Numerical Integration of Constitutive Damage Models Using Matlab

Computational Solid Mechanics

Christian Rossi

April 2016

1 Continuum Damage Models

The term Continuum Damage Mechanics has been used to models materials which are characterized by loss of stiffness, i.e. by a decrease in their stiffness modulus. Damage models have also been used to simulate different materials (fragile and ductile), which are fundamentally characterized by irreversible material degradation. Physically speaking, we can describe the degradation of mechanical material properties as processes in which the initiation and growth (propagation) of micro-defects such as micro pores and micro-cracks take place

1.1 Description of the Isotropic Damage Model

A consideration between two different types of stresses has to be made in a continuum damage model: apparent stresses (σ) and effective stresses ($\underline{\sigma}$).

The apparent stresses are external and they act on the whole section (s) while the apparent stresses act on the undamaged regions ($\underline{s}$).

Damage variable, denoted by d, is a value which relates the apparent stresses and the effective stresses. For a completely damaged section, the damage variable takes the value of 1, while for a not damage section its value is equal to zero. It depends on time and it can't decrease because the damage surface can't decrease too. It can be interpreted as once the material has suffered damage this will be permanent.

So the constitutive equation for isotropic damage model will be:

$$\sigma = (1-d)\mathbb{C} : \varepsilon \tag{1}$$

Where $\mathbb{C}$ is the elastic tensor (4th order definite positive tensor).

The Damage function,which gives the measures of the amount of damage, is introduced as:

$$f(\sigma, r) = \tau_\sigma - q(r) \tag{2}$$

$$g(\varepsilon, r) = \tau_\varepsilon - r \tag{3}$$

Where equation 2 is related to the stress space and equation 3 is related to the strain space. The value of τ_σ in the equation 2 corresponds to the stress norm while the value of τ_σ in the equation 3 corresponds to the strain norm.

The q(r) is the hardening/softening variable. The hardening/softening variable is a scalar function which depends of the measure of the strains and it controls the size of the elastic domain. When the stress state lies inside said damage surface, the material shows elastic behavior ($f < 0$). The damage criterion requires that the current stress state must be on ($f = 0$) or inside the damage surface.

The elastic domain increases with the increasing of $q(r)$ while as the variable $q(r)$ decreases, the domain is reduced. The hardening/softening parameter H which is the rate of the hardening/softening variable, is characterized by:

- Demage with Hardening $H(r) > 0$
- Perfect Damage $H(r) = 0$
- Damage with Softening $H(r) < 0$

The Linear Hardening/Softening Law, which assuming that q varies linearly with r is:

$$q(r) = r_0 + H(r - r_0) \tag{4}$$

$$H = dq(r)/dr \tag{5}$$

The Exponential Hardening/Softening Law can expressed as:

$$q(r) = q_\infty - (q_\infty - r_0)e^{A(1-r/r_0)} \tag{6}$$

$$H = dq(r)/dr = A\frac{(q_\infty - r_0)}{r_0}e^{A(1-r/r_0)} \tag{7}$$

2 Matlab Code

The executables Matlab files given are called *Main nointeractive*, whit no user interface, and *Main*, which uses an user interface. Input datas are provided togheter with the executable files. the input data are:

- Young Modulus (E)
- Poisson's coefcient (ν)
- Hardening/softening modulus (H)
- Uniaxial elastic limit(σ_u)
- Ratio compression strength/tension strength (n)
- Viscosity parameter (η)
- Alpha coefficient (α)

The user can also modified different conditions of the problems, as:

- Type of analysis (plane stress, plane strain or 3D)
- Type of damage surface (Symmetric model, Tensile-damage-only or Non-symmetric model)
- Type of Hardening/Softening Law (Linear or Exponential)
- Viscous/Inviscid case
- Length of the interval of time (T)

3 Assignment 1

3.1 Part 1 (rate independent models)

In the first part of the assignment is required the implementation in the supplied MATLAB code the integration algorithms (rate independent and plane strain case) for the continuum isotropic damage "non-symmetric tension-compression damage" model and the "tension-only" damage model. In order to implement both codes we have to take into account the positive counterpart of the stress tensor, where the positive part of the stresses are extracted ($\overline{\sigma}^+$).

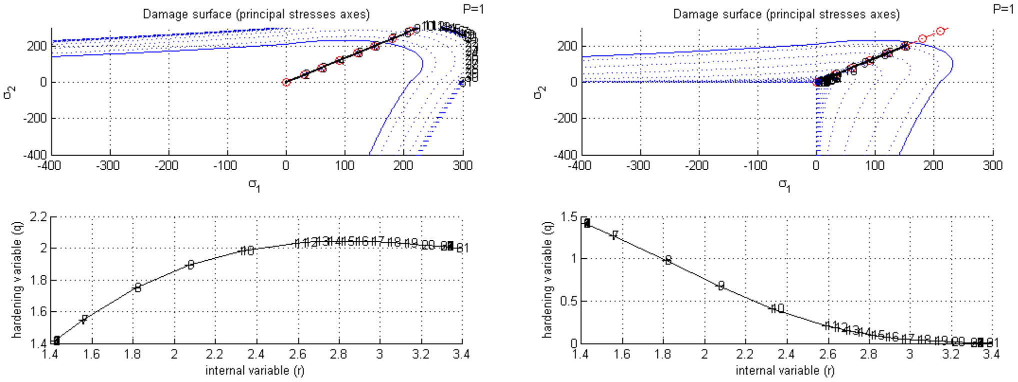
The "tension-only" damage model does not take into account failure by compression and the "non-symmetric tension-compression damage" model is useful in order to simulate materials which tension domain differs with respect to compression.

For the "non-symmetric tension-compression damage" model two new parameter have to be taken into account:

- The Weight factor θ
- The n parameter

The Weight factor parameter depends on the stress state given and the n parameter is defined by means of the ratio of the compression elastic limit to the tension elastic limit.

Once the previous models are implemented, Linear Law, equation 4, and Exponential Hardening/Softening Law ($H < 0$ and $H > 0$), equation 6, are needed. For the linear case the Hardening/Softening Parameter (H) is constant value, while for the exponential case it depends of the internal variable (equation 7).



((a)) Hardening/Softening Parameter (H=1)

((b)) Hardening/Softening Parameter (H=-1)

Figure 1: Exponential Hardening/Softening Law for ”tension-only” damage model

Once all the previous implementations of the damage models are made we proceed by obtaining the path at the stress space and the stress-strain curve, corresponding to appropriate loading paths starting at the point $\sigma_1 = 0$; $\sigma_2 = 0$ and described by three-segment paths in the strain space $\Delta\varepsilon_1 \rightarrow \Delta\varepsilon_2 \rightarrow \Delta\varepsilon_3$.

Inviscid Model (Tension only damage model)

Case 1

$\Delta\sigma_1 = \alpha$; $\Delta\sigma_2 = 0$ *Uniaxial tensile loading*
$\Delta\sigma_1 = -\beta$; $\Delta\sigma_2 = 0$ *Uniaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = 0$ *Uniaxial compressive unloading/tensile loading*

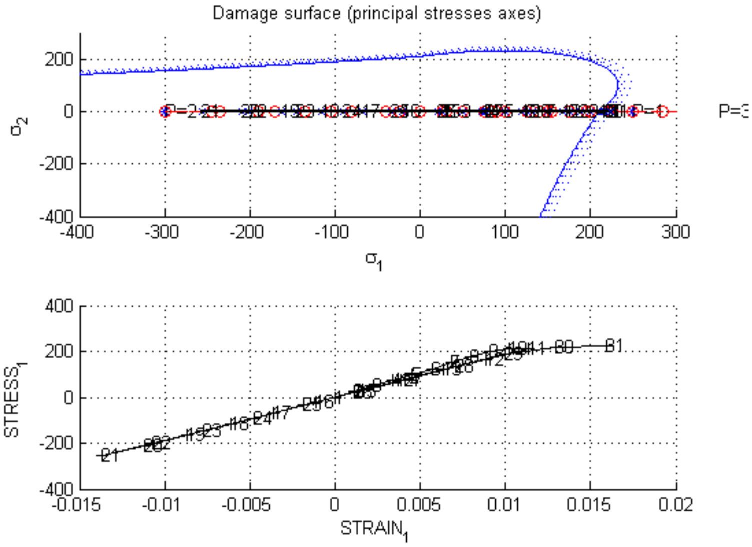


Figure 2: Inviscid Model - Tension only damage model (Case 1)

Figure 13 shows how the material behaves elastically as the tensile load is less then the yield stress, which is fixed up to 100 N/m^2 for all the next simulations. As the the limit is exceed the elastic domain increases and the material experiment deformation. During the next stage (compressive loading) the material experiments compression without no changes of the domain and even for the last stage, during tensile loading, no deformation has been registered.

Case 2

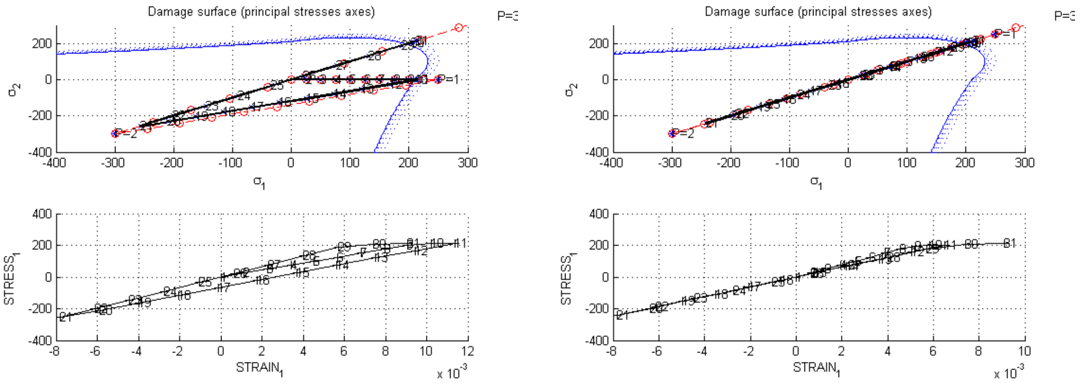
$\Delta\sigma_1 = \alpha$; $\Delta\sigma_2 = 0$ *Uniaxial tensile loading*
$\Delta\sigma_1 = -\beta$; $\Delta\sigma_2 = -\beta$ *Biaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = \gamma$ *Biaxial compressive unloading/tensile loading*

Case 3

$\Delta\sigma_1 = \alpha$; $\Delta\sigma_2 = \alpha$ *Biaxial tensile loading*
$\Delta\sigma_1 = -\beta$; $\Delta\sigma_2 = -\beta$ *Biaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = \gamma$ *Biaxial compressive unloading/tensile loading*

Figure 3(a) and Figure 3(b) show the same behaviour for the two initial stages (tensile loading and compressive loading) registered in the Case 1, but differences have been registered during the last stage: the elastic domain has been increased and and a domain deformation has been marked.

Where for the all cases α=250, β=300 and γ=350, with linear Hardening Modulus of 0.1.



((a)) Case 2

((b)) Case 3

Figure 3: Inviscid Model - Tension only damage model

Inviscid Model (Non Symmetric tension compression damage model)

Case 1

$\Delta\sigma_1 = -\alpha$; $\Delta\sigma_2 = 0$ *Uniaxial tensile loading*
$\Delta\sigma_1 = \beta$; $\Delta\sigma_2 = 0$ *Uniaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = 0$ *Uniaxial compressive unloading/tensile loading*

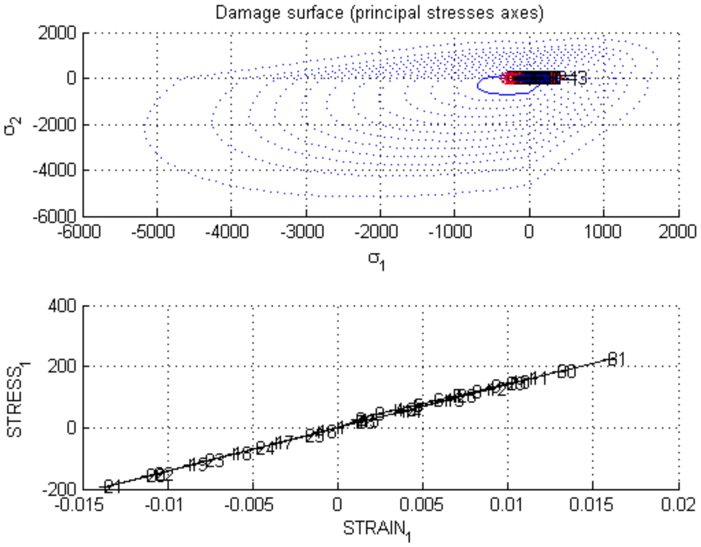
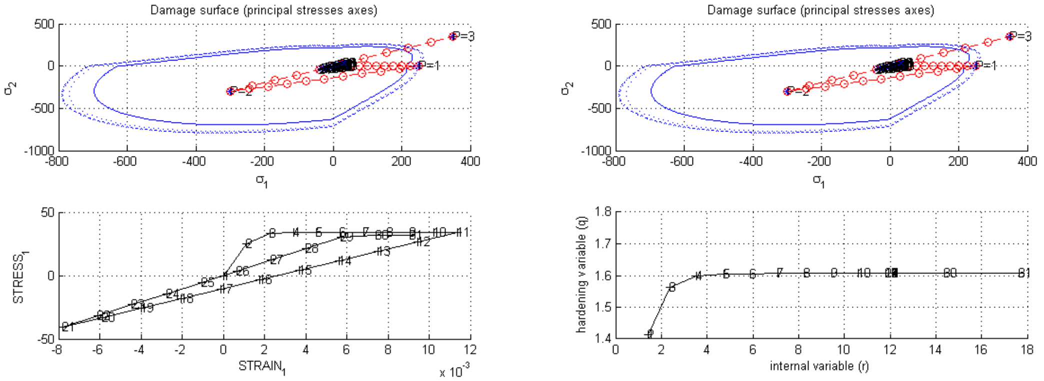


Figure 4: Inviscid Model - Non Symmetric tension compression damage model (Case 1)

Figure 4 shows as for the Case 1 (Tension only damage model) that the material behaves elastically as the tensile load is less then the yield stress. As the the limit is exceed the elastic domain increases and the material experiment deformation. During the next stage (compressive loading) the material experiments compression without no changes of the domain and even for the last stage, during tensile loading, no deformation has been registered.

Case 2

$\Delta\sigma_1 = \alpha$; $\Delta\sigma_2 = 0$ *Uniaxial tensile loading*
$\Delta\sigma_1 = -\beta$; $\Delta\sigma_2 = -\beta$ *Biaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = \gamma$ *Biaxial compressive unloading/tensile loading*



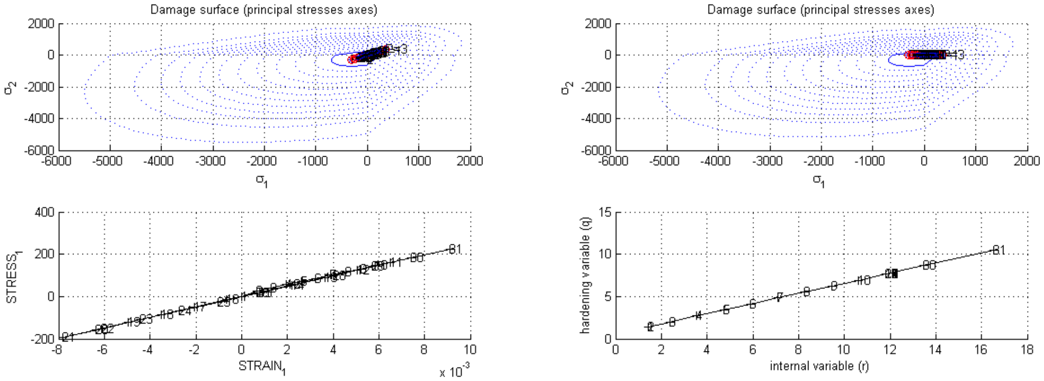
((a)) Stress vs Strain ((b)) Exponential Hardening Behaviour

Figure 5: Inviscid Model - Non Symmetric tension compression damage model (Case 2)

Figure 5(a) shows a deformation of the domain during the first stage (Uniaxial tensile loading and also during the last stage (Biaxial tensile loading).
Figure 5(b) shows an Exponential Hardening Behaviour which relates the Damage variable (q) with the Internal Variable (r). We can see that the function is bounded as the Internal Variable goes to infinity.

Case 3

$\Delta\sigma_1 = \alpha$; $\Delta\sigma_2 = \alpha$ *Biiaxial tensile loading*
$\Delta\sigma_1 = -\beta$; $\Delta\sigma_2 = -\beta$ *Biiaxial tensile unloading/compressive loading*
$\Delta\sigma_1 = \gamma$; $\Delta\sigma_2 = \gamma$ *Biaxial compressive unloading/tensile loading*



((a)) Stress vs Strain ((b)) Linear Hardening Behaviour

Figure 6: Inviscid Model - Non Symmetric tension compression damage model (Case 3)

Figure 6(a) and Figure 6(b) show a Linear Hardening Behaviour which relates the Damage variable (q) with the Internal Variable (r), which means that the material experiment hardening and the domain increase.

Where for the all cases α=250, β=300 and γ=350, with linear Hardening Modulus of 0.5.

3.2 Part 2 (rate dependent models)

The second part of the assignment include the integration algorithm (plane strain case) for the continuum isotropic visco-damage "symmetric tension-compression" model. Once the previous implementation is done the following cases (for a specific given Poisson ratio and linear hardening/softening parameter) are considered:

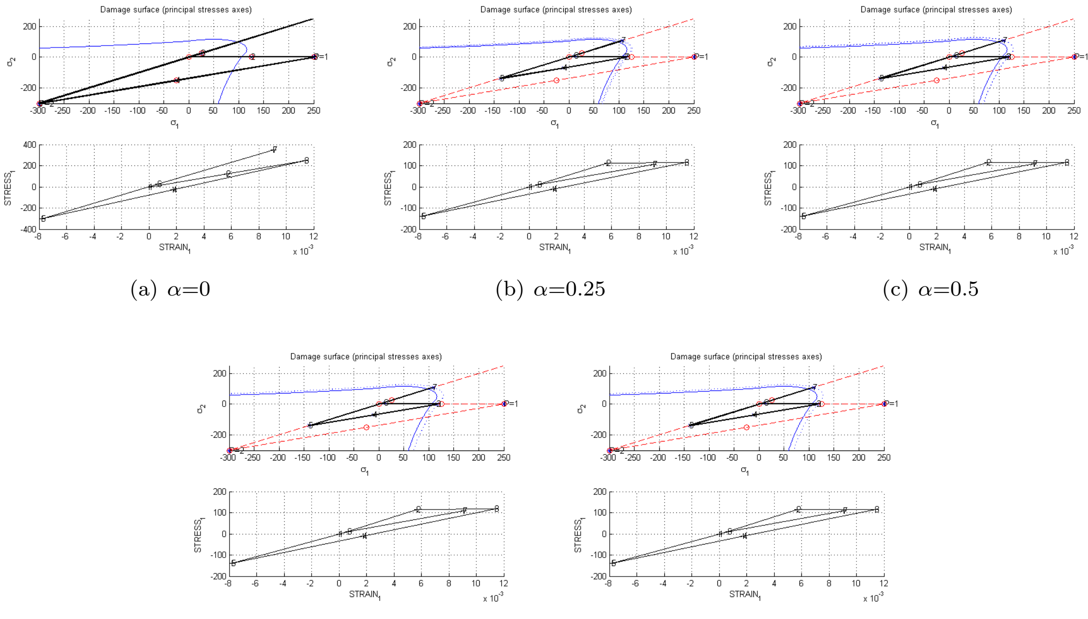
- Different viscosity parameters η: [0 ,0.8]
- Different strain rate $\dot{\varepsilon}$
- Different α values: α = [0, 1/4, 1/2, 3/4, 1] (for the time-integration method)

And the next results are obtained:

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

Results

For the viscosity parameters η=0 and for different values of α we get:



(a) α=0

(b) α=0.25

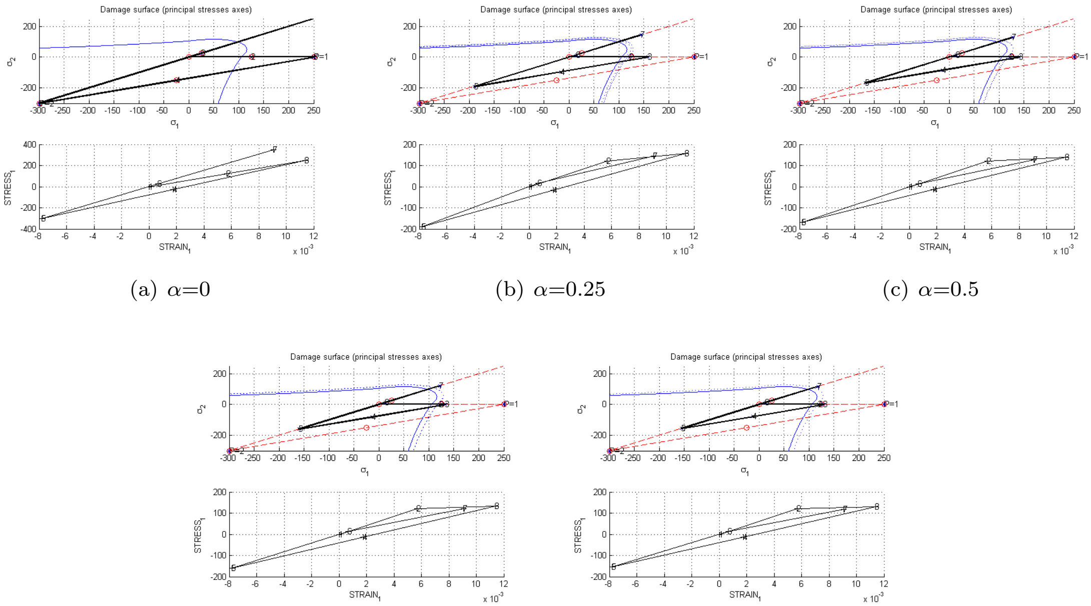
(c) α=0.5

(d) α=0.75

(e) α=1

Figure 7: Results for different values α coefficient and η=0

And for the viscosity parameters η=0.8 and for different values of α we get:



(a) α=0

(b) α=0.25

(c) α=0.5

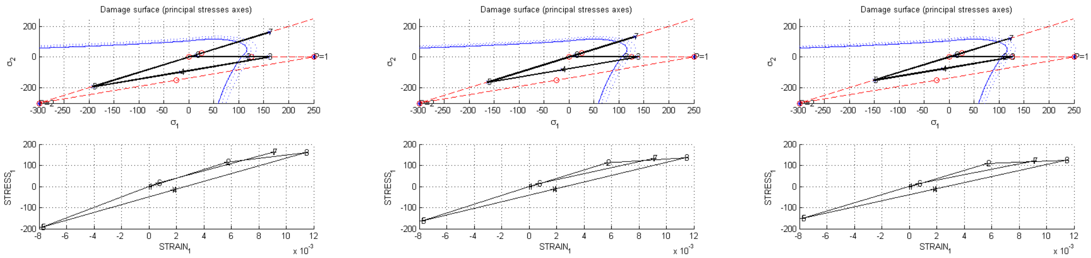
(d) α=0.75

(e) α=1

Figure 8: Results for different values α coefficient and η=0.8

Figure 7 and Figure 8, compared with the inviscid case, shown how η and α determine the behaviour of the numerical simulation. As we know from the theory α is important in order to prevent instabilities. For α=[1/2,1] the solution should not produce instability. Small values η also produce instability. For η=0 and α=1 the inviscid case is recovered with implicit integration.

In order to obtain results for different strain rate values we just need to change the total time in the main code. The simulations has be done for the next total time values (s): [20,50,100].



(a) Total time = 10 s

(b) Total time = 20 s

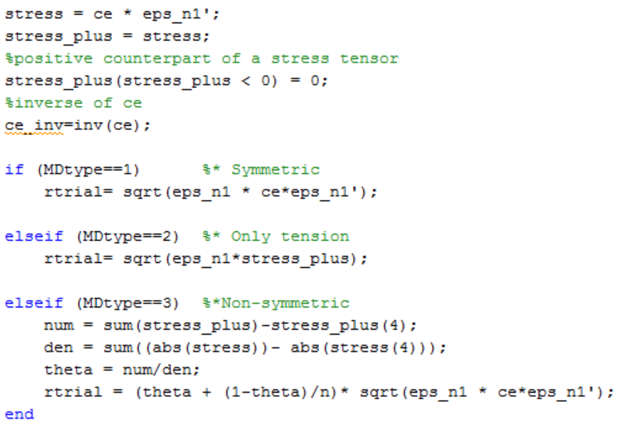
(c) Total time = 50 s

Figure 9: Results for α=0.5 coefficient and η=0.4

Figure 9, compared with the inviscid case, shows that as the total time increases, the viscous solution tends to the inviscid one, which means the small velocity of the applied load.

3.3 Annexes

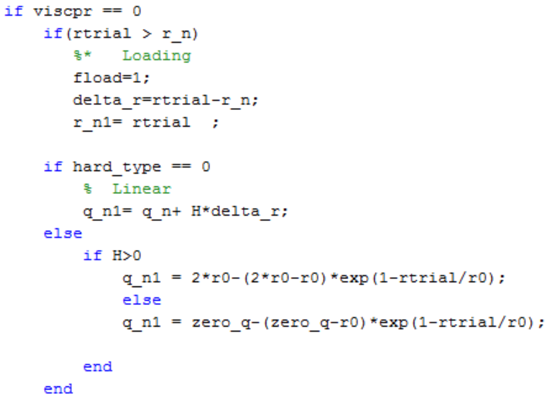
Implementation of the continuum isotropic damage “non-symmetric tension-compression damage” model and he “tension-only” damage model.



```
stress = ce * eps_n1';
stress_plus = stress;
%positive counterpart of a stress tensor
stress_plus(stress_plus < 0) = 0;
%inverse of ce
ce_inv=inv(ce);

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

elseif (MDtype==2)  %* Only tension
    rtrial= sqrt(eps_n1*stress_plus);

elseif (MDtype==3)  %*Non-symmetric
    num = sum(stress_plus)-stress_plus(4);
    den = sum((abs(stress))- abs(stress(4)));
    theta = num/den;
    rtrial = (theta + (1-theta)/n)* sqrt(eps_n1 * ce*eps_n1');
end
```

Figure 10

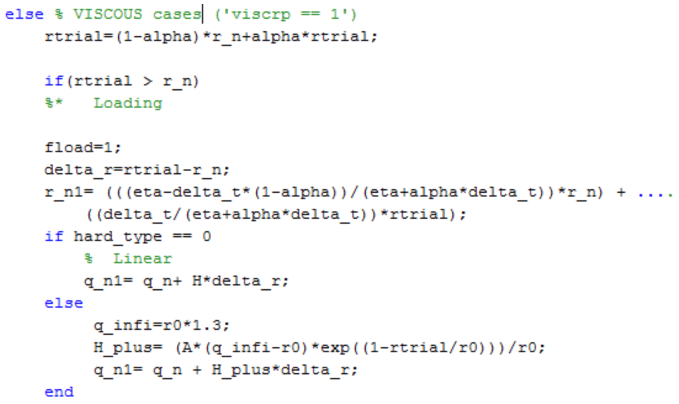
Implementation of the linear and Exponential Hardening/Softening Law (H<0 and H>0)



```
if viscpr == 0
    if(rtrial > r_n)
       %*   Loading
       fload=1;
       delta_r=rtrial-r_n;
       r_n1= rtrial  ;

    if hard_type == 0
        %  Linear
        q_n1= q_n+ H*delta_r;
    else
        if H>0
            q_n1 = 2*r0-(2*r0-r0)*exp(1-rtrial/r0);
            else
            q_n1 = zero_q-(zero_q-r0)*exp(1-rtrial/r0);

        end
    end
```

Figure 11

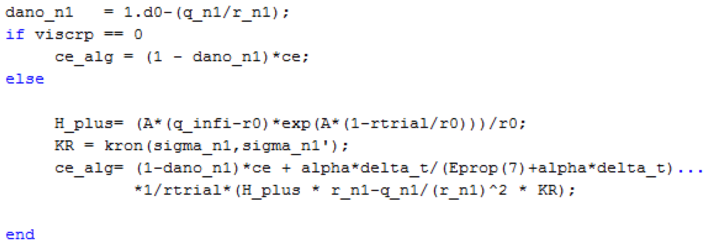
Implementation of the Viscous case



```
else % VISCOUS cases ('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
         q_infi=r0*1.3;
         H_plus= (A*(q_infi-r0)*exp((1-rtrial/r0)))/r0;
         q_n1= q_n + H_plus*delta_r;
    end
```

Figure 12

Implementation of the Consistent (algorithmic) tangent operator:



```
dano_n1   = 1.d0-(q_n1/r_n1);
if viscrp == 0
     ce_alg = (1 - dano_n1)*ce;
else

     H_plus= (A*(q_infi-r0)*exp(A*(1-rtrial/r0)))/r0;
     KR = kron(sigma_n1,sigma_n1');
     ce_alg= (1-dano_n1)*ce + alpha*delta_t/(Eprop(7)+alpha*delta_t)...
             *1/rtrial*(H_plus * r_n1-q_n1/(r_n1)^2 * KR);

end
```

Figure 13