Implimenting Codes in MATLAB for Rate Dependent and Independent Models

Assignment1: Continuum Damage Models

Nadim Saridar

Department Name
Universitat Politecnica de Catalunya
March 23, 2020

Contents

List of Figures

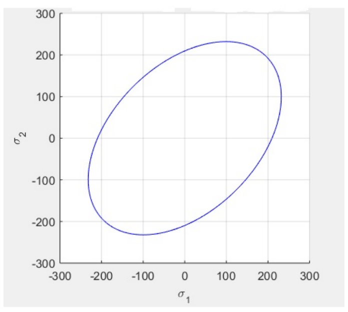
1 Part 1: Rate Independent Models

In the first part, all calculations were made for inviscid models, therefore viscosity will not be considered in the calculations.

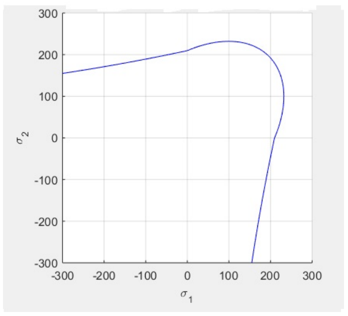
1.1 Damage Models

The figures below represent the different damage models in this assignment. In Figure 1(a) is the Symmetric model where the elastic region of tension and compression are equal. The Only Tension model in Figure 1(b) has an elastic region for the tension, and in the compression region it is always elastic. The Non-Symmetric model in Figure 1(c) has the characteristics for the Symmetric model but with a compression to tension ratio in the compression region.
The following values were taken for the figures below:

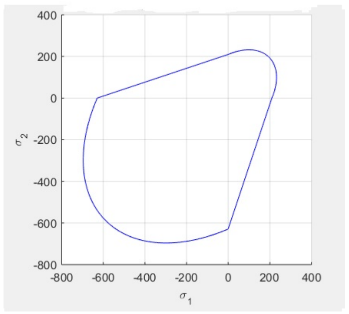
$$\sigma_y = 200$$
$$\nu = 0.3$$
$$n = 3$$



(a) Symmetric



(b) Only Tension



(c) Non-Symmetric

Figure 1: The Three Damage Models

1.2 Hardening and Softening Models

Figure 2 shows the two hardening laws which are linear (green) and exponential (black). It shows how the material behaves depending on its hardening and softening modulus. In the figure below, an extreme case was taken of hardening and softening modulus of $|H| = 1$.

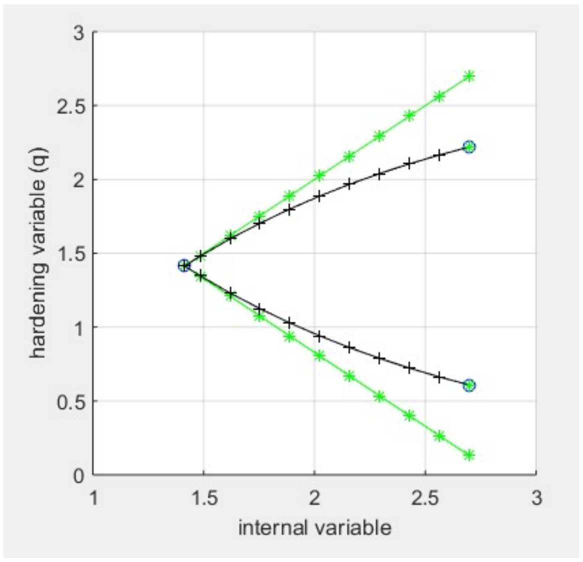


Figure 2: Hardening Models

1.3 Checking the Correctness

The same material properties were taken for all the cases:

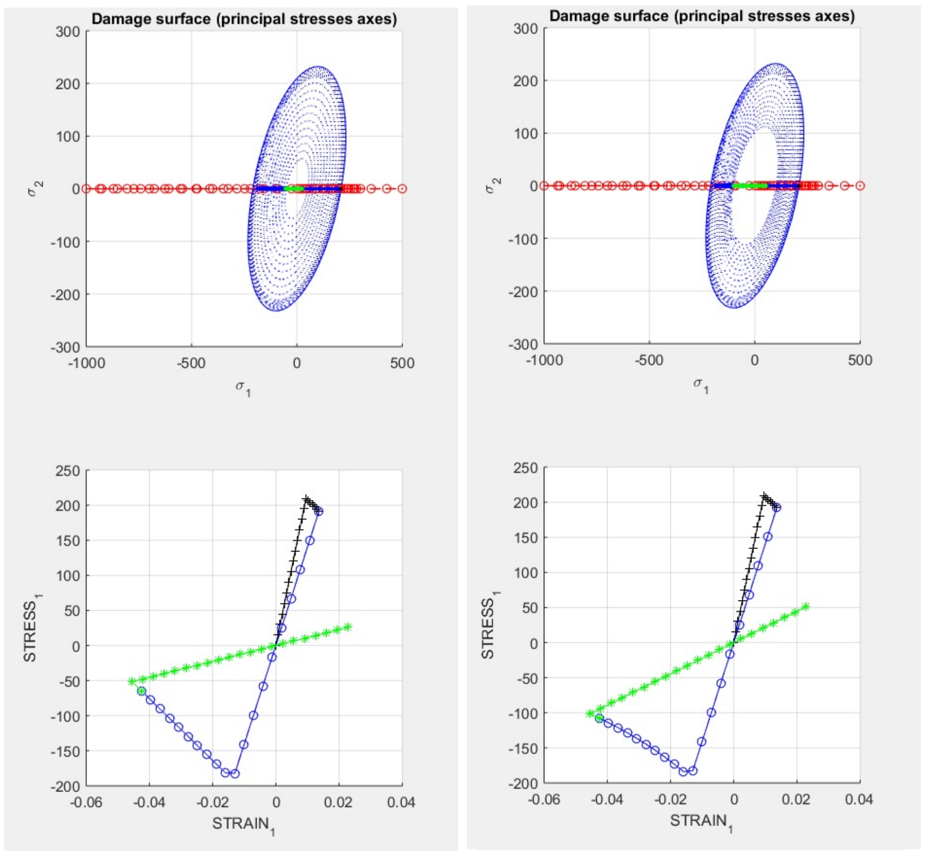
$$\begin{aligned}
&\sigma_y = 200\\
&E = 20000\\
&\nu = 0.3\\
&H = -0.2 \quad (Softening)\\
&n = 3
\end{aligned}$$

1.3.1 Case 1

For the first case, purely uniaxial loading and unloading was tested for the material, and checked the values for each damage model and hardening law.

The values taken are the following:

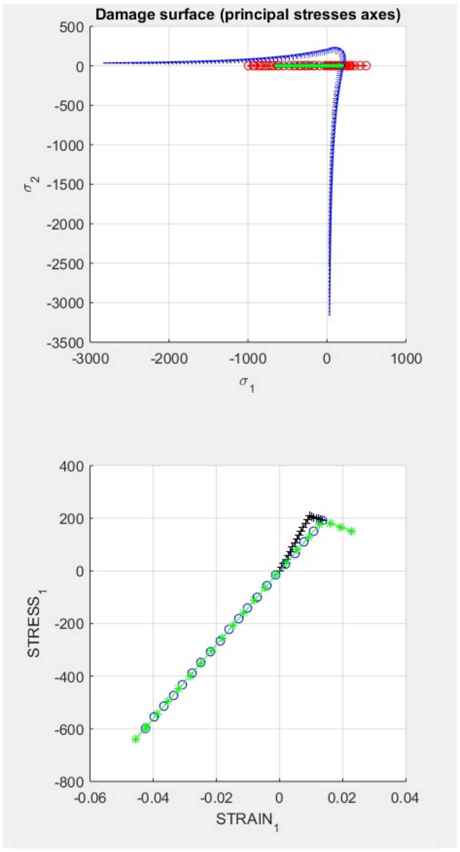
$$\Delta\sigma_1^{(1)} = 300 \qquad \Delta\sigma_2^{(1)} = 0$$
$$\Delta\sigma_1^{(2)} = -1300 \quad \Delta\sigma_2^{(2)} = 0$$
$$\Delta\sigma_1^{(3)} = 1500 \qquad \Delta\sigma_2^{(3)} = 0$$



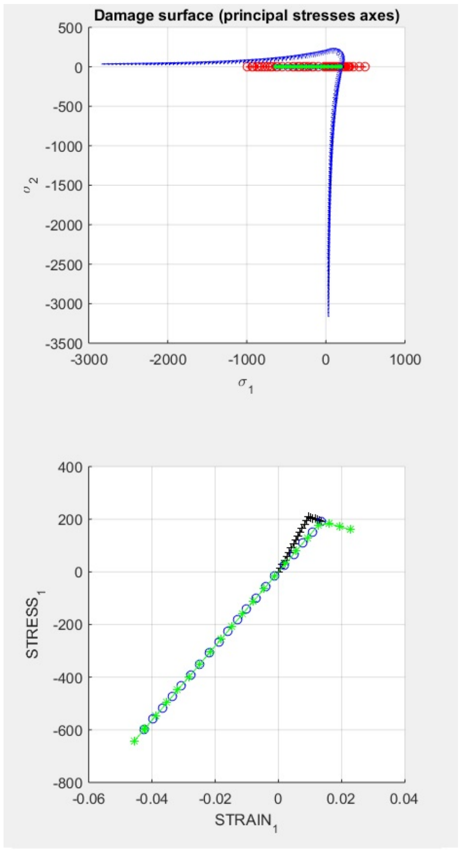
(a) Linear (b) Exponential

Figure 3: Case 1 for a Symemtric Model

The answers for the symmetric model (Figure 3) are logical because after the first loading, the yield stress and strain are the same after compression. Also using the exponential law, it is visible that the yield stress after the compressive load is bigger than the yield stress using the linear load (in absolute values), which is logical refering back to Figure 2.



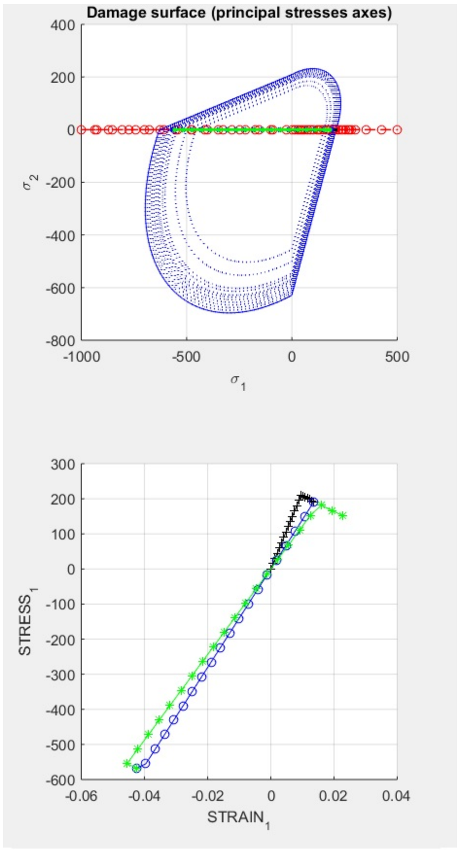
(a) Linear



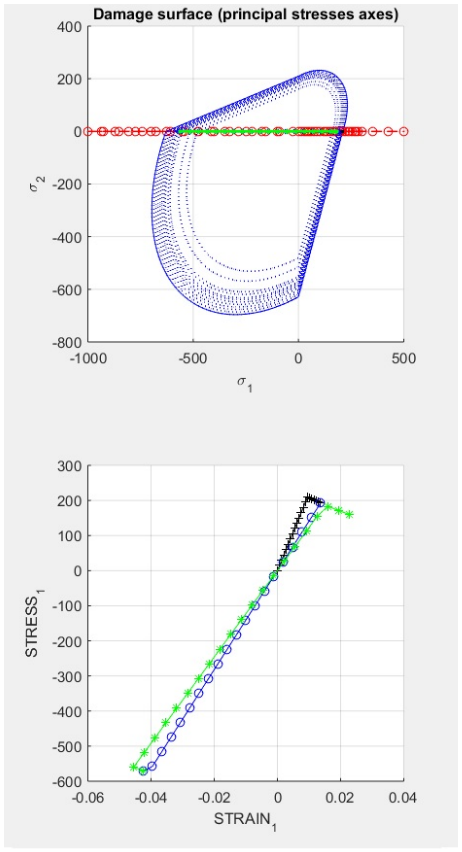
(b) Exponential

Figure 4: Case 1 for a Tension-Only Model

The results of the tension only model (Figure 4) shows its difference from the symmetric model, where it compresses indefinitely without any damage, and it is proven when a tensile load is applied after the compressive loading, as shown in the figure above.



(a) Linear



(b) Exponential

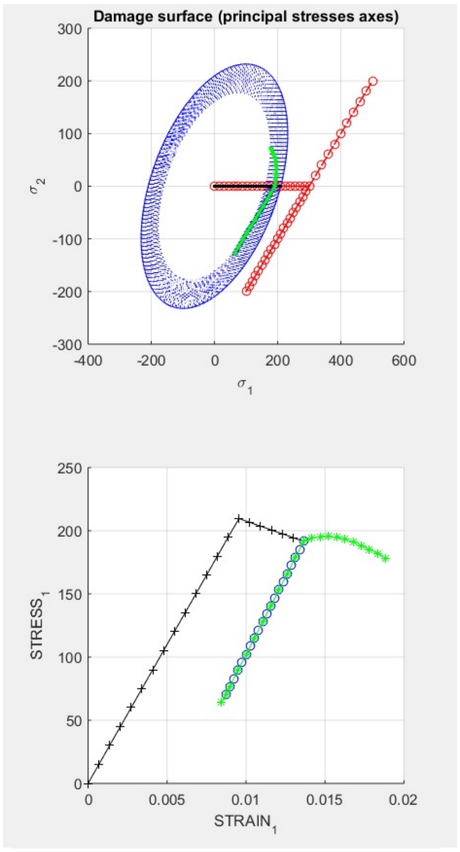
Figure 5: Case 1 for a Non-Symemtric Model

As shown in Figure 5, the compression/tension ratio is visible in the compressive loading, where the material needed a compressive load equal to three times (n times) the tensile load for it to be damaged.

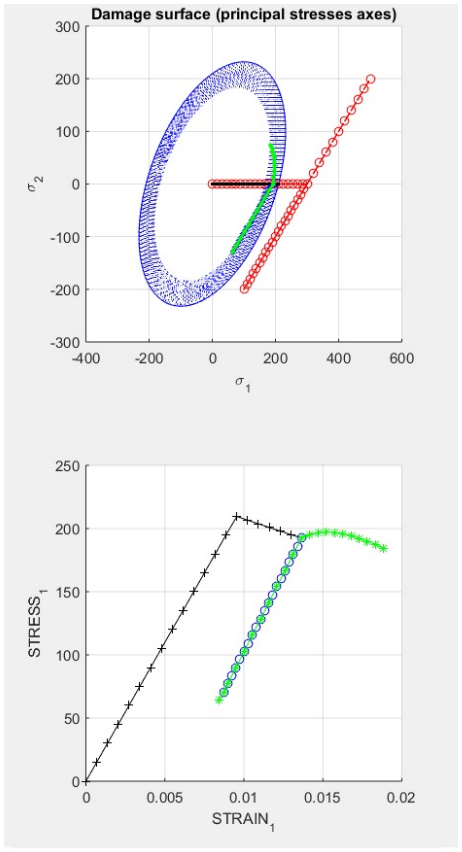
1.3.2 Case 2

For the second case, it starts with uniaxial loading and continues with biaxial unloading and then biaxial loading, and stress1 is plotted function of strain1 because it is the one mostly affected by the loads.
The values taken are the following:

$$\begin{aligned} \Delta\sigma_1^{(1)} &= 300 & \Delta\sigma_2^{(1)} &= 0 \\ \Delta\sigma_1^{(2)} &= -200 & \Delta\sigma_2^{(2)} &= -200 \\ \Delta\sigma_1^{(3)} &= 400 & \Delta\sigma_2^{(3)} &= 400 \end{aligned}$$



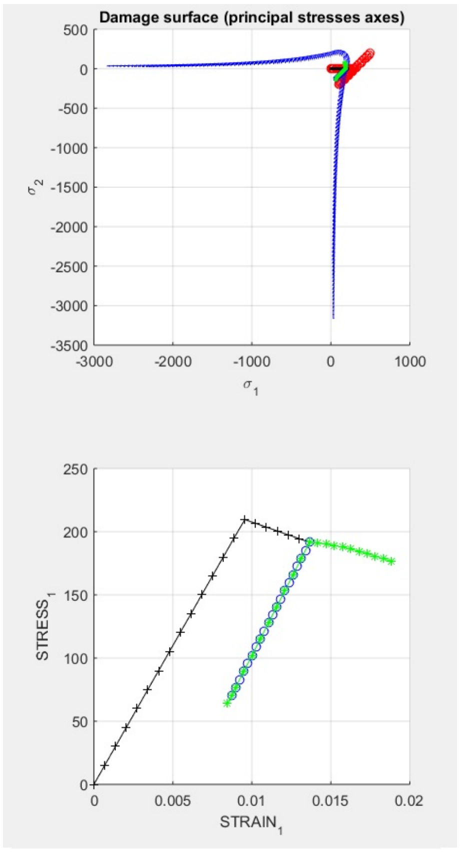
(a) Linear



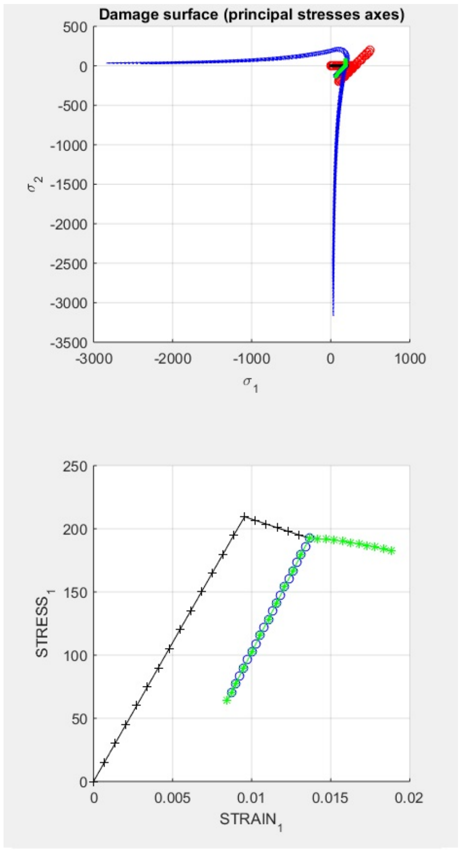
(b) Exponential

Figure 6: Case 2 for a Symemtric Model

In the case of a symmetric model, the first tensile loading (Figure 6) is similar to the load in case 1 (Figure 3). While the compressive and tensile biaxial loading have different results due to the compression and tension of in the other axial direction, which damages the model, therefore affecting the results of the stress1 to strain1 graph.



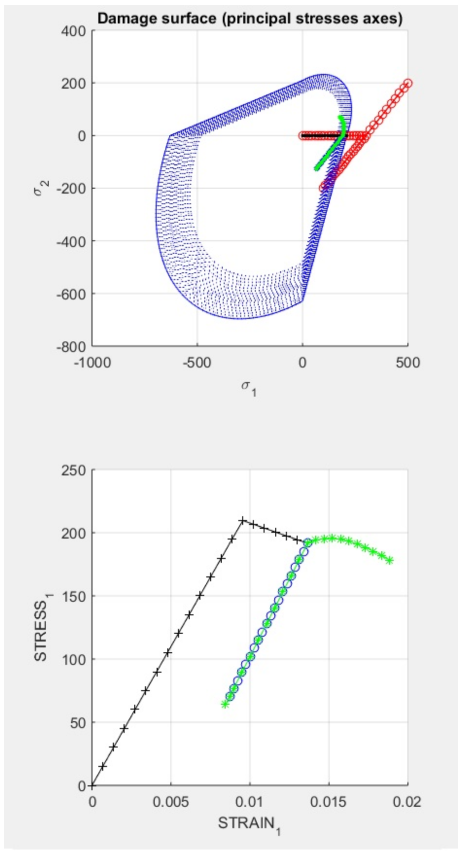
(a) Linear



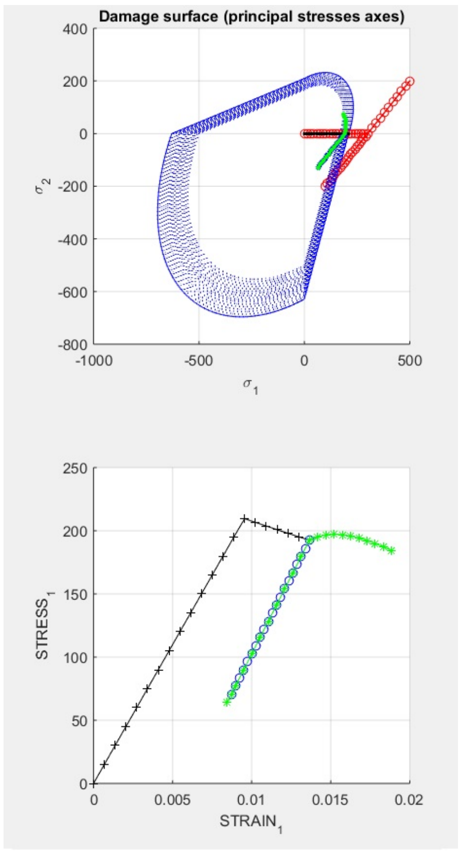
(b) Exponential

Figure 7: Case 2 for a Tension-Only Model

In the case of a only tension model (Figure 7), the results are similar to the case of the symmetric model, but there is no damage to the model during the compressive loading, therefore this explains the lines with the similar slope after the last loading path (biaxial tensile loading) in both the linear and exponential laws.



(a) Linear



(b) Exponential

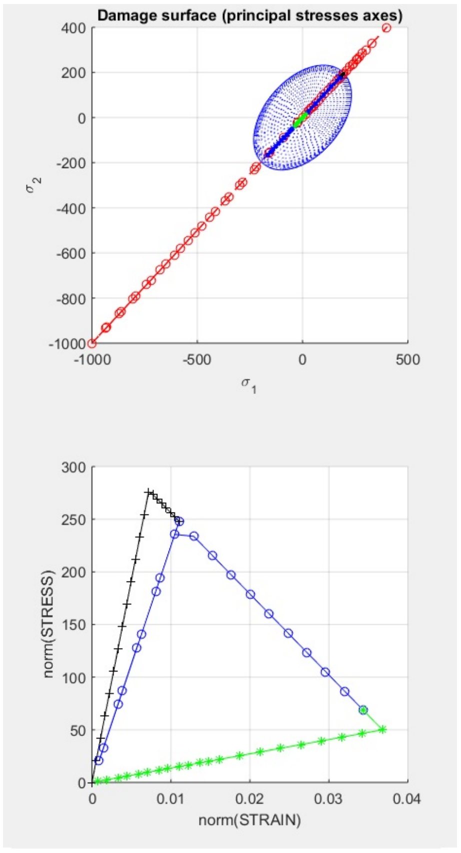
Figure 8: Case 2 for a Non-Syememtric Model

In the case of a non-symmetric model (Figure 8), the results are similar to the symmetric model, which means that there is a mistake in the code running the compressive loading (the only part that is different from the symmetric model).

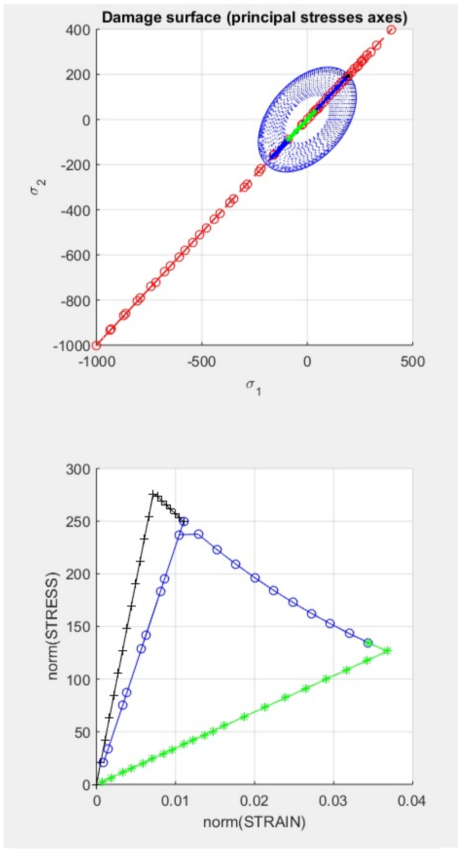
1.3.3 Case 3

For the second case, only biaxial loading and unloading was made. In this case, the norm of the stress and strain are taken to include the loads in both axis. The values taken are the following:

$$\Delta\sigma_1^{(1)} = 300 \qquad \Delta\sigma_2^{(1)} = 300$$
$$\Delta\sigma_1^{(2)} = -1300 \quad \Delta\sigma_2^{(2)} = -1300$$
$$\Delta\sigma_1^{(3)} = 1400 \qquad \Delta\sigma_2^{(3)} = 1400$$



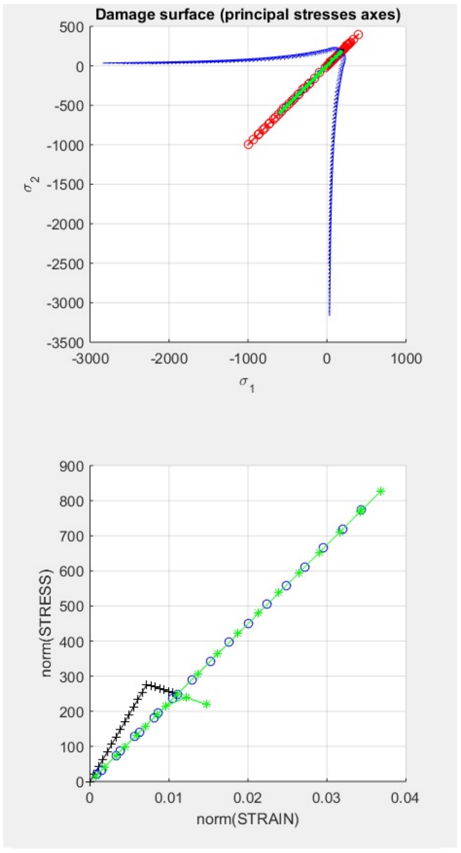
(a) Linear



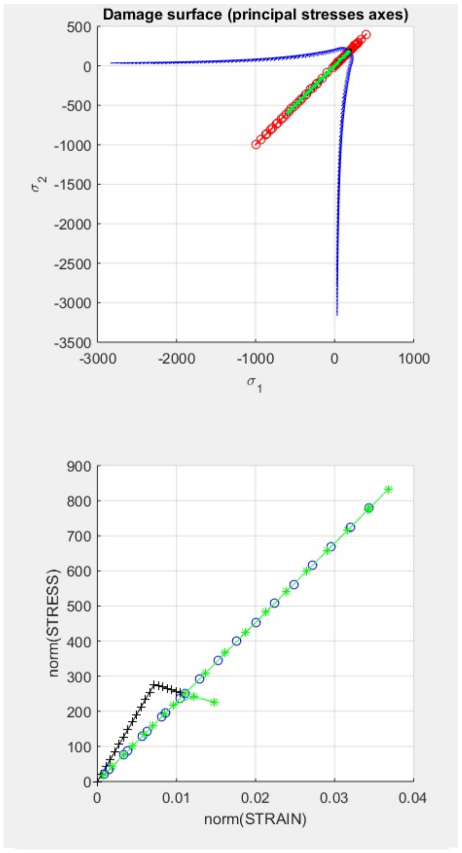
(b) Exponential

Figure 9: Case 3 for a Symemtric Model

The results in Figure 9 for the symmetric model are logical because the yield stress is the same after unloading and after the compressive loading.



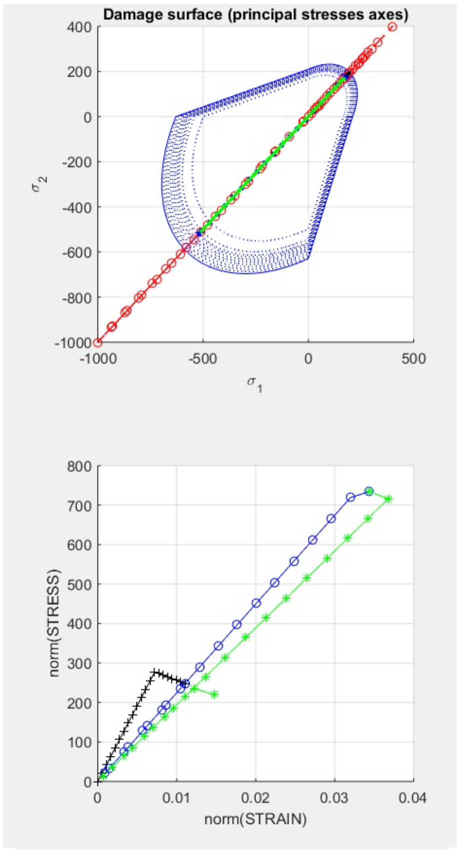
(a) Linear



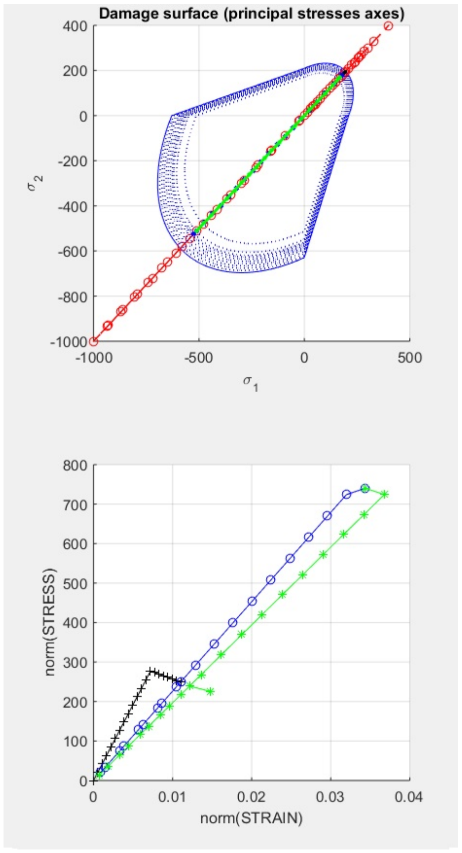
(b) Exponential

Figure 10: Case 3 for a Tension-Only Model

In the case of an only tension model (Figure 10), it is visible that after the compressive loading there is no damage, and there is no yield stress while applying a compressive load.



(a) Linear



(b) Exponential

Figure 11: Case 3 for a Non-Symemtric Model

In the case of a non-symmetric model, it is visible that the model is damage at a higher yield stress after applying the compressive load.

2 Part 2: Rate Dependent Models

In the second part, the viscosity and the integration constant is considered only for the symmetric model with a linear hardening law. For the next studies, one uniaxial loading path was taken which is $\sigma_1 = 600$, with a hardening constant of $H = 0.2$. All other constants are kept the same fron Part 1.
Three new variables are added to the previous part:

- η: Viscosity ($\eta \geq 0$)
- t: Time ($t > 0$)
- α: Integration Constant ($0 \leq \alpha \leq 1$)

2.1 Viscosity and its Effect

The viscosity of the material η affects the way the model is damaged. Therefore different values for the viscosity are taken and tested for the same given stress. Note that the additional constants taken are $t = 3.33s$ and $\alpha = 1$

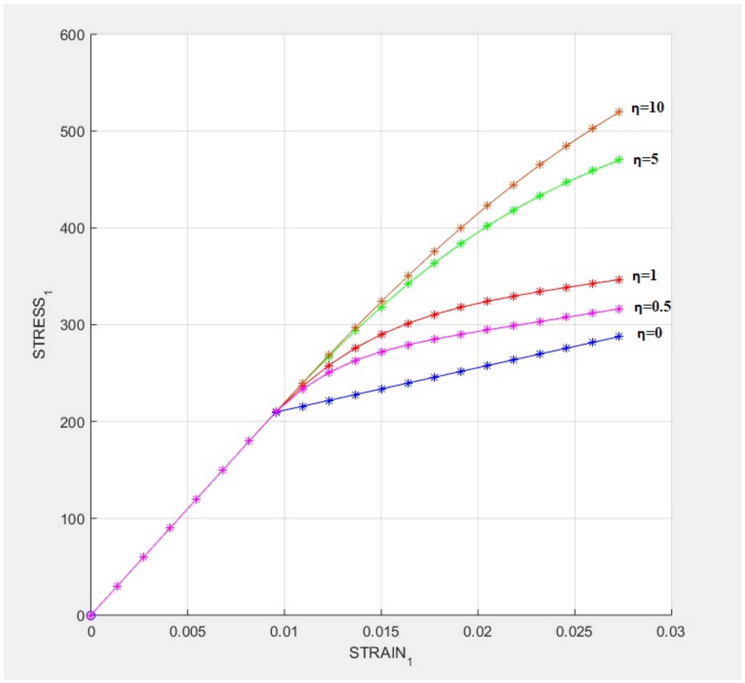


Figure 12: Stress-Strain Curve for Different Values of the Viscosity

As shown in Figure 12, if the viscosity is equal to 0, it is similar to an inviscid model (rate independent model) which confirms the correctness of the application of the viscosity. Referring to the graph above, the higher the viscosity, the more time it needs to get to the linear damage phase. Therefore one can deduct that when $\eta \to \infty$, the model will become fully elastic.

2.2 Time and its Effect

The time t also affects the way the model is damaged. Therefore different values for the time are taken and tested for the same given stress. Note that the additional constants taken are $\eta = 1$ and $\alpha = 1$

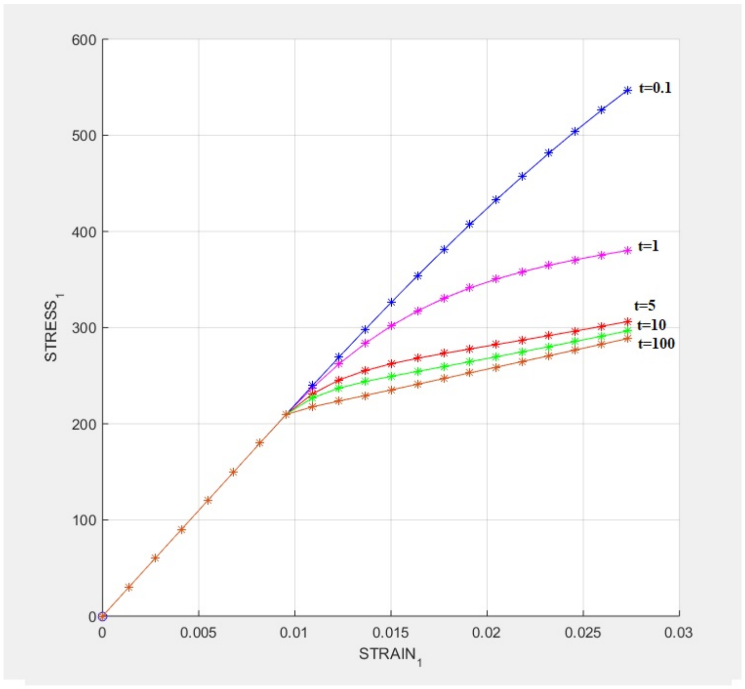


Figure 13: Stress-Strain Curve for Different Values of Time

As Figure 13 shows, as $t \to \infty$, the model acts like an inviscid model, because the increments of the stress are so small with respect to time, which neglects the effect of the viscosity. It also shows that when $t \to 0$, the model acts like a fully elasctic model.

2.3 Integration Constant α and its Effect

The integration constant α affects how the calculations are made. Therefore different values for this constant are taken and tested for the same given stress. Note that the additional constants taken are $\eta = 0.5$ and $t = 50$

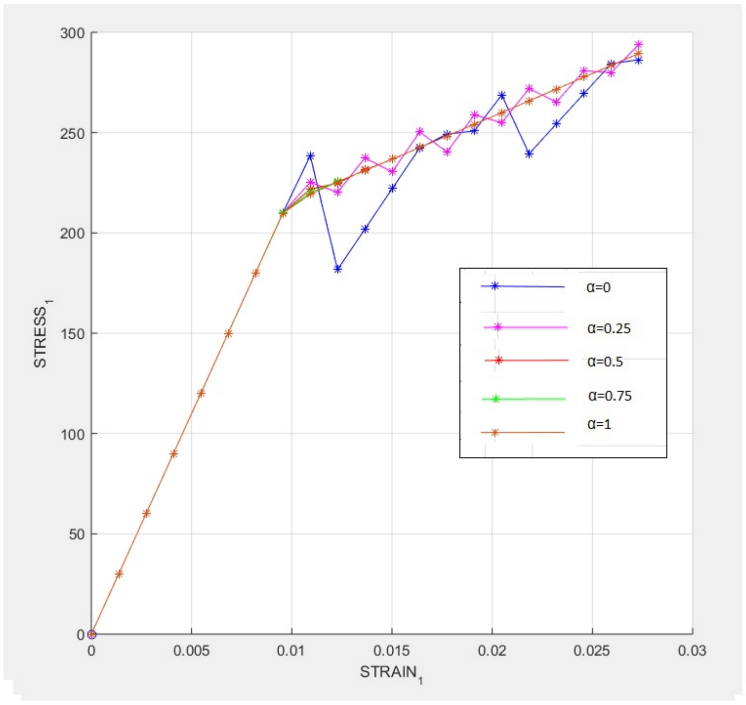


Figure 14: Stress-Strain Curve for Different Values of the Integration Constant α

As shown in Figure 14, when $\alpha < 0.5$, the integration is not stable therefore the answers are not accurate. However when $\alpha \geq 0.5$, the integration is stable and more or less the same. Therefore it is better to use Backward Euler ($\alpha = 1$) and Crank-Nicholson ($\alpha = 0.5$) for rate dependent models.

3 Appendix: Matlab Codes

In this section, the codes that are edited are shown below.

```
function [sigma_v,vartoplot,LABELPLOT,TIMEVECTOR]=damage_main(Eprop,
    ntype,istep,strain,MDtype,n,TimeTotal)
global hplotSURF
% SET LABEL OF "vartoplot" variables  (it may be defined also outside
    this function)
% ----------------------------------
 LABELPLOT = {'hardening variable (q)','internal 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
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
% -------------
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 = 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 = 3 ;
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)

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,:) ;
        eps_n = strain(i-1,:) ;
        % ********************************************************************
        %*       DAMAGE MODEL
        % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        [sigma_n1,hvar_n,aux_var] = rmap_dano1(eps_n1,eps_n,hvar_n,Eprop,ce,MDtype,n,delta_t);
        % 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)
    end
end
```

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

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];
    %* 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) 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);

elseif MDtype==2
     tetha=[-pi/2+0.01:0.01:pi-0.01];
    %* 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
        m1t=m1(i)*(m1(i)>0);
        m2t=m2(i)*(m2(i)>0);
        radio(i)= q/sqrt([m1t m2t 0 nu*(m1t+m2t)]*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
    tetha1=[0:0.01:pi/2];
    tetha2=[pi:0.01:3*pi/2];
    tetha3=[2*pi];
    tetha=[tetha1 tetha2 tetha3];
    %* 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
        if tetha(i)<=(3*pi/2) && tetha(i)>=pi
        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)*n;
        s2(i)=radio(i)*m2(i)*n;
        else
        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
    end
    hplot =plot(s1,s2,tipo_linea);
end
    axis equal square;
return
```

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

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

elseif (MDtype==2)  %* Only tension
eps_pos=eps_n1.*(eps_n1>0);
rtrial= sqrt(eps_pos*ce*eps_n1');

elseif (MDtype==3)  %*Non-symmetric
    sigma=ce*eps_n1';
    sigma_pos=sigma.*(sigma>0);
    sigma_abs=abs(sigma);
    S_pos=sum(sigma_pos);
    S_abs=sum(sigma_abs);
    theta=S_pos/S_abs;
rtrial= (theta+(1-theta)/n)*sqrt(eps_n1*ce*eps_n1') ;
end

return
```

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

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);
viscpr    = Eprop(6);
eta       = Eprop(7);
ALPHA     = Eprop(8);
%*       initializing
 r0 = sigma_u/sqrt(E);
 zero_q=1.d-6*r0;
%*       Damage surface
[rtrial_n1] = Modelos_de_dano1 (MDtype,ce,eps_n1,n);
%*   ---------->    fload=0 : elastic unload
%*   ---------->    fload=1 : damage (compute algorithmic constitutive
    tensor)
fload=0;
if viscpr==0 %If inviscid
    if(rtrial_n1 > r_n)
        %*   Loading
        fload=1;
        delta_r=rtrial_n1-r_n;
        r_n1= rtrial_n1  ;
        if hard_type == 0
            %  Linear
            q_n1= q_n+ H*delta_r;
        else
            % Exponentianl
            q_inf=2*r0-zero_q;
            q_n1=q_n+ (H*((q_inf-r0)/r0)*exp(abs(H)*(1-(rtrial_n1/r0))))
                *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
else    %If viscous(+add the integration constant)
    [rtrial_n] = Modelos_de_dano1 (MDtype,ce,eps_n,n);
    rtrial=rtrial_n*(1-ALPHA)+rtrial_n1*ALPHA;
    if(rtrial > r_n)
        %*   Loading
        fload=1;
        r_n1= ((eta-(1-ALPHA)*delta_t)*r_n+delta_t*rtrial)/(eta+ALPHA*
            delta_t);
        delta_r=r_n1-r_n;
        if hard_type == 0
            %  Linear
            q_n1= q_n+ H*delta_r;
        else
            % Exponentianl
            q_inf=2*r0-zero_q;
            q_n1=q_n+ (H*((q_inf-r0)/r0)*exp(abs(H)*(1-(rtrial/r0))))*
                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
end
% Damage variable
% ---------------
dano_n1   = 1.d0-(q_n1/r_n1);
%  Computing stress
%  ***************
sigma_n1  =(1.d0-dano_n1)*ce*eps_n1';
%hold on
%plot(sigma_n1(1),sigma_n1(2),'bx')
%* 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;
```