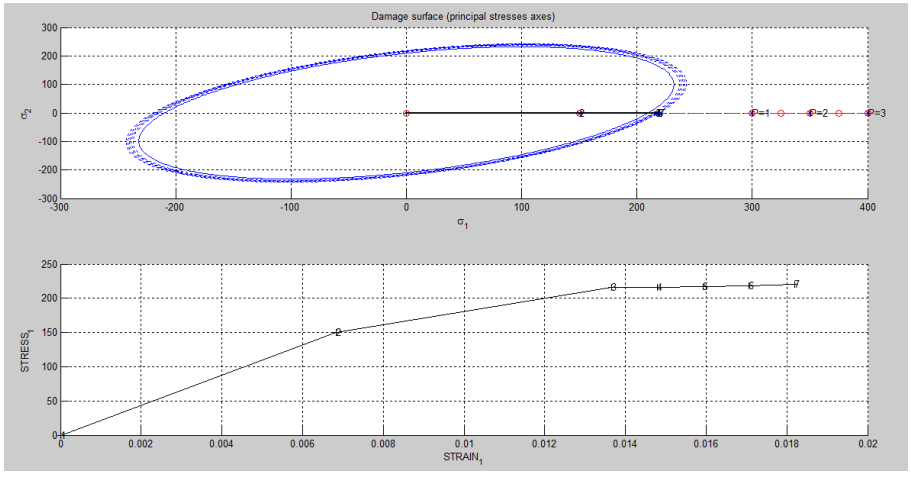
Young modulus	20000
Poisson's coefficient	0,3
Hard/Softening Modulus	0,1
Yield Stress	200
Problem type	Plane Strain
Type of damage surfaces	Symmetric
Type of Hard/softening law	Linear
Viscosity parameter	0,3
Lenght of the interval of time	10
Alpha coefficient	0



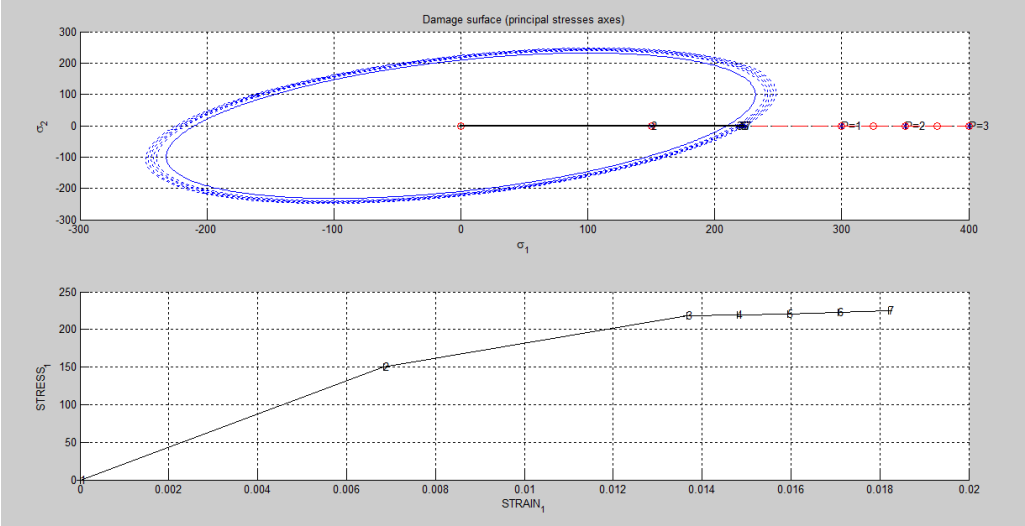
For alpha = 0.25



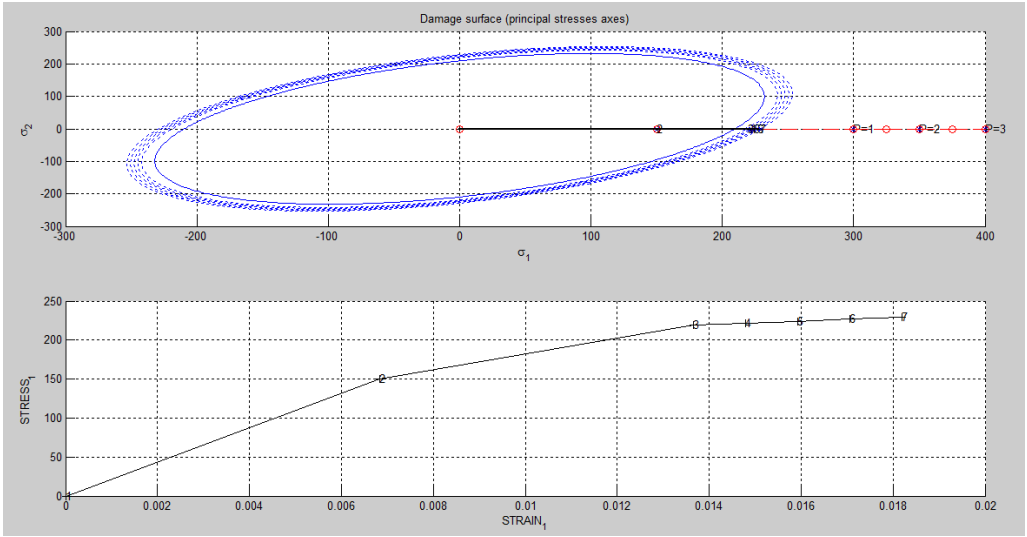
For alpha=0.5



For alpha= 0.75

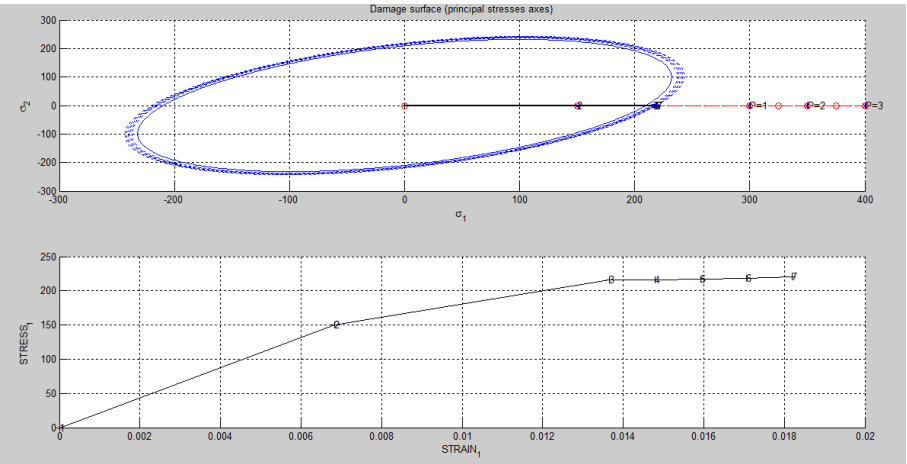


For alpha=1

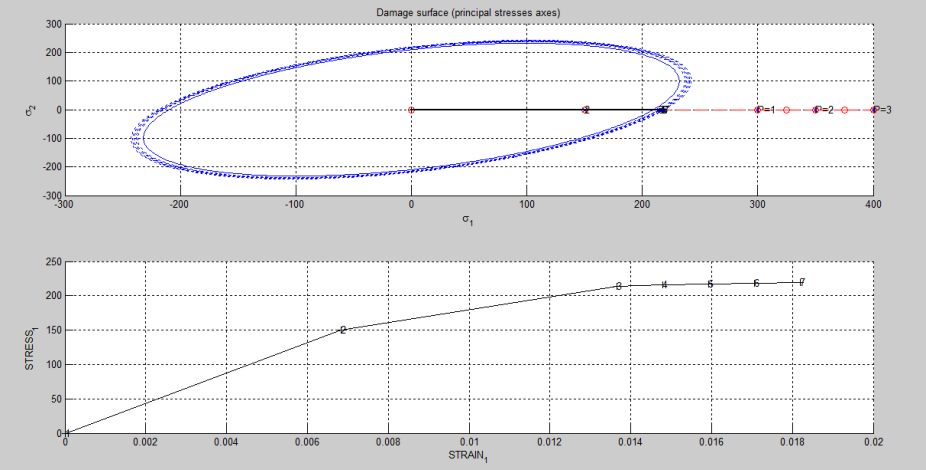


As we can see for an alpha equal to 0.5 is when we start to get good results. Therefore we take as default alpha equal to 0.5.

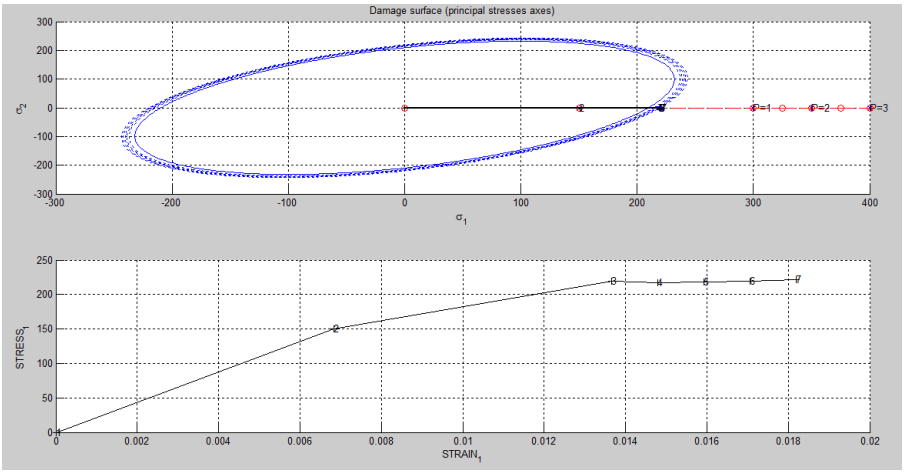
It is also interesting to see how it changes our solution based on the coefficient of viscosity. Taking alpha = 0.5 and the same data in the table above, with a viscosity of 0.3 have



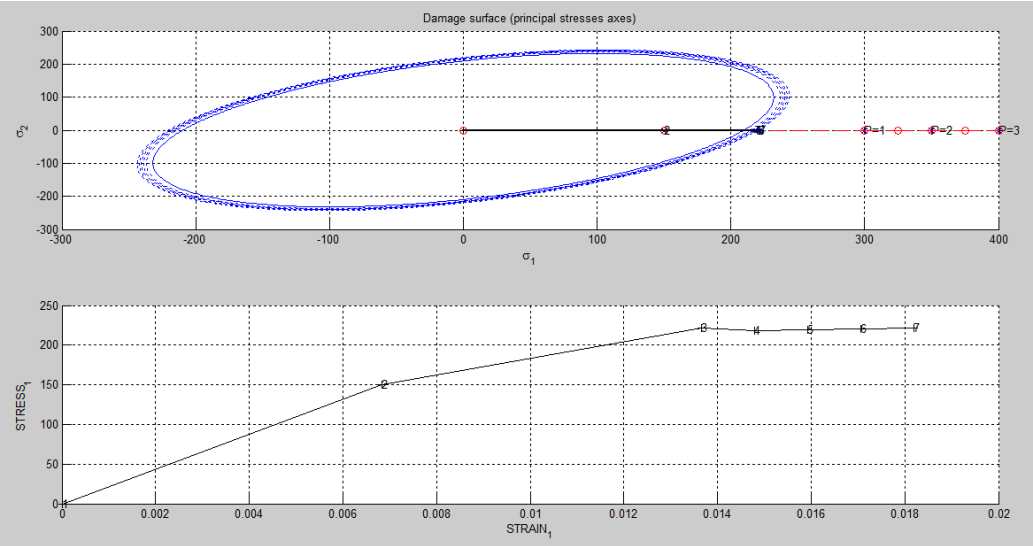
now let's see what happens with an inviscid, ie viscosity 0



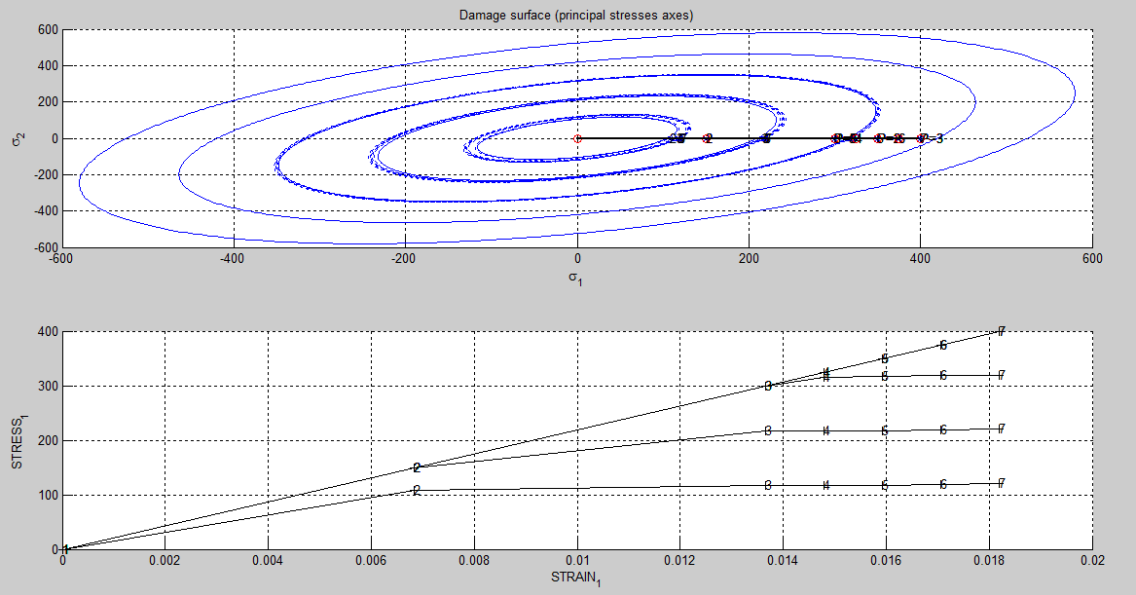
for a higher viscosity than we have by default, that is greater than 0.3, for example 0.7 we have results like these:



and for a viscosity of 1



On the other hand we analyze the results for different stresses. In the following graph we can see how the material behaves when it is subjected to different stresses. The result is displayed for stresses of 100, 200, 300, 400 and 500.



The smaller ellipse corresponds to a stress of 100 and the largest to 500. We can see also that with a stress of 100, we get a response as the lowest line in the second figure. While with a stress of 500 we get a behavior as the highest line in the figure below.

6.Conclusions

We assessed the rate dependent and independent implementations rate through several figures.
We can see that for purposes of low viscosity (viscosity of slow parameters, or application time large enough load) results depend on the type of tends to independent rate, it makes sense.

For alpha greater than half of the results show instabilities, and larger for small values of alpha. We can see also that this instability appears just for low viscosity parameter. This is a very good point, because the theory says that for alpha equal to half second order error is achieved, but there is no instabilities.

The first component of algorithmic constitutive tensor which coincides with the operator constitutive tangent of alpha equals zero, and algorithmic constitutive tensor coincides with the constitutive tensor alpha equal 1 and the viscosity parameter equal to 0. Reduce time discretization is a good strategy to reduce instabilities.

6.Annex

In the matlab file: rmap_dano

We have introduced

```
viscrp= Eprop(6); %if viscrp= 1 then its viscous
eta = Eprop(7); % Viscosity coefficient
alpha = Eprop(8); %
%*********************************************************************
```

We implement the exponential law in the code as

```
menu({'Hardening/Softening exponential law has not been implemented
yet. '; ...
% 'Modify file "rmap_dano1" ' ; ...
% 'to include this option'}, ...
% 'STOP');
% error('OPTION NOT AVAILABLE')
q_infi=r0*1.3;
A = (H*r0)/(q_infi-r0);
H_new= (A*(q_infi-r0)*exp(A*(1-rtrial/r0)))/r0;
q_n1= q_n + H_new*delta_r;
end

if(q_n1<zero_q)
q_n1=zero_q;
end
else
%* Elastic load/unload
fload=0;
r_n1= r_n ;
q_n1= q_n ;
end
%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%5
%%%%%%%%%%%%%%%%%%%%%
else %FOR VISCOUS ('viscrp == 1')
rtrial=(1-alpha)*r_n+alpha*rtrial;
if(rtrial > r_n)
%* Loading
fload=1;
delta_r=rtrial-r_n;
r_n1= (((eta-delta_t*(1-alpha))/(eta+alpha*delta_t))*r_n) + ....
((delta_t/(eta+alpha*delta_t))*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')
q_infi=r0*1.3;
A = (H*r0)/(q_infi-r0);
H_new= (A*(q_infi-r0)*exp(A*(1-rtrial/r0)))/r0;
q_n1= q_n + H_new*delta_r;
end
if(q_n1<zero_q)
q_n1=zero_q;
end
```

```
else
%* Elastic load/unload
fload=0;
r_n1= r_n ;
q_n1= q_n ;
end
```

……

In the main_nointeractive file:

main_nointeractive

```
SIGMAP = zeros(nloadstates,2) ;
SIGMAP(1,:) =[300 0];
SIGMAP(2,:) =[350 0];
SIGMAP(3,:) =[400 0];

% Number of time increments for each load state
% ---------------------------------------
istep = 2*ones(1,nloadstates) ;
% VARIABLES TO PLOT
vpx = 'STRAIN_1' ; % AVAILABLE OPTIONS: 'STRAIN_1', 'STRAIN_2'
% '|STRAIN_1|', '|STRAIN_2|'
% 'norm(STRAIN)', 'TIME'
vpy = 'STRESS_1'; % AVAILABLE OPTIONS: 'STRESS_1', 'STRESS_2'
% '|STRESS_1|', '|STRESS_2|'
% 'norm(STRESS)', 'TIME', 'DAMAGE VAR.','hardening variable
(q)','damage variable (d)'
```

NOTE: The full code can be seen in the zip attachment.