Universitat Politècnica de Catalunya
Numerical Methods in Engineering
Computational Solid Mechanics

J2 computational plasticity

Eduard Gómez
April 29, 2020

Contents

1 Perfect plasticity

We'll start off perfect plasticity. Much like on the case for one-dimensional plasticity, plastic deformation offers no resistance, hence stress will be bounded by the yield stress. Unlike 1D, however, stress and strain are tensors, and while strain might be free unopposed in one direction, it will still act plastically in another. More specifically, with a J2 model plasticity only affects the deviatoric component of strain and stress, while the spherical component will always be in the elastic domain.

For this reason we must ensure that our imposed strain path is non-spherical. For simplicity purposes, I chose the following path:

$$\begin{array}{lllllllll} \varepsilon & : & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \rightarrow & 10^{-3}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \rightarrow & -10^{-3}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \rightarrow & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ t & : & 1.0s & & 2.0s & & 3.0s & & 4.0s \end{array} \tag{1}$$

The major expected difference between inviscid and viscous plasticity is the curvature of the stress-strain path within the plastic domain, as well as the fact that the viscous stress is unbounded, unlike its inviscid counterpart which must stay inside the yield surface.

According to the theory, the stess path must be as shown in this figure. In following sections we'll make references to the geometrical properties of the yield surface, which here we can see is a cylinder.

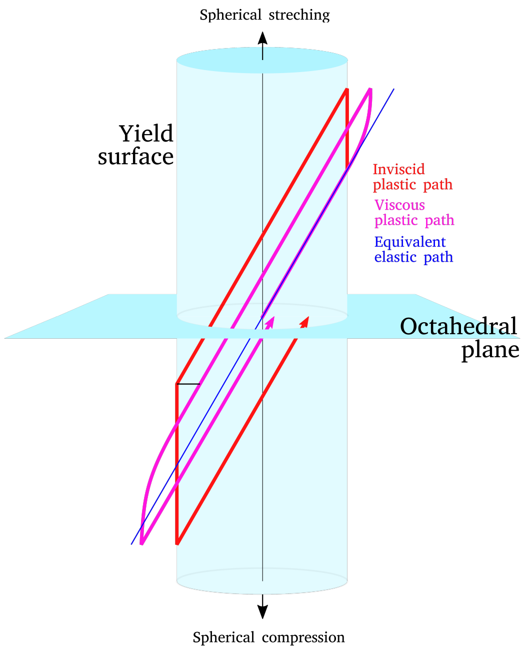
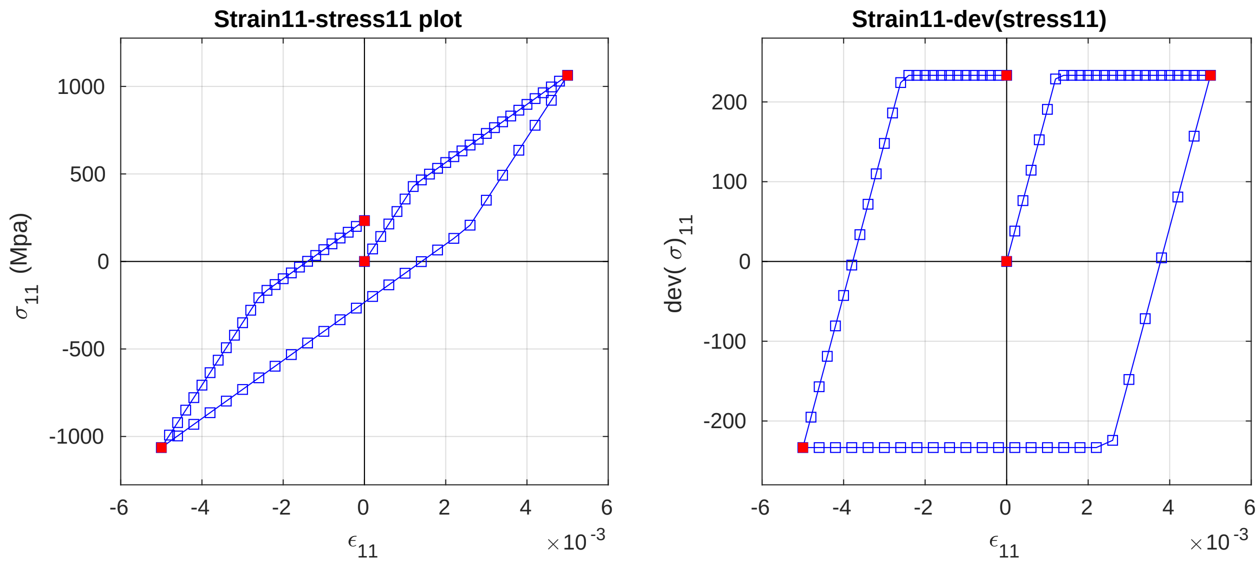


Figure 1: Expected stress paths for perfect plasticity. Note that the cylinder extends to infinity.

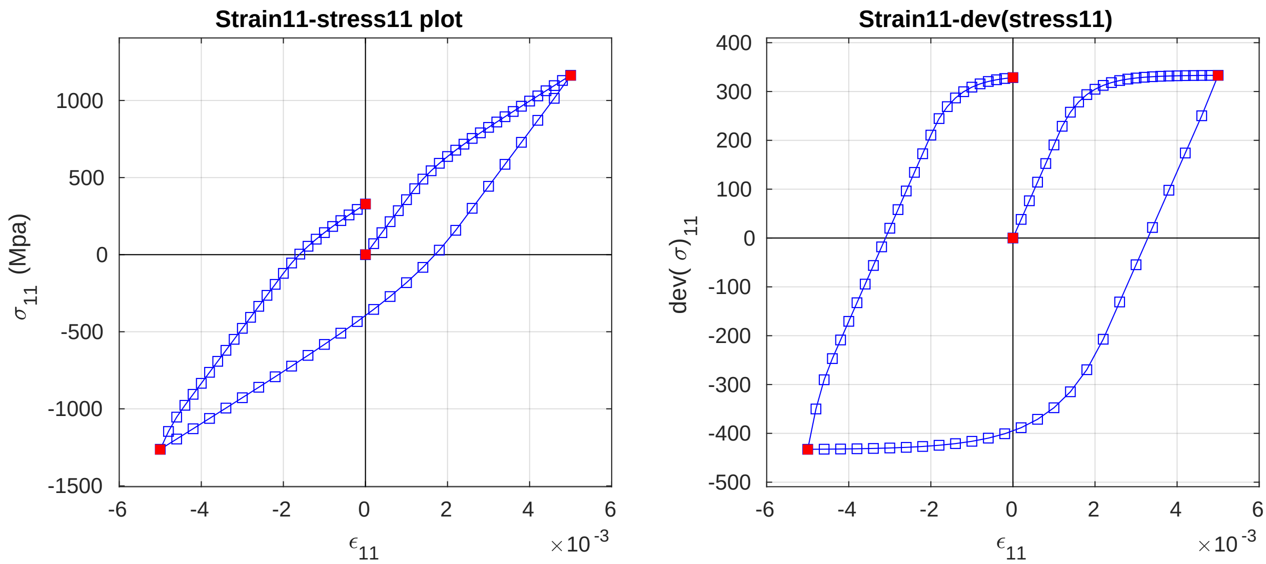
And the result is in figure 2. Further analysis of the effects of η and load rate is explored later in section 4.

Inviscid perfect plasticity



$E = 200$ GPa $\quad \nu = 0.25 \quad \sigma_y = 350$ Mpa $\quad H = K = \eta = 0$

Viscous perfect plasticity



$E = 200$ GPa $\quad \nu = 0.25 \quad \sigma_y = 350$ Mpa $\quad \eta = 10$ MPa·s $\quad H = K = 0$

Figure 2: Perfect plasticity analysis

2 Linear kinematic plasticity

This type of plasticity causes the yield surface to move. This can be represented by considering $\bar{\boldsymbol{q}}$ the centerline of the yield cylinder. The radius of the cylinder, however, will remain constant. This means that stress is now unbounded, since it pushes the cylinder around. In the viscous case, the stress can move outside the cylinder and "pull it" towards itself. In any case, what we expect in the dev $\boldsymbol{\sigma} - \varepsilon$ diagram is that the difference between stretch and compression yield stresses be constant, at least for the inviscid case.

Figure 4 shows the strain-stress plots when following the path outline in equation 5. We can see how the cylinder radius does not expand in figure 3. We define the yield radius as R_y and the stress radius as R_σ:

$$R_y = \sqrt{\frac{2}{3}}(\sigma_y - q) \tag{2}$$

$$R_\sigma = ||\text{dev}\,\boldsymbol{\sigma} - \bar{\boldsymbol{q}}|| \tag{3}$$

In this plot kinematic plasticity looks the same as perfect plasticity, as would be expected. Note that figure 3 uses the same physical properties shown in figure 4.

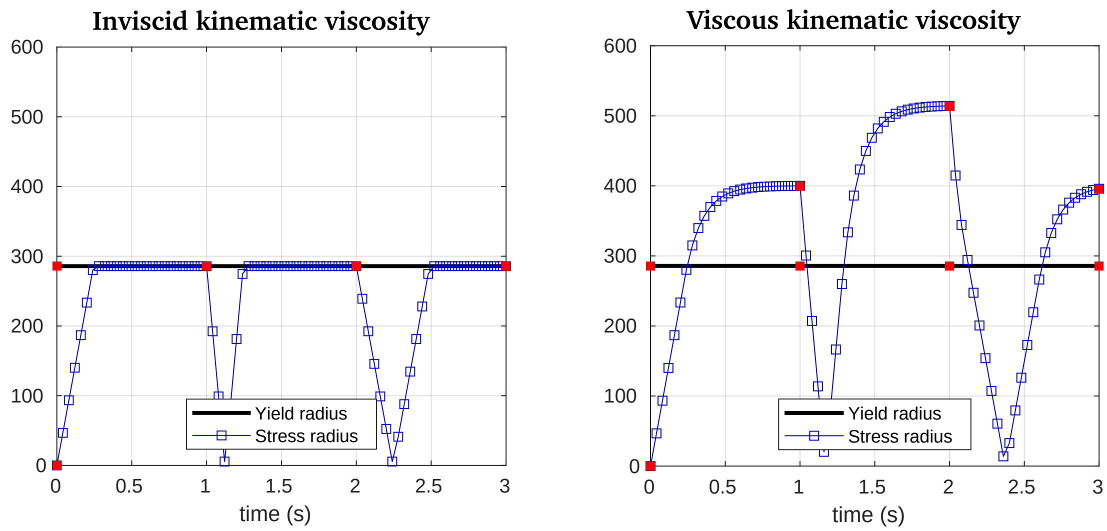
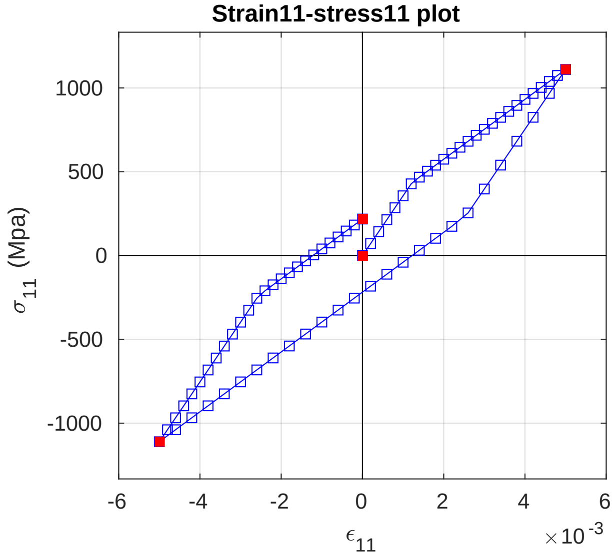
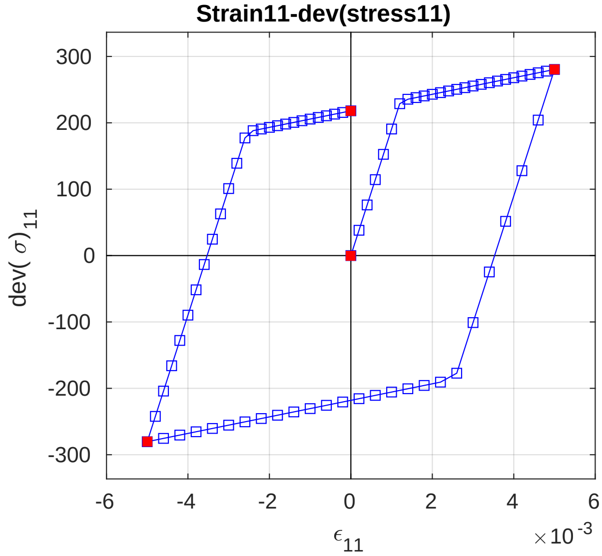


Figure 3: Radii of stress and yield cylinder for kinematic viscosity

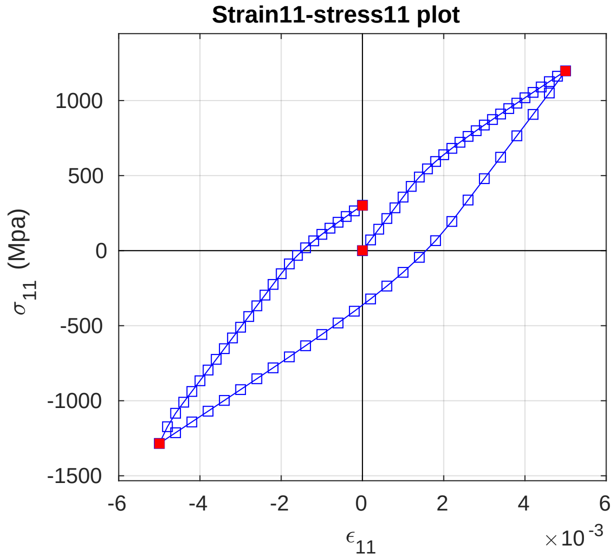
Inviscid kinematic plasticity

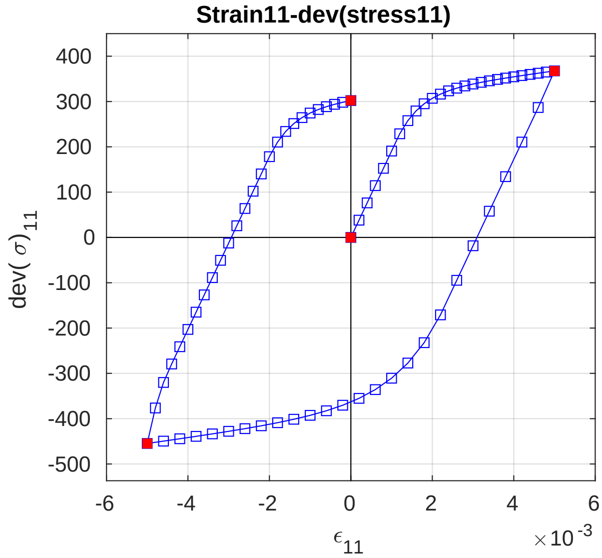




$E = 200$ GPa $\nu = 0.25$ $\sigma_y = 350$ Mpa $K = 30$ MPa $H = \eta = 0$

Viscous kinematic plasticity





$E = 200$ GPa $\nu = 0.25$ $\sigma_y = 350$ Mpa $K = 30$ MPa $\eta = 30$ MPa·s $H = 0$

Figure 4: Kinematic plasticity analysis

3 Isotropic plasticity

3.1 Linear isotropic plasticity

This type of plasticity maintains the centerline of the yield cylinder static but changes its radius. This expansion is manifested with the introduction of q, which expresses the addition of yield stress to σ_y. The expansion of the radius is shown in equation 2. Keep in mind that q is negative.

Once again we use the strain path shown in equation 5. The effect on the radius is clear in figure 5. The stress-strain plots are shown in figure 6.

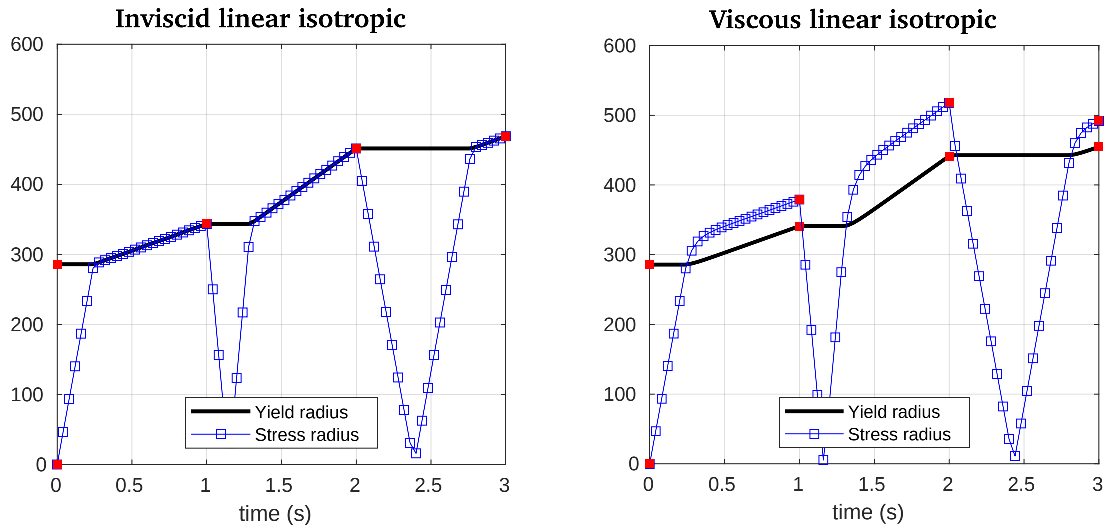
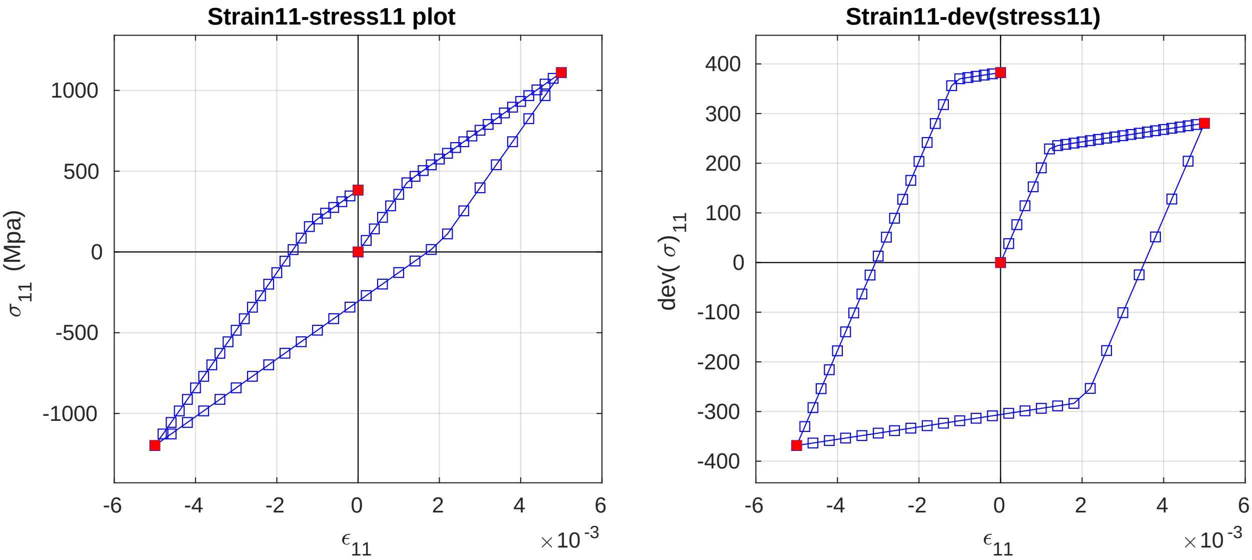


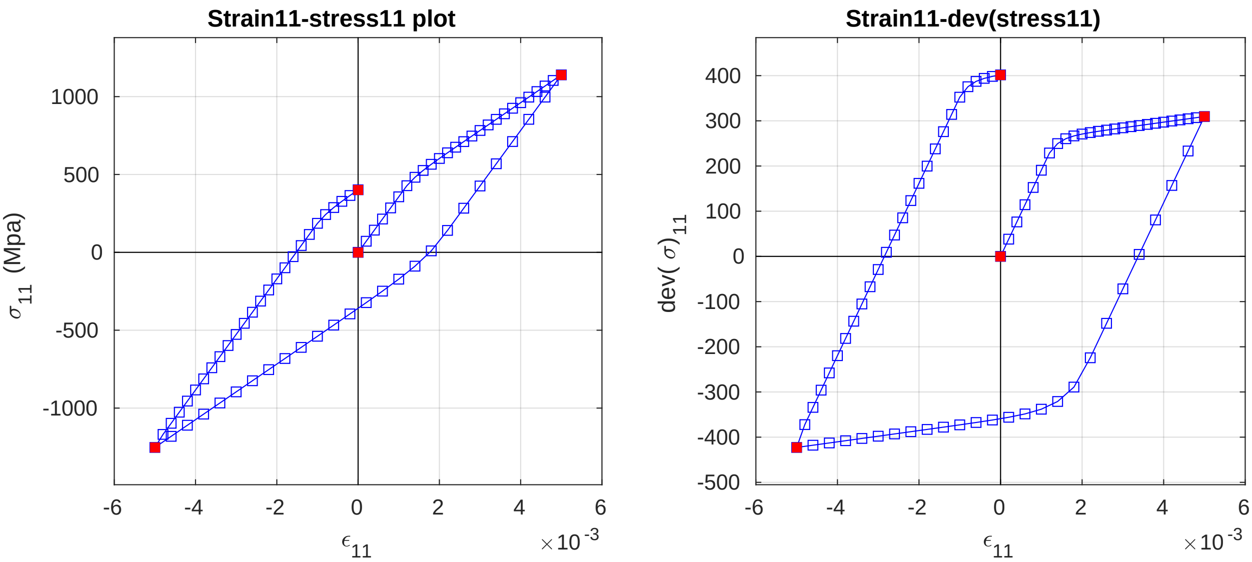
Figure 5: Radii of stress and yield cylinder for linear isotropic viscosity

Inviscid linear isotropic plasticity



$E = 200$ GPa $\quad \nu = 0.25 \quad \sigma_y = 350$ Mpa $\quad K = 30$ MPa $\quad H = \eta = 0$

Viscous linear isotropic plasticity



$E = 200$ GPa $\quad \nu = 0.25 \quad \sigma_y = 350$ Mpa $\quad K = 30$ MPa $\quad \eta = 10$ MPa·s $\quad H = 0$

Figure 6: Kinematic plasticity analysis

3.2 Non-linear isotropic plasticity

For a more exhaustive simmulation of the effects of viscosity, we can introduce a non-linear viscous term. In particular, we use an exponential model:

$$q = -(\sigma_\infty - \sigma_y)\exp(-\delta\xi) + K\xi \tag{4}$$

the linear term is the same as in the previus section, however we introduced two new ones. σ_∞ is an assymptotic value and δ is the speed at which it is approached. Once $\sigma \approx \sigma_\infty$, the underlying linear (or perfect) model resurfaces. In this calculation we set the value of $K = 0$ to highlight this effect.

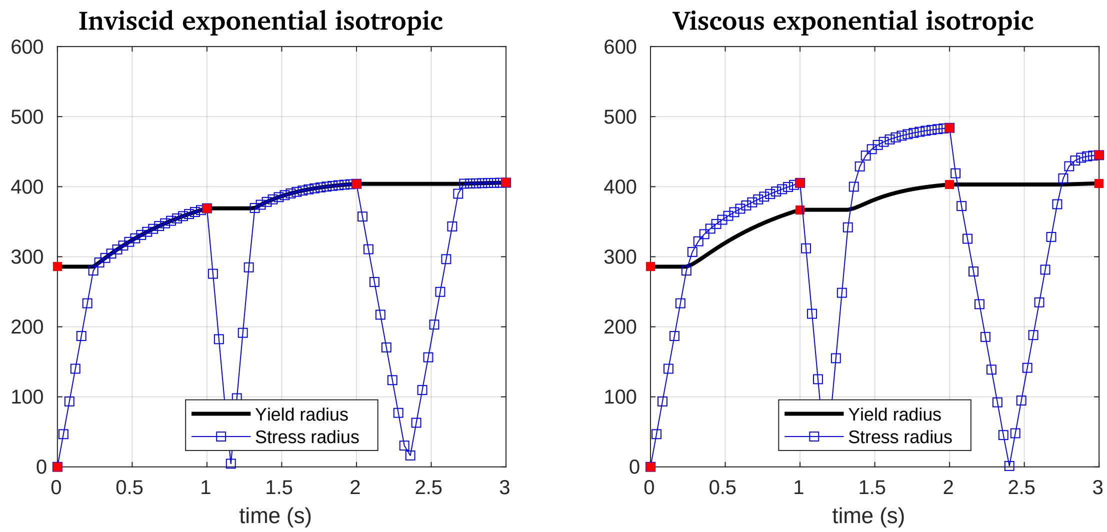
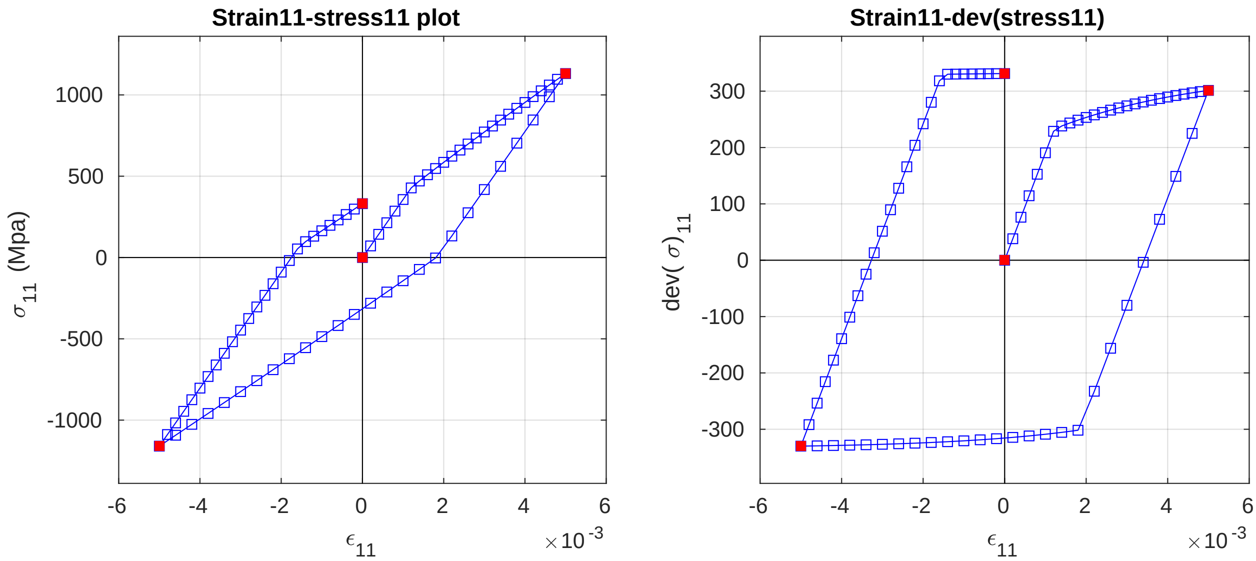


Figure 7: Radii of stress and yield cylinder for exponential isotropic viscosity

Figure 7 shows the expansion of the radius. Notice that the assymptote apears to be close to 400 MPa, however it is set at $\sigma_\infty = 500$ MPa. This is because the radius has a factor of $\sqrt{2/3}$ with stress. This puts the radius of σ_∞ at about 408 MPa, which matches the visual result much better. This also explains why despite working with $\sigma_y = 350$ MPa, it appears to be bellow 300 at $t = 0$. Its radius is actualy only 286 MPa.

Finally, the stress-strain results are shown in figure 8, where we see how the stress-strain paths curve even in the inviscid case.

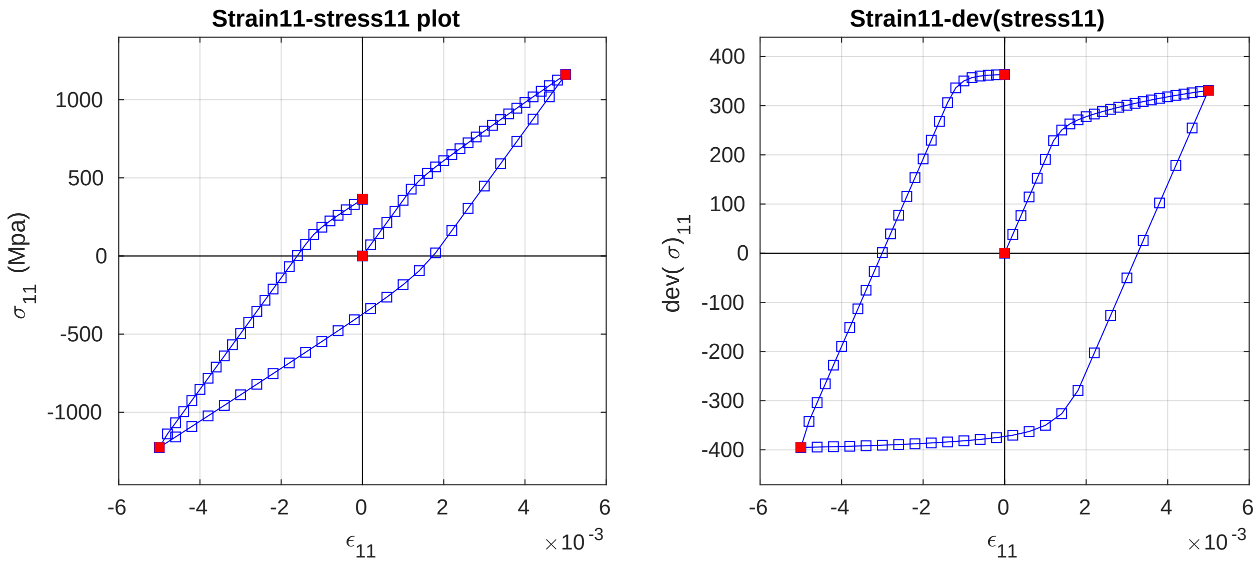
Inviscid exponential isotropic plasticity



$E = 200$ GPa $\nu = 0.25$ $K = H = \eta = 0$

$\sigma_y = 350$ Mpa $\sigma_\infty = 500$ MPa $\delta = 500$

Viscous exponential isotropic plasticity



$E = 200$ GPa $\nu = 0.25$ $\eta = 10$ MPa·s $K = H = 0$

$\sigma_y = 350$ Mpa $\sigma_\infty = 500$ MPa $\delta = 500$

Figure 8: Kinematic plasticity analysis

4 Viscosity

So far we've seen the different viscosity models but we have not compared the effect of viscosity or load rate. This is studied in this section. We'll work with a perfect model to decouple the effects of plasticity models with those of plasticity. To start we'll change the strain path to one more apt to show the effects of viscosity:

$$\begin{array}{ccccccc} \varepsilon & : & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \rightarrow & 10^{-3}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \rightarrow & 10^{-3}\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \\ t & : & t_0 & & t_0 + \Delta t & & t_0 + 2\Delta t \end{array} \tag{5}$$

That is, it goes from rest to stressed in Δt, and then it stays there for one more period of length Δt. We now have this time step as a way to control for load rate. This way we see the effects during loading and during relaxation. The effects during unloading are similar to those during relaxation, so there is no need to lengthen the strain path even further.

4.1 Load-rate comparison

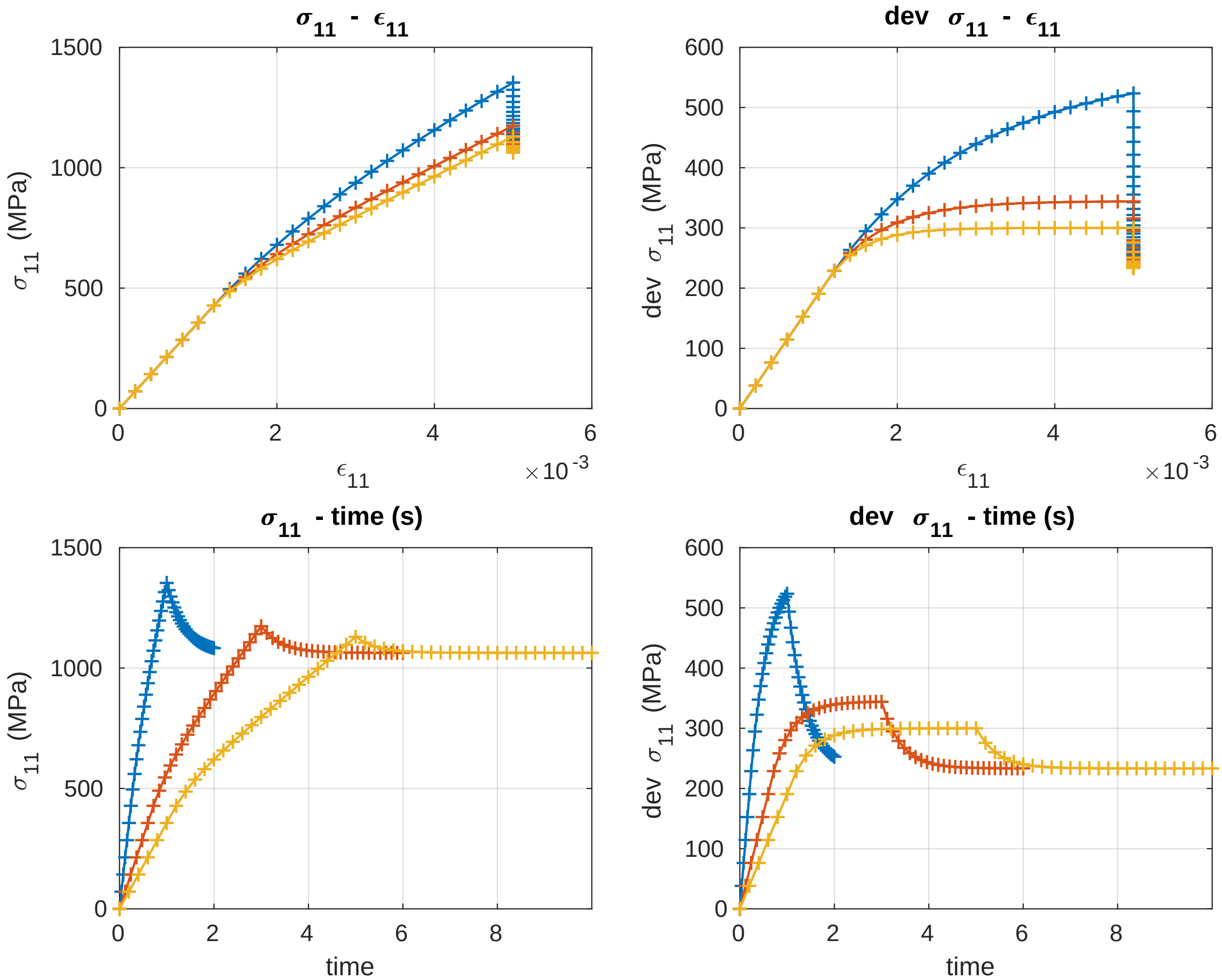
Because the effects of viscosity are more pronounced with higher $\dot{\varepsilon}$, we expect smaller values of Δt to have more noticeable effects. Figure 9 proves this intuition correct. We see how stress peaks at higher values for lower values of Δt. We also see that during relaxation they all approach the same value, which about $300MPa$ in the deviatoric space.

4.2 Viscosity comparisson

The effect of the viscosity is reciprocal to that of Δt. Hence we would expect the effects of viscosity to become more apparent for larger viscosity parameters. Similarly, we would expect smaller viscosity parameters to approach the inviscid case.

Figure 10 shows that is the case. The case for $\eta = 1$ MPa·s is almost indistinguishable from the inviscid case; in all subplots the stress peak is hard to find. Inviscid plasticity makes it so there is no peak stress.

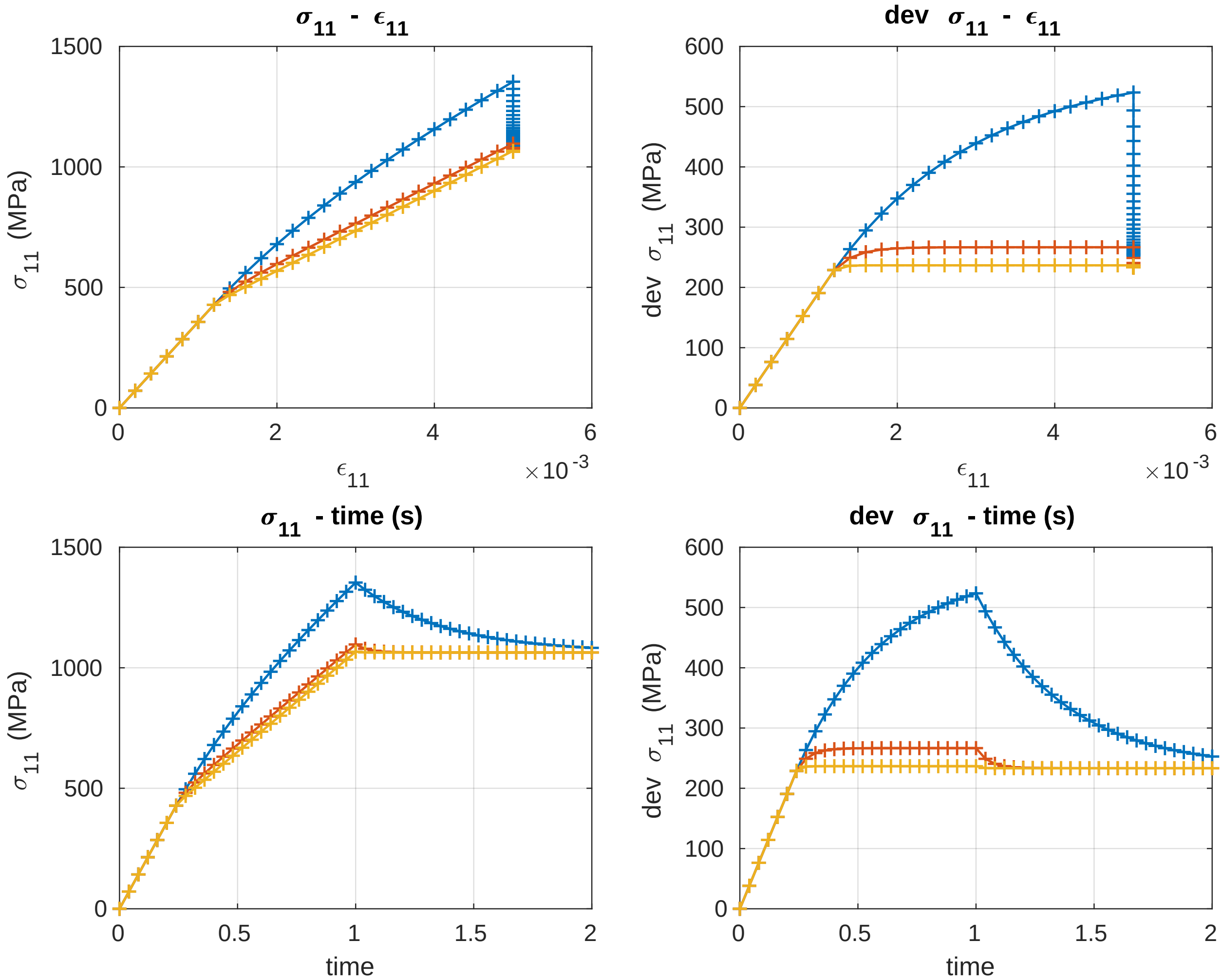
Blue	**Red**	**Yellow**
$\Delta t = 1$s	$\Delta t = 3$s	$\Delta t = 5$s



$E = 200$ GPa $\qquad \nu = 0.25 \qquad \sigma_y = 350$ Mpa $\qquad \eta = 100$ MPa·s $\qquad H = K = 0$

Figure 9: Comparison of the same process at different time scales

Blue	**Red**	**Yellow**
$\eta = 100$ MPa $\cdot$ s	$\eta = 10$ MPa $\cdot$ s	$\eta = 1$ MPa $\cdot$ s



$E = 200$ GPa $\nu = 0.25$ $\sigma_y = 350$ Mpa $\Delta t = 1$s $H = K = 0$

Figure 10: Comparison of the same process with different viscosities

5 Appendix

Here the code is appended, except for the post-processing routines since they are only there for aesthetic purposes.

5.1 Main file

main.m

```
clearvars -except results
close all;
%% Data entry
% Material properties
mat.kappa = 166e3;
mat.mu = 143e3;
mat.yield = 350;
mat.K = 0e3;
mat.H = 0e3;
mat.visc = 1e3;

% Strain path
steps_per_trip = 25;
time_per_trip = 1;

% Hardening
hardening.is_linear = true;

mat.sigma_infty = 500;
mat.delta = 500;

hardening.maxIter = 30;
hardening.tol = 1e-10;

%% Pre-processing
% Computing full strain path
strain = set_strain_path(steps_per_trip);

% Support variables
n_steps = length(strain);
dt = time_per_trip / steps_per_trip;
mat = generalized_contitutive_tensor(mat);

% Initialization of arrays

Strain_p = cell(3,n_steps);
Strain_p{1,1} = zeros(3,3); % Strain tensor
Strain_p{2,1} = 0;          % xi
Strain_p{3,1} = zeros(3,3); % xi bar

```

```
Stress = cell(3,n_steps);
Stress{1,1} = zeros(3,3); % Cauchy stress tensor
Stress{2,1} = 0;          % q
Stress{3,1} = zeros(3,3); % q bar

S_trial = cell(1,3);
elastoplastic_modulus = cell(n_steps,1);
elastoplastic_modulus{1} = mat.D;

%% Computation
for i = 2:n_steps
    % Trial stress
    S_trial{1} = product42(mat.C{1}, (strain{i} - Strain_p{1,i-1}));
    S_trial{3} = - mat.C{3} * Strain_p{3,i-1};

    if hardening.is_linear
        S_trial{2} = - mat.C{2} * Strain_p{2,i-1};
    else
        S_trial{2} = Stress{2,i-1};
    end

    % Trial yield function
    radial_tensor = deviatoric(S_trial{1}) - S_trial{3};
    stress_radius = sqrt(sum(radial_tensor.^2,'all')); % Euclidean norm
    f_trial = stress_radius - sqrt(2/3) * (mat.yield - S_trial{2});

    if f_trial < 0
        % Elastic load/unload
        for k=1:3
            Strain_p{k,i} = Strain_p{k,i-1};
            Stress{k,i} = S_trial{k};
        end
        elastoplastic_modulus{i} = mat.D;
    else
        % Plastic load
        normal = radial_tensor / stress_radius;
        gamma = calc_hardening(hardening, mat, f_trial, Strain_p{2,i-1}, dt);

        % Strain update
        Strain_p{1,i} = Strain_p{1,i-1} + gamma * normal;
        Strain_p{2,i} = Strain_p{2,i-1} + gamma * sqrt(2/3);
        Strain_p{3,i} = Strain_p{3,i-1} - gamma * normal;

        % Stress update
        Stress{1,i} = S_trial{1} - gamma * mat.mu * 2 * normal;
        Stress{3,i} = S_trial{3} + gamma * mat.H * 2/3 * normal;

        if hardening.is_linear
            Stress{2,i} = S_trial{2} - gamma * mat.K * sqrt(2/3);
```

```
        else
            Stress{2,i} = - Pi(Strain_p{2,i},mat,1);
        end

        % Elastoplastic modulus
        elastoplastic_modulus{i} = get_elastoplastic(hardening, mat, Strain_p{2,i}, ...
                                                     dt, gamma, normal, stress_radius);
    end
end

run post_processing
```

5.2 Choice of stress path

set_strain_path.m

```
function strain = set_strain_path(steps_per_trip)
    % First corner must be zeros(3).
    corners{1} = zeros(3);

    %% Customizeable code
    a = 0.005;

    corners{2} = zeros(3);
    corners{2}(1,1) = a;

    corners{3} = corners{2};

    %% Computation of path
    n_steps = steps_per_trip * (length(corners)-1) + 1;
    strain = cell(1,n_steps);
    strain{1} = zeros(3);
    s = 1;
    dx = 1 / steps_per_trip;
    for i = 1:length(corners)-1
        x = 0;
        while x<1
            strain{s} = (1-x)*corners{i} + x*corners{i+1};
            s = s+1;
            x = x+dx;
        end
    end
    strain{end} = corners{end};
end
```

5.3 Obtaining gamma

calc_hardening.m

```
function gamma = calc_hardening(hardening, mat, f_trial, xi, dt)
    if hardening.is_linear
        gamma = f_trial / (2*mat.mu + 2/3*mat.K + 2/3*mat.H + mat.visc/dt);
    else
        gamma = 0;
        for k=1:hardening.maxIter
            g = f_trial - gamma * (2*mat.mu + 2/3* mat.H + mat.visc/dt) ...
                - sqrt(2/3)*(Pi(xi+sqrt(2/3)*gamma,mat,1) - Pi(xi,mat,1));
            if abs(g) < hardening.tol
                break;
            end
            Dg = -(2*mat.mu + 2/3*mat.H + mat.visc/dt ...
                        + 2/3*Pi(xi+sqrt(2/3)*gamma,mat,2));
            gamma = gamma - g/Dg;
        end
        if(k==hardening.maxIter)
           warning(['Maximum number of iterations reached' ...
                     'before convergence. Error %f'],abs(g))
        end
    end
end
```

5.4 Non-linear isotropic hardening

Pi.m

```
function z = Pi(xi, mat, derivative)
    switch derivative
        case 1
            % Pi'(xi)
            z = (mat.sigma_infty - mat.yield) * (1 -exp(-mat.delta * xi))...
                + mat.K*xi;
        case 2
            % Pi"(xi)
            z = mat.delta * (mat.sigma_infty - mat.yield)...
                * exp(-mat.delta*xi) + mat.K;
        otherwise
            error('Only 1st and 2nd derivatives are implemented');
    end
end
```

5.5 Assembly of the constitutive tensor

generalized_contitutive_tensor.m

```
function mat = generalized_contitutive_tensor(mat)
    global I2_x_I2 I4 % They will be reused to obtain the elastoplastic tensor
    I2_x_I2 = zeros(3,3,3,3); % 3x3 identity texsor outer product'd with itself
    I4 = zeros(3,3,3,3);      % 4rth order identity tensor
    for i=1:3
        I4(i,i,i,i) = 1;
        for j=1:3
            I2_x_I2(i,i,j,j) = 1;
        end
    end
    mat.D = mat.kappa * I2_x_I2 + 2*mat.mu*(I4 - 1/3 * I2_x_I2);
    mat.C{1} = mat.D;
    mat.C{2} = mat.K;
    mat.C{3} = mat.H *2/3*eye(3);
end
```

5.6 Elastoplatic modulus

get_elastoplastic.m

```
function D = get_elastoplastic(hardening, mat, xi, dt, gamma, normal, stress_radius)
    delta = 1 - 2 * mat.mu * gamma / stress_radius;
    if hardening.is_linear
        delta_bar = 2 * mat.mu / ...
                    ( 2*mat.mu + 2/3*mat.K + 2/3*mat.H + mat.visc/dt)...
                    - (1 - delta);
    else
        delta_bar = 2*mat.mu / ...
                    (2*mat.mu + 2/3*Pi(xi,mat,2) + 2/3*mat.H + mat.visc/dt)...
                    -(1 - delta);
    end
    global I2_x_I2  I4 %Recycling them to avoid re-calculating them every iteration
    outer_nn = zeros(3,3,3,3); % n (x) n
    for i=1:3
        for j=1:3
            outer_nn(:,:,i,j) = normal * normal(i,j);
        end
    end
    D =   mat.kappa * I2_x_I2 + 2*mat.mu * delta * (I4 - I2_x_I2/3) ...
        - 2*mat.mu * delta_bar * outer_nn;
end
```

5.7 Deviatoric

deviatoric.m

```
function [D, S] = deviatoric(A)
    % Works only on 3x3 matrices
    % Returns the deviatoric and spherical tensors
    S = 1/3 * trace(A) * eye(3);
    D = A - S;
end
```

5.8 Tensor double-dot product

product42.m

```
function C = product42(A, B)
    % Tensor product C = A:B
    % - A is a 4th order tensor
    % - B is a 2nd order tensor
    % - C_ij = A_ijkm*B_km
    C = zeros(3);
    for k=1:3
        for m=1:3
            C = C + A(:,:,k,m) * B(k,m);
        end
    end
end
```