UPC - BARCELONA TECH
Master on Numerical Methods in Engineering
Spring 2017

Computational Solid Mechanics

Assignment 1

Magdalena Pérez Lanfranco

Due date: April, 7th 2017

Contents

1 Rate Independent Models

In order to assess the correctness of the models implemented, the path at the stress space and the stress-strain curve were obtained for different representative loading paths for both models. The loading paths used were defined as follows:

Case 1: $\alpha = 900 \quad ; \quad \beta = -1000 \quad ; \quad \gamma = 750$

Case 2: $\alpha = 500 \quad ; \quad \beta = -500 \quad ; \quad \gamma = 150$

Case 3: $\alpha = 600 \quad ; \quad \beta = -3000 \quad ; \quad \gamma = 2200$

The next sections present the results obtained for those 3 cases for the non-symmetric model and the tension-only model. For both models it was chosen to use the exponencial hardening-softening law with $H = -0.8$, in order to prove its correct implementation.

1.1 Case 1

Figure 1.1 shows the strain path in the stress space and Figure 1.2 shows the corresponding stress-strain curve for the non-symmetric model. Figures 1.3 and 1.4 show the same results for the tension-only model. As the results for both models are similar due to the Case evaluated, the following comments can be applied to the two of them.

The first step of the path is a tensile loading process that goes out of the elastic domain corresponding to the black line in all mentioned Figures. The stress reaches a maximum and then remains constant (perfect plasticity - no viscous effects). The next step represents a compressive loading process that produces a negative strain inside the elastic domain, represented in green in all Figures. The third and last step is another tensile loading, but due to the fact that the material has already been damaged in the first step, the maximum stress reached is lower than before and the slope of the plot is modify by a factor of $(1 - d)$.

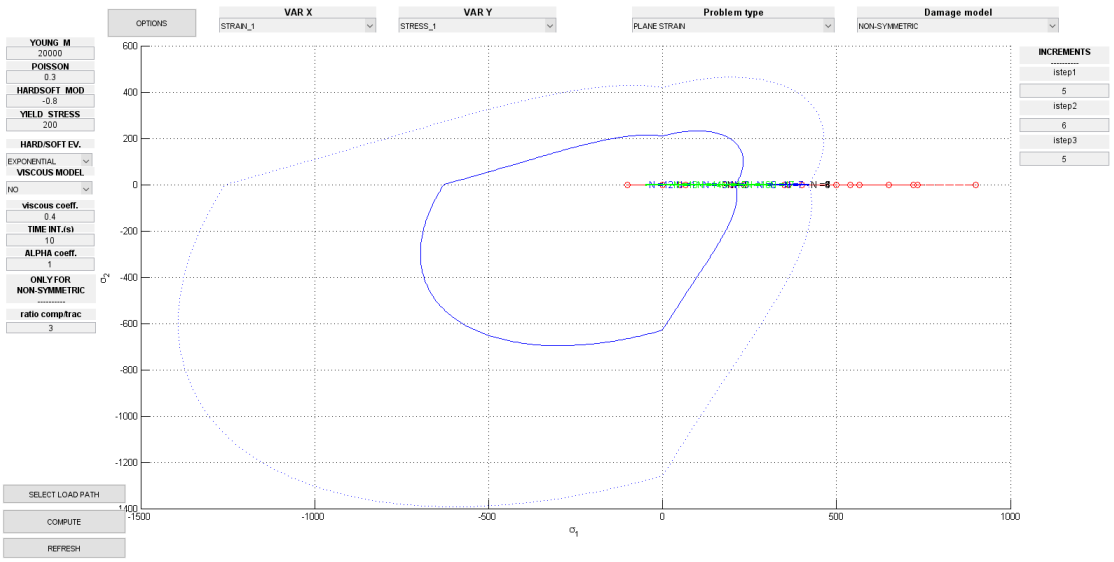

Figure 1.1: Stress space for the non-symmetric model: Case 1

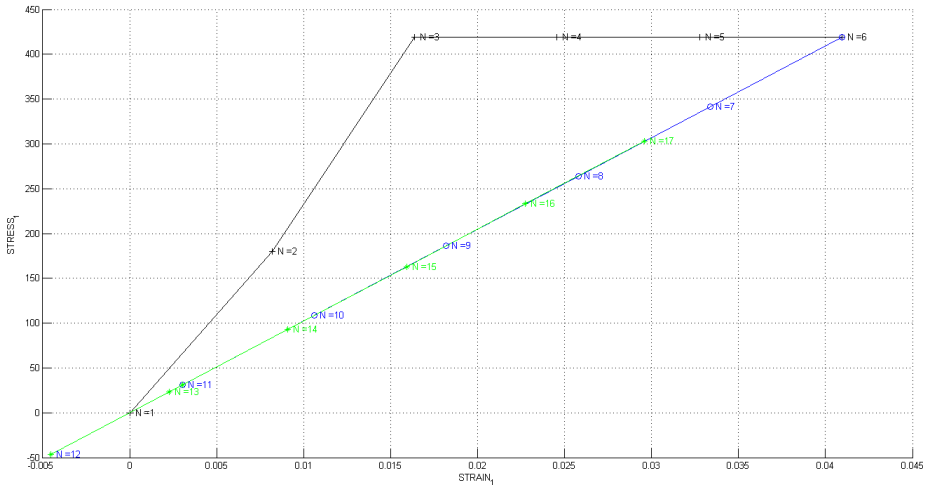

Figure 1.2: Stress-strain plot for the non-symmetric model: Case 1

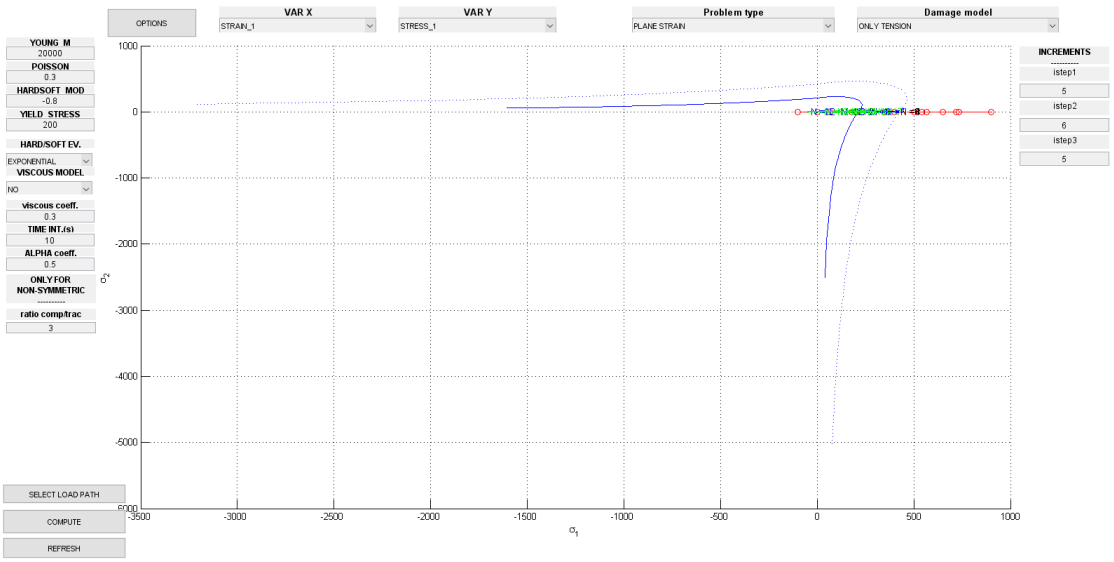

Figure 1.3: Stress space for the tension-only model: Case 1

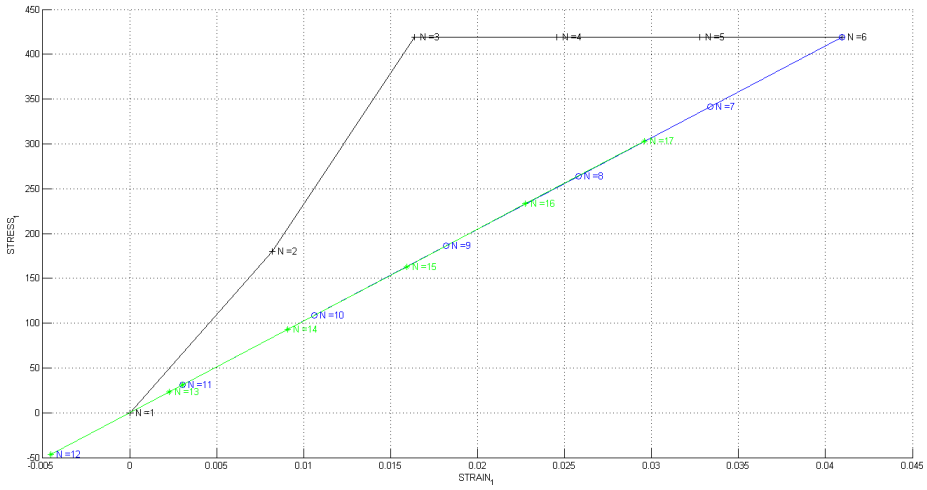

Figure 1.4: Stress-strain plot for the only-tension model: Case 1

1.2 Case 2

As before, the behaviour of both models is the same for this chosen path, so the same comments can be made about both of them. The stress spaces are presented in Figures 1.5 and 1.7 and the corresponding stress-strain curves can be seen in Figures 1.6 and 1.8.

The first step of this path is equivalent to the first of Case 1, and it is represented by the black lines in plots 1.5 and 1.6. Steps 2 and 3 are now biaxial instead of uniaxial but their consequences in terms of elastic behavior of the domain are the same as before, with the only difference being that now the unloading step (step 2) produces a positive strain.

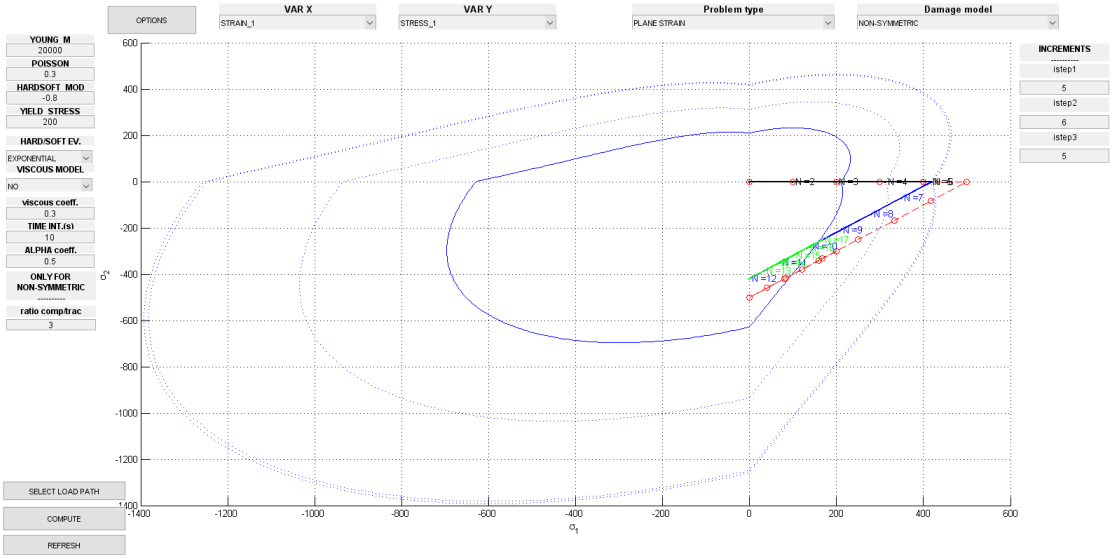

Figure 1.5: Stress space for the non-symmetric model: Case 2

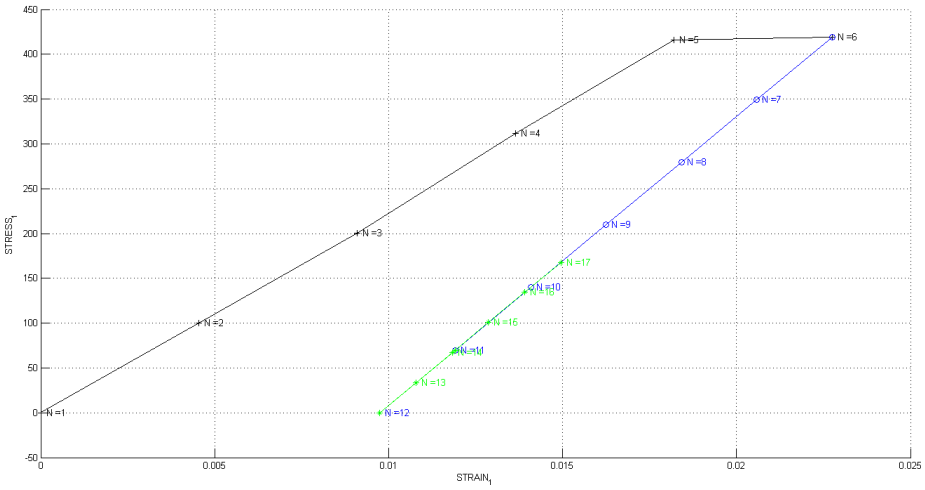

Figure 1.6: Stress-strain plot for the non-symmetric model: Case 2

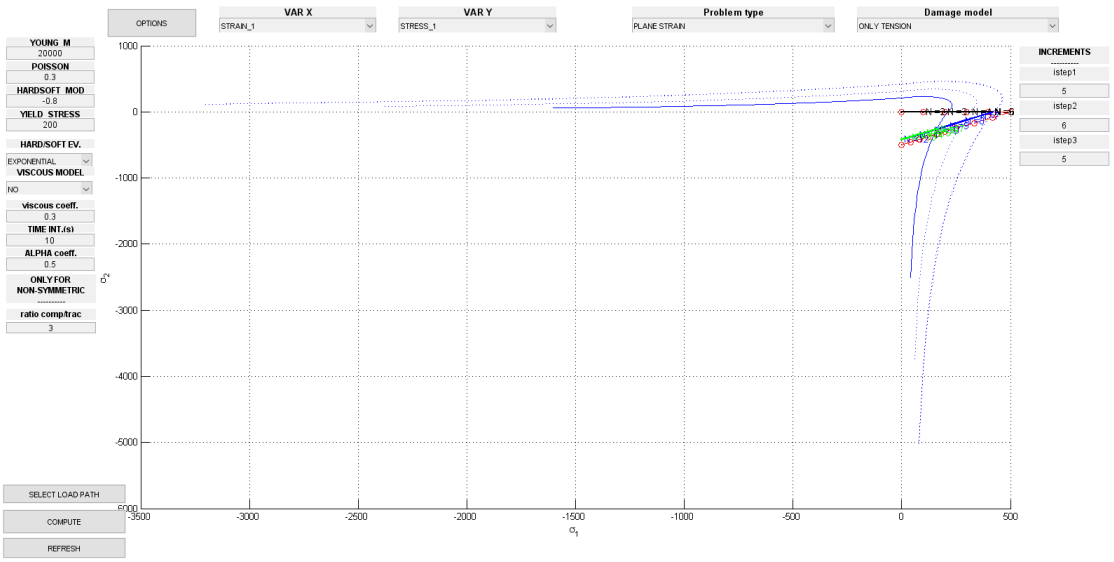

Figure 1.7: Stress space for the tension-only model: Case 2

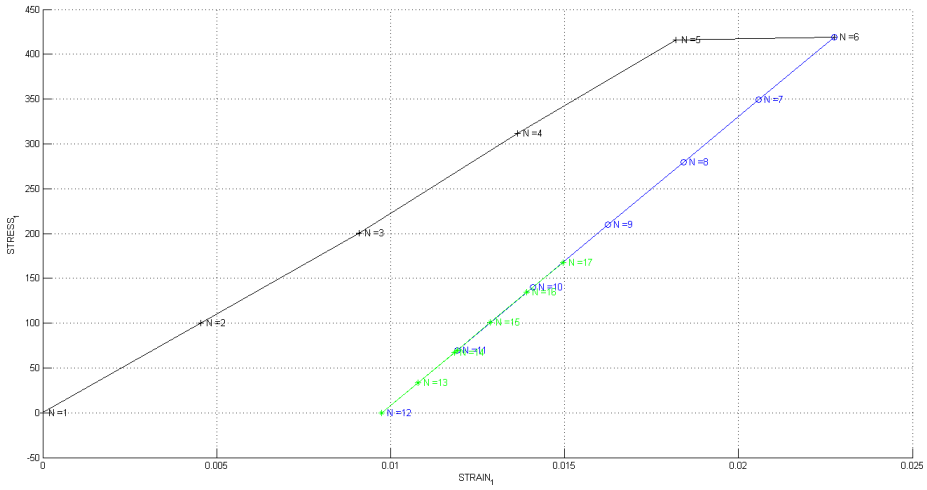

Figure 1.8: Stress-strain plot for the only-tension model: Case 2

1.3 Case 3

In case 3 there are 3 biaxial loading processes. The first step is the same as in the previous cases, a load that goes out of the elastic domain and produces some damage in the material. However, for this case there is a difference between models for the second step of the path. In the case of the non-symmetric model, the second step also produces damage in the material as it lies outside of the compression limit of the domain, as seen in Figure 1.10. In the case of the tension-only model, as its name says, there is no compression limit so the second step relies inside the domain. That means that the response is elastic and there is no further damage, as shown in the plot 1.12. As a remark, if the load in the second step is very close to the compression limit of the non-symmetric model, then the difference between both models will be smaller and in some cases even hard to notice.

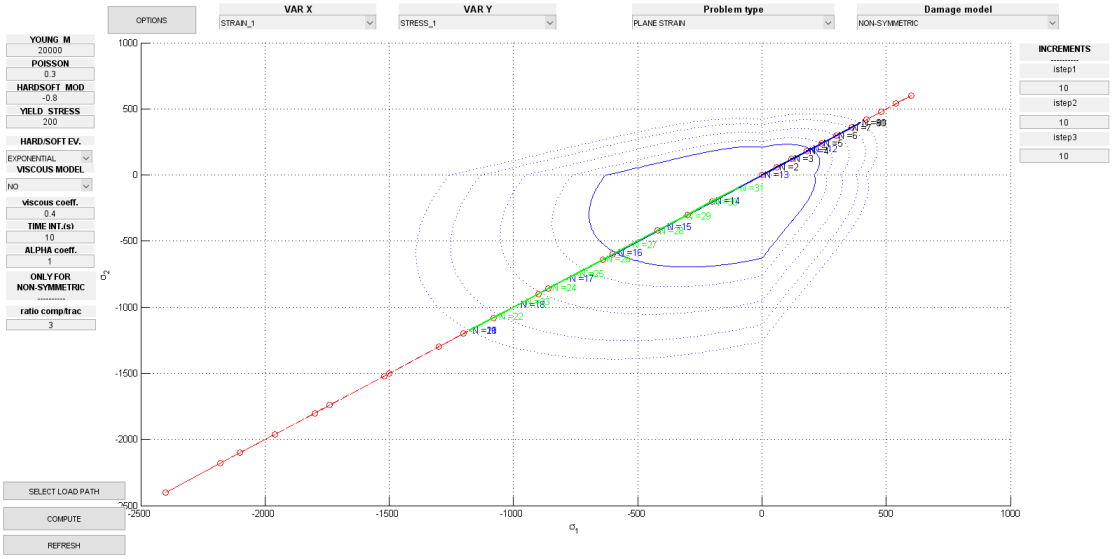

Figure 1.9: Stress space for the non-symmetric model: Case 3

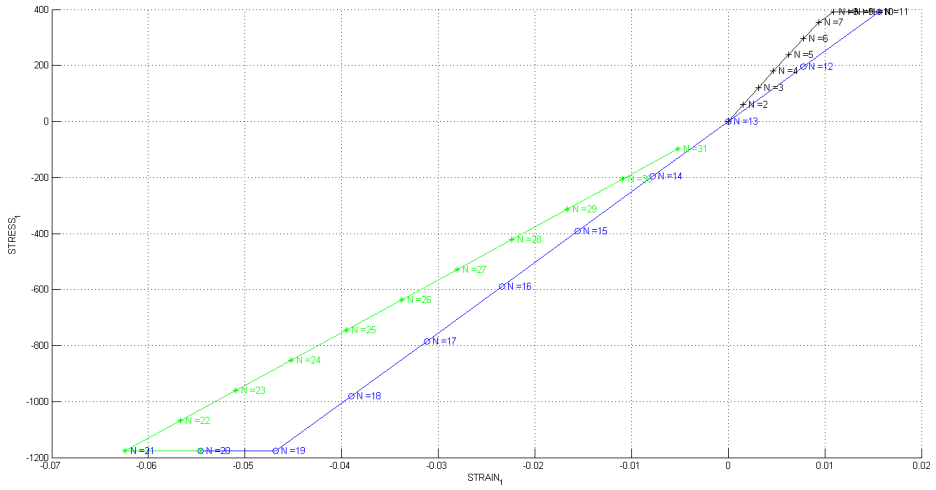

Figure 1.10: Stress-strain plot for the non-symmetric model: Case 3

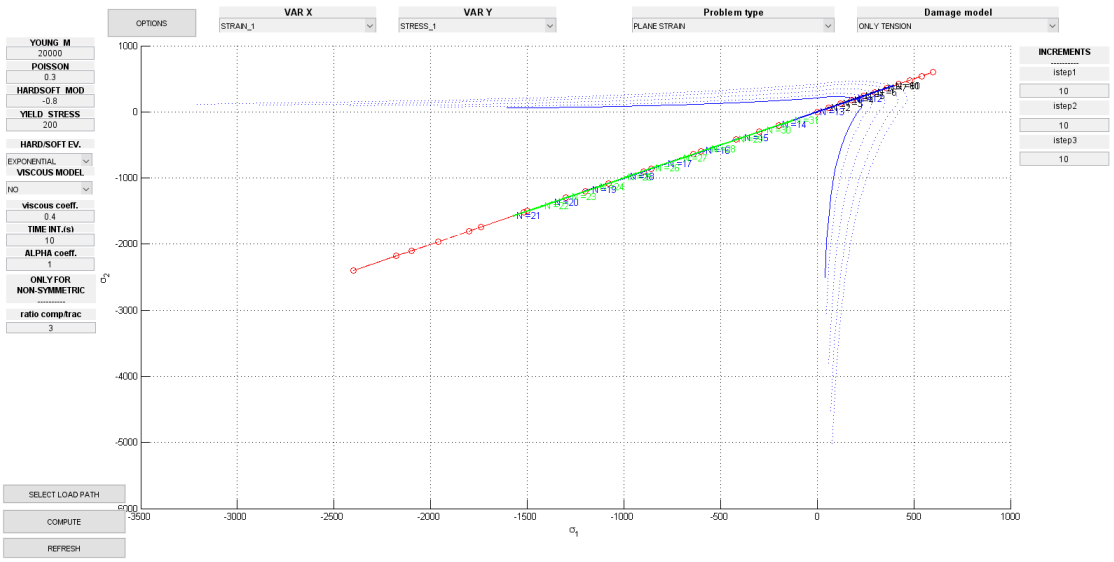

Figure 1.11: Stress space for the tension-only model: Case 3

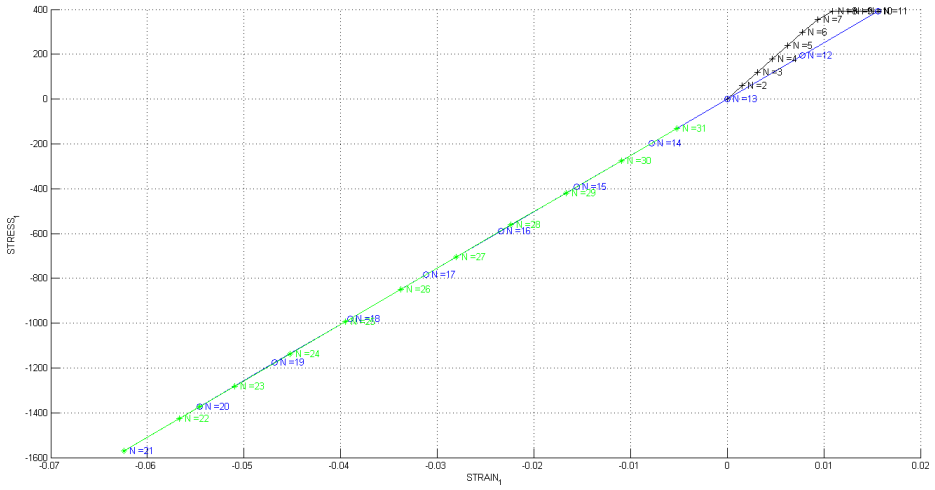

Figure 1.12: Stress-strain plot for the only-tension model: Case 3

2 Rate Dependent Models

2.1 Effect of viscosity η

In the implemented model, the viscosity affects how the material behaves outside its elastic domain. In order to visualize the effect of the parameter η in the material's response, the model was run for three values of viscosity ($\eta = 0.4, \eta = 0.8, \eta = 1.2$) for the same loading path. The loading path used can be seen in Figure 2.1.

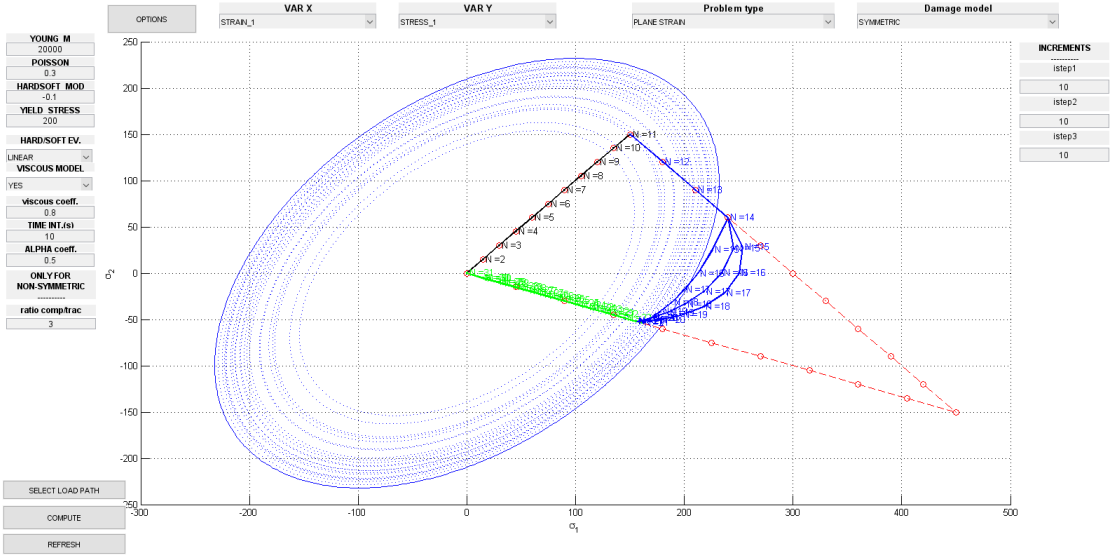

Figure 2.1: Effect of viscosity η in the stress space

Figure 2.2 shows the effect of the viscosity in the stress-strain curve. As it can be seen in the plot, the area surrounded by the curve is larger for bigger η, which means that more energy is dissipated for more viscous materials. Also, for the same strain, the more viscous the material, the higher value of maximum stress it can reach.

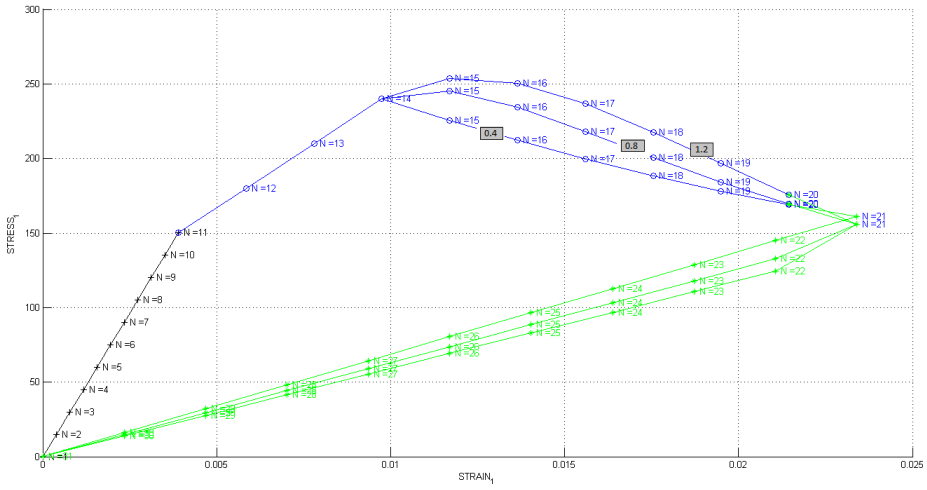

Figure 2.2: Effect of viscosity η in the stress-strain curve

2.2 Effect of strain rate $\dot{\varepsilon}$

The strain rate can be modified by changing the integration time in the provided console. To study the effect of the strain rate in the behavior of the material, several different times were considered: $t = 1, t = 10, t = 20, t = 30$. The loading path used was the same than the one used for testing the effect of viscosity, seen in Figure 2.1. That means that the same amount of load will be applied during different time periods.

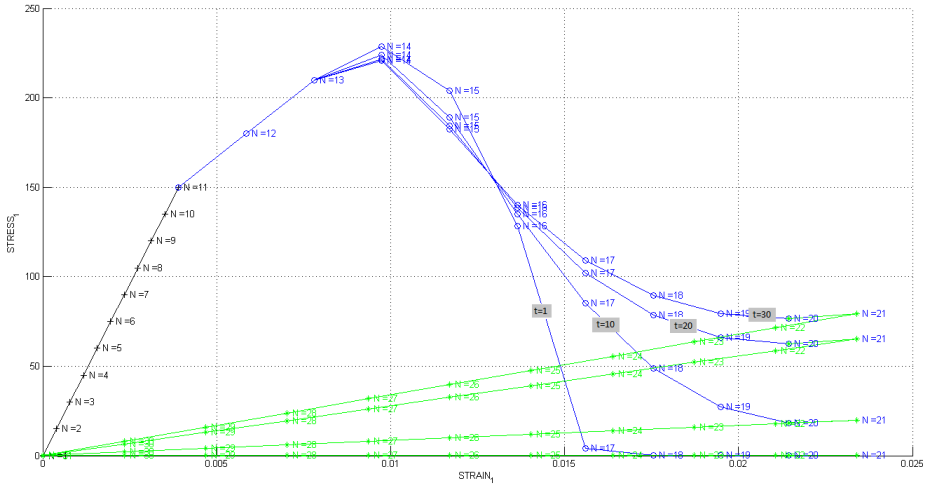

Figure 2.3: Effect of the strain rate $\dot{\varepsilon}$ in the stress-strain curve

Figure 2.3 shows the effect that the strain rate has in the stress-strain curve. As it could be expected, the strain rate has no influence inside the elastic domain. However, it does have an effect when viscosity plays its role. As it can be seen in the graph, the higher the final time (for the same loading path) the smoother the curve. This means that the material is taking the same amount of load in different time windows, and the material behaves in a more viscous way for longer time windows.

2.3 Effect of integration parameter α

The parameter α controls the integration scheme used in the model, with $\alpha = 1$ for implicit (backward) Euler, $\alpha = 0.5$ for Crank-Nicholson and $\alpha = 0$ for explicit (forward) Euler. The Crank-Nicholson scheme is unconditionally stable and the Euler methods are conditionally stable. The stability condition is specially critical for the explicit Euler scheme.

2.3.1 On the stress-strain curves

Figure 2.4 shows the results in the stress-strain curve using different values of α for the same loading path used in 2.1. The values of α used were 0, 0.25, 0.5, 0.75 and 1. All schemes are stable except for the one integrated using the explicit Euler scheme ($\alpha = 0$), that shows spurious oscillations that are not representative of the actual behavior. This could be stabilized by using smaller time steps in each part of the loading path.

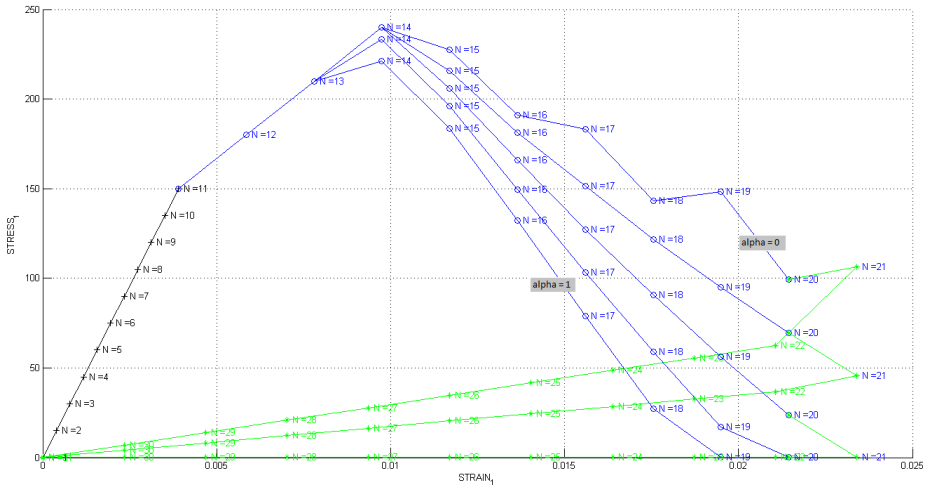

Figure 2.4: Effect of the integration parameter α in the stress-strain curve

2.3.2 On the algorithmic constitutive operator

Another way of visualizing the effect of the different integration schemes is by plotting the algorithmic tangent operator for each of the values of α stated previously. Figure 2.5 shows the results obtained using the same loading path as in the previous sections.

As for the stress-strain curves, it is easy to see that the explicit Euler scheme is unstable because it presents numerical oscillations that are not a good representation of the actual material's behavior.

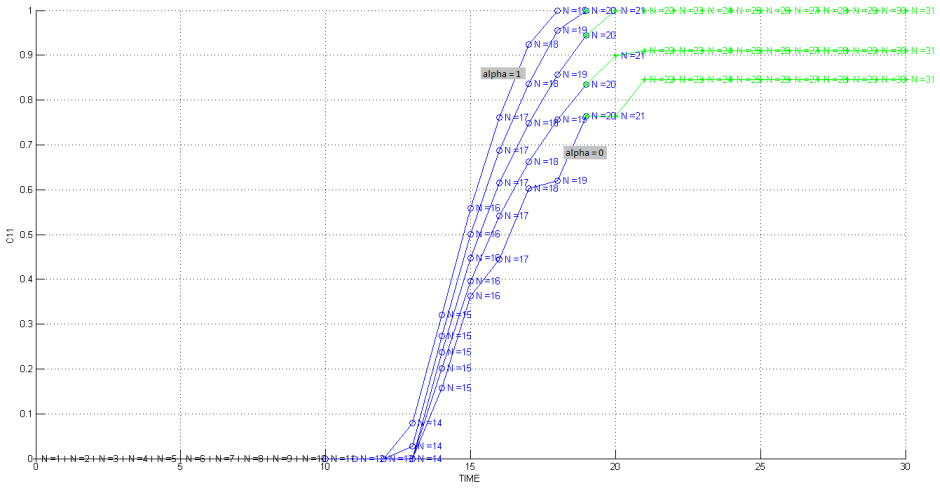

Figure 2.5: Effect of the integration parameter α in the C_{11} component of the tangent operator

APPENDIX

The code developed for this assignment is attached in this Appendix. Only the lines of the provided functions that were modified are included here for practicity purposes.

Damage_main

```
% INITIALIZING  (i = 1) !!!!
i = 1 ;
r0 = sigma_u/sqrt(E);
hvar_n(5) = r0; % r_n
hvar_n(6) = r0; % q_n
eps_n1 = strain(i,:) ;
sigma_n1 =ce*eps_n1'; % Elastic
sigma_v{i} = [sigma_n1(1)  sigma_n1(3) 0;sigma_n1(3) sigma_n1(2) 0 ; 0 0  ...
    sigma_n1(4)];

nplot = 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) = C_tan(1,1);
for  iload = 1:length(istep)
    % Load states
    for iloc = 1:istep(iload)
        i = i + 1 ;
        TIMEVECTOR(i) = TIMEVECTOR(i-1)+ Δ_t(iload) ;
        % Total strain at step "i"
        % ------------------------
        eps_n1 = strain(i,:) ;
        %      DAMAGE MODEL
        if viscpr ==1
            eps_n = strain(i-1,:);
            [sigma_n1,hvar_n,aux_var, C_tan] = ...
                rmap_dano2(eps_n,eps_n1,Δ_t,hvar_n,Eprop,ce);
        else
        [sigma_n1,hvar_n,aux_var] = ...
            rmap_dano1(eps_n1,hvar_n,Eprop,ce,MDtype,n);
        end
        % 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)
```

```
48          %vartoplot{i}(4) = C_tan(1,1);
49      end
50  end
```

Dibujar_criterio_dano1

```
1   elseif MDtype==2
2       limitINF = -pi/2*0.99;
3       limitSUP = pi*0.99;
4       tetha=[limitINF:0.01:limitSUP];
5       % RADIUS
6       D=size(tetha);                          %  Range
7       m1=cos(tetha);                          %
8       m2=sin(tetha);                          %
9       Contador=D(1,2);                        %
10
11      radio = zeros(1,Contador) ;
12      s1    = zeros(1,Contador) ;
13      s2    = zeros(1,Contador) ;
14
15      for i=1:Contador
16          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 ...
17              nu*(m1(i)+m2(i))]');
18
19          s1(i)=radio(i)*m1(i);
20          s2(i)=radio(i)*m2(i);
21
22      end
23      hplot =plot(s1,s2,tipo_linea);
24
25  elseif MDtype==3
26
27      tetha=[0:0.01:2*pi];
28
29      % RADIUS
30      D=size(tetha);                          %  Range
31      m1=cos(tetha);                          %
32      m2=sin(tetha);                          %
33      Contador=D(1,2);                        %
34
35
36      radio = zeros(1,Contador) ;
37      s1    = zeros(1,Contador) ;
38      s2    = zeros(1,Contador) ;
39
40
41      for i=1:Contador
42          tetha_aux = (m1(i)*(m1(i)>0) + m2(i)*(m2(i)>0))/(abs(m1(i)) + ...
                abs(m2(i))) ;
43          radio(i)= q/sqrt([m1(i) m2(i) 0 nu*(m1(i)+m2(i))]*ce_inv*[m1(i) ...
                m2(i) 0 ...
44              nu*(m1(i)+m2(i))]')/(tetha_aux + ((1 - tetha_aux)/n));
45
46          s1(i)=radio(i)*m1(i);
47          s2(i)=radio(i)*m2(i);
48
49      end
50      hplot =plot(s1,s2,tipo_linea);
51
52  end
```

```
return
```

Modelos_de_dano1

```
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
rtrial = sqrt(eps_n1.*(eps_n1>0)*ce*eps_n1');

elseif (MDtype==3)  % Non-symmetric
s_n1 = ce*eps_n1';
s1=s_n1(1); s2=s_n1(2);
tetha_aux = (s1*(s1>0) + s2*(s2>0))/(abs(s1)+abs(s2));
rtrial = (tetha_aux +(1 - tetha_aux)/n)* sqrt(eps_n1*ce*eps_n1');
end
return
```

Rmap_dano1

```
    else
        % Exponential
         %if H>0
          dqdr = H*((inf_q - r_n)/r_n)*exp(H*(1-(r_n1/r_n)));
          q_n1 = q_n + dqdr*Δ_r;
        %elseif H<0
           %dqdr = H*((zero_q - r_n)/r_n)*exp(H*(1-(r_n1/r_n)));
           %q_n1 = q_n + dqdr*Δ_r;
        %end
    end

    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1>inf_q) %Acotar r por arriba
        q_n1 = inf_q;
    end
else
    % Elastic load/unload
    fload=0;
    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(5)= r_n1 ;
hvar_n1(6)= q_n1 ;

% Auxiliar variables
aux_var(1) = fload;
aux_var(2) = q_n1/r_n1;
```

Rmap_dano2

```
if(rtrial_alpha > r_n)
    fload=1;
    %Loading
    Δ_r=rtrial_alpha-r_n;
    r_n1= (eta-Δ_t*(1-alpha))/(eta+alpha*Δ_t)*r_n + ...
          Δ_t/(eta+alpha*Δ_t)*rtrial_alpha  ;
    if hard_type == 0
        %  Linear hardening
        q_n1= q_n+ H*Δ_r;
    else
        % Exponential hardening
        dqdr = H*(q_inf - r_n)/r_n*exp(H*(1-r_n1/r_n));
        q_n1 = q_n + dqdr*Δ_r;
    end
    % Restrict value to q_infinite if needed
    if(q_n1<zero_q)
        q_n1=zero_q;
    elseif (q_n1 > q_inf)
        q_n1=q_inf;
    end
else
    %Elastic load/unload
    fload=0;
    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';

% Algorithmic tangent operator
if (rtrial_alpha > r_n)
sigma_eff = ce*eps_n1'; %effective stress
C_tan = (1-dano_n1)*ce + ...
    (alpha*Δ_t)/((eta+alpha*Δ_t)*rtrial_n1)*...
    ((H*r_n1-q_n1)/r_n1^2)*(sigma_eff'*sigma_eff);
    %Only linear case is considered
else
    C_tan = (1-dano_n1)*ce ;
end

% Updating historic variables
```

```
46 hvar_n1(5)= r_n1 ;
47 hvar_n1(6)= q_n1 ;
48
49 % Auxiliar variables
50 aux_var(1) = fload;
51 aux_var(2) = q_n1/r_n1;
```