

Finite elastic-plastic deformations
(BMEGEMMDKPL)
II. Homework

Szász Zsolt
KRCH5Q

May 3, 2025

Loading

We have an $L \text{ mm} \times L \text{ mm} \times L \text{ mm}$ brick element ($L = 1 \text{ mm}$), which node numbering is shown on Figure 1.

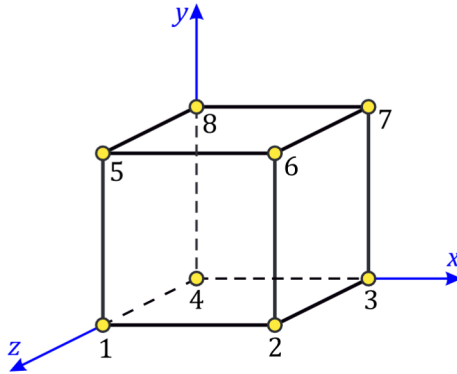


Figure 1: Nodal layout of the brick element.

We have a prescribed displacements on the upper nodes, defined by displacement vector as

$$[\mathbf{U}_1] = [\mathbf{U}_2] = [\mathbf{U}_3] = [\mathbf{U}_4] = [0 \ 0 \ 0]^T, \quad (1)$$

$$[\mathbf{U}_5] = [\mathbf{U}_6] = [\mathbf{U}_7] = [\mathbf{U}_8] = [u_x \ u_y \ 0]^T, \quad (2)$$

where the time evolution of u_x and u_y and their time derivatives are shown on Figure 2.

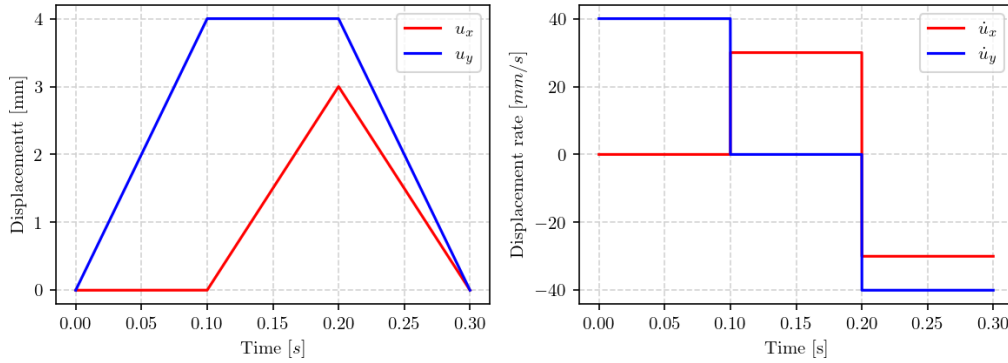


Figure 2: The parameters of the loading

From the displacement vectors we can determine the deformations gradient regarding to that motion.

$$[\mathbf{F}] = \begin{bmatrix} 1 & u_x/L & 0 \\ 0 & 1 + u_y/L & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

For the further calculations we will need the velocity gradiend, which can be calculated from the deformation gradient as

$$[l] = [\dot{\mathbf{F}}\mathbf{F}^{-1}] = \begin{bmatrix} 0 & \dot{u}_x & 0 \\ 0 & \dot{u}_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -\frac{u_x}{1+u_y} & 0 \\ 0 & \frac{1}{1+u_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \frac{1}{1+u_y} \begin{bmatrix} 0 & \dot{u}_x & 0 \\ 0 & \dot{u}_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4)$$

The rate of deformation (\mathbf{d}) is the symmetric and the spin tensor (\mathbf{w}) is the skew symmetric part of the velocity gradient:

$$\mathbf{d} = \frac{1}{2}(\mathbf{l} + \mathbf{l}^T) \quad \& \quad \mathbf{w} = \frac{1}{2}(\mathbf{l} - \mathbf{l}^T). \quad (5)$$

Material behaviour

This element has an elastic-plastic behaviour. We have finite deformations, therefore we need to use hypoelastic-plastic modelling approach, which is based on the additive decomposition of the rate of deformation tensor into elastic and plastic part

$$\mathbf{d} = \mathbf{d}^e + \mathbf{d}^p. \quad (6)$$

Following the ABAQUS formulation, the elastic behaviour can be defined by the Hooke's law as follows

$$\dot{\boldsymbol{\sigma}} = \frac{E}{1+\nu} \left(\mathbf{d}^e + \frac{\nu}{1-2\nu} (\text{tr} \mathbf{d}^e) \mathbf{I} \right) := \mathbb{D} : \mathbf{d}^e, \quad (7)$$

where the $\dot{\boldsymbol{\sigma}}$ denotes the Jaumann rate of the Cauchy stress tensor. The discrete form of the Hooke's law in the rotated coordinate system looks like

$$\tilde{\boldsymbol{\sigma}}_{n+1} = \tilde{\boldsymbol{\sigma}}_n + \mathbb{D} \Delta \tilde{\boldsymbol{\epsilon}}^e \quad (8)$$

To be able to rotate the quantities back and forth, we need to determine the \mathbf{Q} rotation tensor in each step, which can be calculated from the \mathbf{w} spin tensor as

$$\dot{\mathbf{Q}} = \mathbf{w}\mathbf{Q}, \quad (9)$$

where we can conclude that we will need time integration to determine the values. We will use the Hughes-Winget algorithm as it is required, which is defined as

$$\mathbf{Q}_{n+1} = \left(\mathbf{I} - \frac{1}{2} \mathbf{w}_{n+1/2} \Delta t \right)^{-1} \left(\mathbf{I} + \frac{1}{2} \mathbf{w}_{n+1/2} \Delta t \right) \mathbf{Q}_n. \quad (10)$$

The values of the elastic material parameters are the followings

$$E = 200 \text{ GPa} \quad \& \quad \nu = 0.3 \quad (11)$$

Furthermore the characteristic of the Yield stress is defined by the following Swift model:

$$Y(\lambda) = a(\varepsilon_0 + \lambda)^n \quad , \text{where} \quad a = 400 \text{ MPa}, \quad \varepsilon_0 = 0.05, \quad n = 0.25. \quad (12)$$

Solution

My implementation of the radial-return algorithm for this hypoelastic-plastic model in Python language is the following:

```
1 Q = np.tile(I, (N, 1, 1))
2 sig_tilde = np.tile(0, (N, 1, 1))
3 sig = np.tile(0, (N, 1, 1))
4 Y = np.ones(N) * Y_F(0)
5 _lambda = np.zeros(N)
6
7 for n in range(N-1):
8     dt = t[n+1] - t[n]
9     w_mid = (w[n+1] + w[n])/2
10    d_mid = (d[n+1] + d[n])/2
11
12    # Hughes-Winget
13    Q[n+1] = inv(I - 1/2*w_mid*dt) @ (I + 1/2*w_mid*dt) @ Q[n]
14    Q_mid = Q[n+1]
15
16    deps_tilde = Q_mid.T @ d_mid* dt @ Q_mid
17
18    # Step 3: trial stress
19    sig_trial_tilde = sig_tilde[n] + D(deps_tilde)
20    s_trial_tilde = dev(sig_trial_tilde)
21
22    # Step 4: trial yield function
23    F_trial = (3/2)**(1/2) * norm(s_trial_tilde) - Y[n]
24
25    if F_trial <= 0:
26        # Step 6: Elastic increment
27        sig_tilde[n+1] = sig_trial_tilde
28        Y[n+1] = Y[n]
29        _lambda[n+1] = _lambda[n]
30    else:
31        # Step 7: Accumulated plastic strain increment
32        def eq(d_lambda):
33            ret = (3/2)**(1/2) * norm(s_trial_tilde)
34            ret -= 3*G*d_lambda
35            ret -= Y_F(_lambda[n]+d_lambda)
36            return ret
37        d_lambda = (root(eq,1e-8).x)[0]
38
39        # Step 8: Plastic strain increment and yield stress update
40        deps_p_tilde = (3/2)**(1/2) * d_lambda * s_trial_tilde /
41            norm(s_trial_tilde)
42        _lambda[n+1] = _lambda[n] + d_lambda
43        Y[n+1] = Y_F(_lambda[n+1])
44
45        # Step 9: Update the rotated stress
46        sig_tilde[n+1] = sig_trial_tilde - D(deps_p_tilde)
47
48        # Step 10: Compute the stress tensor
49        sig[n+1] = Q[n+1] @ sig_tilde[n+1] @ Q[n+1].T
```

5 increments per step

The values of the plotted quantities are also attached as a CSV file.

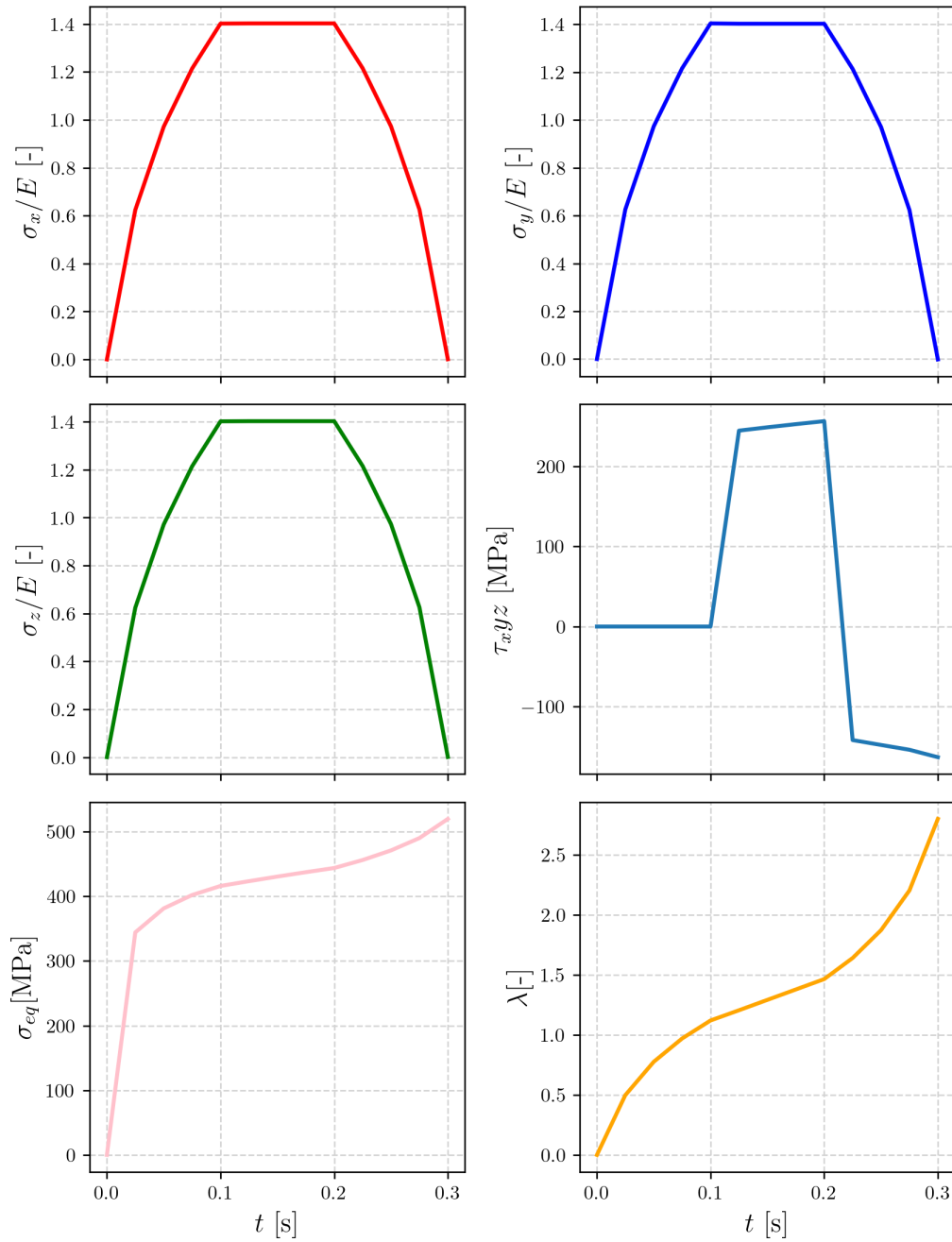


Figure 3: The solution of the problem, by applying 5 increments in each step.

1000 increments per step

The values of the plotted quantities are also attached as a CSV file.

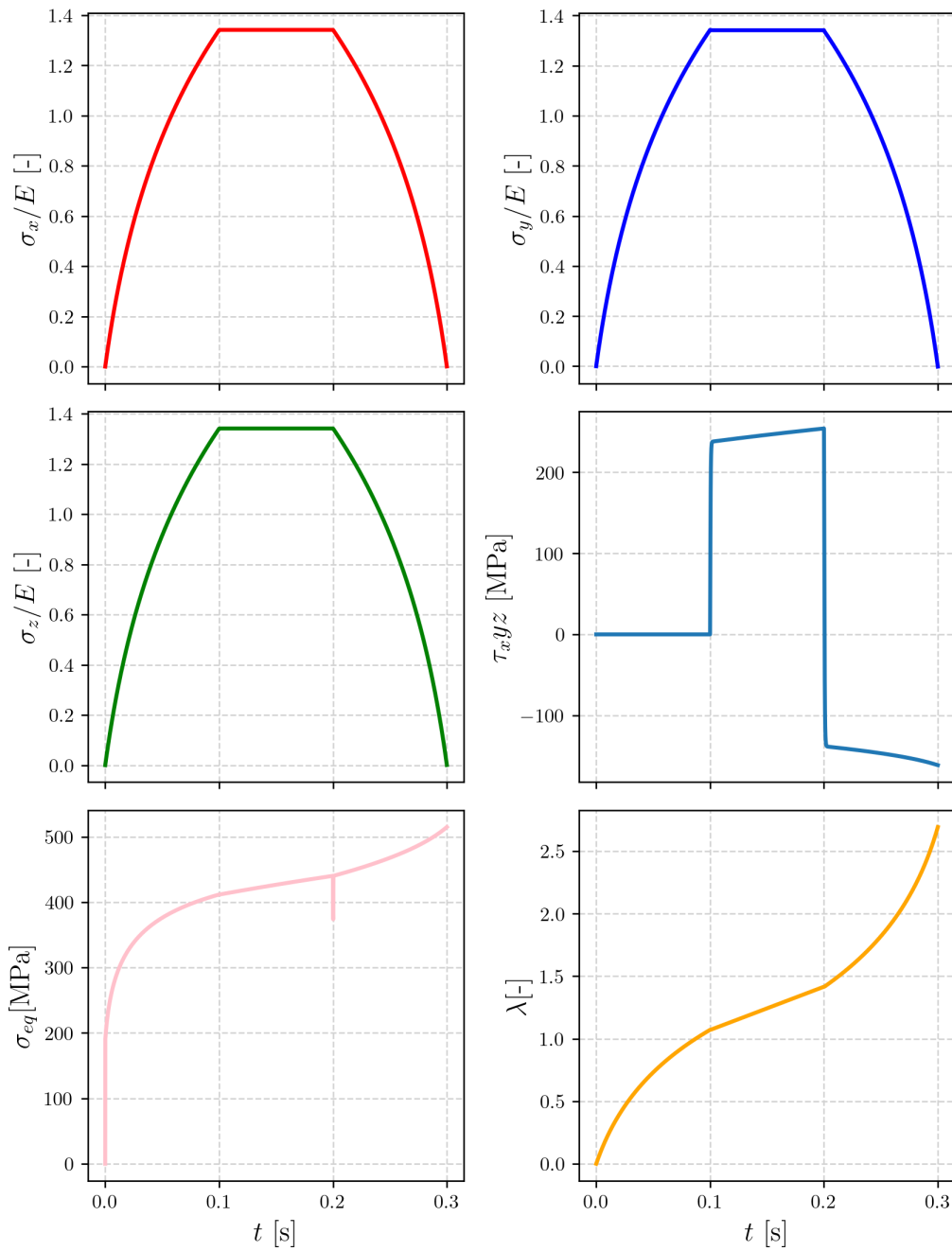


Figure 4: The solution of the problem, by applying 1000 increments in each step.

References

You can find the detailed code on [GitHub](#).