J. Numer. Math., Vol. 10, No. 3, pp. 157–192 (2002)

Elastoviscoplastic Finite Element analysis in 100 lines of Matlab

C. Carstensen * and R. Klose †

Received 3 July, 2002

Abstract — This paper provides a short Matlab implementation with documentation of the P_1 finite element method for the numerical solution of viscoplastic and elastoplastic evolution problems in 2D and 3D for von-Mises yield functions and Prandtl-Reuß flow rules. The material behaviour includes perfect plasticity as well as isotropic and kinematic hardening with or without a viscoplastic penalisation in a dual model, i.e. with displacements and the stresses as the main variables. The numerical realisation, however, eliminates the internal variables and becomes displacement-oriented in the end. Any adaption from the given three time-depending examples to more complex applications can easily be performed because of the shortness of the program and the given documentation. In the numerical 2D and 3D examples an efficient error estimator is realized to monitor the stress error.

Keywords: finite element method, viscoplasticity, elastoplasticity, Matlab

1. INTRODUCTION

Elastoplastic time-evolution problems usually require universal, complex, commercial computer codes running on workstations or even super computers [11,15]. The argument of keeping commercial secrets hidden inside a black-box, leaves the typical user without any idea what *exactly* is going on behind the program's user-friendly surface. The difference between a mathematical and a numerical model is fuzzy. Often, users do not care about nasty details such as quadrature rules or the exact material laws realised. But sometimes it does matter whether a regularised or penalised discrete model is solved, how the termination of an iterative process is steered, and what post-processing led to both, equilibrium and admissibility of the approximate stress field. In particular, if the solution process is part of guaranteed error control, it is quite important to know if discrete equilibrium is fulfilled exactly or not. Finally, the use of more than one black-box in a chain is likely to be less efficient. In summary, to aim efficiency (e.g. by well-adapting automatic mesh-refinements) and reliability we have to prevent, at least confess, all

*Institute for Applied Mathematics and Numerical Analysis, Vienna University of Technology, Wiedner Hauptstraße 8-10, A-1040 Vienna, Austria

†Mathematisches Seminar, Christian-Albrechts-Universität zu Kiel, Ludewig-Meyn-Str. 4, D-24098 Kiel, Germany

the numerical crimes. Those range from modelling errors (e.g. data errors, coarse geometrical resolution, wrong mathematical model, etc.) to variational crimes (e.g. material laws fulfilled at discrete points only or just in some approximate sense). This work aims a clear presentation of clean algorithms to compute a few model examples. Their implementation in Matlab (as one possible higher scientific computing tool) is the content of this paper. The provided software may serve as both, reference material for future numerical examples and as scientificly proved public domain source for modifications in other applications for research and educational purposes.

This paper continues our efforts on the Laplace and Navier–Lamé problem [2,3] towards more complicated material behaviour to advertise and to open the field of computational plasticity to a larger community of applied mathematicians and engineers. Within a small strain framework, the rheological models of this work include perfect plasticity, elastoplastic evolution with kinematic or isotropic hardening in the setting of von-Mises yield conditions and a Prandl-Reuß flow rule [11,15]. All variants are available for a viscoplastic regularisation with a Yosida-regularisation, known as viscoplasticity due to Perzyna.

The time discretisation yields a minimisation problem under severe constraints for each time step. The discrete problem is solved with a Newton-Raphson scheme and Matlab's standard direct solver. The main principle of the presented method is, that, in each time step, the stress tensor is expressed explicitly and applied in Cauchy's first law of motion.

We expect that the reader is familiar with Matlab as well as with linear elasticity and otherwise refer, e.g. to [3,13].

The paper is organised as follows. In Section 2 we introduce the viscoplastic and elastoplastic problem and their mathematical model. In Section 3 we perform a time discretisation and calculate the explicit expression of the stress tensor for the different models of hardening. In Section 4 the discretisation in space together with the data representation of the triangulation, the Dirichlet and Neumann boundary is considered. Furthermore the iterative solution of the discrete problem is explained. Section 5 concerns the calculation of the stiffness matrix. Section 6 discusses post processing routines, one for previewing the numerical solution and one for an a posteriori error estimation. Numerical examples in Section 7 illustrate the usage of the provided tools for four benchmark examples in two and three space dimensions.

The programs are written for Matlab 6.1 but adaption for previous versions is possible. For a numerical calculation it is necessary to run the program `main.m` with (user-specified) files `coordinates.dat`, `elements.dat`, `dirichlet.dat`, `neumann.dat`, as well as the subroutines `f.m`, `g.m`, and `u_D.m`. Their meaning and usage is explained throughout the paper and examples are provided in the later sections. The graphical representation is performed with the function `show.m` and the error is estimated in `aposteriori.m`. All files with the code and the data for all examples can be downloaded from `www.math.tuwien.ac.at/~carsten/`.

2. RHEOLOGICAL MODEL

2.1. Material models

The material body occupies a bounded Lipschitz domain Ω in $\mathbb{R}^d$ and deforms any time t, $0 \leqslant t \leqslant T$. Given a displacement field $u = u(t) \in H^1(\Omega)^d$ (i.e., the functional matrix Du exists almost everywhere in Ω, is measurable, and satisfies $\int_\Omega |Du|^2\,\mathrm{d}x < \infty$) the total strain reads

$$\varepsilon(u) = (Du + Du^T)/2, \qquad (\varepsilon(u)_{jk} = (u_{j,k} + u_{k,j})/2,\ j,k = 1,\dots,d).$$

In the engineering literature, $\varepsilon(u)$ is known as the linear Green strain tensor. The default model for small-strain plasticity involves an additive split

$$\boldsymbol{\varepsilon} = \mathbf{e}(\boldsymbol{\sigma}) + \mathbf{p}(\boldsymbol{\xi})$$

of the total strain in an elastic part, $\mathbf{e}(\boldsymbol{\sigma}) = \mathbb{C}^{-1}\boldsymbol{\sigma}$, and an irreversible part $\mathbf{p}(\boldsymbol{\xi})$. The plastic strain $\mathbf{p}(\boldsymbol{\xi})$ depends on further internal variables $\boldsymbol{\xi}$. The stress field $\boldsymbol{\sigma} = \boldsymbol{\sigma}(t) \in L^2(\Omega;\mathbb{R}^{d\times d}_{\mathrm{sym}})$ of the Cauchy stress tensor $\boldsymbol{\sigma}(x,t)$ is linked to the elastic, reversible strain. The linear relation is provided by $\mathbb{C}^{-1}$, the compliance tensor, which is the inverse of the fourth order elasticity tensor $\mathbb{C}$. In the numerical examples, a two-parameter model (λ and μ are the Lamé constants) in linear isotropic, homogeneous elasticity, $\mathbb{C}\mathbf{e} = \lambda\,\mathrm{tr}(\mathbf{e})\mathbf{I} + 2\mu\,\mathbf{e}$, is employed.

The evolution law for the plastic strain $\mathbf{p}$ is more involved and requires the concept of admissible stresses, a yield function, and an associated flow rule.

The kinematic variables $\mathbf{p}$ and $\boldsymbol{\xi}$ form the generalised strain $\mathbf{P} = (\mathbf{p},\boldsymbol{\xi})$. The corresponding generalised stress reads $\boldsymbol{\Sigma} = (\boldsymbol{\sigma},\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} \in \mathbb{R}^{m\times m}_{\mathrm{sym}}$ describes internal stresses.

We denote by K the set of admissible stresses, which is a closed, convex set, containing 0, and is defined by

$$K = \{\boldsymbol{\Sigma} : \Phi(\boldsymbol{\Sigma}) \leqslant 0\}.$$

The yield function Φ describes admissible stresses by $\Phi(\boldsymbol{\Sigma}) \leqslant 0$ and models either perfect plasticity or hardening. The dissipation functional $\varphi(\boldsymbol{\Sigma})$, defined by

$$\varphi(\boldsymbol{\sigma},\boldsymbol{\alpha}) = \frac{1}{2\nu}\inf\{|(\boldsymbol{\sigma}-\boldsymbol{\tau},\boldsymbol{\alpha}-\boldsymbol{\beta})|^2 : (\boldsymbol{\tau},\boldsymbol{\beta}) \in \mathbb{R}^{d\times d}_{\mathrm{sym}} \times \mathbb{R}^{m\times m}_{\mathrm{sym}},\ \ \Phi(\boldsymbol{\tau},\boldsymbol{\beta}) \leqslant 0\} \tag{2.1}$$

approaches the indicator function of the set K, namely

$$\varphi(\boldsymbol{\Sigma}) = \begin{cases} 0, & \Phi(\boldsymbol{\Sigma}) \leqslant 0 \\ \infty, & \Phi(\boldsymbol{\Sigma}) > 0 \end{cases} \tag{2.2}$$

as $\nu \to 0$. The time derivative of $\mathbf{P}$ is given by the flow rule

$$\begin{aligned}\dot{\mathbf{P}} \in \partial\varphi(\boldsymbol{\Sigma}) = \{(\boldsymbol{\sigma}^*,\boldsymbol{\alpha}^*) \in \mathbb{R}^{d\times d}_{\mathrm{sym}} \times \mathbb{R}^{m\times m}_{\mathrm{sym}} : \forall(\boldsymbol{\tau},\boldsymbol{\beta}) \in \mathbb{R}^{d\times d}_{\mathrm{sym}} \times \mathbb{R}^{m\times m}_{\mathrm{sym}},\\ \varphi(\boldsymbol{\sigma},\boldsymbol{\alpha}) + \boldsymbol{\sigma}^* : (\boldsymbol{\tau}-\boldsymbol{\sigma}) + \boldsymbol{\alpha}^* : (\boldsymbol{\beta}-\boldsymbol{\alpha}) \leqslant \varphi(\boldsymbol{\tau},\boldsymbol{\beta})\}.\end{aligned} \tag{2.3}$$

Here, $A : B = \sum_{j,k=1}^{m} A_{jk} B_{jk}$ denotes the scalar products of (symmetric) matrices $A, B \in \mathbb{R}^{m\times m}$. For $\nu > 0$ the flow rule results in

$$\dot{\mathbf{P}} = (\dot{\mathbf{p}}, \dot{\xi}) = 1/\nu\,(\boldsymbol{\sigma} - \boldsymbol{\tau}, \boldsymbol{\alpha} - \boldsymbol{\beta}) \tag{2.4}$$

for $(\boldsymbol{\tau}, \boldsymbol{\beta})$ uniquely determined as the projection of $(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ onto K, i.e., $(\boldsymbol{\tau}, \boldsymbol{\beta}) \in K$ with $(\boldsymbol{\tau}, \boldsymbol{\beta}) := \Pi(\boldsymbol{\sigma}, \boldsymbol{\alpha})$ uniquely determined by

$$\begin{aligned} \operatorname{dist}((\boldsymbol{\sigma}, \boldsymbol{\alpha}), K) &= |(\boldsymbol{\sigma} - \boldsymbol{\tau}, \boldsymbol{\alpha} - \boldsymbol{\beta})| = |(\boldsymbol{\sigma}, \boldsymbol{\alpha}) - \Pi(\boldsymbol{\sigma}, \boldsymbol{\alpha})| \\ &= \inf\{|(\boldsymbol{\sigma} - \boldsymbol{\tau}', \boldsymbol{\alpha} - \boldsymbol{\beta}')| : (\boldsymbol{\tau}', \boldsymbol{\beta}') \in \mathbb{R}^{d\times d}_{\mathrm{sym}} \times \mathbb{R}^{m\times m}_{\mathrm{sym}},\ \Phi(\boldsymbol{\tau}', \boldsymbol{\beta}') \leqslant 0\}. \end{aligned} \tag{2.5}$$

The following examples involve the von-Mises yield function in various modifications.

Example 2.1 (Viscoplasticity). In the case of perfect viscoplasticity, there is no hardening and the internal variables $\boldsymbol{\xi}$ are absent. The yield function is given by

$$\Phi(\boldsymbol{\sigma}) = |\operatorname{dev}(\boldsymbol{\sigma})| - \sigma_y. \tag{2.6}$$

With (2.6) in (2.4) and $(s)_+ := \max\{0, s\}, s \in \mathbb{R}$ we obtain

$$\dot{\mathbf{p}} = 1/\nu\,(1 - \sigma_y/|\operatorname{dev}(\boldsymbol{\sigma})|)_+ \operatorname{dev}(\boldsymbol{\sigma}). \tag{2.7}$$

Example 2.2 (Viscoplasticity with isotropic hardening). Isotropic hardening is characterised by the scalar hardening parameter $\alpha \geqslant 0$ and the constant $H > 0$, the modulus of hardening. The yield function is

$$\Phi(\sigma, \alpha) = |\operatorname{dev}(\boldsymbol{\sigma})| - \sigma_y(1 + H\alpha). \tag{2.8}$$

With (2.8) in (2.4) we obtain

$$\begin{pmatrix} \dot{\mathbf{p}} \\ \dot{\xi} \end{pmatrix} = \frac{1}{\nu} \frac{1}{1 + H^2\sigma_y^2} \left(1 - \frac{(1 + \alpha H)\sigma_y}{|\operatorname{dev}(\sigma)|}\right)_+ \begin{pmatrix} \operatorname{dev}(\boldsymbol{\sigma}) \\ -H\sigma_y |\operatorname{dev}(\boldsymbol{\sigma})| \end{pmatrix}. \tag{2.9}$$

In our model the plastic part of the free energy is given by $\psi^p = \frac{1}{2} H_1 \xi^2$. The internal force is defined by $\alpha = -\partial\psi^p/\partial\xi$, which means $\alpha = -H_1\xi$, where H_1 is a positive hardening parameter.

Example 2.3 (Viscoplasticity with linear kinematic hardening). In kinematic hardening the yield function is given by

$$\Phi(\boldsymbol{\sigma}, \boldsymbol{\alpha}) = |\operatorname{dev}(\boldsymbol{\sigma}) - \operatorname{dev}(\boldsymbol{\alpha})| - \sigma_y. \tag{2.10}$$

With (2.10) in (2.4) we obtain

$$\begin{pmatrix} \dot{\mathbf{p}} \\ \dot{\xi} \end{pmatrix} = \frac{1}{2\nu} \left(1 - \frac{\sigma_y}{|\operatorname{dev}(\sigma - \alpha)|}\right)_+ \begin{pmatrix} \operatorname{dev}(\boldsymbol{\sigma} - \boldsymbol{\alpha}) \\ -\operatorname{dev}(\boldsymbol{\sigma} - \boldsymbol{\alpha}) \end{pmatrix}. \tag{2.11}$$

For linear kinematic hardening, the plastic part of the free energy has the form $\psi^p = \frac{1}{2}k_1|\xi|^2$. The internal force is defined by $\alpha = -\partial\psi^p/\partial\xi$, which means $\alpha = -k_1\xi$, with a positive parameter k_1.

Remark 2.1 (Elastoplasticity for $\nu \to 0$). In the present situation the positive number ν may be seen as a viscose penalty which leads to equalities (such as (2.7), (2.9), (2.11)). The elastoplastic limit for $\nu \to 0$ leads in (2.3) to a variational inequality [11,15]. This model will be included subsequently as one may consider $\nu \to 0$ in the formulae below.

2.2. Equilibrium equations

The stress field $\boldsymbol{\sigma} \in L^2(\Omega;\mathbb{R}^{d\times d}_{\text{sym}})$ and the volume force $\mathbf{f} \in L^2(\Omega;\mathbb{R}^d)$ are related by the local quasi-static balance of forces,

$$\operatorname{div}\boldsymbol{\sigma} + \mathbf{f} = 0. \tag{2.12}$$

On some closed subset Γ_D of the boundary with positive surface measure, we assume Dirichlet conditions while we have Neumann boundary conditions on the (possible empty) part Γ_N. The two components of the displacement $\mathbf{u}$ need not satisfy simultaneously Dirichlet or Neumann conditions, i.e., Γ_D and Γ_N need not be disjoint. With $\mathbf{M} \in L^\infty(\Gamma_D)^{d\times d}$, $\mathbf{w} \in H^1(\Omega)^d$, and a surface force $g \in L^2(\Gamma_N)$ we have

$$\mathbf{Mu} = \mathbf{w} \quad \text{on } \Gamma_D, \qquad \boldsymbol{\sigma}\mathbf{n} = \mathbf{g} \quad \text{on } \Gamma_N. \tag{2.13}$$

Let Π denote the projection onto the set of admissible stresses K. The plastic problem is then determined by the weak formulation: Seek $\mathbf{u} \in H^1(\Omega)^d$ that satisfies $\mathbf{Mu} = \mathbf{w}$ on Γ_D that, for all $\mathbf{v} \in H^1_D(\Omega)^d := \{\mathbf{v} \in H^1(\Omega)^d : \mathbf{Mv} = 0 \text{ on } \Gamma_D\}$,

$$\begin{aligned} &\int_\Omega \boldsymbol{\sigma}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\,\mathrm{d}x = \int_\Omega \mathbf{f}\cdot\mathbf{v}\,\mathrm{d}x + \int_{\Gamma_N} \mathbf{g}\cdot\mathbf{v}\,\mathrm{d}s \\ &\begin{bmatrix} \boldsymbol{\varepsilon}(\dot{\mathbf{u}}) - \mathbb{C}^{-1}\dot{\boldsymbol{\sigma}} \\ \xi(\dot{\boldsymbol{\alpha}}) \end{bmatrix} = \frac{1}{\nu}\begin{bmatrix} \boldsymbol{\sigma} - \Pi\boldsymbol{\sigma} \\ \boldsymbol{\alpha} - \Pi\boldsymbol{\alpha} \end{bmatrix} \quad \text{a.e. in } \Omega. \end{aligned} \tag{2.14}$$

3. TIME DISCRETISATION AND ANALYTIC EXPRESSION OF THE STRESS TENSOR

3.1. Time discretisation scheme

A generalised midpoint rule serves as a time-discretisation. In each time step there is a spatial problem with given variables $(u(t),\sigma(t),\alpha(t))$ at time t_0 denoted as (u_0,σ_0,α_0) and unknowns at time $t_1 = t_0 + k$ denoted as (u_1,σ_1,α_1). Time derivatives are replaced by backward difference quotients. The time discrete problem reads: Seek $\mathbf{u}_\vartheta \in H^1(\Omega)^d$ that satisfies $\mathbf{Mu}_\vartheta = \mathbf{w}$ on Γ_D that, for all

$\mathbf{v} \in H^1_D(\Omega)^d := \{\mathbf{v} \in H^1(\Omega)^d : \mathbf{M}\mathbf{v} = 0 \text{ on } \Gamma_D\}$,

$$\begin{aligned} &\int_\Omega \boldsymbol{\sigma}(\mathbf{u}_\vartheta) : \boldsymbol{\varepsilon}(\mathbf{v})\,\mathrm{d}x = \int_\Omega \mathbf{f}_\vartheta \cdot \mathbf{v}\,\mathrm{d}x + \int_{\Gamma_N} \mathbf{g}_\vartheta \cdot \mathbf{v}\,\mathrm{d}s \\ &\frac{1}{\vartheta k}\begin{bmatrix} \boldsymbol{\varepsilon}(\mathbf{u}_\vartheta - \mathbf{u}_0) - \mathbb{C}^{-1}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\sigma}_0) \\ \xi(\boldsymbol{\alpha}, t_\vartheta) - \xi(\boldsymbol{\alpha}, t_0) \end{bmatrix} = \frac{1}{\nu}\begin{bmatrix} \boldsymbol{\sigma}_\vartheta - \Pi\boldsymbol{\sigma}_\vartheta \\ \boldsymbol{\alpha}_\vartheta - \Pi\boldsymbol{\alpha}_\vartheta \end{bmatrix} \end{aligned} \tag{3.1}$$

where $\boldsymbol{\sigma}_\vartheta = (1-\vartheta)\boldsymbol{\sigma}_0 + \vartheta\boldsymbol{\sigma}_1$ with $1/2 \leqslant \vartheta \leqslant 1$. We define

$$\mathbf{A} := \boldsymbol{\varepsilon}\left(\frac{\mathbf{u}_\vartheta - \mathbf{u}_0}{\vartheta k}\right) + \mathbb{C}^{-1}\frac{\boldsymbol{\sigma}_0}{\vartheta k}. \tag{3.2}$$

With this notation in the flow law, we have

$$\mathbf{A} - \mathbb{C}^{-1}\frac{\boldsymbol{\sigma}_\vartheta}{\vartheta k} = \frac{1}{\nu}(id - \Pi)\boldsymbol{\sigma}_\vartheta. \tag{3.3}$$

To obtain an expression of $\mathbb{C}^{-1}\boldsymbol{\sigma}_\vartheta$ in terms of $\operatorname{tr}\boldsymbol{\sigma}_\vartheta$ and $\operatorname{dev}\boldsymbol{\sigma}_\vartheta$ we note that $\boldsymbol{\sigma}_\vartheta = \lambda \operatorname{tr}\mathbf{e}_\vartheta\mathbf{I} + 2\mu\,\mathbf{e}_\vartheta$. There are constants α, β, γ and δ such that

$$\boldsymbol{\sigma}_\vartheta = \gamma\operatorname{tr}\mathbf{e}_\vartheta\mathbf{I} + \delta\operatorname{dev}\mathbf{e}_\vartheta, \qquad \mathbf{e}_\vartheta = \alpha\operatorname{tr}\boldsymbol{\sigma}_\vartheta I + \beta\operatorname{dev}\boldsymbol{\sigma}_\vartheta. \tag{3.4}$$

Inserting of (3.4a) in (3.4b) we have

$$\mathbf{e}_\vartheta = d\alpha\gamma\operatorname{tr}\mathbf{e}_\vartheta\mathbf{I} + \beta\delta\operatorname{dev}\mathbf{e}_\vartheta\,, \qquad \alpha = 1/(d^2\gamma), \quad \beta = 1/\delta. \tag{3.5}$$

According to (3.4a) the stress tensor $\boldsymbol{\sigma}_\vartheta$ reads

$$\boldsymbol{\sigma}_\vartheta = \gamma\operatorname{tr}\mathbf{e}_\vartheta\mathbf{I} + \delta(\mathbf{e}_\vartheta - 1/d\operatorname{tr}\mathbf{e}_\vartheta\mathbf{I}) = \left(\gamma - \frac{\delta}{d}\right)\operatorname{tr}\mathbf{e}_\vartheta\,\mathbf{I} + \delta\,\mathbf{e} = \lambda\operatorname{tr}\mathbf{e}_\vartheta\,\mathbf{I} + 2\mu\,\mathbf{e}_\vartheta. \tag{3.6}$$

This shows $\delta = 2\mu$ and $\gamma = \lambda + 2\mu/d$. Inserting this in (3.4a) we deduce

$$\mathbb{C}^{-1}\boldsymbol{\sigma}_\vartheta = \frac{1}{d^2\lambda + 2d\mu}\operatorname{tr}\boldsymbol{\sigma}_\vartheta\mathbf{I} + \frac{1}{2\mu}\operatorname{dev}\boldsymbol{\sigma}_\vartheta. \tag{3.7}$$

3.2. Analytic expression of the stress tensor

In this subsection we derive explicit expressions for the stress tensor $\boldsymbol{\sigma}_\vartheta$ for the different cases of hardening. With the connection $\boldsymbol{\sigma}_\vartheta = \boldsymbol{\sigma}_\vartheta(\mathbf{A}(\mathbf{u}_\vartheta))$ between $\boldsymbol{\sigma}_\vartheta$ and $\mathbf{u}_\vartheta$ the elastoplastic problem is determined by a nonlinear variational problem: Seek $\mathbf{u}_\vartheta \in H^1(\Omega)^d$ that satisfies $\mathbf{M}\mathbf{u}_\vartheta = \mathbf{w}$ on Γ_D and, for all $\mathbf{v} \in H^1_D(\Omega)^d := \{\mathbf{v} \in H^1(\Omega)^d : \mathbf{M}\mathbf{v} = 0 \text{ on } \Gamma_D\}$,

$$\int_\Omega \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{u}_\vartheta - \mathbf{u}_0) + \mathbb{C}^{-1}\boldsymbol{\sigma}_0) : \boldsymbol{\varepsilon}(\mathbf{v})\,\mathrm{d}x = \int_\Omega \mathbf{f}_\vartheta \cdot \mathbf{v}\,\mathrm{d}x + \int_{\Gamma_N} \mathbf{g}_\vartheta \cdot \mathbf{v}\,\mathrm{d}s. \tag{3.8}$$

We solve (3.3) for $\boldsymbol{\sigma}_\vartheta$ in the flow laws of Examples 2.1–2.3.

Theorem 3.1 (Viscoplasticity and plasticity without hardening). *For perfect viscoplasticity and perfect plasticity there exist constants C_1, C_2, and C_3 such that the stress tensor reads*

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + (C_2 + C_3/|\operatorname{dev}\vartheta k\mathbf{A}|)\operatorname{dev}\vartheta k\mathbf{A}. \tag{3.9}$$

For perfect viscoplasticity these constants are

$$C_1 := \lambda + 2\mu/d, \quad C_2 := \nu/(\beta\nu + \vartheta k), \quad C_3 := \vartheta k\sigma_y/(\beta\nu + \vartheta k) \tag{3.10}$$

and in the limit $\nu \to 0$ for perfect plasticity these constants read

$$C_1 := \lambda + 2\mu/d, \quad C_2 := 0, \quad C_3 := \sigma_y. \tag{3.11}$$

The plastic phase occurs for

$$|\operatorname{dev}\vartheta k\mathbf{A}| > \frac{\sigma_y}{2\mu}. \tag{3.12}$$

In the elastic phase the stress tensor reads

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + 2\mu\operatorname{dev}\vartheta k\mathbf{A}. \tag{3.13}$$

Proof. The discretised version of the flow law reads

$$\mathbf{A} - \mathbb{C}^{-1}\frac{\boldsymbol{\sigma}_\vartheta}{\vartheta k} = \frac{1}{\nu}\left(1 - \frac{\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)_+ \operatorname{dev}\boldsymbol{\sigma}_\vartheta. \tag{3.14}$$

We consider the plastic phase $(1 - \sigma_y/|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|) > 0$. With $\mathbb{C}^{-1}\boldsymbol{\sigma}_\vartheta = \alpha\operatorname{tr}\sigma_\vartheta\mathbf{I} + \beta\operatorname{dev}\boldsymbol{\sigma}_\vartheta$ we obtain

$$\vartheta k\operatorname{dev}\mathbf{A} = \left(\beta + \frac{\vartheta k}{\nu}\left(1 - \frac{\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)\right)\operatorname{dev}\boldsymbol{\sigma}_\vartheta\,, \quad |\operatorname{dev}\boldsymbol{\sigma}_\vartheta| = \frac{\vartheta k(\nu|\operatorname{dev}\mathbf{A}| + \sigma_y)}{\vartheta k + \beta\nu}. \tag{3.15}$$

Inserting (3.15b) in (3.15a) shows

$$\operatorname{dev}\boldsymbol{\sigma}_\vartheta = (C_2 + C_3/|\operatorname{dev}\vartheta k\mathbf{A}|)\operatorname{dev}\vartheta k\mathbf{A}. \tag{3.16}$$

From (3.14) we obtain $\operatorname{tr}\boldsymbol{\sigma}_\vartheta = C_1 d\operatorname{tr}\vartheta k\mathbf{A}$. The expression for $\operatorname{dev}\boldsymbol{\sigma}_\vartheta$ in the elastic phase is obtained from (3.14) with $1 - \sigma_y/|\operatorname{dev}\boldsymbol{\sigma}_\vartheta| \leqslant 0$.

The plastic phase occurs for $(1 - \sigma_y/|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|) > 0$. With (3.15b) this is equivalent to $|\operatorname{dev}\vartheta k\mathbf{A}| > \sigma_y/(2\mu)$. □

Theorem 3.2 (Viscoplasticity and plasticity with isotropic hardening). *For viscoplasticity and plasticity with isotropic hardening there exist constants C_1, C_2, C_3, and C_4 such that the stress tensor reads*

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + (C_3/(C_2|\operatorname{dev}\vartheta k\mathbf{A}|) + C_4/C_2)\operatorname{dev}\vartheta k\mathbf{A}. \tag{3.17}$$

For viscoplasticity with isotropic hardening these constants are

$$\begin{aligned}C_1 &:= \lambda + 2\mu/d, \quad C_2 := \beta\nu(1+H^2\sigma_y^2) + \vartheta k(1+\beta H_1 H^2\sigma_y^2)\\ C_3 &:= \vartheta k\sigma_y(1+\alpha_0 H), \quad C_4 := H_1 H^2\vartheta k\sigma_y^2 + \nu(1+H^2\sigma_y^2)\end{aligned} \tag{3.18}$$

and in the limit $\nu \to 0$ *for plasticity with isotropic hardening these constants read*

$$\begin{aligned}C_1 &:= \lambda + 2\mu/d, \quad C_2 := \vartheta k(1+\beta H_1 H^2\sigma_y^2)\\ C_3 &:= \vartheta k\sigma_y(1+\alpha_0 H), \quad C_4 := H_1 H^2\vartheta k\sigma_y^2.\end{aligned} \tag{3.19}$$

The plastic phase occurs for

$$|\operatorname{dev}\vartheta k\mathbf{A}| > \beta(1+\alpha_0 H)\sigma_y. \tag{3.20}$$

In the elastic phase the stress tensor reads

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + 2\mu \operatorname{dev}\vartheta k\mathbf{A}. \tag{3.21}$$

Proof. The discretised version of the flow law is

$$\mathbf{A} - \mathbb{C}^{-1}\frac{\boldsymbol{\sigma}_\vartheta}{\vartheta k} = \frac{1}{\nu}\frac{1}{1+H^2\sigma_y^2}\left(1 - \frac{(1+\alpha_\vartheta H)\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)_+ \operatorname{dev}\boldsymbol{\sigma}_\vartheta \tag{3.22}$$

$$-H_1^{-1}\frac{\alpha_\vartheta - \alpha_0}{\vartheta k} = -\frac{1}{\nu}\frac{1}{1+H^2\sigma_y^2}\left(1 - \frac{(1+\alpha_\vartheta H)\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)_+ H\sigma_y|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|. \tag{3.23}$$

We consider the plastic phase $1 - (1+\alpha_\vartheta H)\sigma_y/|\operatorname{dev}\boldsymbol{\sigma}_\vartheta| > 0$. With $\mathbb{C}^{-1}\boldsymbol{\sigma}_\vartheta = \alpha \operatorname{tr}\boldsymbol{\sigma}_\vartheta\mathbf{I} + \beta \operatorname{dev}\boldsymbol{\sigma}_\vartheta$ in (3.22) we infer

$$\vartheta k \operatorname{dev}\mathbf{A} = \left(\beta + \frac{\vartheta k}{\nu}\frac{1}{1+H^2\sigma_y^2}\left(1 - \frac{(1+\alpha_\vartheta H)\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)\right)\operatorname{dev}\boldsymbol{\sigma}_\vartheta \tag{3.24}$$

and so

$$|\operatorname{dev}\vartheta k\mathbf{A}| = \left(\beta + \frac{\vartheta k}{\nu}\frac{1}{1+H^2\sigma_y^2}\left(1 - \frac{(1+\alpha_\vartheta H)\sigma_y}{|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|}\right)\right)|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|. \tag{3.25}$$

The second component of the flow rule, (3.23), can be solved for α_ϑ. Inserting of α_ϑ in (3.25), solving for $|\operatorname{dev}\boldsymbol{\sigma}_\vartheta|$ and inserting the resulting expression in (3.24) yields an expression for $\operatorname{dev}\boldsymbol{\sigma}_\vartheta$. We obtain

$$\operatorname{dev}\boldsymbol{\sigma}_\vartheta = \left(\frac{C_3}{C_2|\operatorname{dev}\vartheta k\mathbf{A}|} + \frac{C_4}{C_2}\right)\operatorname{dev}\vartheta k\mathbf{A}. \tag{3.26}$$

From the first component of the flow rule we conclude $\operatorname{tr}\boldsymbol{\sigma}_\vartheta = C_1 d \operatorname{tr}\vartheta k\mathbf{A}$. With $|\operatorname{dev}\boldsymbol{\sigma}_\vartheta| \leqslant (1+\alpha_\vartheta H)\sigma_y$ the expression for $\operatorname{dev}\boldsymbol{\sigma}_\vartheta$ in the elastic results from (3.22).

The plastic phase occurs for $|\operatorname{dev}\boldsymbol{\sigma}_\vartheta| > (1+\alpha_\vartheta H)\sigma_y$. With (3.26) this is equivalent to $1 - \beta(1+\alpha_0 H)\sigma_y/|\operatorname{dev}\vartheta k\mathbf{A}| > 0$. $\square$

Theorem 3.3 (Viscoplasticity and plasticity with kinematic hardening). *For viscoplasticity and plasticity with kinematic hardening there exit constants C_1, C_2, and C_3 such that the stress tensor reads*

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + (C_2 + C_3/|\operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\alpha}_0)|) \operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\alpha}_0) + \operatorname{dev}\boldsymbol{\alpha}_0. \tag{3.27}$$

For viscoplasticity with kinematic hardening these constants are

$$C_1 := \lambda + 2\mu/d, \quad C_2 := \frac{\vartheta k k_1 + 2\nu}{\vartheta k + \vartheta k k_1/(2\mu) + \nu/\mu}, \quad C_3 := \frac{\vartheta k \sigma_y}{\vartheta k + \vartheta k k_1/(2\mu) + \nu/\mu} \tag{3.28}$$

and in the limit $\nu \to 0$ for plasticity with kinematic hardening these constants read

$$C_1 := \lambda + 2\mu/d, \quad C_2 := \frac{\vartheta k k_1}{\vartheta k + \vartheta k k_1/(2\mu)}, \quad C_3 := \frac{\vartheta k \sigma_y}{\vartheta k + \vartheta k k_1/(2\mu)}. \tag{3.29}$$

The plastic phase occurs for

$$|\operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\alpha}_0)| > \beta\sigma_y. \tag{3.30}$$

In the elastic phase the stress tensor reads

$$\boldsymbol{\sigma}_\vartheta = C_1 \operatorname{tr}(\vartheta k\mathbf{A})\mathbf{I} + 2\mu \operatorname{dev} \vartheta k\mathbf{A}. \tag{3.31}$$

Proof. The discretised version of the flow law reads

$$\mathbf{A} - \mathbb{C}^{-1}\frac{\boldsymbol{\sigma}_\vartheta}{\vartheta k} = \frac{1}{2\nu}(1 - \sigma_y/|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_\vartheta)|)_+ \operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_\vartheta) = \frac{1}{k_1}\frac{\boldsymbol{\alpha}_\vartheta - \boldsymbol{\alpha}_0}{\vartheta k}. \tag{3.32}$$

We consider the plastic phase $1 - \sigma_y/|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_\vartheta)| > 0$. With $\mathbb{C}^{-1}\boldsymbol{\sigma}_\vartheta = \alpha \operatorname{tr}\boldsymbol{\sigma}_\vartheta \mathbf{I} + \beta \operatorname{dev}\boldsymbol{\sigma}_\vartheta$,

$$\boldsymbol{\alpha}_\vartheta = \boldsymbol{\alpha}_0 + k_1(\vartheta k\mathbf{A} - \alpha \operatorname{tr}\boldsymbol{\sigma}_\vartheta\mathbf{I} - \beta \operatorname{dev}\boldsymbol{\sigma}_\vartheta). \tag{3.33}$$

This yields in the flow rule that

$$\begin{aligned}\operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\sigma}_\vartheta) \;=\; & \frac{\vartheta k}{2\nu}\left(1 - \frac{\sigma_y}{|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_0 - k_1\vartheta k\mathbf{A} + k_1\beta\boldsymbol{\sigma}_\vartheta)|}\right) \\ & \times \operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_0 - k_1\vartheta k\mathbf{A} + k_1\beta\boldsymbol{\sigma}_\vartheta).\end{aligned}$$

The absolute value on both sides results in an equation for $|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_0 + k_1\vartheta k\mathbf{A} - k_1\beta\boldsymbol{\sigma}_\vartheta)|$. We solve for this modulus and substitute it in the above identity. This shows

$$\begin{aligned}\operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\sigma}_\vartheta) \;=\; & \frac{\vartheta k}{2\nu}\left(1 - \frac{\sigma_y}{2\nu/(\vartheta k)|\operatorname{dev}(\vartheta k\mathbf{A} - \beta\boldsymbol{\sigma}_\vartheta)| + \sigma_y}\right) \\ & \times \operatorname{dev}(\boldsymbol{\sigma}_\vartheta - \boldsymbol{\alpha}_0 - k_1\vartheta k\mathbf{A} + k_1\beta\boldsymbol{\sigma}_\vartheta).\end{aligned} \tag{3.34}$$

Multiplying both sides of (3.34) with $(2\nu)/(\vartheta k)|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta)|+\sigma_y$ together with some basic transformations shows

$$\left(\left(\frac{2\nu}{\vartheta k}+k_1\right)|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta)|+\sigma_y\right) \times \operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta) = |\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta)|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0). \quad (3.35)$$

On both sides of (3.35) we build the absolute value and obtain

$$|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta)| = \frac{|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|-\sigma_y}{k_1+2\nu/(\vartheta k)} \quad (3.36)$$

which can be inserted in (3.35),

$$|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\sigma}_\vartheta) = \frac{|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|-\sigma_y}{k_1+2\nu/(\vartheta k)}\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0). \quad (3.37)$$

Adding the term $|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|\beta\operatorname{dev}\boldsymbol{\alpha}_0$ on both sides of (3.37) yields with some basic transformations

$$|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\alpha}_0) = \left(\beta|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|+\frac{|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|-\sigma_y}{k_1+2\nu/(\vartheta k)}\right) \times \operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0). \quad (3.38)$$

Because of $2\nu/\vartheta k+k_1>0$ and with $|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)|-\sigma_y>0$ from (3.36) we can calculate the absolute value of $\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)$ as

$$|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_0)| = \frac{(\vartheta k k_1+2\nu)|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\alpha}_0)|+\vartheta k\sigma_y}{\vartheta k+\beta\vartheta k k_1+2\beta\nu}. \quad (3.39)$$

We insert (3.39) in (3.38) to obtain

$$\operatorname{dev}\boldsymbol{\sigma}_\vartheta = (C_2+C_3/|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\alpha}_0)|)\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\alpha}_0)+\operatorname{dev}\boldsymbol{\alpha}_0. \quad (3.40)$$

From the first component of the flow rule we have $\operatorname{tr}\boldsymbol{\sigma}_\vartheta = C_1 d\operatorname{tr}(\vartheta k\mathbf{A})$. The expression for $\operatorname{dev}\boldsymbol{\sigma}_\vartheta$ in the elastic phase is obtained from (3.32) with $1-\sigma_y/|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_\vartheta)|\leqslant 0$.

The plastic phase occurs for $1-\sigma_y/|\operatorname{dev}(\boldsymbol{\sigma}_\vartheta-\boldsymbol{\alpha}_\vartheta)|>0$. With (3.36), (3.37) and (3.39) this is equivalent to $|\operatorname{dev}(\vartheta k\mathbf{A}-\beta\boldsymbol{\alpha}_0)|>\beta\sigma_y$. □

4. NUMERICAL SOLUTION OF THE TIME-DISCRETE PROBLEM

This section is devoted to the framework of the spatial discretisation of one timestep (3.1) with data structures in Subsection 4.1 and the iteration in Subsection 4.2. The heart of the matter is the assembling of the tangential local stiffness matrix explained in Section 5.

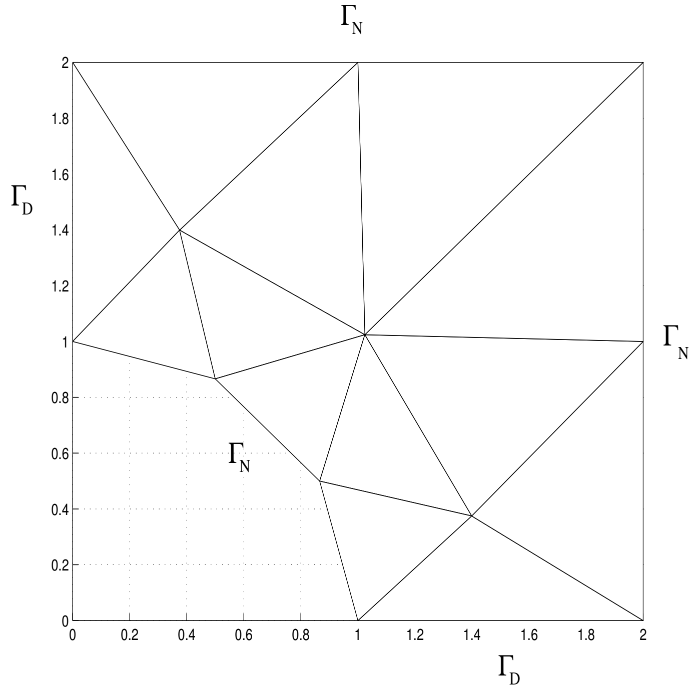


Figure 1. Plot of initial triangulation for the example of Subsection 7.1.

4.1. Discretisation in space

Suppose the domain Ω has a polygonal boundary Γ, we can cover $\bar{\Omega}$ by a regular triangulation $\mathscr{T}$, i.e., $\bar{\Omega} = \bigcup_{T \in \mathscr{T}} T$, where the elements of $\mathscr{T}$ are triangles for $d = 2$ and tetrahedrons for $d = 3$. Regular triangulation in the sense of Ciarlet [10] means that the nodes $\mathscr{N}$ of the mesh lie on the vertices of the elements, the elements of the triangulation do not overlap, no node lies on an edge of an element, and each edge $E \subset \Gamma$ of an element $T \in \mathscr{T}$ belongs either to $\bar{\Gamma}_N$ or to $\bar{\Gamma}_D$.

Matlab supports reading data from files given in ASCII format by `*.dat` files. Figure 1 shows the initial mesh of a two dimensional example and Figure 2 shows the corresponding data files. The file `coordinates.dat` contains the coordinates of each node. The two real numbers per row are the x- and y-coordinates of each node. The file `elements.dat` contains for each element the node numbers of the vertices, numbered anti-clockwise.

The files `neumann.dat` and `dirichlet.dat` contain the two node numbers which bound the corresponding edge on the boundary.

In the discrete version of (2.14), $H^1(\Omega)$ and $H^1_D(\Omega)$ are replaced by finite dimensional subspaces $\mathscr{S}$ and $\mathscr{S}_D = \{\mathbf{V} \in \mathscr{S} : \mathbf{V} = 0 \text{ on } \Gamma_D\}$. The discrete problem `coordinates.dat`

1.0000	0.0000
2.0000	0.0000
2.0000	1.0000
2.0000	2.0000
1.0000	2.0000
0.0000	2.0000
0.0000	1.0000
0.5000	0.8660
0.8660	0.5000
1.3989	0.3748
1.0241	1.0241
0.3748	1.3989

`elements.dat`

1	2	10
2	3	10
3	11	10
3	4	11
4	5	11
5	12	11
5	6	12
6	7	12
7	8	12
8	11	12
8	9	11
9	10	11
9	1	10

`dirichlet.dat`

1	2
6	7

`neumann.dat`

2	3
3	4
4	5
5	6
7	8
8	9
9	1

Figure 2. Data files `coordinates.dat`, `elements.dat`, `dirichlet.dat`, and `neumann.dat` for the initial triangulation in the example of Subsection 7.1. The files `u_d.m`, `f.m`, and `g.m` are listed in Section 7.

reads: Seek $\mathbf{U}_\vartheta \in \mathscr{S}$ with

$$\int_\Omega \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbb{C}^{-1}\boldsymbol{\sigma}_0) : \boldsymbol{\varepsilon}(\mathbf{V})\,\mathrm{d}x = \int_\Omega \mathbf{f}_\vartheta \cdot \mathbf{V}\,\mathrm{d}x + \int_\Gamma \mathbf{g}_\vartheta \cdot \mathbf{V}\,\mathrm{d}s \quad \forall \mathbf{V} \in \mathscr{S}. \tag{4.1}$$

Let $\mathscr{T}$ denote a triangulation of Ω and $\mathscr{N}$ the set of all nodes in $\mathscr{T}$, then let

$$(\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_{dN}) = (\varphi_1\mathbf{e}_1, \ldots \varphi_1\mathbf{e}_d, \ldots, \varphi_N\mathbf{e}_1, \ldots, \varphi_N\mathbf{e}_d)$$

be the nodal basis of the finite dimensional space $\mathscr{S}$, where N is the number of nodes of the mesh and φ_z is the scalar hat function of node z in the triangulation $\mathscr{T}$, i.e., $\varphi_z(z) = 1$ and $\varphi_z(y) = 0$ for all $y \in \mathscr{N}$ with $y \neq z$. Then (4.1) is equivalent to

$$F_p := \int_\Omega \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbf{C}^{-1}\boldsymbol{\sigma}_0) : \boldsymbol{\varepsilon}(\boldsymbol{\varphi}_p)\,\mathrm{d}x - \int_\Omega \mathbf{f}_\vartheta \cdot \boldsymbol{\varphi}_p\,\mathrm{d}x - \int_\Gamma \mathbf{g}_\vartheta \cdot \boldsymbol{\varphi}_p\,\mathrm{d}s = 0 \tag{4.2}$$

for $p = 1,\ldots,dN$. F_p can be decomposed into a sum of a part Q_p which depends on $\mathbf{u}_\vartheta$ and a part P_p which is independent of $\mathbf{u}_\vartheta$. Thus we have $F_p := Q_p - P_p$ with

$$Q_p := \int_\Omega \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbf{C}^{-1}\boldsymbol{\sigma}_0) : \boldsymbol{\varepsilon}(\boldsymbol{\varphi}_p)\,\mathrm{d}x \tag{4.3}$$

$$P_p := \int_\Omega \mathbf{f}_\vartheta \cdot \boldsymbol{\varphi}_p\,\mathrm{d}x + \int_\Gamma \mathbf{g}_\vartheta \cdot \boldsymbol{\varphi}_p\,\mathrm{d}s. \tag{4.4}$$

4.2. Iterative solution

This subsection describes the iterative solution of (4.2) by a Newton–Raphson scheme which is realised in the Matlab program `fem.m`, listed at the end of this section. One step of the Newton iteration reads

$$DF(\mathbf{U}_\vartheta^k)\mathbf{U}_\vartheta^{k+1} = DF(\mathbf{U}_\vartheta^k)\mathbf{U}_\vartheta^k - F(\mathbf{U}_\vartheta^k). \tag{4.5}$$

The discrete displacement vector $\mathbf{U}_\vartheta^k$ is expressed in the nodal basis as $\mathbf{U}_\vartheta^k = \sum_{\ell=1}^{dN} U_{\vartheta,\ell}^k \boldsymbol{\varphi}_\ell$, $(U_{\vartheta,1}^k,\dots,U_{\vartheta,dN}^k)$ are the components of $\mathbf{U}_\vartheta^k$. The matrix DF is defined as

$$(DF(U_{\vartheta,1}^k,\dots,U_{\vartheta,dN}^k))_{pq} := \partial F_p(U_{\vartheta,1}^k,\dots,U_{\vartheta,dN}^k)/\partial U_{\vartheta,q}^k. \tag{4.6}$$

At the end of this section the Matlab program `fem.m`, a finite element program for the two dimensional case is listed. In the following line 1

```
 1 function ut=FEM(coordinates,elements,dirichlet,neumann,ut,u0,t0,t1,th,N)
```

shows the function call with input `coordinates`, `elements`, `dirichlet` and `neumann` together with the vectors `ut` and `u0`, which are the start-vector $\mathbf{u}_\vartheta$ of the Newton iteration and the displacement $\mathbf{u}_0$ at time t_0. The last three variables in the parameter list are the times t_0, t_1, and the variable ϑ. Lines 2–7,

```
%Initialisation
 2 DF=sparse(2*N,2*N);Q=zeros(2*N,1);P=zeros(2*N,1);
 3 list=2*elements(:,[1,1,2,2,3,3])-ones(size(elements,1),1)*[1,0,1,0,1,0];
%Assembly
 4 for j=1:size(elements,1)
 5  L=list(j,:);
 6  [DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
 7 end
```

show the assembling of the stiffness matrix DF and the stiffness vector Q. Line 2 initialises DF, Q, and P, line 3 defines a vector `list` which contains the numbers of degrees of freedom for each element. Lines 4–7 perform the assembling of DF and Q with a call of subroutine `stima.m`. The element with number j contributes the degrees of freedom `L=list(j,:)`. The local stiffness matrix and the local stiffness vector of element number j have to be added at the positions (L,L) of the global stiffness matrix and (L) of the global stiffness vector.

In the following lines 8–22

```
%Volume forces
 8 for j=1:size(elements,1)
 9  lcoordinates=coordinates(elements(j,:),:);area=det([1,1,1;lcoordinates'])/2;
10  b=area*(1-th)*f(sum(lcoordinates,1)/3,t0)'/3+th*f(sum(lcoordinates,1)/3,t1)'/3;
11  P(list(j,:))=P(list(j,:))+repmat(b,3,1);
12 end
%Neumann conditions
13 if ~isempty(neumann)
14  Nlist=2*neumann(:,[1,1,2,2])-ones(size(neumann,1),1)*repmat([1,0],1,2);
15  EV=coordinates(neumann(:,2),:)-coordinates(neumann(:,1),:);
16  Lg=sqrt(sum(EV.*EV,2));EV=EV./[Lg,Lg];NV=EV*[0,-1;1,0];
17  for j=1:size(neumann,1)
18   gs=Lg(j)*((1-th)*g(sum(coordinates(neumann(j,:),:))/2,NV(j,:),t0)'/2 ...
19      +th*g(sum(coordinates(neumann(j,:),:))/2,NV(j,:),t1)'/2);
20   P(Nlist(j,:))=P(Nlist(j,:))+repmat(gs,2,1);
21  end
```

```
22 end
23 F=Q-P;
```

the vector $F = Q - P$ is determined. P contains the work of the volume forces $\mathbf{f}$ and the surface forces $\mathbf{g}$ at time t_ϑ. The values of $\mathbf{f}$ are provided by the subroutine `f.m` and evaluated in the centre of gravity (x_S, y_S) of T, to approximate the integral $\int_T \mathbf{f}\cdot\boldsymbol{\varphi}_j\,\mathrm{d}x$ in (4.2),

$$\int_T \mathbf{f}\cdot\boldsymbol{\varphi}_r\,\mathrm{d}x \approx \frac{1}{3}\,|T|\,f_k(x_S,y_S), \qquad k := \mathrm{mod}(r-1,2)+1. \tag{4.7}$$

The integral $\int_E \mathbf{g}\cdot\boldsymbol{\varphi}_k\,\mathrm{d}s$ in (4.2) that involves the Neumann conditions is approximated with the value of $\mathbf{g}$ in the centre (x_M, y_M) of the edge E with length $|E|$,

$$\int_E \mathbf{g}\cdot\boldsymbol{\varphi}_r\,\mathrm{d}s \approx \tfrac{1}{2}|E|\,g_k(x_M,y_M), \qquad k := \mathrm{mod}(r-1,2)+1. \tag{4.8}$$

The following lines 24–33

```
%Dirichlet conditions
24 dnodes=unique(dirichlet);
25 [W0,M0]=u_D(coordinates(dnodes,:),t0);[W1,M1]=u_D(coordinates(dnodes,:),t1);
26 W=(1-th)*W0+th*W1;M=(1-th)*M0+th*M1;B=sparse(size(W,1),2*N);
27 for k = 0:1
28  for j = 0:1
29   B(1+j:2:size(M,1),2*dnodes-1+k)=diag(M(1+j:2:size(M,1),1+k));
30  end
31 end
32 mask=find(sum(abs(B)'));freeNodes=find(~sum(abs(B)));
33 B=B(mask,:);W=W(mask,:);norm0=norm(F(freeNodes));nit=0;
```

take into account Dirichlet conditions whereas gliding boundary conditions are allowed as well. Gliding boundary conditions such as those along symmetry axes, are implemented different from Dirichlet conditions (for all components). The displacement is fixed merely in one specified direction and possibly free in others. A general approach to this type of condition reads, for each node on the Dirichlet boundary,

$$\begin{pmatrix}\mathbf{m}_1\\ \vdots\\ \mathbf{m}_d\end{pmatrix}\begin{pmatrix}U_1\\ \vdots\\ U_d\end{pmatrix} = \begin{pmatrix}w_1\\ \vdots\\ w_d\end{pmatrix} \tag{4.9}$$

where $\mathbf{m}_j \in \mathbb{R}^{1\times d}$, $w_j \in \mathbb{R}$, and $(U_1,\ldots,U_d)$ are the degrees of freedom for the considered node on the Dirichlet boundary. With the normal vector $\mathbf{n}$ along Γ and the canonical basis $\mathbf{e}_j$, $j = 1,\ldots,d$, in $\mathbb{R}^d$, gliding and classical Dirichlet conditions and the lack of Dirichlet-type conditions on certain parts of Γ can all be included by

$$\begin{pmatrix}\mathbf{n}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix}\begin{pmatrix}U_1\\ \vdots\\ U_d\end{pmatrix} = \mathbf{0}, \qquad \begin{pmatrix}\mathbf{e}_1^T\\ \vdots\\ \mathbf{e}_d^T\end{pmatrix}\begin{pmatrix}U_1\\ \vdots\\ U_d\end{pmatrix} = \mathbf{u_D}, \qquad \begin{pmatrix}\mathbf{0}\\ \vdots\\ \mathbf{0}\end{pmatrix}\begin{pmatrix}U_1\\ \vdots\\ U_d\end{pmatrix} = \mathbf{0} \tag{4.10}$$

respectively. Line 24 determines all nodes on the Dirichlet boundary and stores them in the variable `dnodes`. Let there be n Dirichlet nodes. Line 26 creates a matrix `M` of size $2n \times 2$ and a vector `W` of size $2n \times 1$. The rows $2(j-1)+1$ up to $2j$ for $j = 1,...,n$ of `M` and `W` contain for each Dirichlet node the matrix $(\mathbf{m}_1,...,\mathbf{m}_d)^T$ and the vector (w_1, w_2). For the discrete problem, the generalised Dirichlet conditions lead to the linear system

$$B\mathbf{U} = \mathbf{w} \tag{4.11}$$

where each row of $B \in \mathbb{R}^{2n\times 2N}$ contains the value of some $\mathbf{m}_j$ at a node on the boundary and $\mathbf{w} \in \mathbb{R}^{2n}$ contains the corresponding values w_j at this node. The matrix `B` is calculated in lines 27–31 from the entries of `M`. A row $\mathbf{m}_j = \mathbf{0}$ is not included in B and W, which is realised in line 33.

Between lines 34–45

```
34 while (norm(F(freeNodes))>10^(-10)+10^(-6)*norm0) & (nit<100)
35  nit=nit+1
% calculation of the solution
36  F=[DF*ut-F;W];DF=[DF,B';B,sparse(size(B,1),size(B,1))];
37  x=DF\F;ut=x(1:2*N);lb=x(2*N+1:size(x,1));
 % Assembly
38  DF=sparse(2*N,2*N);Q=zeros(2*N,1);
39  for j=1:size(elements,1)
40   L=list(j,:);
41   [DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
42  end
43  F=Q-P;
44 end
45 if (nit==100)  disp('Fails to converge within 100 iteration steps!'); end
```

the Newton iteration takes place. Coupling the set of boundary conditions through Lagrange parameters with (4.5) leads to the extended system

$$\begin{pmatrix} DF(\mathbf{U}_\vartheta^k) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{U}_\vartheta^{k+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{w} \end{pmatrix} \tag{4.12}$$

with

$$\mathbf{b} := DF(U_\vartheta^k)\mathbf{U}_\vartheta^k - \mathbf{f}(U_\vartheta^k). \tag{4.13}$$

This system of equations is solved in line 37 by the Matlab solver. Matlab makes use of the properties of a sparse, symmetric matrix for solving the system of equations efficiently. Line 37 determines the parts $\mathbf{U}_\vartheta^{k+1}$ and λ of the solution. Line 38 to 43 assemble the local stiffness matrix and stiffness vector. Line 32 calculates the set $\mathscr{K}$ of free nodes. There are m free nodes $(z_1,...,z_m) \in \mathscr{K}$. The iteration process only continues in line 34, if the relative residual $F(U_{\vartheta,z_1}^k,...,U_{\vartheta,z_m}^k)$ is greater than a given tolerance and if the maximum number of iterations is less than 100.

4.3. 2D Finite Element program

```
function ut=FEM(coordinates,elements,dirichlet,neumann,ut,u0,t0,t1,th,N)
%Initialisation
```

```
DF=sparse(2*N,2*N);Q=zeros(2*N,1);P=zeros(2*N,1);
list=2*elements(:,[1,1,2,2,3,3])-ones(size(elements,1),1)*[1,0,1,0,1,0];
%Assembly
for j=1:size(elements,1)
 L=list(j,:);
 [DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
end
%Volume forces
for j=1:size(elements,1)
 lcoordinates=coordinates(elements(j,:),:);area=det([1,1,1;lcoordinates'])/2;
 b=area*(1-th)*f(sum(lcoordinates,1)/3,t0)'/3+th*f(sum(lcoordinates,1)/3,t1)'/3;
 P(list(j,:))=P(list(j,:))+repmat(b,3,1);
end
%Neumann conditions
if ~isempty(neumann)
 Nlist=2*neumann(:,[1,1,2,2])-ones(size(neumann,1),1)*repmat([1,0],1,2);
 EV=coordinates(neumann(:,2),:)-coordinates(neumann(:,1),:);
 Lg=sqrt(sum(EV.*EV,2));EV=EV./[Lg,Lg];NV=EV*[0,-1;1,0];
 for j=1:size(neumann,1)
  gs=Lg(j)*((1-th)*g(sum(coordinates(neumann(j,:),:))/2,NV(j,:),t0)'/2 ...
     +th*g(sum(coordinates(neumann(j,:),:))/2,NV(j,:),t1)'/2);
  P(Nlist(j,:))=P(Nlist(j,:))+repmat(gs,2,1);
 end
end
F=Q-P;
%Dirichlet conditions
dnodes=unique(dirichlet);
[W0,M0]=u_D(coordinates(dnodes,:),t0);[W1,M1]=u_D(coordinates(dnodes,:),t1);
W=(1-th)*W0+th*W1;M=(1-th)*M0+th*M1;B=sparse(size(W,1),2*N);
for k = 0:1
 for j = 0:1
  B(1+j:2:size(M,1),2*dnodes-1+k)=diag(M(1+j:2:size(M,1),1+k));
 end
end
mask=find(sum(abs(B)'));freeNodes=find(~sum(abs(B)));
B=B(mask,:);W=W(mask,:);norm0=norm(F(freeNodes));nit=0;
while (norm(F(freeNodes))>10^(-10)+10^(-6)*norm0) & (nit<100)
 nit=nit+1
% calculation of the solution
 F=[DF*ut-F;W];DF=[DF,B';B,sparse(size(B,1),size(B,1))];
 x=DF\F;ut=x(1:2*N);lb=x(2*N+1:size(x,1));
 % Assembly
 DF=sparse(2*N,2*N);Q=zeros(2*N,1);
 for j=1:size(elements,1)
  L=list(j,:);
  [DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
 end
 F=Q-P;
end
if (nit==100)  disp('Fails to converge within 100 iteration steps!'); end
```

4.4. 3D Finite Element program

```
function ut=FEM3_d(coordinates,elements,dirichlet,neumann,ut,u0,t0,t1,th,N,NJ)
%Initialisation
DF=sparse(3*N,3*N);
Q=zeros(3*N,1);P=zeros(3*N,1);
list=3*elements(:,[1,1,1,2,2,2,3,3,3,4,4,4])-ones(NJ,1)*[2,1,0,2,1,0,2,1,0,2,1,0];
%Assembly
for j=1:size(elements,1)
L=list(j,:);
[DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
```

```
end
%Volume forces
for j=1:size(elements,1)
 lcoordinates=coordinates(elements(j,:),:);vol=det([1,1,1,1;lcoordinates'])/6;
 b=vol*(1-th)*f(sum(lcoordinates,1)/3,t0)'/4+th*f(sum(lcoordinates,1)/3,t1)'/4;
 P(list(j,:))=P(list(j,:))+repmat(b,4,1);
end
%Neumann conditions
if ~isempty(neumann)
 Nlist=3*neumann(:,[1,1,1,2,2,2,3,3,3])-ones(size(neumann,1),1)*repmat([2,1,0],1,3);
 for j=1:size(neumann,1)
  NV=cross(coordinates(neumann(j,2),:)-coordinates(neumann(j,1),:), ...
           coordinates(neumann(j,3),:)-coordinates(neumann(j,1),:));
  area=norm(NV)/2;NV=NV/norm(NV);
  gs=area*((1-th)*g(sum(coordinates(neumann(j,:),:))/2,NV,t0)'/2 ...
     +th*g(sum(coordinates(neumann(j,:),:))/2,NV,t1)'/2);
  P(Nlist(j,:))=P(Nlist(j,:))+repmat(gs,3,1);
 end
end
F=Q-P;
%Dirichlet conditions
dnodes=unique(dirichlet);
[W0,M0]=u_D(coordinates(dnodes,:),t0);[W1,M1]=u_D(coordinates(dnodes,:),t1);
W=(1-th)*W0+th*W1;M=(1-th)*M0+th*M1;B=sparse(size(W,1),3*N);
for k = 0:2
 for j = 0:2
  B(1+j:3:size(M,1),3*dnodes-2+k)=diag(M(1+j:3:size(M,1),1+k));
 end
end
mask=find(sum(abs(B)'));freeNodes=find(~sum(abs(B)));
B=B(mask,:);W=W(mask,:);norm0=norm(F(freeNodes));nit=0;
while (norm(F(freeNodes))>10^(-10)+10^(-6)*norm0) & (nit<100)
 nit=nit+1
% Calculating the solution
 F=[DF*ut-F;W];DF=[DF,B';B,sparse(size(B,1),size(B,1))];
 x=DF\F;ut=x(1:3*N);lb=x(3*N+1:size(x,1));
% Assembly
 DF=sparse(3*N,3*N);Q=zeros(3*N,1);
 for j=1:size(elements,1)
  L=list(j,:);
  [DF(L,L),Q(L)]=stima(DF(L,L),Q(L),coordinates(elements(j,:),:),ut(L),u0(L),j);
 end
 F=Q-P;
end
if (nit==100)  disp('Fails to converge within 100 iteration steps!'); end
```

5. TANGENTIAL LINEAR SYSTEM OF EQUATIONS

The tangential stiffness matrix DF in (4.6) as well as the vector F in (4.2) are expressed as a sum over all elements T in $\mathscr{T}$. Each part of the sum defines the local stiffness matrix M and the vector of the right-hand side R for the corresponding element. Let k_1 through k_K be the numbers of the nodes of an element T. There are dK basis functions with support on T, namely, $\boldsymbol{\varphi}_{\pi(T,1)} = \varphi_{k_1}\mathbf{e}_1, \dots, \boldsymbol{\varphi}_{\pi(T,d)} = \varphi_{k_1}\mathbf{e}_d, \dots, \boldsymbol{\varphi}_{\pi(T,2K-1)} = \varphi_{k_K}\mathbf{e}_1, \dots, \boldsymbol{\varphi}_{\pi(T,dK)} = \varphi_{k_K}\mathbf{e}_d$, where $\mathbf{e}_j$ is the j-th unit vector. The function φ_{k_j} is the local scalar hat function of node k_j. The function π maps the indices $1,\dots,dK$ of the local numeration with respect to T to the global numeration. In the sequel we want to compute the local stiffness matrix M and a local vector R for element T. For simplicity we write $\boldsymbol{\varphi}_1,\dots,\boldsymbol{\varphi}_{dK}$ instead of $\boldsymbol{\varphi}_{\pi(T,1)},\dots,\boldsymbol{\varphi}_{\pi(T,dK)}$.

The local vector R is defined as

$$R_r := \int_T \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbb{C}^{-1}\boldsymbol{\sigma}_0) : \boldsymbol{\varepsilon}(\boldsymbol{\varphi}_r)\,\mathrm{d}x, \qquad r = 1,\dots,dK. \tag{5.1}$$

The local stiffness matrix for element T is defined, for $r,s = 1,\dots,dK$, by

$$M_{rs} = \frac{\partial R_r}{\partial U^k_{\vartheta,s}} = \frac{\partial}{\partial U^k_{\vartheta,s}} \left(\int_T \sigma \left(\varepsilon \left(\sum_{\ell=1}^{dK} U^k_{\vartheta,\ell} \boldsymbol{\varphi}_\ell - \mathbf{U}_0 \right) + \mathbb{C}^{-1}\sigma_0 \right) : \varepsilon(\boldsymbol{\varphi}_r)\,\mathrm{d}x \right). \tag{5.2}$$

5.1. Notation for triangular elements

The Matlab implementation of M and R can be done in a compact way by defining a matrix `epsilon` by

$$\begin{aligned} \texttt{epsilon:} &= [\varepsilon_{11}(\varphi_j)\ \varepsilon_{12}(\varphi_j)\ \varepsilon_{21}(\varphi_j)\ \varepsilon_{22}(\varphi_1)]_{j=1}^6 = \frac{1}{2}(\texttt{Deta1} + \texttt{Deta2}) \\ &= \frac{1}{2}([(D\varphi_j)_{11}, (D\varphi_j)_{12}, (D\varphi_j)_{21}, (D\varphi_j)_{22}]_{j=1}^6 \\ &\quad + [(D\varphi_j)_{11}, (D\varphi_j)_{21}, (D\varphi_j)_{12}, (D\varphi_j)_{22}]_{j=1}^6) \end{aligned} \tag{5.3}$$

with

$$\texttt{Deta1} = \begin{bmatrix} \frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} \\ \frac{\partial\varphi_{k_2}}{\partial x} & \frac{\partial\varphi_{k_2}}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial\varphi_{k_2}}{\partial x} & \frac{\partial\varphi_{k_2}}{\partial y} \\ \frac{\partial\varphi_{k_3}}{\partial x} & \frac{\partial\varphi_{k_3}}{\partial y} & 0 & 0 \\ 0 & 0 & \frac{\partial\varphi_{k_3}}{\partial x} & \frac{\partial\varphi_{k_3}}{\partial y} \end{bmatrix}, \quad \texttt{Deta2} = \begin{bmatrix} \frac{\partial\varphi_{k_1}}{\partial x} & 0 & \frac{\partial\varphi_{k_1}}{\partial y} & 0 \\ 0 & \frac{\partial\varphi_{k_1}}{\partial x} & 0 & \frac{\partial\varphi_{k_1}}{\partial y} \\ \frac{\partial\varphi_{k_2}}{\partial x} & 0 & \frac{\partial\varphi_{k_2}}{\partial y} & 0 \\ 0 & \frac{\partial\varphi_{k_2}}{\partial x} & 0 & \frac{\partial\varphi_{k_2}}{\partial y} \\ \frac{\partial\varphi_{k_3}}{\partial x} & 0 & \frac{\partial\varphi_{k_3}}{\partial y} & 0 \\ 0 & \frac{\partial\varphi_{k_3}}{\partial x} & 0 & \frac{\partial\varphi_{k_3}}{\partial y} \end{bmatrix}. \tag{5.4}$$

With the coordinates (x_1,y_1), (x_2,y_2), (x_3,y_3) of the vertices of the element T, the entries of `Deta1` and `Deta2` are stored in the matrix `Dphi`,

$$\texttt{Dphi} = \begin{bmatrix} \frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} \\ \frac{\partial\varphi_{k_2}}{\partial x} & \frac{\partial\varphi_{k_2}}{\partial y} \\ \frac{\partial\varphi_{k_3}}{\partial x} & \frac{\partial\varphi_{k_3}}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.5}$$

With `Dphi` from (5.5) the following Matlab lines calculate the matrices `Deta1` and `Deta2`.

```
Deta1=zeros(6,4);Deta1(1:2:5,1:2)=Dphi;Deta1(2:2:6,3:4)=Dphi;
Deta2=zeros(6,4);Deta2(1:2:5,[1,3])=Dphi;Deta2(2:2:6,[2,4])=Dphi;
```

5.2. Notation for tetrahedral elements

In the three-dimensional case the matrix `epsilon` is defined by

$$
\begin{aligned}
\texttt{epsilon} &:= [\varepsilon_{11}(\varphi_j)\ \varepsilon_{12}(\varphi_j)\ \varepsilon_{13}(\varphi_j)\ \varepsilon_{21}(\varphi_j)\ \varepsilon_{22}(\varphi_j)\ \varepsilon_{23}(\varphi_j)\ \varepsilon_{31}(\varphi_j)\ \varepsilon_{32}(\varphi_j)\ \varepsilon_{33}(\varphi_j)]_{j=1}^{12} \\
&= \frac{1}{2}(\texttt{Deta1} + \texttt{Deta2}) \qquad (5.6)
\end{aligned}
$$

with

$$
\begin{aligned}
\texttt{Deta1} &= [(D\varphi_j)_{11},(D\varphi_j)_{12},(D\varphi_j)_{13},(D\varphi_j)_{21},(D\varphi_j)_{22},(D\varphi_j)_{23},(D\varphi_j)_{31},(D\varphi_j)_{32},(D\varphi_j)_{33}]_{j=1}^{12} \\
&= \begin{bmatrix}
\frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} & \frac{\partial\varphi_{k_1}}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} & \frac{\partial\varphi_{k_1}}{\partial z} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} & \frac{\partial\varphi_{k_1}}{\partial z} \\
\vdots & & & & & & & & \vdots \\
\frac{\partial\varphi_{k_4}}{\partial x} & \frac{\partial\varphi_{k_4}}{\partial y} & \frac{\partial\varphi_{k_4}}{\partial z} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial x} & \frac{\partial\varphi_{k_4}}{\partial y} & \frac{\partial\varphi_{k_4}}{\partial z} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial x} & \frac{\partial\varphi_{k_4}}{\partial y} & \frac{\partial\varphi_{k_4}}{\partial z}
\end{bmatrix} \qquad (5.7)
\end{aligned}
$$

and

$$
\begin{aligned}
\texttt{Deta2} &= [D\varphi_j)_{11},(D\varphi_j)_{21},(D\varphi_j)_{31},(D\varphi_j)_{12},(D\varphi_j)_{22},(D\varphi_j)_{32},(D\varphi_j)_{13},(D\varphi_j)_{23},(D\varphi_j)_{33}]_{j=1}^{12} \\
&= \begin{bmatrix}
\frac{\partial\varphi_{k_1}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial z} & 0 & 0 \\
0 & \frac{\partial\varphi_{k_1}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial z} & 0 \\
0 & 0 & \frac{\partial\varphi_{k_1}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_1}}{\partial z} \\
\vdots & & & & & & & & \vdots \\
\frac{\partial\varphi_{k_4}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial z} & 0 & 0 \\
0 & \frac{\partial\varphi_{k_4}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial z} & 0 \\
0 & 0 & \frac{\partial\varphi_{k_4}}{\partial x} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial y} & 0 & 0 & \frac{\partial\varphi_{k_4}}{\partial z}
\end{bmatrix}. \qquad (5.8)
\end{aligned}
$$

The entries of `Deta1` and `Deta2` are stored in the matrix `Dphi`,

$$
\texttt{Dphi} = \begin{bmatrix}
\frac{\partial\varphi_{k_1}}{\partial x} & \frac{\partial\varphi_{k_1}}{\partial y} & \frac{\partial\varphi_{k_1}}{\partial z} \\
\frac{\partial\varphi_{k_2}}{\partial x} & \frac{\partial\varphi_{k_2}}{\partial y} & \frac{\partial\varphi_{k_2}}{\partial z} \\
\frac{\partial\varphi_{k_3}}{\partial x} & \frac{\partial\varphi_{k_3}}{\partial y} & \frac{\partial\varphi_{k_3}}{\partial z} \\
\frac{\partial\varphi_{k_4}}{\partial x} & \frac{\partial\varphi_{k_4}}{\partial y} & \frac{\partial\varphi_{k_4}}{\partial z}
\end{bmatrix}
= \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix}^{-1}
\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (5.9)
$$

where (x_1,y_1,z_1), (x_2,y_2,z_2), (x_3,y_3,z_3), (x_4,y_4,z_4) are the coordinates of the vertices of the element T. The following Matlab lines calculate the matrices `Deta1` and `Deta2`.

```
Deta1=zeros(12,9);Deta2=zeros(12,9);
Deta1(1:3:10,1:3)=Dphi;Deta1(2:3:11,4:6)=Dphi;Deta1(3:3:12,7:9)=Dphi;
Deta2(1:3:10,[1,4,7])=Dphi;Deta2(2:3:11,[2,5,8])=Dphi;Deta2(3:3:12,[3,6,9])=Dphi;
```

5.3. Local stiffness matrix M and local vector R

The subsequent algorithms perform a fall differentiation between the elastic and the plastic phase. In the plastic phase the local stiffness matrix M and the local vector R for element T are obtained by evaluating (5.2) and (5.1) for the different cases of hardening. In the elastic phase M and R are well known, c.f. [3]. The Matlab routine `stima.m` calculates the local stiffness matrix M according to (5.2) and the first summand of (5.1). With M and R a step of the assembling procedure is performed in `stima.m`.

Example 5.1 (Perfect viscoplasticity for 2D and 3D). The Matlab program calculating the local stiffness matrix and the vector R depending on the element T is listed at the end of this example for the 2D and 3D case, together with a special implementation of the deviator and trace function, called `dev2` and `tr2` in the 2D case and `dev3` and `tr3` in the 3D case. The 2D program is described below. In line 2 global variables are transfered to the routine,

$$C_1 := \lambda + 2\mu/d, \quad C_2 := \nu/(\beta\nu + \vartheta k), \quad C_3 := \vartheta k \sigma_y/(\beta\nu + \vartheta k), \quad C_4 := 2\mu.$$

In lines 4–7 the matrix `epsilon` is calculated as described in the previous section. In line 7 we define the matrix $v := \varepsilon(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbb{C}^{-1}\boldsymbol{\sigma}_0$ and store it row wise. In line 8 the area of element T is determined. In line 9 a fall differentiation is performed. If $|\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0$ the plastic phase occurs, otherwise the elastic phase. Depending on the result of this distinction, a constant C_5 and a vector C_6 are defined by

$$C_5 := \begin{cases} C_2 + C_3/|\operatorname{dev}(v)| & \text{if } |\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0 \\ 2\mu & \text{else} \end{cases} \tag{5.10}$$

$$C_6 := \begin{cases} C_3/|\operatorname{dev}(v)|^3 [\mathtt{dev2}(\varepsilon(\varphi_j)) : \mathtt{dev2}(v)]_{j=1}^6 & \text{if } |\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0 \\ [0\,0\,0\,0\,0\,0]^T & \text{else.} \end{cases} \tag{5.11}$$

In lines 14 and 15 an update of the local stiffness matrix and the first summand of the vector R is performed. The components of the local stiffness matrix read

$$M_{jk} = |T|(C_1 \operatorname{tr}(\varepsilon(\varphi_j)) \operatorname{tr}(\varepsilon(\varphi_k)) + C_5 \operatorname{dev}(\varepsilon(\varphi_j)) : \varepsilon(\varphi_k) - (C_6)_j \operatorname{dev}(v) : \varepsilon(\varphi_k)). \tag{5.12}$$

The components of the first summand of the local vector R read

$$R_j = |T|(C_1 \operatorname{tr}(v) \operatorname{tr}(\varepsilon(\varphi_j)) + C_5 \operatorname{dev}(v) : \varepsilon(\varphi_j)). \tag{5.13}$$

The Matlab routines `stima.m`, `tr2.m` and `dev2.m` for the 2D case are listed below.

```
function [M,F]=stima(M0,F0,lcoordinates,ut,u0,j);
global lambda mu sigma_y C1 C2 C3 e0
Dphi=inv([1,1,1;lcoordinates'])*[0,0;eye(2)];
eps=zeros(6,4);Deta1=zeros(6,4);Deta2=zeros(6,4);
Deta1(1:2:5,1:2)=Dphi;Deta1(2:2:6,3:4)=Dphi;
Deta2(1:2:5,[1,3])=Dphi;Deta2(2:2:6,[2,4])=Dphi;
eps=(Deta1+Deta2)/2;v=(ut-u0)'*eps+e0(j,:);
T=det([1,1,1;lcoordinates'])/2;
if  norm(dev2(v))-sigma_y/(2*mu)>0
 C5=C2+C3/norm(dev2(v));C6=C3/norm(dev2(v))^3*(dev2(eps)*dev2(v)');
else
 C5=2*mu;C6=zeros(6,1);
end
M=M0+T*(C1*tr2(eps)*tr2(eps)'+C5*dev2(eps)*eps'-C6*dev2(v)*eps');
F=F0+T*(C1*tr2(v)*tr2(eps)+C5*eps*dev2(v)');

function tr2=tr2(A);
 tr2=A(:,1)+A(:,4);

function dev2=dev2(A);
 dev2=[(A(:,1)-A(:,4))/2,A(:,2),A(:,3),(A(:,4)-A(:,1))/2];
```

The Matlab routines `stima.m`, `tr3.m` and `dev3.m` for the 3D case are listed below.

```
function [M,F]=stima(M0,F0,loccoordinates,u1,u0,j);
global mu sigma_y C1 C2 C3 C4 e0
Dphi=inv([1,1,1,1;loccoordinates'])*[zeros(1,3);eye(3)];
eps=zeros(12,9);Deta1=zeros(12,9);Deta2=zeros(12,9);
Deta1(1:3:10,1:3)=Dphi;Deta1(2:3:11,4:6)=Dphi;Deta1(3:3:12,7:9)=Dphi;
Deta2(1:3:10,[1,4,7])=Dphi;Deta2(2:3:11,[2,5,8])=Dphi;Deta2(3:3:12,[3,6,9])=Dphi;
eps=(Deta1+Deta2)/2;v=(u1-u0)'*eps+e0(j,:);
T=det([1,1,1,1;loccoordinates'])/6;
if  norm(dev3(v))-sigma_y/(2*mu)>0
 C5=C2+C3/norm(dev3(v));C6=C3/norm(dev3(v))^3*(dev3(eps)*dev3(v)');
else
 C5=C4;C6=zeros(12,1);
end
M=M0+T*(C1*tr3(eps)*tr3(eps)'+C5*dev3(eps)*eps'-C6*dev3(v)*eps');
F=F0+T*(C1*tr3(v)*tr3(eps)+C5*eps*dev3(v)');

function tr=tr3(A);
 tr=A(:,1)+A(:,5)+A(:,9)

function dev=dev3(A);
 dev=A-tr3(A)*[1,0,0,0,1,0,0,0,1];
```

Example 5.2 (Isotropic hardening in 2D). The Matlab program calculating the local stiffness matrix and the local vector R is listed at the end of this example. In line 2 global variables C_1 up to C_6 are transfered to the routine,

$$C_1 := \lambda + \mu, \qquad C_2 := \frac{\nu}{2\mu}(1 + H^2\sigma_y^2) + \vartheta k \left(1 + \frac{1}{2\mu} H_1 H^2 \sigma_y^2\right)$$

$$C_3 := \vartheta k \sigma_y (1 + \alpha_0 H), \qquad C_4 := H_1 H^2 \vartheta k \sigma_y^2 + \nu(1 + H^2\sigma_y^2)$$

$$C_5 := 2\mu, \qquad C_6 := \frac{1}{2\mu}(1 + \alpha_0 H)\sigma_y.$$

Lines 3–8 are analog to the last example and perform the calculation of `epsilon`, define the vector v and determine the element area T. In line 9 the fall differentiation is performed. If $|\operatorname{dev}(v)| - C_6 > 0$ the plastic phase occurs, otherwise the elastic phase. Depending on the result of this distinction, constants C_7 and a vector C_8 are defined by

$$C_7 := \begin{cases} C_3/(C_2|\operatorname{dev}(v)|) + C_4/C_2 & \text{if } |\operatorname{dev}(v)| - C_6 > 0 \\ C_5 & \text{else} \end{cases} \tag{5.14}$$

$$(C_8)_j := \begin{cases} C_3/(C_2|\operatorname{dev}(v)|^3)\mathtt{dev2}(\varepsilon(\varphi_j)) : \mathtt{dev2}(v) & \text{if } |\operatorname{dev}(v)| - C_6 > 0 \\ [0\,0\,0\,0\,0\,0]^T & \text{else.} \end{cases} \tag{5.15}$$

In lines 14 and 15 an update of the local stiffness matrix and the first summand of the local vector R is performed. The components of the local stiffness matrix read

$$M_{jk} = |T|(C_1 \operatorname{tr}(\varepsilon(\varphi_j)) \operatorname{tr}(\varepsilon(\varphi_k)) + C_7 \operatorname{dev}(\varepsilon(\varphi_j)) : \varepsilon(\varphi_k) - (C_8)_j \operatorname{dev}(v) : \varepsilon(\varphi_k))\,. \tag{5.16}$$

The components of the first summand of the vector R read

$$R_j = |T|(C_1 \operatorname{tr}(v) \operatorname{tr}(\varepsilon(\varphi_j)) + C_7 \operatorname{dev}(v) : \varepsilon(\varphi_j))\,. \tag{5.17}$$

The Matlab routine `stima.m` is listed below.

```
 1 function [M,R]=stima(M0,R0,loccoordinates,u1,u0,j);
 2 global C1 C2 C3 C4 C5 C6 e0
 3 Dphi=inv([1,1,1;loccoordinates'])*[0,0;eye(2)];
 4 eps=zeros(6,4);Deta1=zeros(6,4);Deta2=zeros(6,4);
 5 Deta1(1:2:5,1:2)=Dphi;Deta1(2:2:6,3:4)=Dphi;
 6 Deta2(1:2:5,[1,3])=Dphi;Deta2(2:2:6,[2,4])=Dphi;
 7 eps=(Deta1+Deta2)/2;v=(u1-u0)'*eps+e0(j,:);
 8 T=det([1,1,1;loccoordinates'])/2;
 9 if norm(dev2(v))-C6(j)>0
10  C7=C3(j)/(C2*norm(dev2(v)))+C4/C2;C8=C3(j)/(C2*norm(dev2(v))^3)*dev2(eps)*dev2(v)';
11 else
12  C7=C5;C8=zeros(6,1);
13 end
14 M=M0+T*(C1*tr2(eps)*tr2(eps)'+C7*dev2(eps)*eps'+C8*dev2(v)*eps');
15 R=R0+T*(C1*tr2(v)*tr2(eps)+C7*eps*dev2(v)');
```

Example 5.3 (Kinematic hardening in 2D). The Matlab program calculating the local stiffness matrix and the local vector R is listed at the end of this example. In line 2 global variables C_1 up to C_4 are transfered to the routine,

$$C_1 := \lambda + \mu, \qquad C_2 := \frac{\vartheta k k_1 + 2\nu}{\vartheta k + \vartheta k k_1/(2\mu) + \nu/\mu}$$

$$C_3 := \frac{\vartheta k \sigma_y}{\vartheta k + \vartheta k k_1/(2\mu) + \nu/\mu}, \qquad C_4 := 2\mu\,.$$

Lines 3–8 are analog to the last example. The strain tensor `epsilon` is calculated, and matrices v and $v1$ are defined, with $v := \boldsymbol{\varepsilon}(\mathbf{U}_\vartheta - \mathbf{U}_0) + \mathbb{C}^{-1}\boldsymbol{\sigma}_0$, $v1 :=$

$-1/(2\mu)\alpha_0$. In line 9 the fall differentiation is performed. If $|\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0$ the plastic phase occurs, otherwise the elastic phase. Depending on the result of this distinction, a constant C_5, a vector C_6 and a Kronecker delta d are defined by

$$C_5 := \begin{cases} C_3/(|\operatorname{dev}(v)|) + C_2 & \text{if } |\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0 \\ C_4 & \text{else} \end{cases} \tag{5.18}$$

$$(C_6)_j := \begin{cases} C_3/|\operatorname{dev}(v)|^3 \mathtt{dev2}(\varepsilon(\varphi_j)) : \mathtt{dev2}(v) & \text{if } |\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0 \\ [0\,0\,0\,0\,0\,0]^T & \text{else} \end{cases} \tag{5.19}$$

$$d := \begin{cases} 1 & \text{if } |\operatorname{dev}(v)| - \sigma_y/(2\mu) > 0 \\ 0 & \text{else.} \end{cases} \tag{5.20}$$

In lines 14 and 15 an update of the local stiffness matrix and the first summand of the local vector R is performed. The components of the local stiffness matrix read

$$M_{jk} = |T|(C_1 \operatorname{tr}(\varepsilon(\varphi_j)) \operatorname{tr}(\varepsilon(\varphi_k)) + C_5 \operatorname{dev}(\varepsilon(\varphi_j)) : \varepsilon(\varphi_k) - (C_6)_j \operatorname{dev}(v) : \varepsilon(\varphi_k)). \tag{5.21}$$

The components of the first summand of the vector R read

$$R_j = |T|(C_1 \operatorname{tr}(v) \operatorname{tr}(\varepsilon(\varphi_j)) + C_5 \operatorname{dev}(v) : \varepsilon(\varphi_j) + d \operatorname{dev}(\alpha_0) : \varepsilon(\varphi_j)). \tag{5.22}$$

The Matlab routine `stima.m` is listed below.

```
 1 function [M,R]=stima(M0,R0,loccoordinates,u1,u0,j);
 2 global mu sigma_y C1 C2 C3 C4 e0 al0
 3 Dphi=inv([1,1,1;loccoordinates'])*[0,0;eye(2)];
 4 eps=zeros(6,4);Deta1=zeros(6,4);Deta2=zeros(6,4);
 5 Deta1(1:2:5,1:2)=Dphi;Deta1(2:2:6,3:4)=Dphi;
 6 Deta2(1:2:5,[1,3])=Dphi;Deta2(2:2:6,[2,4])=Dphi;
 7 eps=(Deta1+Deta2)/2;v0=(u1-u0)'*eps+e0(j,:);v1=-1/(2*mu)*al0(j,:);v=v0+v1;
 8 T=det([1,1,1;loccoordinates'])/2;
 9 if norm(dev2(v))-sigma_y/(2*mu)>0
10  C5=C2+C3/norm(dev2(v));C6=(C3/norm(dev2(v))^3)*dev2(eps)*dev2(v)';d=1;
11 else
12  C5=C4;C6=zeros(6,1);d=0;
13 end
14 M=M0+T*(C1*tr2(eps)*tr2(eps)'+C5*dev2(eps)*eps'-C6*dev2(v)*eps');
15 R=R0+T*(C1*tr2(v0)*tr2(eps)+C5*eps*dev2(v+d*v1)'+d*eps*dev2(al0(j,:))');
```

6. POSTPROCESSING

6.1. Displaying the solution

Our two dimensional problems model the plain stress condition. In that case, the complete stress tensor $\sigma \in \mathbb{R}^{3\times 3}_{\text{sym}}$ has the form

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & 0 \\ \sigma_{12} & \sigma_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

with $\sigma_{33} = \frac{1}{2}\lambda/(\mu+\lambda)(\sigma_{11}+\sigma_{22})$. It then follows for $|\operatorname{dev}\sigma|^2$, where $\operatorname{dev} A := A - \frac{1}{3}\operatorname{tr} A Id_{3\times 3}$ and $|A| := \left(\sum_{j,k=1}^{n} A_{jk}^2\right)^{1/2}$ is the Frobenius norm, that

$$|\operatorname{dev}\sigma|^2 = \left(\frac{\mu^2}{6(\mu+\lambda)^2} + \frac{1}{2}\right)(\sigma_{11}+\sigma_{22})^2 + 2(\sigma_{12}^2 - \sigma_{11}\sigma_{22}).$$

The function `show.m`, listed at the end of this subsection, calculates the variable `AvS` in lines 2–9, which stores the nodal values of σ_h^*. Here σ_h is the stress calculated by the finite element method and σ_h^* is a smoother approximation of the discrete stress σ_h, which may be used for error control [8]. Line 10 determines the shear energy density $|\operatorname{dev}\sigma_h^*|^2/(4\mu)$ and stores it in the variable `AvC`. In line 13 the Matlab routine `trisurf` is used to draw the deformed mesh with a magnification of the displacement, which is fixed in line 11. Grey tones are attached to the displayed meshes which are proportional to the elastic shear energy density $|\operatorname{dev}\sigma_h^*|^2/(4\mu)$.

```
 1  function show(coordinates,elements,Sigma,u,lambda,mu)
 2   AreaOmega=zeros(size(coordinates,1),1);
 3   AvS=zeros(size(coordinates,1),4);
 4   for j=1:size(elements,1)
 5    area=det([1,1,1;coordinates(elements(j,:),:)'])/2;
 6    AreaOmega(elements(j,:))=AreaOmega(elements(j,:))+area;
 7    AvS(elements(j,:),:)=AvS(elements(j,:),:) +area*[1;1;1]*Sigma(j,:);
 8   end;
 9   AvS=AvS./(AreaOmega*[1,1,1,1]);
10   AvC=(mu/(24*(mu+lambda)^2)+1/(8*mu))*(AvS(:,1) ...
         +AvS(:,4)).^2+1/(2*mu)*(AvS(:,2).^2-AvS(:,1).*AvS(:,4));
11   magnify=20;
12   colormap(1-gray);
13   trisurf(elements,magnify*u(1:2:size(u,1))+coordinates(:,1),...
            magnify*u(2:2:size(u,1))+coordinates(:,2),...
14   zeros(size(coordinates,1),1),AvC,'facecolor','interp');
15   view(0,90);
16   colorbar('vert')
```

The Matlab routine `trisurf(elements,x,y,z,AvC,'facecolor','interp')` is used to draw triangulations for equal types of elements. Every row of the matrix `elements` determines one polygon where the *x*-, *y*- and *z*-coordinate of each corner of this polygon is given by the corresponding entry in `x`, `y` and `z`. For a 2D-problem, we use for the *z*-coordinate the same value (zero) in all mesh points. The values `AvC` together with the options `'facecolor','interp'` determine the grey tone of the area. The grey tone of each node is determined by `AvC`, that of the rest of the area is linearly interpolated.

6.2. *A posteriori* error indication

The averaged stress field $\boldsymbol{\sigma}_h^*$, allows an a posteriori error estimation by comparing it to the discrete (discontinuous) stress $\boldsymbol{\sigma}_h$. For one time step, the computable quantity

$$\eta_h = \|\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_h^*\|_{L^2(\Omega)} \tag{6.1}$$

defines the error estimate [8] (for pure Dirichlet conditions – Neumann boundary conditions require modifications [9])

$$\| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \|_{L^2(\Omega)} \leqslant C\eta_h. \tag{6.2}$$

Here, $\boldsymbol{\sigma}_h^*$ may be any piecewise affine and globally continuous stress approximation. It is emphasized that the error bound (6.2) holds only if the time-discretization error is neglected. It is generally believed that the accumulation error (in time) cannot be captured by averaging (in space). However the quantity η_h may monitor the local spatial approximation error and a large size of η_h indicates a larger spatial error and motivates refinements.

Theorem 6.1 (Efficeny). *Assume that the stress field $\boldsymbol{\sigma}$ at time t is smooth, i.e. $\boldsymbol{\sigma} \in H^1(\Omega;\mathbb{R}^{d\times d}_{\mathrm{sym}})$, then η_h is an efficient error estimator up to higher order terms in the sense that*

$$\eta_h \leqslant 4\| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \|_{L^2(\Omega)} + h.o.t. \tag{6.3}$$

where generically $h.o.t. = O(\| h^2 \mathbf{D}\boldsymbol{\sigma} \|_{L^2(\Omega)}) \ll \| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \|_{L^2(\Omega)}$.

Proof. The triangle inequality provides, for all $\boldsymbol{\tau}_h$ which are globally continous and $\mathscr{T}$-piecewise affine (like $\boldsymbol{\sigma}_h^*$), that

$$\| \boldsymbol{\sigma}_h - \boldsymbol{\tau}_h \|_{L^2(\Omega)} \leqslant \| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \|_{L^2(\Omega)} + \| \boldsymbol{\sigma} - \boldsymbol{\tau}_h \|_{L^2(\Omega)} \leqslant \| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \|_{L^2(\Omega)} + h.o.t. \tag{6.4}$$

for the nodal interpolation $\boldsymbol{\tau}_h = I\boldsymbol{\sigma}$ of $\boldsymbol{\sigma}$ [5]. One can prove that (with $\boldsymbol{\tau}_h$ globaly continous and $\mathscr{T}$-piecewise affine but arbitrary otherwise)

$$\eta_M := \min_{\boldsymbol{\tau}_h} \| \boldsymbol{\sigma}_h - \boldsymbol{\tau}_h \|_{L^2(\Omega)} \leqslant \| \boldsymbol{\sigma}_h - \boldsymbol{\sigma}_h^* \|_{L^2(\Omega)} \leqslant 4\eta_M .$$

This is shown for ab arbitrary constant in [6,14,16] and for the constant 4 and quite general boundary conditions in [7]. The main argument is a (local) inverse estimate, caused by equivalence of norms on finite dimensional vector spaces.

The combination of the two inequalities leads to

$$\eta_h = \| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h^* \|_{L^2(\Omega)} \leqslant 4\eta_M \leqslant 4\| \boldsymbol{\sigma} - \boldsymbol{\sigma}_h^* \|_{L^2(\Omega)} + 4\| \boldsymbol{\sigma} - \boldsymbol{\tau}_h \|_{L^2(\Omega)}$$

for any $\boldsymbol{\tau}_h$, in particular for $\boldsymbol{\tau}_h = I\boldsymbol{\sigma}$. □

It is emphasized once more that the reliability (6.2) is problematic and holds only for one time step and for large hardening and large viscosity [8].

The Matlab routine at the end of this subsection calculates `eta1` at time $t = t_1$. Lines 3–4 initialise $|\omega_z|$, the patch of the hat function at node z, which is stored in `AreaOmega` and $\boldsymbol{\sigma}_h^*$, which is stored in `AvS`. Line 5 determines weights and barycentric coordinates by a two dimensional quadrature rule. In our algorithm we have $\boldsymbol{\sigma}_h^*(z) = \int_{\omega_z} \boldsymbol{\sigma}_h \, \mathrm{d}x / |\omega_z|$. Lines 6–10 calculate the area of the elements (line 7), the patch size $|\omega_z|$ (line 8) and $\int_{\omega_z} \boldsymbol{\sigma}_h \, dx$ (line 9) by a loop over all elements. In line 11, $\boldsymbol{\sigma}_h^*$ is obtained by dividing $\int_{\omega_z} \boldsymbol{\sigma}_h \, dx$ through the patch size. Lines 12–16 compute `eta1`, where for $\boldsymbol{\sigma}_h^*$ four interpolation points are used on each element T.

```
1  function eta=APosteriori(coordinates,elements,neumann,sigma,t);
2  eta=0;
3  AreaOmega = zeros(size(coordinates,1),1);
4  AvS = zeros(size(coordinates,1),4);
5  [weights,baryc]=quadratur2D(2);
6  for j = 1:size(elements,1)
7   area(j)=det([1,1,1;coordinates(elements(j,:),:)'])/2;
8   AreaOmega(elements(j,:))=AreaOmega(elements(j,:))+area(j);
9   AvS(elements(j,:),:)=AvS(elements(j,:),:) +area(j)*[1;1;1]*sigma(j,:);
10 end;
11 AvS=AvS./(AreaOmega*[1,1,1,1]);
12 for j=1:size(elements,1)
13  eta=eta+area(j)*weights*sum((ones(size(weights,2),1)*sigma(j,:)-...
14                     baryc'*AvS(elements(j,:),:)).^2,2);
15 end
16 eta=eta^(1/2);
```

7. NUMERICAL EXAMPLES

7.1. On a plate with circular hole

A two dimensional squared plate with a hole, $\Omega = (-2,2)^2 \setminus B(0,1)$ is submitted to time dependent surface forces $g(t) = (600t)n$ at the top ($y = 2$) and the bottom ($y = -2$), where n denotes the outer normal to $\partial\Omega$. The rest of the boundary is traction free. As the problem is symmetric only a quarter of Ω is discretised. The boundary conditions are specified in files `u_d.m`, `f.m`, and `g.m`:

```
 function [W,M]=u_D(x,t)
 M=zeros(2*size(x,1),2);
 W=zeros(2*size(x,1),1);
% symmetry condition on the x-axis
 tmp=find(x(:,1)>0 & x(:,2)==0);
 M(2*tmp-1,1:2)=ones(size(tmp,1),1)*[0 1];
 W(2*tmp-1,1)=zeros(size(tmp,1),1);
% symmetry condition on the y-axis
 tmp=find(x(:,2)>0 & x(:,1)==0);
 M(2*tmp-1,1:2)=ones(size(tmp,1),1)*[1 0];
 W(2*tmp-1,1)=zeros(size(tmp,1),1);

 function volforce = f(x);
 volforce=zeros(size(x,1),2);

 function sforce=g(x,n,t)
 sforce=zeros(size(x,1),2);
 if (n(2)==1)
 sforce(:,2)=600*t;
 end
```

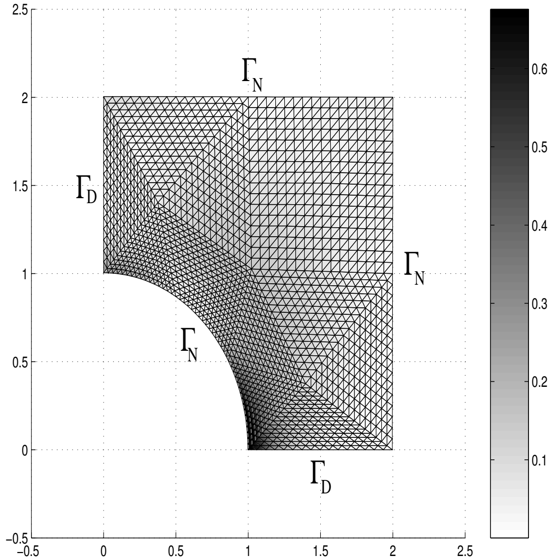


Figure 3. Deformed mesh for membrane with hole in the example of Subsection 7.1 calculated with 3474 degrees of freedom. The grey tone visualises $|\operatorname{dev}\sigma_h^*|^2/(4\mu)$ as described in Subsection 6.1.

This numerical example from [17] models perfect viscoplasticity with Young's modulus $E = 206900$, Poisson's ratio $\nu = 0.29$, yield stress $\sigma_y = 450$ and vanishing initial data for the stress tensor σ_0.

The solution is calculated with 3474 degrees of freedom in the time interval from $t = 0$ to $t = 0.4$ by the implicit Euler method in 8 uniform steps of length $k = 1/20$. The material remains elastic between $t = 0$ and $t = 0.2$. Figure 3 shows the deformed mesh at $t = 0.2$, Table 1 shows the number of iterations in Newton's method and the estimated error for each time step.

Table 1.
Number of iterations and indicated error for membrane with hole in the example of Subsection 7.1 for various time steps.

step	iterations	$\eta = \|\|\sigma_h - \sigma_h^*\|\|_2/\|\|\sigma_h\|\|_2$
1	1	0.0432
2	1	0.0432
3	1	0.0432
4	5	0.0436
5	5	0.0462
6	5	0.0505
7	6	0.0565
8	6	0.0642

7.2. Numerical examples for Cooks' membrane problem

In the second numerical example we solve the viscoplastic problem with isotropic hardening for a two-dimensional model of a tapered panel of plexi glass described by $\Omega = \text{conv}\,\{(0,0),\,(48,44),\,(48,60),\,(0,44)\}$. The panel is clamped at one end ($x=0$) and subjected to a shearing load $g=(0,t)$ on the opposite end ($x=48$) with vanishing volume force f. The boundary conditions are specified in files `u_d.m`, `f.m`, and `g.m`:

```
function [W,M]=u_d(x,t)
M= zeros(2*size(x,1),2);
W= zeros(2*size(x,1),1);
M(1:2:2*size(x,1)-1,1)=ones(size(x,1),1);
M(2:2:2*size(x,1),2)=ones(size(x,1),1);;
value=zeros(size(x,1),1);
W(1:2:2*size(u,1)-1,1)=value(:,1);
W(2:2:2*size(u,1),1)=value(:,2);

function volforce=f(x);
volforce= zeros(size(x,1),2);

function sforce=g(u,n,t)
sforce=zeros(size(x,1),2);
sforce(find(n(:,2)==1),2)=exp(2*t);
```

For plexi glass we set $E = 2900$ and $\nu = 0.4$. This example is often referred to as the Cooks' membrane problem and constitutes a standard test for bending dominated response. We calculate the solution with 8450 degrees of freedom in the time interval [0,0.1] with vanishing initial data for the displacement u_0, the stress tensor σ_0 and the hardening parameter α_0. The time step in the implicit Euler method is $k = 0.01$. The plastic parameters are $H = 1000$, $H_1 = 1$ and $\sigma_y = 0.1$. The problem becomes plastic at $t = 0.02$. Figure 4 shows the deformed mesh at $t = 0.1$ for 2178 degrees of freedom, while Table 2 shows the number of iterations in Newton's method and the estimated error for each time step.

Table 2.
Number of iterations and indicated error for Cook's membrane problem in the example of Subsection 7.2 for various time steps.

step	iterations	$\eta = \|\|\sigma_h - \sigma_h^*\|\|_2/\|\|\sigma_h\|\|_2$
1	1	0.0674
2	7	0.0674
3	11	0.0675
4	12	0.0675
5	13	0.0680
6	14	0.0684
7	14	0.0695
8	14	0.0706
9	14	0.0723
10	14	0.0737

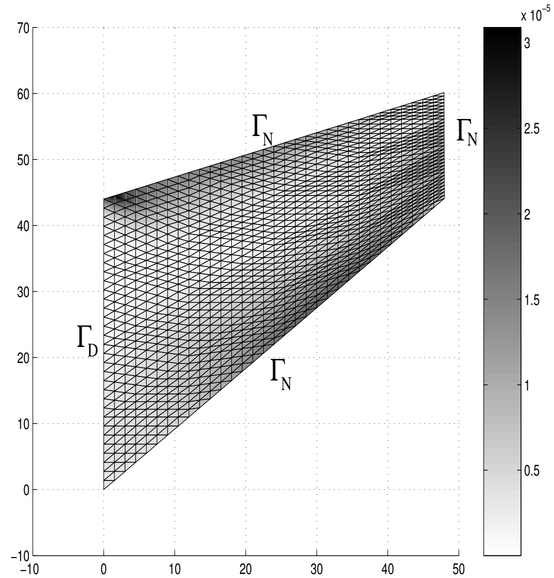


Figure 4. Deformed mesh for Cooks' membrane problem in the example of Subsection 7.2 calculated with 2178 degrees of freedom. The grey tone visualises $|\operatorname{dev}\sigma_h^*|^2/(4\mu)$ as described in Subsection 6.1.

7.3. Numerical example for axissymmetric ring

The third numerical example is taken from [4]. We solve a viscoplastic model with kinematic hardening. The geometry is a two-dimensional section of a long tube with inner radius 1 and outer radius 2 (Fig. 6). We have no volume force, $f = 0$ but time depending surface forces $g_1(r,\varphi,t) = te_r$ and $g_2(r,\varphi,t) = -t/4e_r$. The system is required to keep centered at the origin and rotation is prohibited. The boundary conditions are specified in files `u_d.m`, `f.m`, and `g.m`:

```
 function [W,M] = u_D(x,t)
 M = zeros(2*size(x,1),2);
 W = zeros(2*size(x,1),1);
% symmetry condition on the x-axis
 tmp = find(x(:,1)>0 & x(:,2)==0);
 M(2*tmp-1,1:2) = ones(size(tmp,1),1)*[0 1];
 W(2*tmp-1,1) = zeros(size(tmp,1),1);
% symmetry condition on the y-axis
 tmp = find(x(:,2)>0 & x(:,1)==0);
 M(2*tmp-1,1:2) = ones(size(tmp,1),1)*[1 0];
 W(2*tmp-1,1) = zeros(size(tmp,1),1);

 function volforce=f(x,t)
 volforce=zeros(size(x,1),2);
```

```
function sforce = g(x,n,t)
[phi,r]=cart2pol(x(:,1),x(:,2));
phiNeg=find(phi<0);
phi(phiNeg)=phi(phiNeg)...
           +2*pi*ones(size(phiNeg));
for i=1:size(x,1)
 if r(i)>.5 & r(i)<1.5
  k(i,:)=t*[cos(phi(i)),sin(phi(i))];
 elseif r(i)>1.5 & r(i)<2.5
k(i,:)=-t/4*[cos(phi(i)),sin(phi(i))];
 end
 if x(i,1)==0 | x(i,2)==0
  k(i,:)=[0 0];
 end
end
```

The exact solution is

$$\begin{aligned} u(r,\varphi,t) &= u_r(r,t)e_r \\ \sigma(r,\varphi,t) &= \sigma_r(r,t)e_r\otimes e_r + \sigma_\varphi e_\varphi \otimes e_\varphi \\ p(r,\varphi,t) &= P_r(r,t)(e_r\otimes e_r - e_\varphi\otimes e_\varphi) \end{aligned} \tag{7.1}$$

with $e_r = (\cos\varphi, \sin\varphi)$, $e_\varphi = (-\sin\varphi, \cos\varphi)$ and with $a = \mu+\lambda$, $\varkappa = 2\mu/(2\mu+\lambda)$,

$$u_r(r,t) = \begin{cases} \dfrac{t}{2\mu r} - \dfrac{1}{3}\varkappa I(R(t))\left(r + \dfrac{4a}{\mu r}\right), & r \geqslant R(t) \\ \dfrac{t}{2\mu r} - \dfrac{1}{3}\varkappa I(R(t))\left(4r + \dfrac{4a}{\mu r}\right) + \varkappa I(r) r, & r < R(t) \end{cases} \tag{7.2}$$

$$\sigma_r(r,t) = \begin{cases} -\dfrac{t}{r^2} - \dfrac{8}{3}a\varkappa I(R(t))\left(\dfrac{1}{4} - \dfrac{1}{r^2}\right), & r \geqslant R(t) \\ -\dfrac{t}{r^2} - \dfrac{8}{3}a\varkappa I(R(t))\left(1 - \dfrac{1}{r^2}\right) + 2a\varkappa I(r), & r < R(t) \end{cases} \tag{7.3}$$

$$\sigma_\varphi(r,t) = \frac{\partial(r\sigma_r)}{\partial r} \tag{7.4}$$

$$P_r(r,t) = \begin{cases} 0, & r \geqslant R(t) \\ \dfrac{\sigma_y}{\sqrt{2}(a\varkappa + k_1)}(1 - R^2/r^2), & r < R(t) \end{cases} \tag{7.5}$$

$$I(r) = \frac{\sigma_y}{\sqrt{2}(a\varkappa + k_1)}(\ln r + 1/2(R^2/r^2 - R^2)). \tag{7.6}$$

With $\alpha = 4a\varkappa/(3(a\varkappa + k_1))$, the radius of the plastic boundary $R(t)$ is the positive root of

$$f(R) = -2\alpha \ln R + (\alpha - 1)R^2 - \alpha + \frac{\sqrt{2}}{\sigma_y}t. \tag{7.7}$$

Table 3.
Number of iterations, indicated error, and exact error for axissymmetric ring in the example of Subsection 7.3. This numerical example validates our claim that η can be an accurate error guess.

step	iterations	$e = \|\|\sigma - \sigma_h\|\|_2 / \|\|\sigma\|\|_2$	$\eta = \|\|\sigma_h - \sigma_h^*\|\|_2 / \|\|\sigma_h\|\|_2$	e/η
1	1	0.0512	0.0510	1.00
...	...	...	...	...
14	1	0.0512	0.0510	1.00
15	3	0.0521	0.0515	1.01
16	4	0.0533	0.0529	1.01
17	4	0.0549	0.0543	1.01
18	4	0.0565	0.0560	1.01
19	4	0.0584	0.0579	1.01

The material parameters are $E = 70000$ for Young's modulus, $\nu = 0.33$ for Poisson's ratio and the yield stress is $\sigma_y = 0.2$.

The solution is calculated in the time interval [0,0.2] with 3202 degrees of freedom and vanishing initial data for the displacement u_0, the stress tensor σ_0 and the hardening parameter α_0. The time step in the implicit Euler method is $k = 0.01$. The hardening parameter is $k_1 = 1$. The material becomes plastic at $t = 0.16$. Table 3 shows the error and the estimated error for the different time steps together with the number of iterations in Newton's method. Figure 5 shows the deformed mesh at $t = 0.18$, displacements magnified by the factor 10.

7.4. Numerical example in three dimensions

As a three-dimensional example we solve a perfect viscoplastic problem on the cube $[0,1]^3 \setminus [0,0.5]^3$. The face with $x = 1$ is Dirichlet boundary, the rest is Neumann boundary. The volume force f is zero while the surface force is $g = tn$, if the normal vector n points in y-direction. The boundary conditions are specified in files `u_d.m`, `f.m`, and `g.m`:

```
function [W,M] = u_d(x,t)
M= zeros(3*size(x,1),3);
W= zeros(3*size(x,1),1);
M(1:3:3*size(x,1)-2,1)= zeros(size(x,1),1);
M(2:3:3*size(x,1)-1,2)= zeros(size(x,1),1);
M(3:3:3*size(x,1),3)=zeros(size(x,1),1);
value=zeros(size(x,1),3);
W(1:3:3*size(x,1)-2,1)=value(:,1);
W(2:3:3*size(x,1)-1,1)=value(:,2);
W(3:3:3*size(x,1),1)=value(:,3);

function volforce = f(x);
volforce= zeros(size(x,1),2);

function sforce = g(u,n,t)
sforce=zeros(size(x,1),2);
sforce(find(n(:,2)==1),2)=t;
```

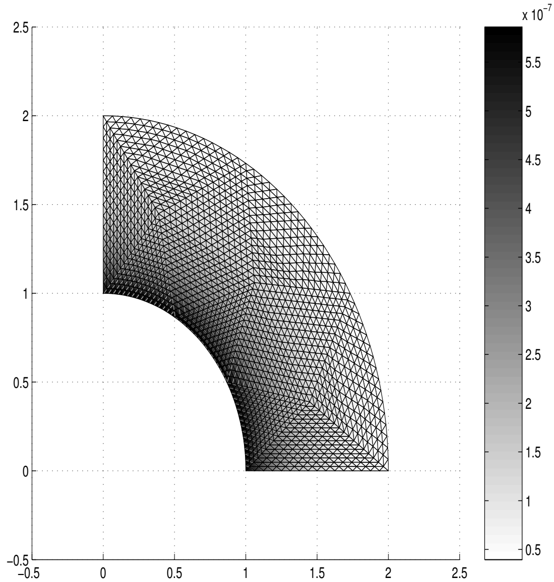
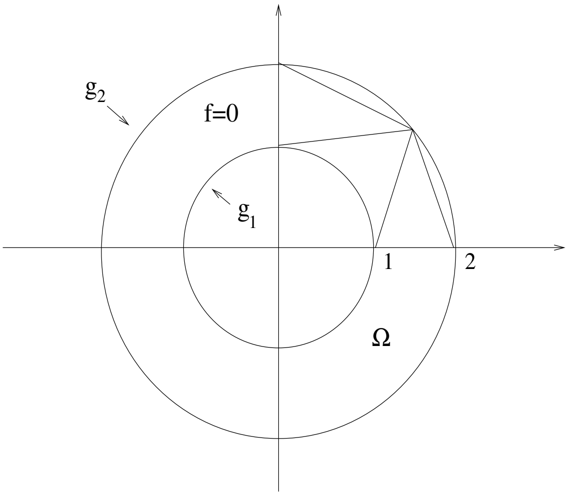


Figure 5. Deformed mesh for axissymmetric ring in the example of Subsection 7.3 calculated with 3202 degrees of freedom. The grey tone visualises $|\operatorname{dev}\sigma_h^*|^2/(4\mu)$ as described in Subsection 6.1.



$$E = 70000$$
$$\nu = 0.33$$
$$\sigma_y = 0.2$$
$$k_1 = 1$$
$$g_1 = te_r$$
$$g_2 = -t/4e_r$$

Figure 6. Axissymmetric ring in the example of Subsection 7.3 with initial triangulation of a quarter of the ring endowed with symmetrie boundary conditions.

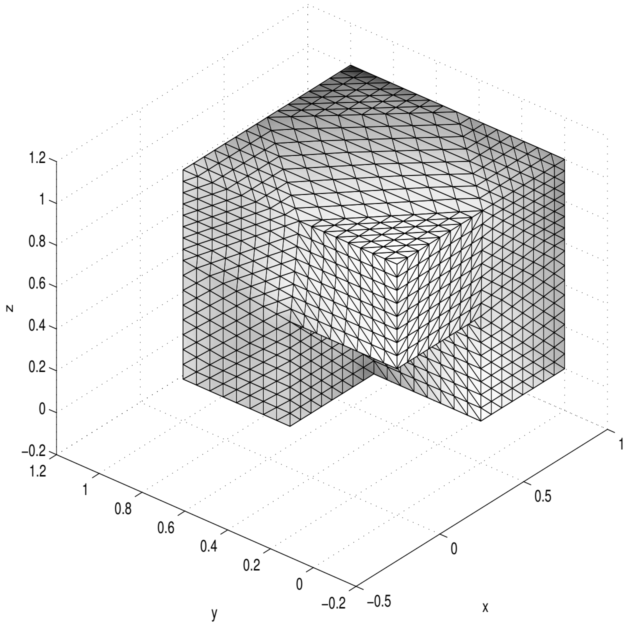


Figure 7. Deformed mesh for 3D example in Subsection 7.4. The grey tone visualises the square mean of the eigenvalues of the stress tensor.

The solution is calculated in the time interval [0,1] with stepsize 0.05 and vanishing initial data for the displacement u_0 and the stress tensor σ_0 and for material paramters $E = 206900$, $\nu = 0.29$, and yield stress $\sigma_y = 450$. Up to $t = 0.2$ s the material is entirely elastic. The deformed mesh for 10359 degrees of freedom at $t = 0.15$ s is shown in Fig. 7.

Acknowledgments. It is our pleasure to thank Dr. Jochen Alberty and Andre Geilenkothen for discussions. The second author (R. K.) thankfully acknowledges the partial support by the German research foundation at the Graduiertenkolleg 357 'Effiziente Algorithmen und Mehrskalenmethoden'. Parts of this work was done at the Institute for Applied Mathematics and Numerical Analysis, Vienna University of Technology whose hospitability is greatfully acknowledged.

REFERENCES

1. M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*. Wiley, 2000.

2. J. Alberty, C. Carstensen, and S. A. Funken, Remarks around 50 lines of Matlab: Finite Element Implementation. *Numer. Algorithms* (1999) **20**, 117 – 137.

3. J. Alberty, C. Carstensen, S. A. Funken, and R. Klose, *Matlab Implementation of the Finite Element Method in Elasticity.* Berichtsreihe des Mathematischen Seminars, Kiel, 2000, 00–21.

4. J. Alberty, C. Carstensen, and D. Zarrabi, Adaptive numerical analysis in primal elastoplasticity with hardening. *Comp. Methods Appl. Mech. Engrg.* (1999) **171**, 175 – 204.

5. S. C. Brenner and L. R. Scott, The mathematical theory of finite element methods. *Texts in Applied Mathematics*, Vol. 15. Springer, New York, 1994.

6. C. Carstensen, On the History and Future of Averaging Techniques in *A Posteriori* FE Error Analysis. In: *ZAMM Proceedings of Annual GAMM Conference*, 2002 (to appear).

7. C. Carstensen, First-order averaging techniques for *a posteriori* Finite Element error control on unstructured grids are efficient and reliable, 2002 (submitted).

8. C. Carstensen and J. Alberty, Averaging techniques for reliable a posteriori FE-error control in elastoplasticity with hardening? *Berichtsreihe des Mathematischen Seminars*, Kiel, 2000, 00–23.

9. C. Carstensen and S. A. Funken, Averaging technique for FE-*a posteriori* error control in elasticity. Part I: Conforming FEM. *Comp. Meth. Appl. Mech. Engrg.* (2001) **190**, 2483 – 2498. Part II: λ-independent estimates. *Comp. Meth. Appl. Mech. Engrg.* (2001) **190**, 4663 – 4675. Part III: Locking-free conforming FEM. *Comp. Meth. Appl. Mech. Engrg.* (2001) **191**, 861 – 877.

10. P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*. North-Holland, 1978.

11. W. Han and B. D. Reddy, *Plasticity: Mathematical Theory and Numerical Analysis*. Springer, New York, 1999.

12. L. Langemyr *et. al.*, *Partial Differential Equation Toolbox User's Guide*. The Math. Works, 1995.

13. D. Redfern and C. Campbell, *The Matlab5 Handbook*. Springer, 1998.

14. R. Rodriguez, Some remarks on Zienkiewicz-Zhu estimator. *Int. J. Numer. Meth. in PDE*, **10**, 625 – 635.

15. J. C. Simo and T. J. R. Hughes, *Computational Inelasticity*. Springer, 2000.

16. R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wily-Teubner, 1996.

17. P. Wriggers *et al.*, Benchmark Perforated Tension Strip. In: *Communication in Talk at ENUMATH Conference in Heidelberg*, 1997.

8. APPENDIX

8.1. Main program in Example 1

```
global lambda mu sigma_y C1 C2 C3 C4 e0
th=1.0; time=[0:0.05:0.4]; nu=0;
%initialization
load coordinates.dat; load elements.dat;load dirichlet.dat;load neumann.dat;
N=size(coordinates,1);initvector=zeros(2*N,1);
sigma0=zeros(size(elements,1),4);u0=zeros(2*N,1);
%material parameters
lambda=1.107438169066076e+05;mu=8.019379844961240e+04;C1=lambda+mu;sigma_y=450;
for step=2:length(time)
 t0=time(step-1);t1=time(step);dt=t1-t0;
 C2=nu/(nu/(2*mu)+th*dt);C3=dt*sigma_y/(nu/(2*mu)+th*dt);
 e0=1/(4*(lambda+mu))*tr2(sigma0)*[1,0,0,1]+1/(2*mu)*dev2(sigma0);
 u_th=fem(coordinates,elements,dirichlet,neumann,initvector,u0,t0,t1,th,N);
 sigma_th=tension(coordinates,elements,u_th,u0,sigma0);
```

```
 u1=1/th*(u_th+(th-1)*u0);sigma1=1/th*(sigma_th+(th-1)*sigma0);
 show(coordinates,elements,sigma1,u1,lambda,mu);
 u0=u1;sigma0=sigma1;
end
```

8.2. Calculation of the stress in Example 1

```
function sigma1=tension(coordinates,elements,u_1,u_0,sigma0);
global lambda mu sigma_y C1 C2 C3 e0
sigma1=sigma0;
for j=1:size(elements,1)
 lcoordinates=coordinates(elements(j,:),:);
 Dphi=inv([1,1,1;lcoordinates'])*[0,0;eye(2)];
 Deta1=zeros(6,4);Deta2=zeros(6,4);u1=zeros(6,1);u0=zeros(6,1);
 Deta1(1:2:5,1:2)=Dphi;Deta1(2:2:6,3:4)=Dphi;
 Deta2(1:2:5,[1,3])=Dphi;Deta2(2:2:6,[2,4])=Dphi;
 u1(1:2:5)=u_1(2*elements(j,:)-1);u1(2:2:6)=u_1(2*elements(j,:));
 u0(1:2:5)=u_0(2*elements(j,:)-1);u0(2:2:6)=u_0(2*elements(j,:));
 eps=(Deta1+Deta2)/2;v=(u1-u0)'*eps+e0(j,:);
 if norm(dev2(v))-sigma_y/(2*mu)>0
  sigma1(j,:)=C1*tr2(v)*[1,0,0,1]+(C2+C3/norm(dev2(v)))*dev2(v);
 else
  sigma1(j,:)=C1*tr2(v)*[1,0,0,1]+2*mu*dev2(v);
 end
end
```

8.3. Main program in Example 4

```
global lambda mu sigma_y C1 C2 C3 e0
th=1.0;time=[0:0.05:1.1];nu=0;
%Initialisation
load coordinates.dat;load elements.dat;load dirichlet.dat;load neumann.dat;
N=size(coordinates,1);initvector=zeros(3*N,1);NJ=size(elements,1);
sigma0=zeros(NJ,9);u0=zeros(3*N,1);
%Material parameters
lambda=1.107438169066076e+05;mu=8.019379844961240e+04;C1=lambda+2*mu/3;sigma_y=450;
for step=2:length(time)
 t0=time(step-1);t1=time(step);dt=t1-t0;
 C2=nu/(nu/(2*mu)+th*dt);C3=th*dt*sigma_y/(nu/(2*mu)+th*dt);
 e0=1/(9*lambda+6*mu)*spur3(sigma0)*[1,0,0,0,1,0,0,0,1]+1/(2*mu)*dev3(sigma0);
 u_th=fem(coordinates,elements,dirichlet,neumann,initvector,u0,t0,t1,th,N,NJ);
 sigma_th=tension(coordinates,elements,u_th,u0,sigma0);
 sigma1=(sigma_th+(th-1)*sigma0)/th;
 u1=(u_th+(th-1)*u0)/th;u0=u1;sigma0=sigma1;
 show(coordinates,elements,dirichlet,neumann,sigma1,u1,lambda,mu);
end
```

8.4. Calculation of the stress in Example 4

```
function sigma1=tension(coordinates,elements,u_1,u_0,sigma0);
global mu sigma_y C1 C2 C3 e0
sigma1=sigma0;
for j=1:size(elements,1)
 lcoordinates=coordinates(elements(j,:),:);
 Dphi=inv([1,1,1,1;lcoordinates'])*[zeros(1,3);eye(3)];
 eps=zeros(12,9);Deta1=zeros(12,9);Deta2=zeros(12,9);
 Deta1(1:3:10,1:3)=Dphi;Deta1(2:3:11,4:6)=Dphi;Deta1(3:3:12,7:9)=Dphi;
 Deta2(1:3:10,[1,4,7])=Dphi;Deta2(2:3:11,[2,5,8])=Dphi;Deta2(3:3:12,[3,6,9])=Dphi;
```

```
 u1=u_1(3*elements(j,reshape(repmat([1:4],3,1),1,12))-repmat([2,1,0],1,4));
 u0=u_0(3*elements(j,reshape(repmat([1:4],3,1),1,12))-repmat([2,1,0],1,4));
 eps=(Deta1+Deta2)/2;v=(u1-u0)'*eps+e0(j,:);
 if norm(dev3(v))-sigma_y/(2*mu)>0
  sigma1(j,:)=C1*tr3(v)*[1,0,0,0,1,0,0,0,1]+(C2+C3/norm(dev3(v)))*dev3(v);
 else
  sigma1(j,:)=C1*tr3(v)*[1,0,0,0,1,0,0,0,1]+2*mu*dev3(v);
 end
end
```