UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Erasmus Mundus

UNIVERSITAT POLITÈCNICA DE CATALUNYA, BARCELONA

MSC. COMPUTATIONAL MECHANICS ERASMUS MUNDUS

ASSIGNMENT 1: CONTINUUM DAMAGE MODELS

Computational Solid Mechanics

Author:
Nikhil Dave

Date: March 23, 2018

Contents

1 Objective **1**

2 Part I: Rate independent models **1**

2.1 Case 1: Complete uniaxial path 1

2.2 Case 2: Uniaxial / biaxial path 4

2.3 Case 3: Complete biaxial path 6

3 Part II: Rate dependent models **9**

3.1 Effect of viscosity η on stress-strain curve 9

3.2 Effect of strain rate on stress-strain curve 9

3.3 Effect of time-integration parameter α on stress-strain curve 10

3.4 Effect of time-integration parameter α on tangent and algorithmic constitutive operators 12

4 Conclusion **14**

5 Appendix **15**

1 Objective

The objective of this assignment is to understand, present and implement the theory of continuum damage for elastoplastic materials. In the first part, for strain-rate independent models, a supplied MATLAB code for symmetric model with linear hardening/softening law is to be modified to incorporate the tension-only and non-symmetric tension-compression models along with exponential hardening/softening law. The second part deals with the implementation of the rate-dependent models and viscosity parameter and study their behaviour and correctness.

2 Part I: Rate independent models

In this section we present the result plots obtained - the path in the stress space and the stress-strain curve for three proposed cases. These are analysed for the two specified models with the following basic material properties,

$$E = 200\ GPa, \quad \sigma_y = 200\ MPa, \quad \nu = 0.3, \quad H = \pm 0.2, \quad n = 2$$

where E is the Young modulus, σ_y is the yield stress, ν is the Poisson ratio, H is the hardening/softening modulus and n is the ratio of compression/tension strength.

2.1 Case 1: Complete uniaxial path

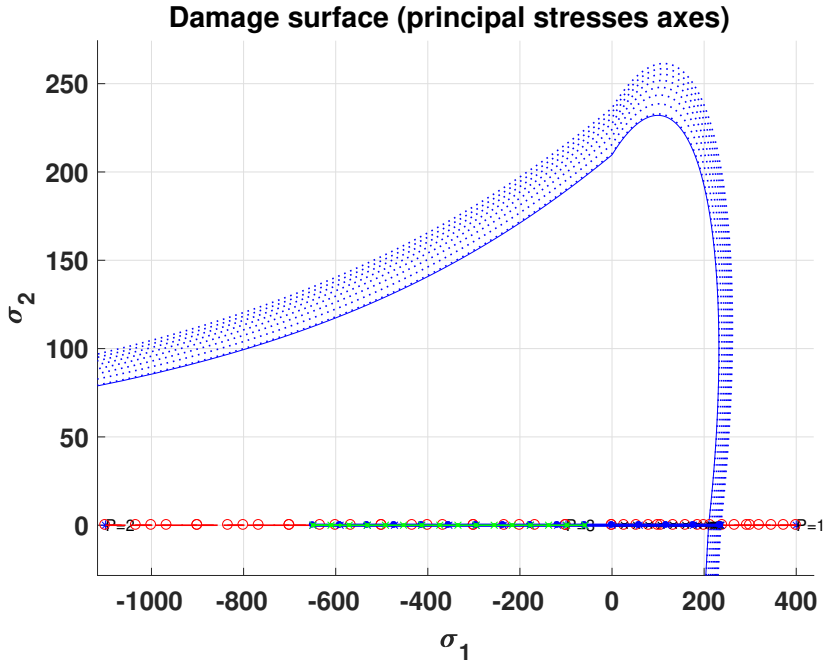


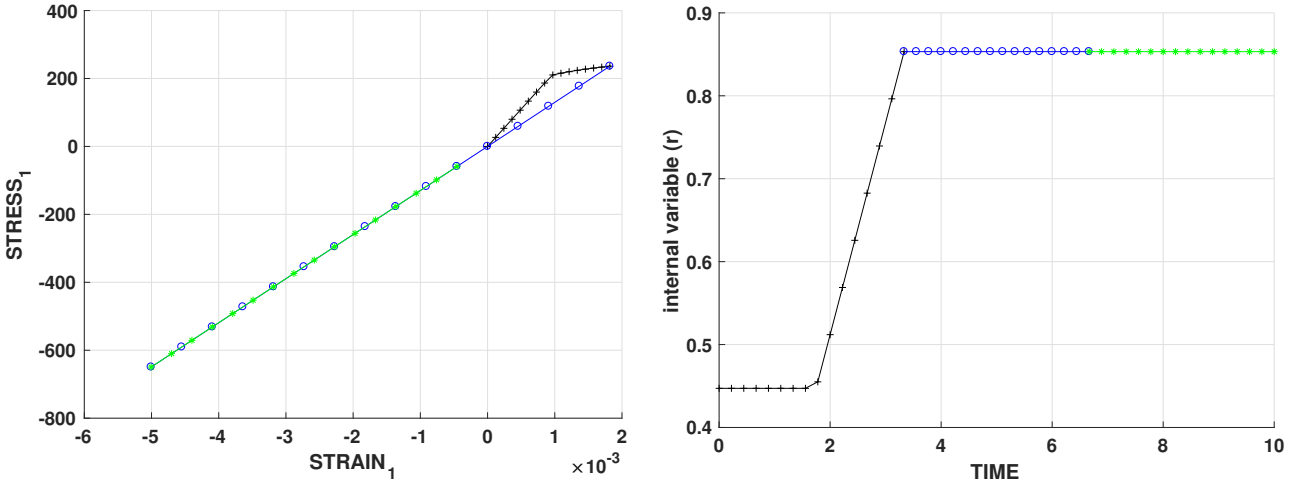
Figure 1: Path at the stress space for case 1: tension-only damage model

The first case of the analysis consisting of uniaxial loading/unloading path is given as,

$$\Delta\bar{\sigma}_1^{(1)} = \alpha, \quad \Delta\bar{\sigma}_2^{(1)} = 0; \quad \Delta\bar{\sigma}_1^{(2)} = -\beta, \quad \Delta\bar{\sigma}_2^{(2)} = 0; \quad \Delta\bar{\sigma}_1^{(3)} = \gamma, \quad \Delta\bar{\sigma}_2^{(3)} = 0 \tag{1}$$

where α, β and γ are arbitrary parameters chosen as $\alpha = 400\ MPa$, $\beta = -1500\ MPa$ and $\gamma = 1000\ MPa$.

Figure 1 shows a graphical representation of the obtained path in the stress space for the tension-only damage model. In this case, stress paths were chosen specifically to check the correctness of our implementation with hardening type as exponential. Firstly the load path goes into the inelastic zone which is then compressed and loaded again. The evolution of the stress space is represented by the dotted blue lines which expands, to keep the points on the boundary, due to hardening. As the material overcomes the elastic regime in the first loading, it undergoes a hardening process and the internal variable r increases. In the following step, the material is compressed with a lower slope but without any restriction on its elastic domain. This is due to the fact that the tension-only model posses a characteristic to allow the material to be compressed up to infinity.



(a) Stress-strain curve

(b) Internal variable evolution

Figure 2: Result plots for case 1: tension-only damage model

Figure 2 shows the stress-strain curve and the evolution of internal variable with time obtained for the loading path explained above where the black line with cross represents the first load stage, the blue line with circles shows the second load stage and the green line with asterisk corresponds to the third load stage. In the final step, the evolution takes place with the same slope as the previous step, as expected. Also, the internal variable remains constant for this step since the compression loading does not overpasses the elastic regime.

We know that the non-symmetrical tension-compression damage model represents materials, like concrete, whose tension and compression domains are different from each other. Figure 3 shows a plot of the obtained path in the stress space. It is interesting to note that the elastic regime is surpassed twice because the material is allowed to fail under compressive loading in addition to the first tensile loading.

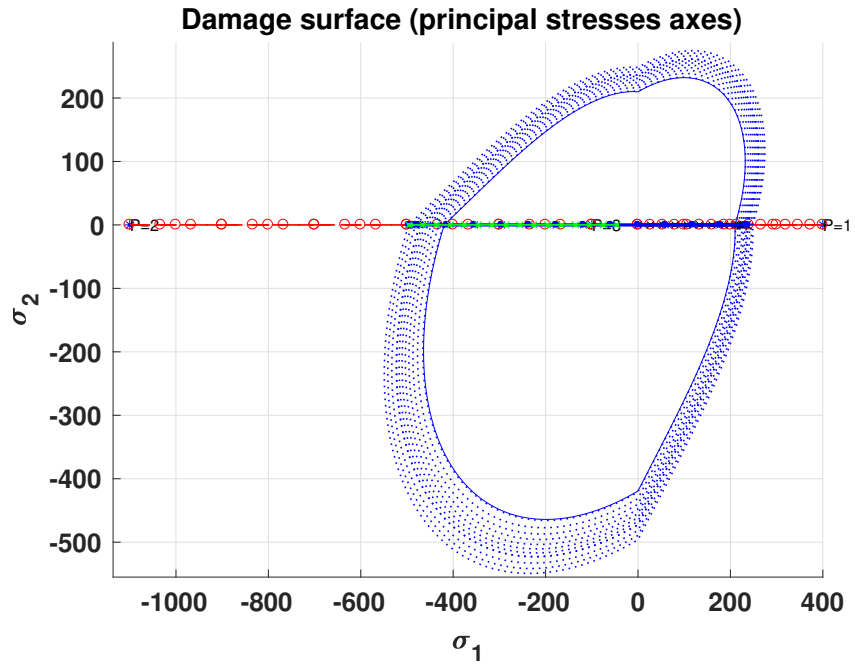
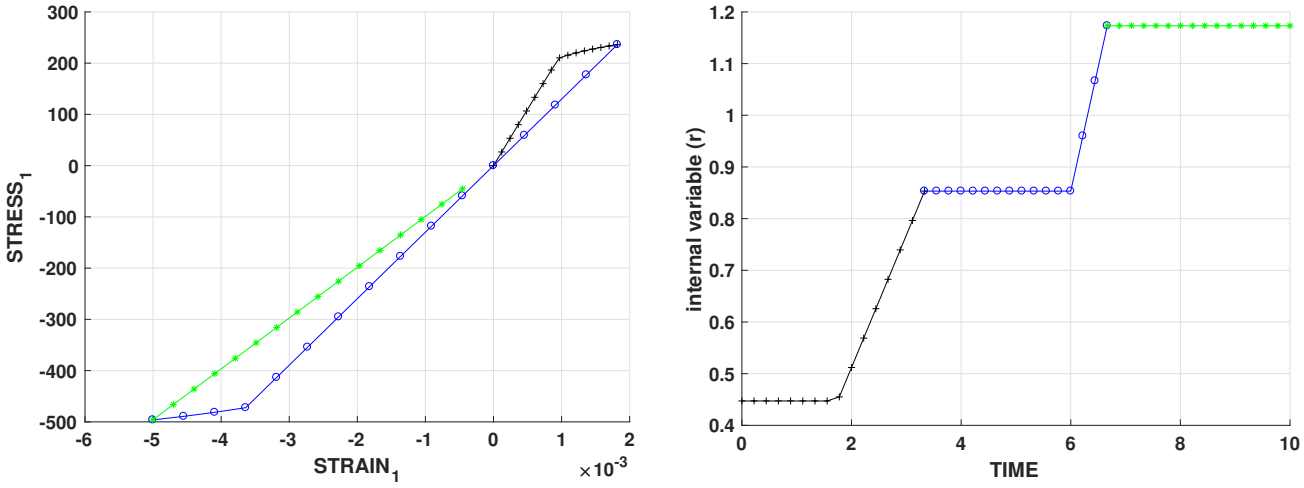


Figure 3: Path at the stress space for case 1: non-symmetrical tension-compression damage model



(a) Stress-strain curve

(b) Internal variable evolution

Figure 4: Result plots for case 1: non-symmetrical tension-compression damage model

Figure 4 shows the stress-strain curve and the evolution of internal variable with time obtained for the loading path where we can observe that the internal variable increases with the first loading (as in tension-only model) and later when the inelastic regime is achieved again under compression. This effect can also be seen in the lower part of the stress-strain curve.

2.2 Case 2: Uniaxial / biaxial path

The second case of the analysis consisting of uniaxial / biaxial path is given as,

$$\Delta\bar{\sigma}_1^{(1)} = \alpha, \quad \Delta\bar{\sigma}_2^{(1)} = 0; \quad \Delta\bar{\sigma}_1^{(2)} = -\beta, \quad \Delta\bar{\sigma}_2^{(2)} = -\beta; \quad \Delta\bar{\sigma}_1^{(3)} = \gamma, \quad \Delta\bar{\sigma}_2^{(3)} = \gamma \tag{2}$$

where $\alpha = 400\ MPa$, $\beta = -1500\ MPa$ and $\gamma = 1000\ MPa$.

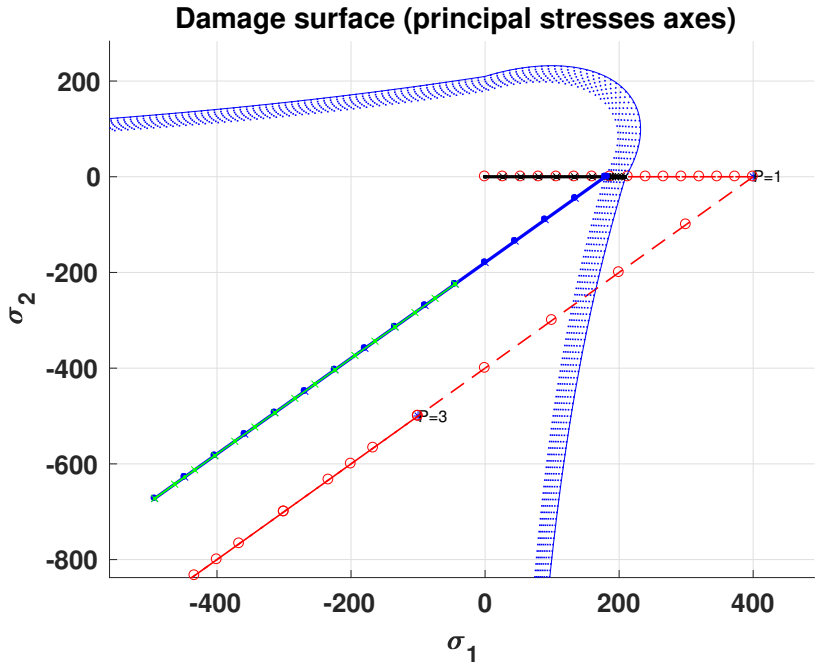
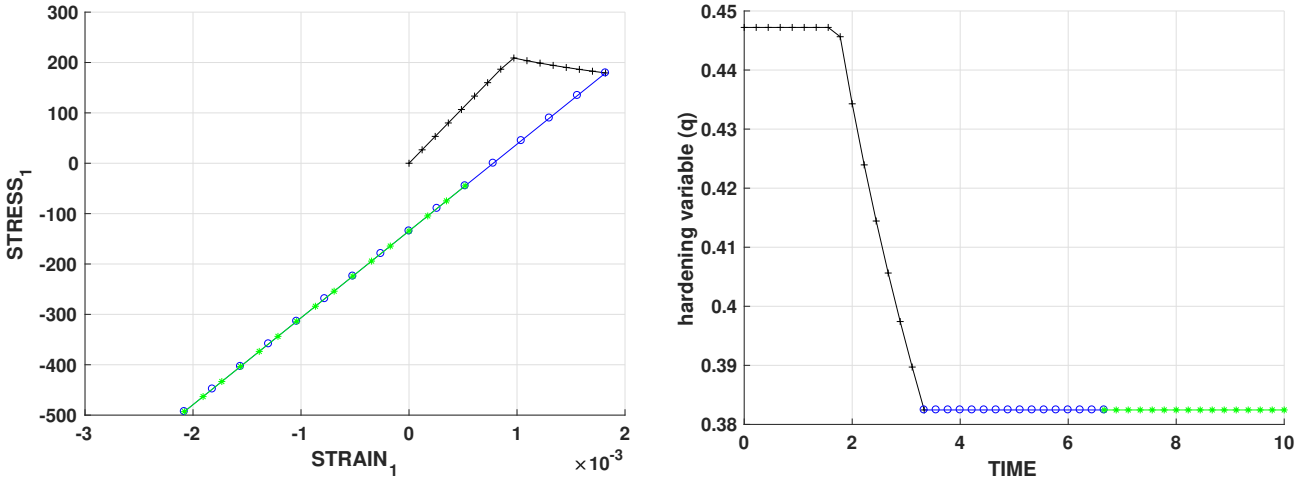


Figure 5: Path at the stress space for case 2: tension-only damage model

Figure 5 shows a graphical representation of the obtained path in the stress space for the tension-only damage model. In this case, since we use the softening parameter, $H = -0.2$, it can be seen that as the material overcomes the elastic regime, the stress space represented by the dotted blue lines contracts in order to fulfil the consistency conditions.

Figure 6 shows the stress-strain curve and the evolution of hardening variable with time obtained for the loading path implemented with softening type as exponential. The hardening variable q varies with the evolution of the domain in the stress space i.e. evolution of the internal variable and is helpful in understanding the contraction behaviour in case of softening.

As the material overcomes the elastic regime in the first loading, it undergoes a softening process and the hardening variable decreases. In the following step, the material is compressed without any restriction on its elastic domain due to the characteristic of the tension-only model. Again, as expected in the final step, the evolution takes place with the same slope as the previous step. Also, the hardening variable remains constant for this step since the compression loading does not overpasses the elastic regime.



(a) Stress-strain curve (b) Hardening variable evolution

Figure 6: Result plots for case 2: tension-only damage model

Figure 7 shows a plot of the obtained path in the stress space for the non-symmetrical tension-compression damage model. Again in this case, the elastic regime is surpassed twice because the material is allowed to fail under compressive loading as well as during the first tensile loading due to which we could notice the unloading and second loading stages are not coincident as in the tensile-only model.

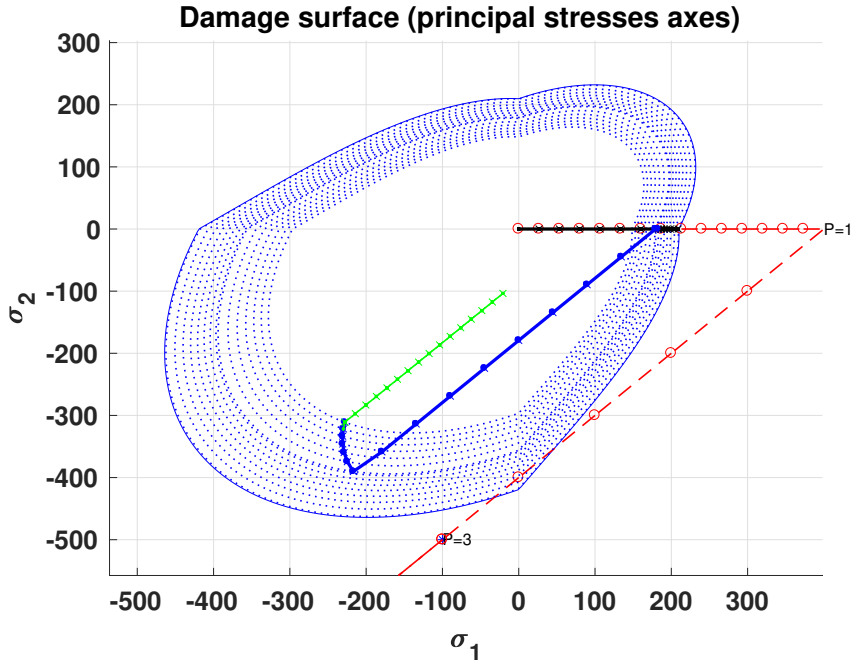
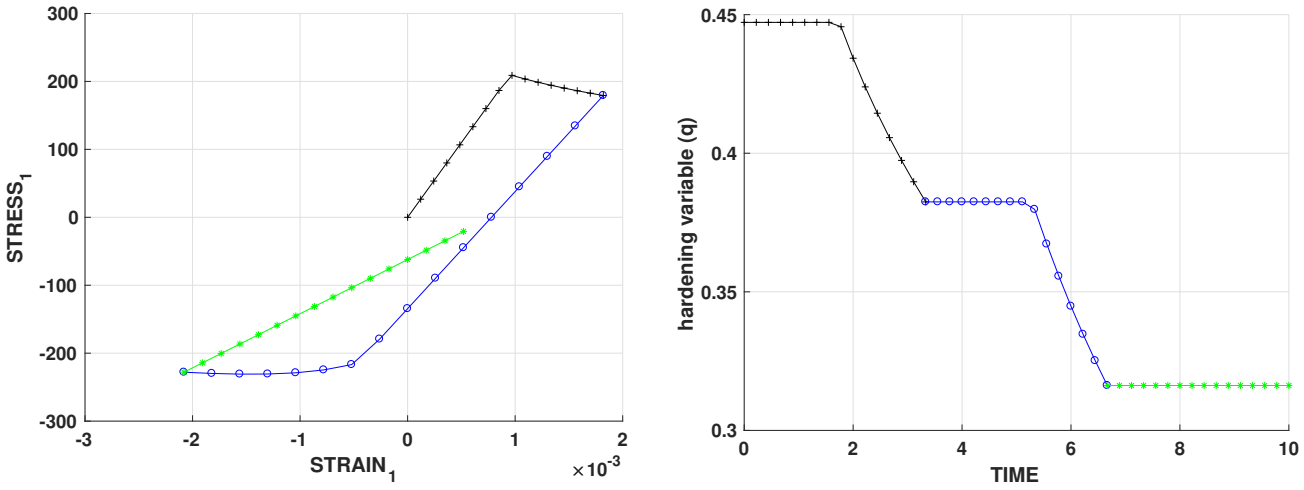


Figure 7: Path at the stress space for case 2: non-symmetrical tension-compression damage model



(a) Stress-strain curve (b) Hardening variable evolution

Figure 8: Result plots for case 2: non-symmetrical tension-compression damage model

Figure 8 shows the stress-strain curve and the evolution of hardening variable with time obtained for the loading path where we can observe that the hardening variable decreases twice evolving (contraction) the size of the elastic surface, first with the tensile loading and then during compression.

By solving different scenarios and loading paths, we could assess the behaviour of our implementation, which relates with the theoretical statements.

2.3 Case 3: Complete biaxial path

The third case of the analysis consisting of complete biaxial path is given as,

$$\Delta\overline{\sigma}_1^{(1)} = \alpha, \quad \Delta\overline{\sigma}_2^{(1)} = \alpha; \quad \Delta\overline{\sigma}_1^{(2)} = -\beta, \quad \Delta\overline{\sigma}_2^{(2)} = -\beta; \quad \Delta\overline{\sigma}_1^{(3)} = \gamma, \quad \Delta\overline{\sigma}_2^{(3)} = \gamma \tag{3}$$

where $\alpha = 400\ MPa$, $\beta = -1500\ MPa$ and $\gamma = 1000\ MPa$.

In this final case, we check the correctness of our implementation with hardening type as linear. Figure 9 shows a graphical representation of the obtained path in the stress space for the tension-only damage model.

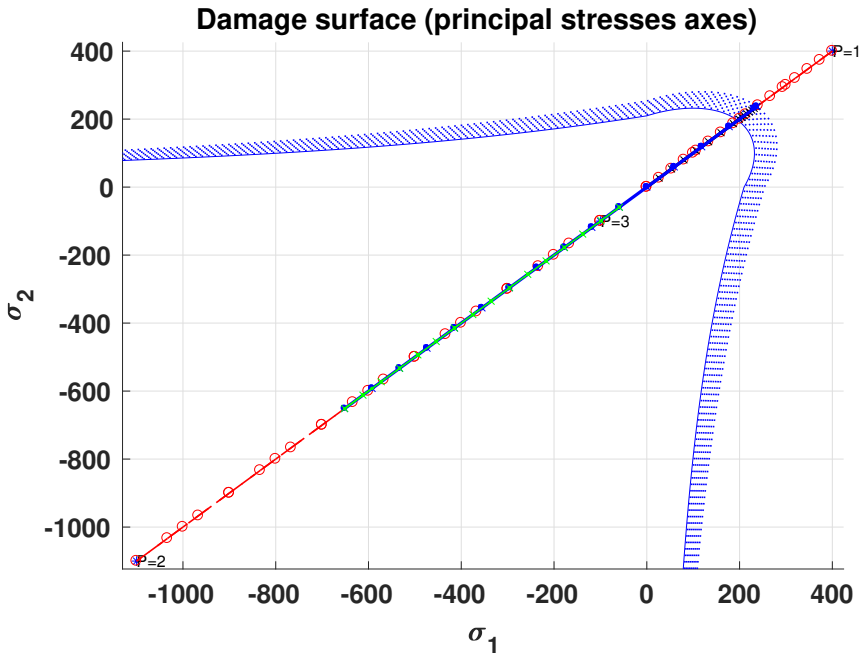
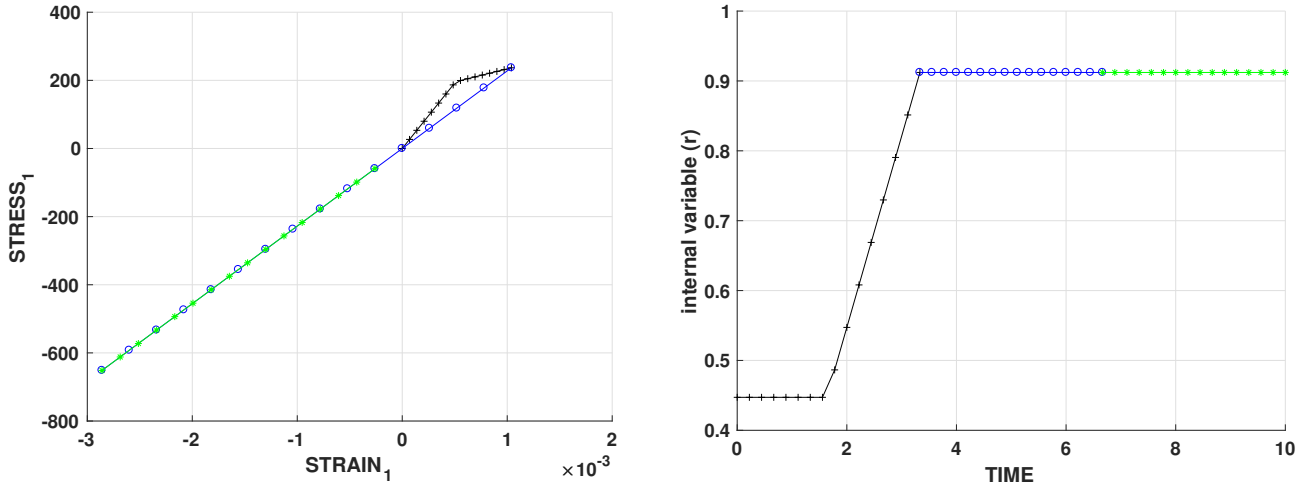


Figure 9: Path at the stress space for case 3: tension-only damage model

Figure 10 shows the stress-strain curve and the evolution of hardening variable with time obtained for the loading path. It could be easily noticed that the results of this case are very similar to the complete uniaxial case, since it is practically the same problem but two-dimensional.



(a) Stress-strain curve

(b) Internal variable evolution

Figure 10: Result plots for case 3: tension-only damage model

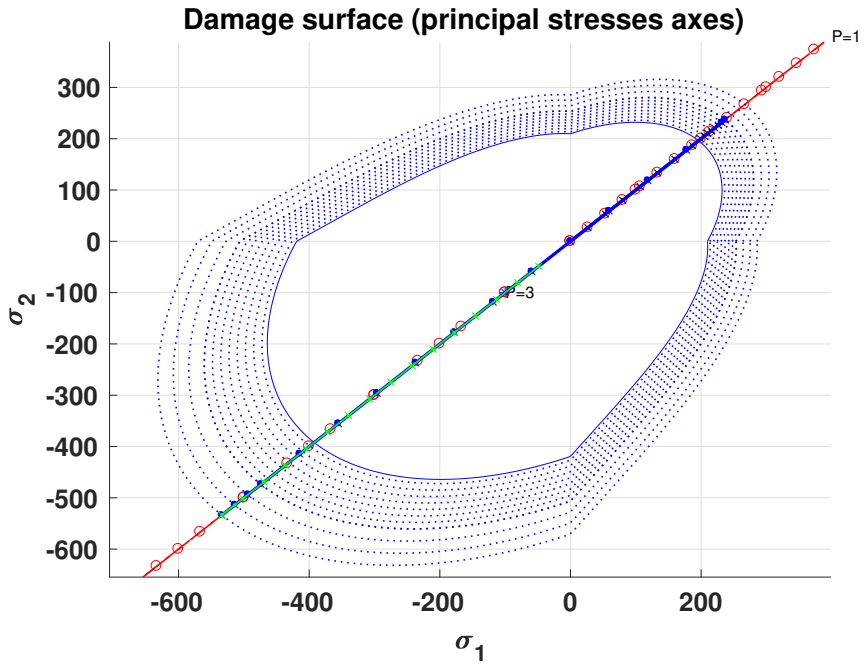
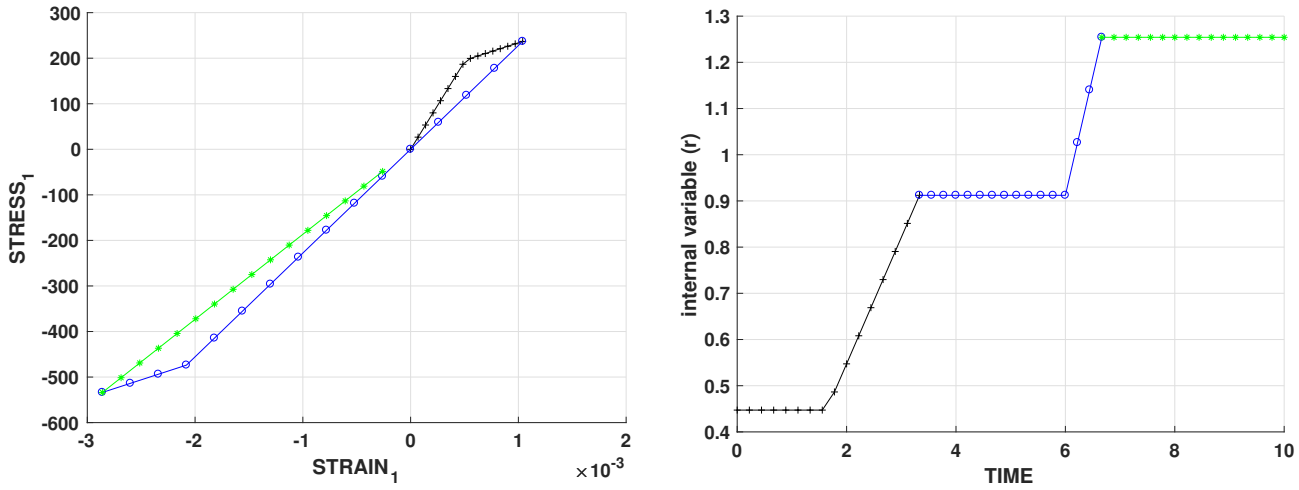


Figure 11: Path at the stress space for case 3: non-symmetrical tension-compression damage model

The results for the non-symmetrical tension-compression damage model given in Figures 11 and 12 also shows that the this case does not differ with the complete uniaxial case. The 2D stress-strain plots in both dimensions are also exactly the same which makes it very similar to case 1. Therefore, the effects of modelling the hardening type as linear or exponential is analysed with all three cases for this task. It could be seen that the variation of the hardening variable in Figure 12 is linear in nature compared to the other two cases, where the variation is curved and effectively more accurate than the linear implementation.



(a) Stress-strain curve

(b) Internal variable evolution

Figure 12: Result plots for case 1: non-symmetrical tension-compression damage model

3 Part II: Rate dependent models

In the second part of this assignment, we assess the correctness of the implementation of the integration algorithm for the continuum isotropic visco-damage symmetric tension-compression model considering the effects of a few properties like viscosity η, strain rate $\dot{\epsilon}$, alpha value for α time-integration method. We consider the same fixed values of the parameters Poisson ratio, linear hardening/softening, Young's modulus and yield stress as in the first part of the assignment.

3.1 Effect of viscosity η on stress-strain curve

We know that the material achieves higher stresses in the inelastic region due to the effect of higher viscosity. To check if our implementation produces the same effect, we select a simple case of uniaxial load path overpassing the elastic region. The results obtained for the stress-strain curve with varying viscosities are shown in Figure 13.

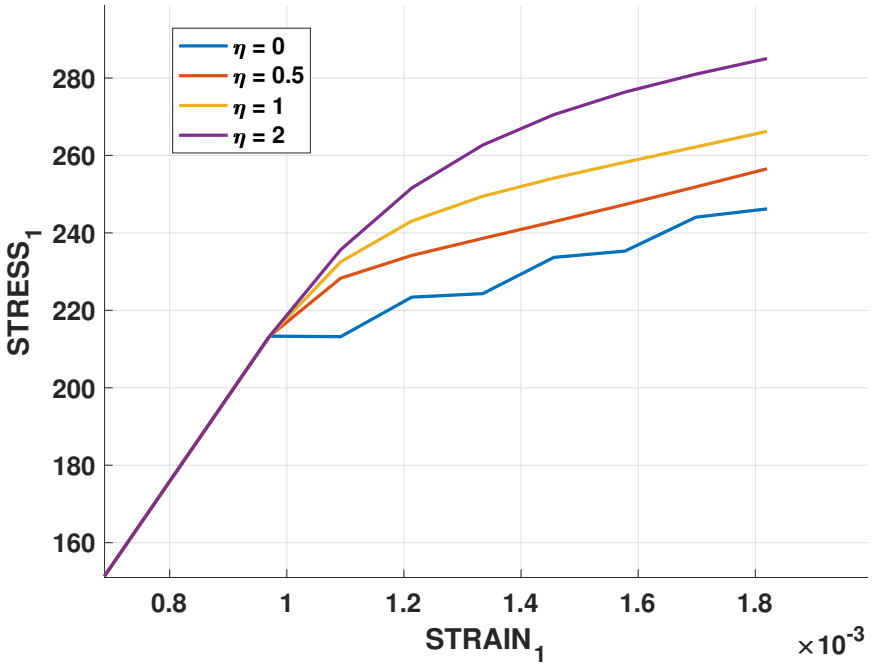


Figure 13: Evolution of stress-strain curve for different values of viscosity η

From the above plot, we see identical behaviour of all the models within the elastic domain since we have zero viscosity associated with the strain in the elastic region. Though, within the inelastic region, higher viscosities account for higher stresses for the same strain value which is consistent with our theoretical knowledge. Hence, our implementation behaves as expected.

3.2 Effect of strain rate on stress-strain curve

In order to show the evolution of the stress-strain curve as a function of strain rate $\dot{\epsilon}$, we fix all parameters and loading path except the total time. This allows us to control the strain rate as a direct relation i.e. for higher strain rate, the loading path should be applied in lesser time.

Figure 14 shows that within the elastic region, there is no variation. But as expected in the inelastic region, a direct dependency of stress on strain rate is observed. This is the main property of the rate dependent models, wherein the stresses not only depend on strains but also on their rate. This important characteristic of the visco-damage models means that even for a constant strain tensor ϵ, a change in stress tensor σ could be observed.

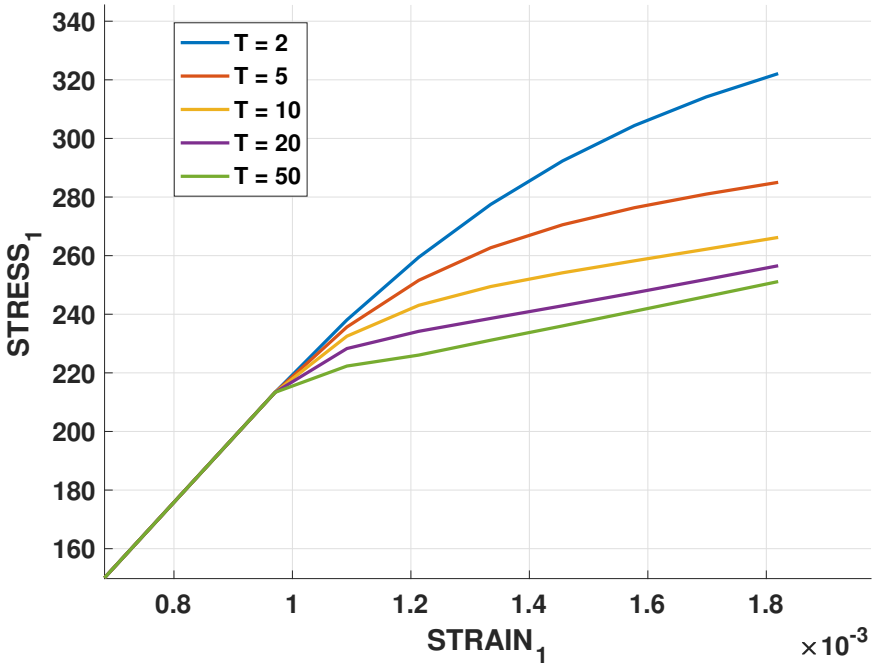


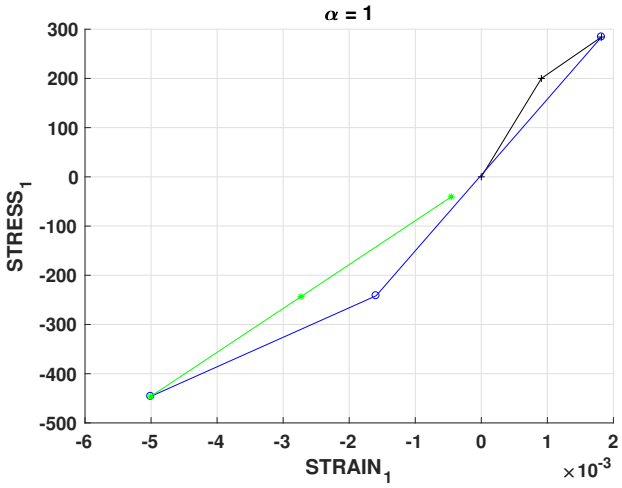
Figure 14: Evolution of stress-strain curve for different values of strain rate $\dot{\epsilon}$

For checking the correctness of the implementation, we compared the stress-strain curves for zero viscosity and varying total time period to give us the same results as the rate independent model. It can also be observed that the model also matches the rate independent case if the total time period is high enough to consider strain rate as zero. This proved that the model behaves as expected.

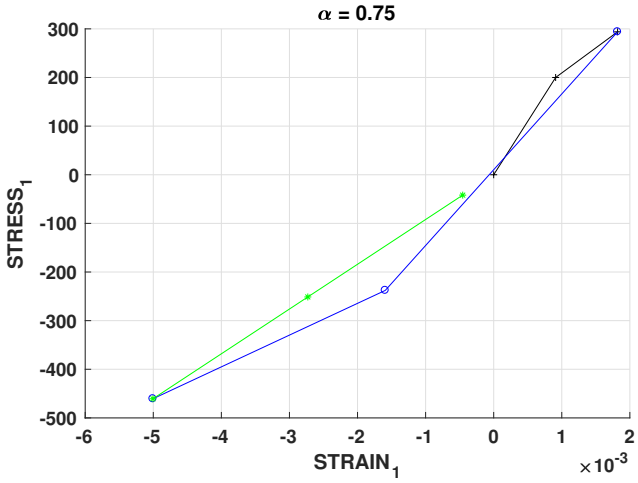
3.3 Effect of time-integration parameter α on stress-strain curve

Now, we analyse the behaviour of the stress-strain curve with various time-integration schemes. We considered different time-integration schemes like explicit Forward-Euler scheme ($\alpha = 0$), explicit scheme with $\alpha = 0.25$, implicit Crank-Nicholson scheme ($\alpha = 0.5$), Galerkin scheme ($\alpha = 0.75$), implicit Barckward-Euler scheme ($\alpha = 1$).

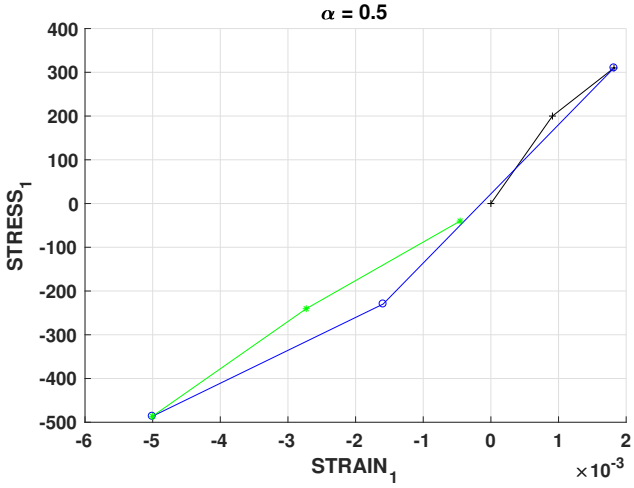
Figure 15 shows the stress-strain curves for different α values. It can be noticed that we get a meaningful solution using the integration schemes with $\alpha > 0.5$, where the loading - unloading behaviour could be easily understood. But with the explicit integration schemes, the solution obtained is does not seems meaningful. It is known from the partial differential equation solution methods that explicit schemes are conditionally stable, produce spurious solutions and should be avoided even if they provide low computational cost. We also know that all these schemes are first order accurate except Crank-Nicholson (second order accurate) which should be preferred and therefore is used for all other results provided in this report.



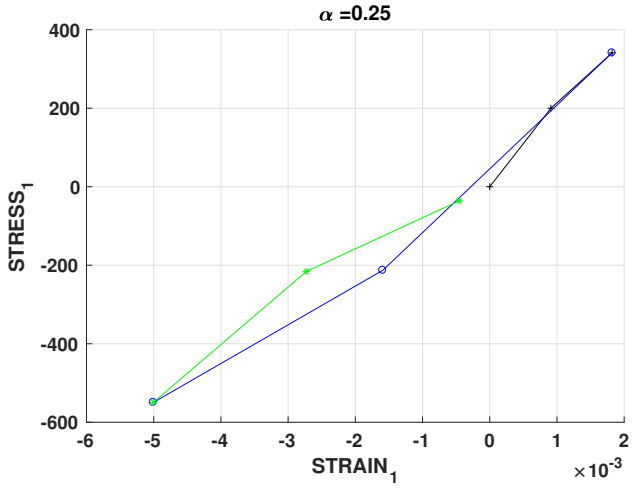
(a) Implicit backward Euler scheme $\alpha = 1$



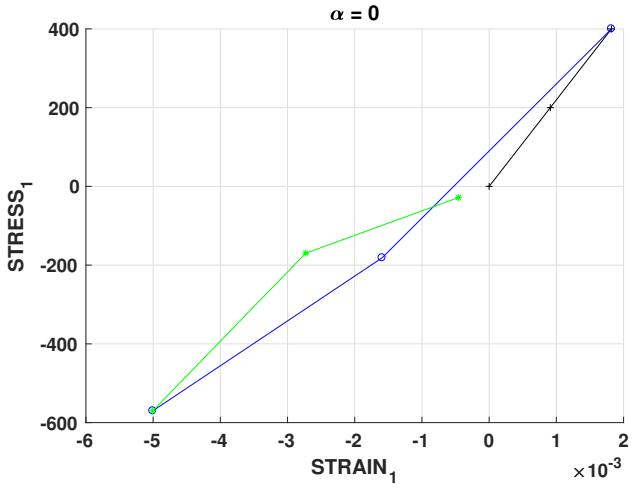
(b) Galerkin scheme $\alpha = 0.75$



(c) Crank-Nicholson scheme $\alpha = 0.5$



(d) Explicit scheme with $\alpha = 0.25$



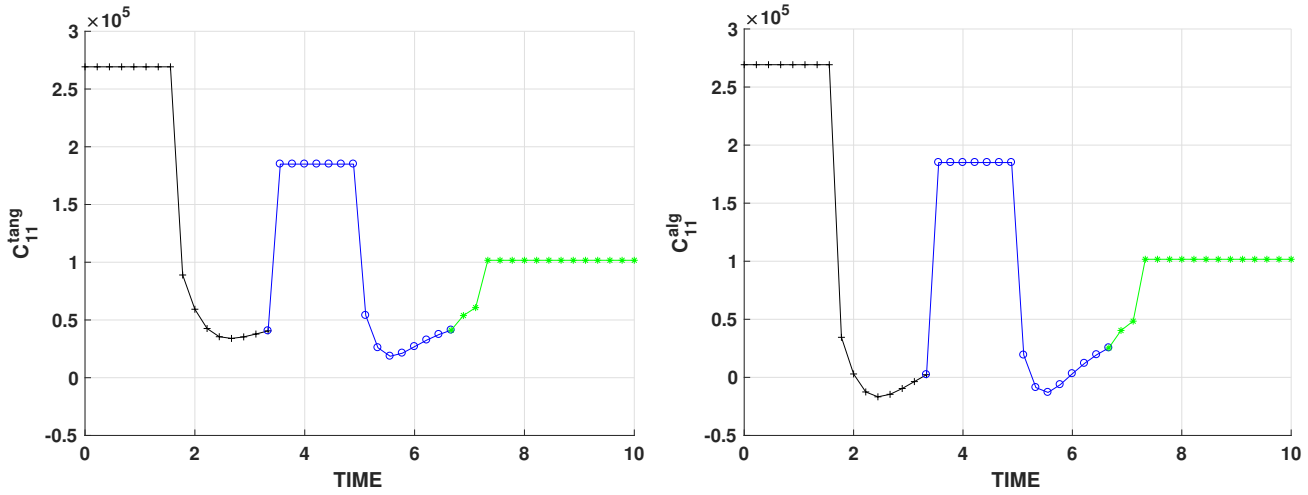
(e) Explicit forward Euler scheme $\alpha = 0$

Figure 15: Evolution of stress-strain curve for different values of α integration parameter

3.4 Effect of time-integration parameter α on tangent and algorithmic constitutive operators

To obtain the influence of α on the evolution of the tangent and algorithmic constitutive operators, we fix all other parameters and evaluate their behaviour. In this section, we use three different integration schemes - Implicit backward Euler scheme ($\alpha = 1$), Crank-Nicholson scheme ($\alpha = 0.5$) and Explicit forward Euler scheme ($\alpha = 0$).

Figure 16 shows the evolution of both tangent and algorithmic constitutive operators using the implicit backward Euler scheme. We know that the tangent constitutive operator relates the stress and strain derivatives and the algorithmic operator relates the derivatives of the current values of stresses and strains. Therefore, the value of these operators are equal whenever they are in the elastic region as seen in the Figure 16 but as soon as the damage appears, the values change.

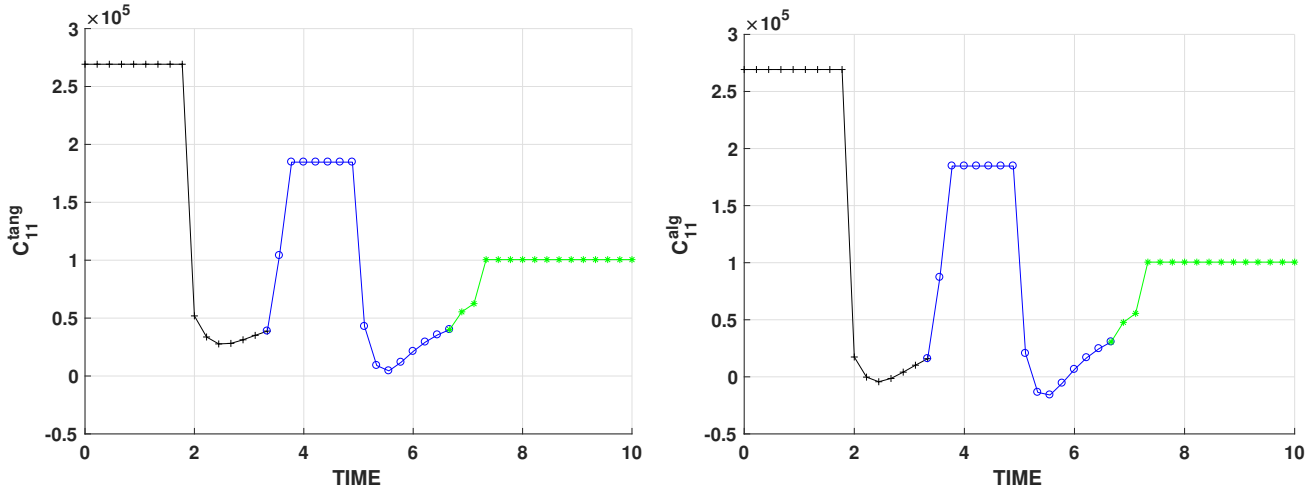


(a) Tangent constitutive operator (b) Algorithmic constitutive operator

Figure 16: Evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators for $\alpha = 1$

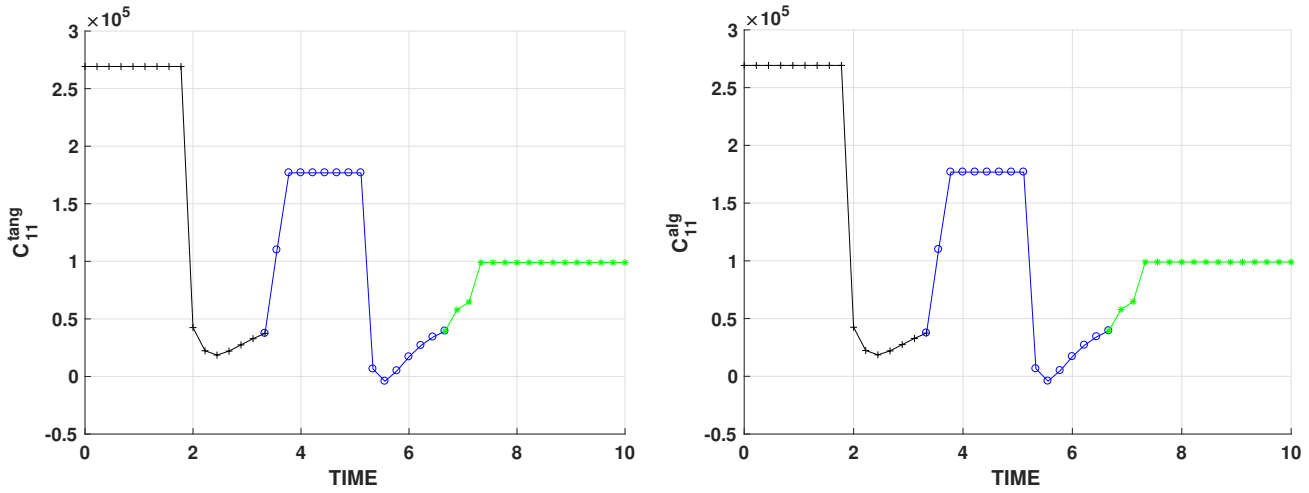
Similarly, the evolution of these operators using the Crank-Nicholson scheme is shown in Figure 17. The results obtained follow the same trend as in the previous case but with lower value of α.

In the final plot, shown in Figure 18, we check the correctness of our model implementation using Explicit forward Euler scheme. Since $\alpha = 0$, both components are identical and the model behaviour is as expected. It is interesting to note that as α increases these components starts deviating from the same value to reproduce the behaviour of the material but with either α or Δt tending to zero, the the differences vanish and both operators tend to match.



(a) Tangent constitutive operator (b) Algorithmic constitutive operator

Figure 17: Evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators for $\alpha = 0.5$



(a) Tangent constitutive operator (b) Algorithmic constitutive operator

Figure 18: Evolution along time of the C_{11} component of the tangent and algorithmic constitutive operators for $\alpha = 0$

4 Conclusion

In this assignment work for continuum damage for elastoplastic materials, the implementation of the several requirements was performed into a given MATLAB code after understanding the theory of rate independent and dependent models. While comparing all obtained results with the theoretical knowledge, the capability of the implementation performed to solve various cases was validated at each step. The model currently works for all required cases specified in the first assignment of Computational Solid Mechanics. Lastly, all modified functions of the implementation can be found in the Appendix of this document.

5 Appendix

Modified functions

This section includes all the modified functions used in the implementation of the code and to generate the presented results. The functions listed below are -
$damage\_main.m$, $rmap\_dano1.m$, $Modelos\_de\_dano1.m$, $dibujar\_criterio\_dano1.m$.

A) damage_main.m

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

```
%
% MDtype     = Damage surface criterion %
%           1 : SYMMETRIC
%           2 : ONLY-TENSION
%           3 : NON-SYMMETRIC
%
% n         = Ratio compression/tension strength (dummy if MDtype is
%                                                 different from 3)
%
% TimeTotal = Interval length
%
%  OUTPUTS <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%  ------------------------------------------------------------------
%  1) sigma_v{itime}(icomp,jcomp) --> Component (icomp,jcomp) of the cauchy
%                                  stress tensor at step "itime"
%                                  REMARK: sigma_v is a type of
%                                  variable called "cell array".
%
%  2) vartoplot{itime} --> Cell array containing variables wishes to plot
%                           ---------------------------------------
%   vartoplot{itime}(1) = Hardening variable (q)
%   vartoplot{itime}(2) = Internal variable (r)%
%
%  3) LABELPLOT{ivar}           --> Cell array with the label string for
%                                   variables of "varplot"
%
%         LABELPLOT{1} => 'hardening variable (q)'
%         LABELPLOT{2} => 'internal variable'
%
%  4) TIME VECTOR - >
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SET LABEL OF "vartoplot" variables (it may be defined also outside...
%                                                        this function)
% ---------------------------------
LABELPLOT = {'hardening variable (q)','internal variable','C_alg','C_tan'};
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
    mhist   = 6    ;
end

```

```
 if viscpr == 1
     Eprop = [Eprop 0];
 else
 end

totalstep = sum(istep) ;

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

% Elastic constitutive tensor
% ---------------------------
[ce]    = tensor_elastico1 (Eprop, ntype);
% Initz.
% -----
% Strain vector
% -------------
hvar_n = cell(mhist) ;

% INITIALIZING (i = 1) !!!!
% ***********i*
i = 1 ;
r0 = sigma_u/sqrt(E);
hvar_n{3} = ce; %
hvar_n{4} = ce; %
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 = cell(1,totalstep+1) ;
vartoplot{i}(1) = hvar_n{6} ; % Hardening variable (q)
vartoplot{i}(2) = hvar_n{5} ; % Internal variable (r)
vartoplot{i}(3) = 1-hvar_n{6}/hvar_n{5} ; % Damage variable (d)
vartoplot{i}(4) = hvar_n{3}(1,1) ; % C_tan
vartoplot{i}(5) = hvar_n{4}(1,1) ; % C_alg

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"
        % ------------------------
        Eprop(end) = TIMEVECTOR(i) - TIMEVECTOR(i-1);
```

```
        eps = strain(i-1,:);
        eps_n1 = strain(i,:) ;

        %* DAMAGE MODEL
           [sigma_n1 ,hvar_n ,aux_var] =...
                    rmap_dano1(eps,eps_n1,hvar_n,Eprop,ce,MDtype,n);

        % PLOTTING DAMAGE SURFACE
        if(aux_var(1)>0)
            hplotSURF(i) = dibujar_criterio_dano1(ce, nu, hvar_n{6},...
                                                    'r:',MDtype,n );
            set(hplotSURF(i),'Color',[0 0 1],'LineWidth',1);
        end

        % GLOBAL VARIABLES
        % ***************
        % Stress
        % ------
        m_sigma=[sigma_n1(1) sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ;...
                                                  0 0  sigma_n1(4)];
        sigma_v{i} = m_sigma ;

        % VARIABLES TO PLOT (set label on cell array LABELPLOT)
        % -----------------
        vartoplot{i}(1) = hvar_n{6} ; % Hardening variable (q)
        vartoplot{i}(2) = hvar_n{5} ; % Internal variable (r)
        vartoplot{i}(3) = 1-hvar_n{6}/hvar_n{5} ; % Damage variable (d)
        vartoplot{i}(4) = hvar_n{3}(1,1);
        vartoplot{i}(5) = hvar_n{4}(1,1);
    end
end
```

B) rmap_dano1.m

```

function [sigma_n1,hvar_n1,aux_var] = rmap_dano1 (eps,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);
H       = Eprop(3);
Aexp = Eprop(3) ;
sigma_u = Eprop(4);
hard_type = Eprop(5) ;
visc = Eprop(6) ;
eta = Eprop(7) ;
alf = Eprop(8) ;

%*******************************************************************
%*      initializing                                              %*
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
 q_inf = r0 +( r0-zero_q);
%*******************************************************************
%*      Damage surface
tau_bf = Modelos_de_dano1(MDtype,ce,eps,n);
tau_af = Modelos_de_dano1(MDtype,ce,eps_n1,n);
%*******************************************************************
if visc == 0
    rtrial = tau_af;
else
    tau = (1-alf)*tau_bf+alf*tau_af;
    dt = Eprop(9);
    if tau <= r_n
    rtrial = r_n;
    else
        rtrial = r_n*(eta-dt*(1-alf))/(eta+alf*dt)+tau*(dt)/(eta+alf*dt);
    end
end

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

```

```
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;
        H_n1 = H;
    else
        % Exponential
        q_n1 = q_inf-(q_inf-q_n)*exp(Aexp*(1-rtrial/r_n));
        H_n1 = H*(q_inf-q_n)/r_n*exp(Aexp*(1-rtrial/r_n));
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
        H_n1 = 0;
    elseif (q_n1>q_inf)
        q_n1 = q_inf;
        H_n1 = 0;
    end
    ce_tang_n1 = (q_n1/r_n1)*ce-(q_n1-H_n1*r_n1)/(r_n1.^3)*((ce*eps_n1')...
                                                    *(ce*eps_n1')');
    if (visc == 1)
        ce_alg_n1 = ce_tang_n1+(alf*dt)/(eta+alf*dt)*(H_n1*r_n1-q_n1)/...
                            (tau_af*r_n1.^2)*((ce*eps_n1')*(ce*eps_n1')');
    end

else

    %*    Elastic load/unload
    fload=0;
    r_n1= r_n ;
    q_n1= q_n ;
    ce_alg_n1 = (q_n1/r_n1)*ce;
    ce_tang_n1 = ce_alg_n1;

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{3}= ce_tang_n1 ;
hvar_n1{5}= r_n1 ;
```

```
hvar_n1{6}= q_n1 ;
if (visc == 1)
    hvar_n1{4} = ce_alg_n1;
end
%*************************************************************************
%* Auxiliar variables                                                    *%
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
%*************************************************************************
```

C) Modelos_de_dano1.m

```

 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
sig_n1 = eps_n1*ce;
sig_n1(sig_n1 < 0) = 0;
rtrial= sqrt(sig_n1*eps_n1');

elseif (MDtype==3) %*Non-symmetric
sig_n1 = eps_n1*ce;
sig_n1p = sig_n1;
sig_n1p(sig_n1p < 0) = 0;
th = (sig_n1p(1)+sig_n1p(2))/(abs(sig_n1(1))+abs(sig_n1(2)));
rtrial= (th+(1-th)/n)*sqrt(eps_n1*ce*eps_n1');
end
%*************************************************************************
return
```

D) dibujar_criterio_dano1.m

```
1
2  function hplot = dibujar_criterio_dano1(ce,nu,q,tipo_linea,MDtype,n)
3  %*************************************************************************
4  %*               PLOT DAMAGE SURFACE CRITERIUM: ISOTROPIC MODEL   %*
5  %*                                                                %*
6  %*     function [ce] = tensor_elastico (Eprop, ntype)             %*
7  %*                                                                %*
8  %*     INPUTS                                                     %*
9  %*                                                                %*
10 %*            Eprop(4)   vector de propiedades de material  %*
11 %*                         Eprop(1)= E------>modulo de Young      %*
12 %*                         Eprop(2)= nu----->modulo de Poisson    %*
13 %*                         Eprop(3)= H----->modulo de Softening/hard.%*
14 %*                         Eprop(4)=sigma_u----->tensile limit %*
15 %*                   ntype                                 %*
16 %*                              ntype=1 plane stress             %*
17 %*                              ntype=2 plane strain             %*
18 %*                              ntype=3 3D                       %*
19 %*                   ce(4,4)    Constitutive elastic tensor (PLANE S.)%*
20 %*                   ce(6,6)                                ( 3D)   %*
21 %*************************************************************************
22 %*       Inverse ce                                               %*
23 ce_inv=inv(ce);
24 c11=ce_inv(1,1);
25 c22=ce_inv(2,2);
26 c12=ce_inv(1,2);
27 c21=c12;
28 c14=ce_inv(1,4);
29 c24=ce_inv(2,4);
30 %*************************************************************************
31 % POLAR COORDINATES
32 if MDtype==1
33     tetha=[0:0.01:2*pi];
34     %*********************************************************************
35     %* RADIUS
36     D=size(tetha);                    %*  Range
37     m1=cos(tetha);                    %*
38     m2=sin(tetha);                    %*
39     Contador=D(1,2);                  %*
40     radio = zeros(1,Contador) ;
41     s1    = zeros(1,Contador) ;
42     s2    = zeros(1,Contador) ;
43     for i=1:Contador
44         radio(i)= q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*...
45                                     [m1(i) m2(i) 0 nu*(m1(i)+m2(i))]');
46         s1(i)=radio(i)*m1(i);
47         s2(i)=radio(i)*m2(i);
48     end
```

```
    hplot =plot(s1,s2,tipo_linea);

elseif MDtype==2

        limitINF = -pi/2*0.99;
        limitSUP = pi*0.99;
        tetha=[limitINF:0.01:limitSUP];
        % 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
            radio(i)= q/sqrt([m1(i)*(m1(i)>0) m2(i)*(m2(i)>0) 0 nu*...
                              (m1(i)+m2(i))]*ce_inv*[m1(i) m2(i) 0 ...
                                                     nu*(m1(i)+m2(i))]');
            s1(i)=radio(i)*m1(i);
            s2(i)=radio(i)*m2(i);
        end
        hplot =plot(s1,s2,tipo_linea);

elseif MDtype==3

        tetha=[0:0.01:2*pi];
        % 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
          tetha_aux = (m1(i)*(m1(i)>0) + m2(i)*(m2(i)>0))/...
                                              (abs(m1(i)) + abs(m2(i))) ;
          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))]')/...
                                        (tetha_aux + ((1 - tetha_aux)/n));
          s1(i)=radio(i)*m1(i);
          s2(i)=radio(i)*m2(i);
      end
      hplot =plot(s1,s2,tipo_linea);
end
%*************************************************************************
return
```