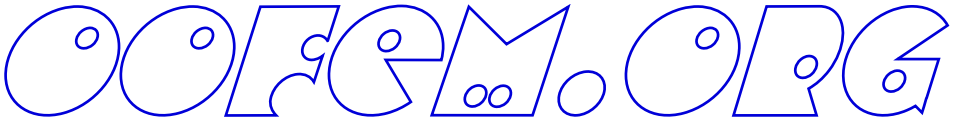

Theory manual

Bořek Patzák, Martin Horák, Filip Kolařík

Czech Technical University
Faculty of Civil Engineering
Department of Mechanics
Thákurova 7, 166 29 Prague, Czech Republic

April 8, 2021

Acknowledgments

This work was supported by the Grant Agency of the Czech Republic - Project No.: 103/97/P106.

Contents

Chapter 1

Introduction and basic equations

1.1 Introduction

The aim of this document is to ducument the theoretical bacground of available solvers, elements, and models in oofem. At present, this manual is a working document, covering only small portion of its capabilities.

1.1.1 Notation

$\boldsymbol{B}^e$	Element strain-displacement matrix
$\boldsymbol{C}$	Elastic constitutive matrix
$\boldsymbol{r}$	Displacement vector
$\boldsymbol{r}^e$	Element displacement vector
$\hat{\boldsymbol{r}}$	Global displacement vector
$\boldsymbol{K}$	Stiffness matrix
N_i^e	Element shape function
$\boldsymbol{n}$	Exterior unit normal vector
Γ	Domain boundary
Γ_u	Dirichlet boundary
Γ_t	Neumann boundary
δ_{ij}	Kronecker symbol
$\boldsymbol{\varepsilon}$	Infinitesimal deformation tensor
ν	POisson's ratio
$\boldsymbol{\sigma}$	Cauchy stress tensor

Table 1.1: Table of symbols

1.2 Linear elasticity

1.2.1 Linear kinematics

Let us consider a deformable body as a collection of points, where position of each point is denoted as $\boldsymbol{x} \in \Omega$. In a deformed configuration the position of each point is identified by its position vector $\boldsymbol{x}^{\varphi}(x) = \boldsymbol{\phi}(\boldsymbol{x})$. The displacement

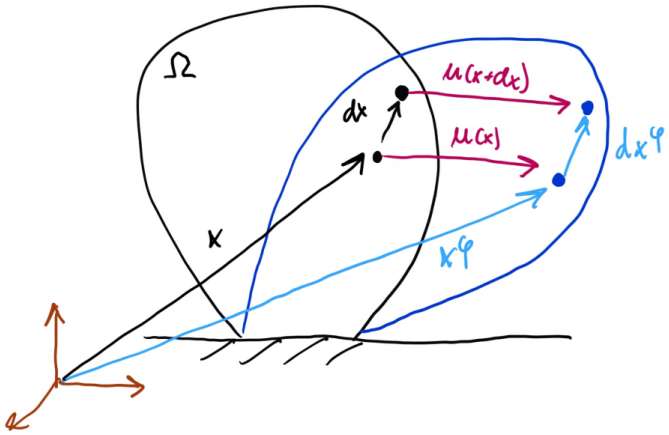

Figure 1.1: Deformed configuration

vector is then defined as:

$$\boldsymbol{x}^{\varphi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}) \tag{1.1}$$

Let us now examine the position in a local neighborhood of a point. The deformed position of such neighbor point with coordinates $\boldsymbol{x} + d\boldsymbol{x}$ (where d$\boldsymbol{x}$is infinitisemally small vector) is

$$\boldsymbol{x}^{\varphi}(\boldsymbol{x} + d\boldsymbol{x}) = \boldsymbol{x} + d\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x} + d\boldsymbol{x}) = \boldsymbol{x}^{\varphi} + d\boldsymbol{x}^{\varphi}$$

, where $d\boldsymbol{x}^{\varphi}$ is the mapping of vector $d\boldsymbol{x}$ onto deformed configuration, see Fig. 1.2.1. Taking into account the definition of displacement vector 1.1 and using Taylor formula we get

$$d\boldsymbol{x}^{\varphi} = \boldsymbol{x} + d\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x} + d\boldsymbol{x}) - \boldsymbol{x}^{\varphi} = d\boldsymbol{x} - \boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x} + d\boldsymbol{x}) \approx [\boldsymbol{I} + \nabla\boldsymbol{u}(\boldsymbol{x})]d\boldsymbol{x} \tag{1.2}$$

where $\nabla\boldsymbol{u}(\boldsymbol{x})$ is the displacement gradient tensor (in small strain theory we assume $||\nabla\boldsymbol{u}(\boldsymbol{x})|| \ll 1$). The displacement gradient tensor can be decomposed into symetric and antisymmetric parts

$$\nabla\boldsymbol{u} = \boldsymbol{\varepsilon} + \boldsymbol{\omega} = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) + \frac{1}{2}(\nabla\boldsymbol{u} - \nabla\boldsymbol{u}^T) = \nabla^s\boldsymbol{u} + \nabla^a\boldsymbol{u}$$

The antisymmetric part corresponds to infinitesimal rotation. The symmeric part of displacement gradient tensor is therefore the measure of infinitesimal deformation

$$d\boldsymbol{x}^{\varphi} = \boldsymbol{\varepsilon}d\boldsymbol{x}$$

1.2.2 Equlibrium equations

Stress is defined as the force across a "small" boundary per unit area of that boundary, for all orientations of the boundary. In the most general case, called triaxial stress, the stress is nonzero across every surface element. Cauchy observed that the stress vector $\boldsymbol{t}$ across a surface is a linear function of the surface's normal vector $\boldsymbol{n}$:

$$\boldsymbol{t}(x) = \boldsymbol{\sigma}(x)\boldsymbol{n}(x)$$

where $\boldsymbol{\sigma}(x)$ is called the (Cauchy) stress tensor, completely describing the stress state at any point.

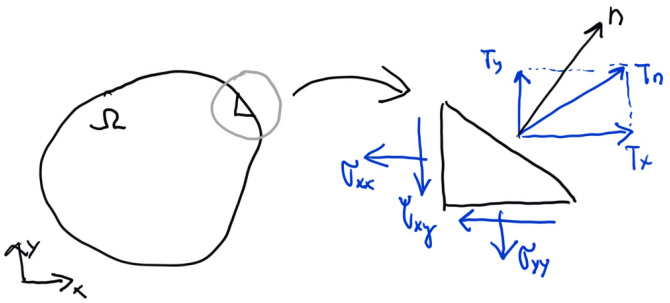

Figure 1.2: Balance between tractions and stresses in 2D

The components of the Cauchy stress tensor at every point in a material satisfy the equilibrium equations (Cauchy's equlibrium equations). From the conservation of angular momentum follows the symmetry of the stress tensor. Therefore, the stress state of the medium at any point and instant can be specified by only six independent parameters, rather than nine. These may be written

$$\begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix}$$

where the elements $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$ are called the normal stresses (relative to the chosen coordinate system), and $\tau_{yz}, \tau_{xz}, \tau_{xy}$ the shear stresses.
In a static equlibrium, the Cauchy stress components in every material point satisfy the equilibrium equations, see ref-fig:stressbalance

$$\sigma_{ji,j} + F_i = 0 \tag{1.3}$$

where we use summation convention over repated indices and F_i are the components of the body force. In a compact tensorial notation we can write the above equation as

$$\nabla \cdot \sigma + F_i = 0 \tag{1.4}$$

1.2.3 Constitutive equations

In this section we present the constituve relations (i.e. relations between stress and strain tensors) for the case of hyperelasticity, which could be defined in terms of strain energy density $W(\boldsymbol{\varepsilon})$, which allows to evaluate stress components as partial derivatives:

$$\sigma_{ij} = \frac{\partial W}{\partial \varepsilon_{ij}}$$

For example, the Hooke's law is defined using following strain energy potential

$$W(\varepsilon) = \frac{1}{2}\varepsilon_{ij}\boldsymbol{C}_{ijkl}\varepsilon_{kl}$$

where $\boldsymbol{C}$ is forth order elasticity tensor. The equality of mixed derivatives ($\frac{\partial^2 W}{\partial \varepsilon_{ij}\varepsilon_{kl}{}^2} = \frac{\partial^2 W}{\partial \varepsilon kl \varepsilon ij^2}$) and symmetry of stress and strain tensors $\frac{\partial \sigma ij}{\partial \varepsilon_{kl}} = C_{ijkl} = C_{jikl} = C_{ijlk}$ imply that there is in general maximum 21 independ components of the elasticity tensor. In the simplest case, the elasticity tensor for isotropic linear elastic material can

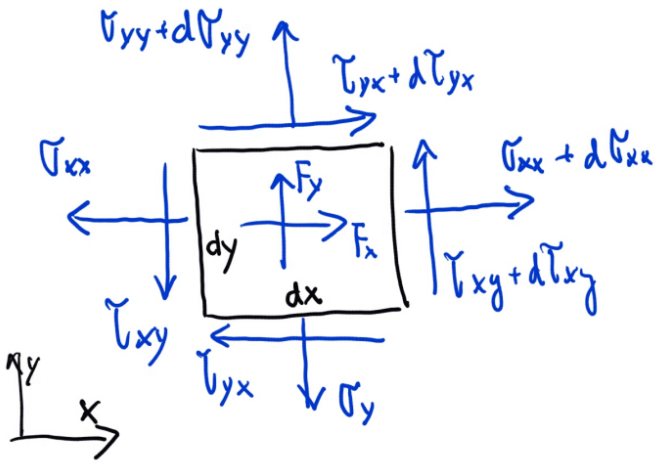

Figure 1.3: Stress balance in 2D

be described by only two parameters: either Lameś parameters (λ, μ) or more usual parametrs being Young's modulus E and Poisson's ratio ν:

$$C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + 2\mu\frac{1}{2}[\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}] \equiv \boldsymbol{C} = \lambda\mathbf{1}\otimes\boldsymbol{I} + 2\mu\boldsymbol{I}$$

1.2.4 Voight notation

The Voigt notation is hrequantly used to take advantage of the symmetry of the stress tensor to express the stress tensor as a six-dimensional vector of the following form:

$$\tilde{\boldsymbol{\sigma}} = [\sigma_x, \sigma_y, \sigma_z, \tau_{yz}, \tau_{xz}, \tau_{xy}]^T \equiv [\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \tau_{yz}, \tau_{xz}, \tau_{xy}]^T$$

The strain tensor, similar in nature to the stress tensor (both symmetric second-order tensors) can be written in Voight notation as

$$\tilde{\boldsymbol{\varepsilon}} = [\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{yz}, \gamma_{xz}, \gamma_{xy}]^T \equiv [\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}, 2\varepsilon_{yz}, 2\varepsilon_{xz}, 2\varepsilon_{xy}]^T$$

The benefit of using different representations for stress and strain is the scalar invariance

$$\boldsymbol{\sigma}\cdot\boldsymbol{\varepsilon} = \sigma_{ij}\varepsilon_{ij} = \tilde{\sigma}\tilde{\varepsilon}$$

Similarly, a three-dimensional symmetric fourth-order tensor can be reduced to a [6,6] matrix.

1.2.5 Boundary value problem in small strain elasticity

Strong form

Starting from the equilibrium equations refeq:staticequlibrium3d, into which we can substitute the constituve equations and strain-displacement relation we obtain the equlibrium equation expressed in terms of displacements:

$$\frac{\partial}{\partial x_j}\left(C_{ijkl}\frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right)\right) + F_i = 0$$

This system of three partial differnetial equations can be solved, provided that appropriate boundary conditions are given. In summary, the strong form is the following:

Find $\boldsymbol{u} \in R^n$, such that

$$\frac{\partial}{\partial x_j}\left(C_{ijkl}\frac{1}{2}(\frac{\partial u_k}{\partial x_l}+\frac{\partial u_l}{\partial x_k})\right)+F_i=0\in\Omega$$

$$\boldsymbol{u}=\bar{\boldsymbol{u}}\in\Gamma_u$$

$$\boldsymbol{t}_i^n=C_{ijkl}\frac{1}{2}(\frac{\partial u_k}{\partial x_l}+\frac{\partial u_l}{\partial x_k})\,n_j=\bar{t}_i\ \in\Gamma_t$$

Weak form

By following the method of weighted residuals, we multiply the governing differential equations 1.4 in residual form by a sutable test functions $\delta\boldsymbol{u}$, satisfying the homogeneous boundary conditions on Γ_u

$$\int_\Omega \delta\boldsymbol{u}\cdot(\nabla\cdot\boldsymbol{\sigma}+F)\ d\Omega=\boldsymbol{0}$$

By applying the Green's formula, we arrive at

$$\int_\Omega \nabla\delta\boldsymbol{u}\cdot\boldsymbol{\sigma}\ d\Omega=\int_\Omega \delta\boldsymbol{u}\cdot\boldsymbol{F}\ d\Omega+\int_\Gamma \delta\boldsymbol{u}\cdot\boldsymbol{\sigma n}\ d\Gamma$$

Then we can substitute for the stresses and tractions and taking into account the symmetry of stress tensor ($\sigma=C\boldsymbol{\varepsilon}$)

$$\int_\Omega \nabla^s\delta\boldsymbol{u}\cdot\boldsymbol{C\varepsilon}\ d\Omega=\int_\Omega \delta\boldsymbol{u}\cdot\boldsymbol{F}\ d\Omega+\int_\Gamma \delta\boldsymbol{u}\cdot\boldsymbol{t}\ d\Gamma \tag{1.5}$$

Or, equaivalently using Voight's notation

$$\int_\Omega \delta\tilde{\varepsilon}^T\tilde{\boldsymbol{D}}\tilde{\boldsymbol{\varepsilon}}\ d\Omega=\int_\Omega \delta\boldsymbol{u}^T\boldsymbol{F}\ d\Omega+\int_\Gamma \delta\boldsymbol{u}^T\boldsymbol{t}\ d\Gamma \tag{1.6}$$

Note: this is equaivalent to the principle of virtual displacements. For hyperelastic material, the weak form is identical to the principle of minimum potential energy.

1.2.6 Finite element discretization

Let us consider discretization of the problem domain Ω into set of nonoverlapping subdomains Ω_e, called elements. Next we will consider the approximation of the unknown displacement field, defined on individual subdomains. Note that the approximation is not arbitrary:

- The weak form kontains only first derivatives of the unknown and test functions, thus only C^0 continuity is required.

The element approxiamtion of the arbitrary function f has the form

$$f=\sum N_j(\boldsymbol{x})r_j=\boldsymbol{Nr}$$

where N_j are so called shape or approximation functions and r_j are nodal values. Note that for the approximation functions to be interpolatory, the shape functions have to satisfy Kronecker-delta property, i.e., $N_j(\boldsymbol{x}_i)=\delta_{ij}$, where $\boldsymbol{x}_i$ is the position vector of the i-th node. Also, the shape functions have to satisfy the condition $\sum N_i=1$, which follows from the requirement to approximate the constant function. The required continuity of element approximations have to

be satisfied. This is typically achieved by enforcing the continuity at the nodal points. In our case, the approximation of displacements and test functions is

$$\boldsymbol{u}^e = \boldsymbol{N}^e(\boldsymbol{x})\boldsymbol{r}^e \tag{1.7}$$
$$\delta\boldsymbol{u}^t = \boldsymbol{N}^e(\boldsymbol{x})\delta\boldsymbol{r}^e \tag{1.8}$$

We will use the weak form 1.5, which using Voight's notation has the form

$$\int_\Omega \nabla^s \delta\boldsymbol{u} \cdot \boldsymbol{C}\boldsymbol{\varepsilon}\ d\Omega = \int_\Omega \delta\boldsymbol{u}\cdot\boldsymbol{F}\ d\Omega + \int_\Gamma \delta\boldsymbol{u}\cdot\boldsymbol{t}\ d\Gamma$$

We will need also the derivatives of the displacement and test functions

$$\tilde{\varepsilon}^e == \boldsymbol{B}^e(\boldsymbol{x})\boldsymbol{r}^e \tag{1.9}$$
$$\delta\tilde{\varepsilon}^e = \boldsymbol{B}^e(\boldsymbol{x})\delta\boldsymbol{r}^e \tag{1.10}$$

where B^e matrix contains the first partial derivatives of the shape functions. By substiituting into the weak form 1.6 we obtain

$$\sum_e \delta\boldsymbol{r}^{e,T}\left[\underbrace{\int_{\Omega^e}\boldsymbol{B}^{e,T}\tilde{\boldsymbol{D}}^e\boldsymbol{B}^e\boldsymbol{r}^e\ d\Omega}_{\boldsymbol{K}^e}\boldsymbol{r}^e - \underbrace{\int_\Omega \boldsymbol{N}^{e,T}\boldsymbol{F}\ d\Omega}_{\boldsymbol{f}^e_\Omega} - \underbrace{\int_{\Gamma_t}\boldsymbol{N}^{e,T}\bar{\boldsymbol{t}}\ d\Gamma}_{\boldsymbol{f}^e_\Gamma}\right] = \boldsymbol{0} \tag{1.11}$$

After introducing a mappig between element displacement vectors $\boldsymbol{r}^e$, nodal vectors of test function values $\delta\boldsymbol{r}$ and their global counterparts $\hat{\boldsymbol{r}}, \delta\hat{\boldsymbol{r}}$ one can obtain

$$\delta\hat{\boldsymbol{r}}^T\left[\hat{\boldsymbol{K}}\hat{\boldsymbol{r}} - \hat{\boldsymbol{f}}_\Omega - \hat{\boldsymbol{f}}_\Gamma\right] = \boldsymbol{0} \tag{1.12}$$

By taking into account that the test fuctions are arbitrary (i.e. $\delta\hat{\boldsymbol{r}} \neq \boldsymbol{0}$), one finnaly obtains the following set of linear algebraic equations for unknonwn nodal displacements $\hat{\boldsymbol{r}}$:

$$\hat{\boldsymbol{K}}\hat{\boldsymbol{r}} = \hat{\boldsymbol{f}}_\Omega + \hat{\boldsymbol{f}}_\Gamma \tag{1.13}$$

Chapter 2

Solution procedures

2.1 Solution procedures for nonlinear systems

We illustrate this on problem of nonlinear mechanics. Our starting point is general form of equilibrium equations expressing the balance between internal $\boldsymbol{f}^{int}$ and external $\boldsymbol{f}^{ext}$

$$\boldsymbol{f}^{int}(\boldsymbol{r}) = \boldsymbol{f}^{ext}$$

Suppose we are looking for an equilibrium at the end of load increment $\Delta\boldsymbol{f}^{ext}$

$$\boldsymbol{f}^{int}(\boldsymbol{r} + \Delta\boldsymbol{r}) = \boldsymbol{f}^{ext} + \Delta\boldsymbol{f}^{ext} \tag{2.1}$$

By the linearization of the nodal force vector around known equilibrium state we can obtain

$$\boldsymbol{f}^{int}(\boldsymbol{r}) + \frac{\partial\,\boldsymbol{f}^{int}}{\partial\,\boldsymbol{r}}\Delta\boldsymbol{r} + O(\|\Delta\boldsymbol{r}\|^2) \tag{2.2}$$

The derivative of internal force vector with respect to nodal displacements is called jacobian matrix and in solid mechanics as tangent stiffness matrix. For the case of material non-linearity

$$\boldsymbol{f}^{e,int}(\boldsymbol{r^e}) = \int_{\Omega^e} \boldsymbol{B}^T\boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\boldsymbol{r^e}))\;d\Omega \Rightarrow \frac{\partial\,\boldsymbol{f}^{e,int}}{\partial\,\boldsymbol{d}^e} = \int_{\Omega^e} \boldsymbol{B}^T\frac{\partial\,\boldsymbol{\sigma}}{\partial\,\boldsymbol{\varepsilon}}\frac{\partial\,\boldsymbol{\varepsilon}}{\partial\,\boldsymbol{r^e}}\;d\Omega = \int_{\Omega^e} \boldsymbol{B}^T\frac{\partial\,\boldsymbol{\sigma}}{\partial\,\boldsymbol{\varepsilon}}\boldsymbol{B}\boldsymbol{r}^e\;d\Omega = \int_{\Omega^e} \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}\boldsymbol{r}^e\;d\Omega$$

2.1.1 Newton-Raphson method

The method is based on splitting of the loading process into series of subsequent incremental loading steps in which the incremental loading vector $\Delta\boldsymbol{f}$ is applied. We are looking for the equilibrium at the end of loading step 2.1 using the iterative procedure outlined in 2.1. The algorithm is graphically outlined in Fig. 2.1.1 for a system with one unknown and summarized in Table 2.2.

Based on update strategy for stiffness matrix, one can obtain different variants of the method. When the stiffness matrix $\boldsymbol{K}^i$ is updated in each iteration, the full Newton-Raphson method is obtained. When stiffness matrix only every n-th iteration, one speaks about modified Newton-Raphson method. Finally, when the stiffness matrix is updated only at the beginning of the loading step, one obtains so called initial stiffness method. For the full Newton-Raphson method a quadratic convergence is obtained.

One can implement two blends of Newton-Raphson algorithm, where the loading can be driven by load control or by displacement control, where the prescribed increments of displacements are applied to selected DOFs.

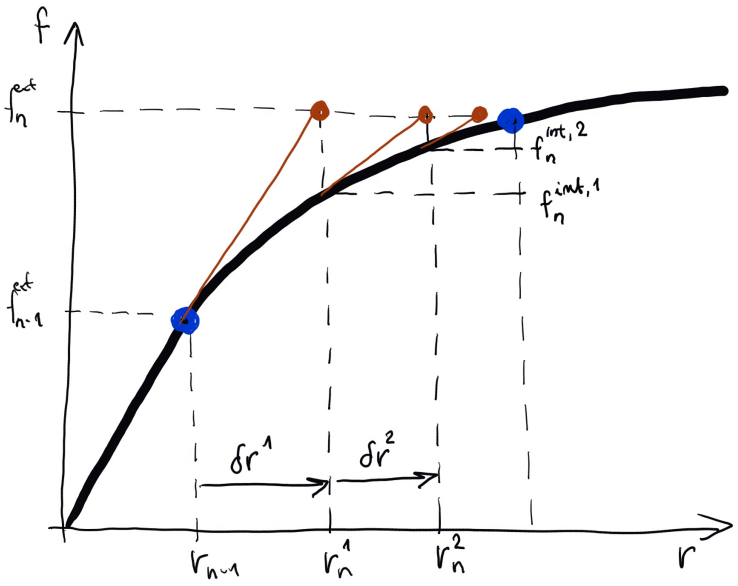

Figure 2.1: Illustration of Newton-Raphson method

Given
$\boldsymbol{f}_{n-1}^{ext}$
$\boldsymbol{f}_{n}^{ext} = \boldsymbol{f}_{n-1}^{ext} + \delta \boldsymbol{f}_{n}^{ext}$
$\boldsymbol{r}_{n}^{0} = \boldsymbol{r}_{n-1}$
Looking for $\boldsymbol{r}_n$, such that $\boldsymbol{f}^{int}(\boldsymbol{r}_n) = \boldsymbol{f}_{n}^{ext}$
Solve for $i = 1, 2, \cdots$
$\boldsymbol{K}^i \delta \boldsymbol{r}^i = \boldsymbol{f}_{n}^{ext} - \boldsymbol{f}_{n}^{int}(\boldsymbol{r}_{n}^{i-1})$
$\boldsymbol{r}_{n}^{i} = \boldsymbol{r}_{n}^{i-1} + \delta \boldsymbol{r}^i$
Until $\|\boldsymbol{f}_{n}^{ext} - \boldsymbol{f}_{n}^{int}(\boldsymbol{r}_{n}^{i-1})\| \leq \varepsilon$

Table 2.1: Newton-Raphson method

2.1.2 Arc-length method

We start by assuming the parametrized loading, in which the total external load vector is expressed as

$$\boldsymbol{f}^{ext}(\lambda) = +\lambda \boldsymbol{f}_{p}^{ext}$$

where $\boldsymbol{f}_p$ is proportional, reference load vector, and λ is load scaling parameter, The arc-length method is based on idea of controlling the length passed along the loading path. For the differential length of loading path we can write

$$\Delta l = \sqrt{\Delta \boldsymbol{r}^T \Delta \boldsymbol{r} + (c^2 \Delta \lambda^2 \boldsymbol{f}_p^T \boldsymbol{f}_p)} \tag{2.3}$$

where c is coefficient of generalized metrics used to define Δl (taking into account different units of displacement and load). For selected increment of loading path length Δl, we are looking for the equilibrium, where the unknowns are

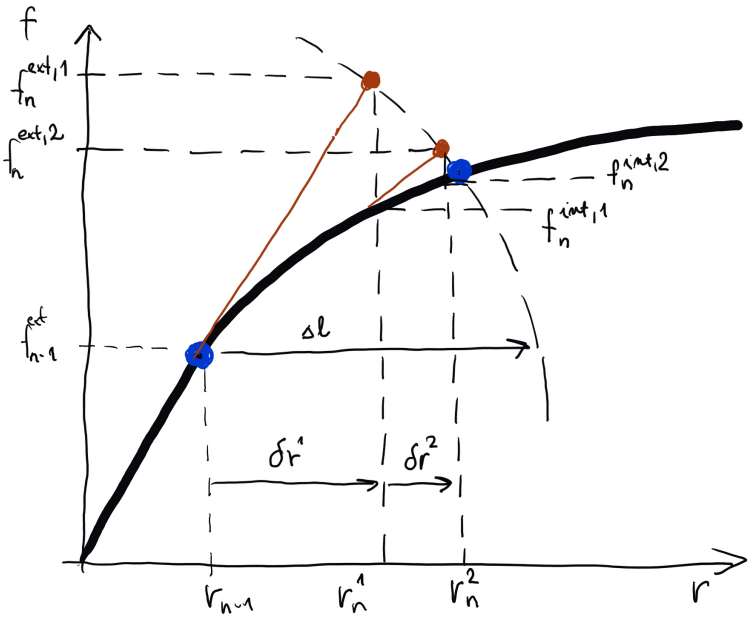

Figure 2.2: Illustration of Acr-length method

nodal displacements $\boldsymbol{r}$ and the load scaling parameter λ. We have the equilibrium equation and additional scalar equation 2.3:

$$\boldsymbol{f}^{int}(\boldsymbol{r}_n) = \boldsymbol{f}^{ext}(\lambda_n \boldsymbol{f}_p) \tag{2.4}$$

$$\Delta l_n^2 = \Delta \boldsymbol{r}_n^T \Delta \boldsymbol{r}_n + c^2 \Delta\lambda^2 \boldsymbol{f}_p^T \boldsymbol{f}_p \tag{2.5}$$

At the end of n-th loading step and i-th iteration the displacement vector can be written as $\boldsymbol{r}_n^i = \boldsymbol{r}_n^{i-1} + \delta\boldsymbol{r}$ and similarly the load scaling parameter as $\lambda_n^i = \lambda_n^{i-1} + \delta\lambda$. Substituting this into equilibrium equation 2.4 we get

$$\boldsymbol{f}^{int}(\boldsymbol{r}_n^{i-1} + \delta\boldsymbol{r}) = \boldsymbol{f}^{ext}((\lambda_n^{i-1} + \delta\lambda)\boldsymbol{f}_p)$$

By linearization of $\boldsymbol{F}^{int}$ around known state $\boldsymbol{r}_n^{i-1}$ we get

$$\boldsymbol{f}_n^{int}(\boldsymbol{r}_n^{i-1}) + \boldsymbol{K}_n^{i-1}\delta\boldsymbol{r} = \boldsymbol{f}^{ext,i-1} + \delta\lambda \boldsymbol{f}_p$$

and finally for unknown $\delta\boldsymbol{r}$

$$\delta\boldsymbol{r} = \underbrace{(\boldsymbol{K}_n^{i-1})^{-1}(\boldsymbol{f}^{ext,i-1} - \boldsymbol{f}^{int}(\boldsymbol{r}_n^{i-1}))}_{\delta\boldsymbol{r}_r} + \delta\lambda \underbrace{(\boldsymbol{K}_n^{i-1})^{-1}\boldsymbol{f}_p}_{\delta\boldsymbol{r}_\lambda} \tag{2.6}$$

Note that the vectors $\delta\boldsymbol{r}_r$ and $\delta\boldsymbol{r}_\lambda$ can be computed and the only unknown remaining is the incremental change of loading parameter $\delta\lambda$, which could be determined from 2.5

$$\Delta l_n^2 = (\Delta\boldsymbol{r}_n^{i-1} + \delta\boldsymbol{r}_r + \delta\lambda\delta\boldsymbol{r}_\lambda)^T(\Delta\boldsymbol{r}_n^{i-1} + \delta\boldsymbol{r}_r + \delta\lambda\delta\boldsymbol{r}_\lambda) + c^2(\Delta\lambda_n^{i-1} + \delta\lambda)^2 \boldsymbol{f}_p^T\boldsymbol{f}_p \tag{2.7}$$

This finally yields a quadratic equation for unknown increment of loading parameter $\delta\lambda$. The algorithm is summarized in Table **??**.

Given
$\boldsymbol{f}_{n-1}^{ext}, \boldsymbol{f}_p$
$\boldsymbol{r}_n^0 = \boldsymbol{r}_{n-1}$
Evaluate
$\delta\boldsymbol{r}_\lambda = (\boldsymbol{K}_n)^{-1}\boldsymbol{f}_p$
$\Delta\lambda^0 = \pm\Delta l/\sqrt{\delta\boldsymbol{r}_\lambda^T\delta\boldsymbol{r}_\lambda + c^2\boldsymbol{f}_p^T\boldsymbol{f}_p}$
$\Delta\boldsymbol{r}_n^0 = (\boldsymbol{K}_n)^{-1}(\Delta\lambda\boldsymbol{f}_p) = (\boldsymbol{K}_n)^{-1}((\lambda_n + \Delta\lambda^0)\boldsymbol{f}_p - \boldsymbol{f}_n^{int,0})$
Repeat for $i = 1, 2, \cdots$
$\delta\boldsymbol{r}_\lambda = (\boldsymbol{K}_n^{i-1})^{-1}\boldsymbol{f}_p$
$\delta\boldsymbol{r}_r = (\boldsymbol{K}_n^{i-1})^{-1}(\lambda_n^{i-1}\boldsymbol{f}_p - \boldsymbol{f}^{int}(\boldsymbol{r}_n^{i-1}))$
Solve quadratic equation 2.7 for $\delta\lambda$
$\delta\boldsymbol{r}^i = \delta\boldsymbol{r}_r + \delta\lambda\delta\boldsymbol{r}_\lambda$
$\Delta\boldsymbol{r}_n^i = \Delta\boldsymbol{r}^{i-1} + \delta\boldsymbol{r}^i,\ \boldsymbol{r}_n^i = \boldsymbol{r}_n^{i-1} + \delta\boldsymbol{r}^i$
$\lambda^i = \lambda^{i-1} + \delta\lambda,\ \Delta\Lambda_n^i = \Delta\Lambda_n^{i-1} + \delta\lambda$
Until convergence reached

Table 2.2: Newton-Raphson method

2.2 Non-stationary linear transport model

The weak form of diffusion-type differential equation leads to

$$\boldsymbol{K}\boldsymbol{r} + \boldsymbol{C}\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{F}, \tag{2.8}$$

where the matrix $\boldsymbol{K}$ is a general non-symmetric conductivity matrix, $\boldsymbol{C}$ is a general capacity matrix and the vector $\boldsymbol{F}$ contains contributions from external and internal sources. The vector of unknowns, $\boldsymbol{r}$, can hold nodal values of temperature, humidity, or concentration fields, for example.
Time discretization is based on a generalized trapezoidal rule. Let us assume that the solution is known at time t and the time increment is Δt. The parameter $\alpha \in \langle 0, 1\rangle$ defines a type of integration scheme; $\alpha = 0$ results in an explicit (forward) method, $\alpha = 0.5$ refers to the Crank-Nicolson method, and $\alpha = 1$ means an implicit (backward) method. The appromation of solution vector and its time derivative yield

$$\tau = t + \alpha\Delta t = (t + \Delta t) - (1 - \alpha)\Delta t, \tag{2.9}$$
$$\boldsymbol{r}_\tau = (1 - \alpha)\boldsymbol{r}_t + \alpha\boldsymbol{r}_{t+\Delta t}, \tag{2.10}$$
$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \frac{1}{\Delta t}\left(\boldsymbol{r}_{t+\Delta t} - \boldsymbol{r}_t\right). \tag{2.11}$$
$$\boldsymbol{F}_\tau = (1 - \alpha)\boldsymbol{F}_t + \alpha\boldsymbol{F}_{t+\Delta t}, \tag{2.12}$$

Let us assume that Eq. (2.8) should be satisfied at time τ. Inserting Eqs. (2.10)-(2.12) into Eq. (2.8) leads to

$$\left[\alpha\boldsymbol{K} + \frac{1}{\Delta t}\boldsymbol{C}\right]\boldsymbol{r}_{t+\Delta t} = \left[(\alpha - 1)\boldsymbol{K} + \frac{1}{\Delta t}\boldsymbol{C}\right]\boldsymbol{r}_t + (1 - \alpha)\boldsymbol{F}_t + \alpha\boldsymbol{F}_{t+\Delta t} \tag{2.13}$$

where the conductivity matrix $\boldsymbol{K}$ contains also a contribution from convection, since it depends on $\boldsymbol{r}_{t+\Delta t}$

$$\boldsymbol{K} = \int_\Omega \boldsymbol{B}^T\lambda\boldsymbol{B}\mathrm{d}\Omega + \underbrace{\int_{\Gamma_{\overline{c}}} \boldsymbol{N}^T h\boldsymbol{N}\mathrm{d}\Gamma}_{\text{Convection}} \tag{2.14}$$

The vectors $\boldsymbol{F}_t$ or $\boldsymbol{F}_{t+\Delta t}$ contain all known contributions

$$\boldsymbol{F}_t = -\underbrace{\int_{\Gamma_{\overline{q}}} \boldsymbol{N}^T \overline{q}_t \mathrm{d}\Gamma}_{\text{Given flow}} + \underbrace{\int_{\Gamma_{\overline{c}}} \boldsymbol{N}^T h T_{\infty,t} \mathrm{d}\Gamma}_{\text{Convection}} + \underbrace{\int_{\Omega} \boldsymbol{N}^T \overline{Q}_t \mathrm{d}\Omega}_{\text{Source}} \tag{2.15}$$

2.3 Non-stationary nonlinear transport model

In a nonlinear model, Eq. (2.8) is modified to

$$\boldsymbol{K}(\boldsymbol{r})\boldsymbol{r} + \boldsymbol{C}(\boldsymbol{r})\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{F}(\boldsymbol{r}), \tag{2.16}$$

Time discretization is the same as in Eqs. (2.9)-(2.11) but the assumption in Eq. (2.15) is not true anymore. Let us assume that Eq. (2.16) should be satisfied at time $\tau \in \langle t, t+\Delta t\rangle$. By substituting of Eqs. (2.10)-(2.11) into Eq. (2.16) leads to the following equation

$$\left[(1-\alpha)\boldsymbol{r}_t + \alpha \boldsymbol{r}_{t+\Delta t}\right] \boldsymbol{K}_\tau(\boldsymbol{r}_\tau) + \left[\frac{\boldsymbol{r}_{t+\Delta t} - \boldsymbol{r}_t}{\Delta t}\right] \boldsymbol{C}_\tau(\boldsymbol{r}_\tau) = \boldsymbol{F}_\tau(\boldsymbol{r}_\tau). \tag{2.17}$$

Eq. (2.17) is non-linear and the Newton method is used to obtain the solution. First, the Eq. (2.17) is transformed into a residual form with the residuum vector $\boldsymbol{R}_\tau$, which should converge to the zero vector

$$\boldsymbol{R}_\tau = \left[(1-\alpha)\boldsymbol{r}_t + \alpha \boldsymbol{r}_{t+\Delta t}\right] \boldsymbol{K}_\tau(\boldsymbol{r}_\tau) + \left[\frac{\boldsymbol{r}_{t+\Delta t} - \boldsymbol{r}_t}{\Delta t}\right] \boldsymbol{C}_\tau(\boldsymbol{r}_\tau) - \boldsymbol{F}_\tau(\boldsymbol{r}_\tau) \to \boldsymbol{0}. \tag{2.18}$$

A new residual vector at the next iteration, $\boldsymbol{R}_\tau^{i+1}$, can determined from the previous residual vector, $\boldsymbol{R}_\tau^i$, and its derivative simply by linearization. Since the aim is getting an increment of solution vector, $\Delta\boldsymbol{r}_\tau^i$, the new residual vector $\boldsymbol{R}_\tau^{i+1}$ is set to zero

$$\boldsymbol{R}_\tau^{i+1} \approx \boldsymbol{R}_\tau^i + \frac{\partial \boldsymbol{R}_\tau^i}{\partial \boldsymbol{r}_t}\Delta\boldsymbol{r}_\tau^i = \boldsymbol{0}, \tag{2.19}$$

$$\Delta\boldsymbol{r}_\tau^i = -\left[\frac{\partial \boldsymbol{R}_\tau^i}{\partial \boldsymbol{r}_t}\right]^{-1} \boldsymbol{R}_\tau^i. \tag{2.20}$$

Deriving Eq. (2.18) and inserting to Eq. (2.20) leads to

$$\tilde{\boldsymbol{K}}_\tau^i = \frac{\partial \boldsymbol{R}_\tau^i}{\partial \boldsymbol{r}_t} = -\alpha \boldsymbol{K}_\tau^i(\boldsymbol{r}) - \frac{1}{\Delta t}\boldsymbol{C}_\tau^i(\boldsymbol{r}), \tag{2.21}$$

$$\Delta\boldsymbol{r}_\tau^i = -\left[\tilde{\boldsymbol{K}}_\tau^i\right]^{-1} \boldsymbol{R}_\tau^i, \tag{2.22}$$

which gives the resulting increment of the solution vector $\Delta\boldsymbol{r}_\tau^i$

$$\begin{aligned}\Delta\boldsymbol{r}_\tau^i = -\left[\tilde{\boldsymbol{K}}_\tau^i\right]^{-1} \Big\{ &\left[(1-\alpha)\boldsymbol{r}_t + \alpha \boldsymbol{r}_{t+\Delta t}\right] \boldsymbol{K}_\tau^i(\boldsymbol{r}_\tau) + \\ &\left[\frac{\boldsymbol{r}_{t+\Delta t} - \boldsymbol{r}_t}{\Delta t}\right] \boldsymbol{C}_\tau^i(\boldsymbol{r}_\tau) - \boldsymbol{F}_\tau(\boldsymbol{r}_\tau)\Big\},\end{aligned} \tag{2.23}$$

and the new total solution vector at time $t+\Delta t$ is obtained in each iteration

$$\boldsymbol{r}_{t+\Delta t}^{i+1} = \boldsymbol{r}_{t+\Delta t}^i + \Delta\boldsymbol{r}_\tau^i. \tag{2.24}$$

There are two options how to initialize the solution vector at time $t+\Delta t$. While the first case applies linearization with a known derivative, the second case simply starts from the previous solution vector. The second method in Eq. (2.26) is implemented in OOFEM.

$$\boldsymbol{r}^0_{t+\Delta t} = \boldsymbol{r}_t + \Delta t \frac{\partial \boldsymbol{r}_t}{\partial t}, \tag{2.25}$$

$$\boldsymbol{r}^0_{t+\Delta t} = \boldsymbol{r}_t. \tag{2.26}$$

Note that the matrices $\boldsymbol{K}(\boldsymbol{r}_\tau), \boldsymbol{C}(\boldsymbol{r}_\tau)$ and the vector $\boldsymbol{F}(\boldsymbol{r}_\tau)$ depend on the solution vector $\boldsymbol{r}_\tau$. For this reason, the matrices are updated in each iteration step (Newton method) or only after several steps (modified Newton method). The residuum $\boldsymbol{R}^i_\tau$ and the vector $\boldsymbol{F}_\tau(\boldsymbol{r}_\tau)$ are updated in each iteration, using the most recent capacity and conductivity matrices.

2.3.1 Heat flux from radiation

Heat flow from a body surrounded by a medium at a temperature T_∞ is governed by the Stefan-Boltzmann Law

$$q(T, T_\infty) = \varepsilon\sigma(T^4 - T^4_\infty) \tag{2.27}$$

where $\varepsilon \in \langle 0, 1\rangle$ represents emissivity between the surface and the boundary at temperature T_∞. $\sigma = 5.67 \cdot 10^{-8}$ $\mathrm{W/m^{-2}K^{-4}}$ stands for a Stefan-Boltzmann constant. Transport elements in OOFEM implement Eq. (2.27) and require non-linear solver.
Alternatively (not implemented), a linearization using Taylor expansion around T_∞ and neglecting higher-order terms results to

$$q(T, T_\infty) \approx q(T = T_\infty) + \frac{\partial q(T, T_\infty)}{\partial T_\infty}(T_\infty - T) = 4\varepsilon\sigma T^3_\infty (T - T_\infty) \tag{2.28}$$

leading to so-called radiation heat transfer coefficient $\alpha_{rad} = 4\varepsilon\sigma T^3_\infty$. The latter resembles similar coefficient as in convective heat transfer. Other methods for Eq. (2.27) could be based on Oseen or Newton-Kantorovich linearization. Also, radiative heat transfer coefficient α_{rad} can be expressed as [**?**, pp.28]

$$q(T, T_\infty) = \varepsilon\sigma\frac{T^4 - T^4_\infty}{T - T_\infty}(T - T_\infty) = \underbrace{\varepsilon\sigma(T^2 + T^2_\infty)(T + T_\infty)}_{\alpha_{rad}}(T - T_\infty) \tag{2.29}$$

2.4 Transient incompressible flow - Lagrangian Particle Method

2.4.1 Lagrangian governing equations of incompressible fluid

The Particle finite element method (PFEM) is based on the Lagrangian form of the Navier-Stokes equation for incompressible Newtonian fluids. Assuming the density does not change in time for an incompressible fluid, the continuity equation reduces to zero requirements for the divergence of the velocity. The Navier-Stokes equations take the form

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} = \rho\boldsymbol{b} + \nabla\cdot\boldsymbol{\sigma}\ , \tag{2.30}$$

$$\nabla\cdot\boldsymbol{u} = 0\ . \tag{2.31}$$

For the deviatoric stress in Newtonian fluids a linear dependency of stress tensor and strain rate tensor is adopted and for the Newtonian fluids. Considering the incompressibility of the fluid, the Cauchy stress reads

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\nabla^s\boldsymbol{u}\ . \tag{2.32}$$

This equation is known as *Stokes' law* and its Cartesian form writes

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) . \tag{2.33}$$

Substituting the expression of Cauchy stress from Stokes' law (2.32) into the momentum equation (2.30) and rewriting gives

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} = \rho\boldsymbol{b} + \mu\nabla^2\boldsymbol{u} - \nabla p . \tag{2.34}$$

The governing equations of the mass (2.31) and momentum conservation (2.34) form can be written in the Cartesian form for the individual component i using Einstein summation convention

$$\rho\frac{\partial u_i}{\partial t} = -\frac{\partial}{\partial x_i}p + \mu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + \rho b_i , \tag{2.35}$$

$$\frac{\partial u_i}{\partial x_i} = 0 . \tag{2.36}$$

The equations are accompanied by a set of standard boundary conditions imposed on the complementary parts of the domain boundary

$$\tau_{ij}\nu_j - p\nu_i = \bar{\sigma}_{ni} \quad \text{on } \Gamma_\sigma , \tag{2.37}$$

$$u_i\nu_i = \bar{u}_n \quad \text{on } \Gamma_n , \tag{2.38}$$

$$u_i\zeta_i = \bar{u}_t \quad \text{on } \Gamma_t , \tag{2.39}$$

where ν or n denotes the normal direction to the boundary and ζ or t the tangential one. The bar sign over a quantity $\bar{x}$ stands for its prescribed value.

2.4.2 Time discretization

For the time discretization of the momentum equation, a general trapezoid rule can be adopted. Using this rule, the time derivative of a generic function ϕ can be approximated by following equation

$$[\phi(x,t)]^{n+\theta} = \theta\phi(x,t^{n+1}) + (1-\theta)\phi(x,t^n) = \theta\phi^{n+1} + (1-\theta)\phi^n . \tag{2.40}$$

Rewriting the time derivative on the left hand side of the momentum balance (2.35) as a finite difference in time and applying the trapezoidal rule on the right hand side, we obtain

$$\rho\frac{\partial u_i}{\partial t} \approx \rho\frac{u_i^{n+1} - u_i^n}{\Delta t} = \left[-\frac{\partial}{\partial x_i}p + \mu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + \rho b_i\right]^{n+\theta} . \tag{2.41}$$

The parameter θ can take values from the interval $[0,1]$. The approximation is considered as a weighted average of the derivative values in the time step n and $n+1$. Using a specific value of the θ parameter, well-known methods can be recovered: The explicit Euler method $\theta = 0$, the backward Euler for $\theta = 1$ or the Crank-Nicolson method $\theta = 1/2$. The current implemantation of PFEM allows the use of explicit and backward (implicit) method.

2.4.3 Fractional step scheme

Beside the three velocity components, the discretized momentum balance equations (2.41) for a three dimensional case includes pressure as a coupling variable. A possible approach to decouple them is the application of so-called *fractional step method*. The main idea of this method consists in introducing an intermediate velocity as supplementary variable

and splitting the momentum equation. The modification introduced by R.Codina [**?**] splits the the discretized time step is split into two sub-steps. The implicit part of the pressure is avoided and assigned to the second step.

$$\frac{\partial u_i}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_i^{n+1} - u_i^* + u_i^* - u_i^n}{\Delta t} = \left[-\frac{1}{\rho}\frac{\partial}{\partial x_i} p + \frac{\mu}{\rho}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + b_i \right]^{n+\theta} . \tag{2.42}$$

where the intermediate velocity u_i^* is introduced. Splitting the equation in the following manner gives the expression for the unknown velocities

$$u_i^* = u_i^n + b_i \Delta t - \frac{\Delta t}{\rho}\frac{\partial}{\partial x_i}\gamma p^n + \frac{\Delta t \mu}{\rho}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i^{n+\theta}}{\partial x_j}\right) , \tag{2.43}$$

$$u_i^{n+1} = u_i^* - \frac{\Delta t}{\rho}\frac{\partial}{\partial x_i}(p^{n+1} - \gamma p^n) . \tag{2.44}$$

The pressure split is here introduced by the new parameter γ defining the amount of splitting and can take values from 0 to 1. The body loads are considered to be constant over time step.
In a similar way, the fractional step method is applied on the mass conservation equation. Here, the time derivative of density would be approximated. As we examine an incompressible flow, whose density does not change in time, merely the intermediate velocity term is incorporated in the divergence of the velocity.

$$\frac{\partial (u_i^{n+1} - u_i^* + u_i^*)}{\partial x_i} = 0 , \tag{2.45}$$

which can be decomposed into two sub-equations

$$\frac{\partial u_i^*}{\partial x_i} = 0 \tag{2.46}$$

$$\frac{\partial (u_i^{n+1} - u_i^*)}{\partial x_i} = 0 . \tag{2.47}$$

By substituting for the velocity difference into the equation (2.44) we obtain

$$\frac{\partial}{\partial x_i}(u_i^{n+1} - u_i^*) = \frac{\partial}{\partial x_i}\left(-\frac{\Delta t}{\rho}\frac{\partial}{\partial x_i}(p^{n+1} - \gamma p^n) \right) . \tag{2.48}$$

Now we can sum the separated mass equations together. This operation gives the coupled mass-momentum equation

$$\frac{\partial u_i^*}{\partial x_i} - \frac{\Delta t}{\rho}\frac{\partial^2}{\partial x_i^2}(p^{n+1} - \gamma p^n) = 0 . \tag{2.49}$$

The final set of equations reads

$$u_i^* = u_i^n + b_i \Delta t - \frac{\Delta t}{\rho}\frac{\partial}{\partial x_i}\gamma p^n + \frac{\Delta t \mu}{\rho}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i^{n+\theta}}{\partial x_j}\right) , \tag{2.50}$$

$$\frac{\partial^2}{\partial x_i^2}(p^{n+1}) = \frac{\rho}{\Delta t}\frac{\partial u_i^*}{\partial x_i} + \frac{\partial^2}{\partial x_i^2}(\gamma p^n) , \tag{2.51}$$

$$u_i^{n+1} = u_i^* - \frac{\Delta t}{\rho}\frac{\partial}{\partial x_i}(p^{n+1} - \gamma p^n) . \tag{2.52}$$

The above PFEM formulation is based on the paper by Idelsohn, Oñate and Del Pin [**?**]. The authors described an approach using arbitrary time discretization scheme and pressure split factor. Their choice of implicit scheme $\theta = 1$ was motivated by better convergence properties, whereas the decision for $\gamma = 0$ leading to greater pressure split was driven by better pressure stabilization.

2.4.4 Spatial discretization

The unknown functions of velocity and pressure are approximated using equal order interpolation for all variables in the final configuration

$$u_i = \boldsymbol{N}^T(X,t)\boldsymbol{U}_i \tag{2.53}$$
$$p = \boldsymbol{N}^T(X,t)\boldsymbol{P}\,. \tag{2.54}$$

By applying the Galerkin weighted residual method on the splitted governing equations, following system of linear algebraic equations is obtained

$$\boldsymbol{M}\boldsymbol{U}^* = \boldsymbol{M}\boldsymbol{U}^n + \Delta t\boldsymbol{F} - \frac{\Delta t\mu}{\rho}\boldsymbol{K}\boldsymbol{U}^{n+\theta}\,, \tag{2.55}$$
$$\boldsymbol{L}\boldsymbol{P}^{n+1} = \frac{\rho}{\Delta t}\left(\boldsymbol{G}^T\boldsymbol{U}^* - \hat{\boldsymbol{U}}\right)\,, \tag{2.56}$$
$$\boldsymbol{M}\boldsymbol{U}^{n+1} = \boldsymbol{M}\boldsymbol{U}^* - \frac{\rho}{\Delta t}\boldsymbol{G}\boldsymbol{P}^{n+1}\,. \tag{2.57}$$

The matrix $\boldsymbol{M}$ denotes the mass matrix in a lumped form, whereas the vector $\boldsymbol{F}$ stands for the load vector. The matrix $\boldsymbol{G}$ represents the gradient operator, which is the transposition of the divergence operator denoted simply as $\boldsymbol{G}^T$. Matrices $\boldsymbol{K}$ and $\boldsymbol{L}$ are build in a similar way however noted differently. Both mean the Laplacian operator. Due to its common use in computational mechanics, the classical notation of the stiffness matrix $\boldsymbol{K}$ is used. Prescribed velocity components are enclosed in vector $\hat{\boldsymbol{U}}$.
In each computational time step, an iteration is performed until the equilibrium is reached. Depending on the value of θ used, the equation system for the components of the auxiliary velocity U_i^* (2.55) can be solved either explicitly $\theta = 0$ or implicitly $\theta \neq 0$. Then, the calculated values of the auxiliary velocity are used as input for the pressure computation (2.56). The last system of equations (2.57) determines the velocity values at the end of the time step, taking auxiliary velocities and pressure or pressure increments into account.
Let us summarize the iterative step. The position of the particles at the end of the previous time step is known, as well as the the value of the velocity u^n and pressure p^n. The set of governing equations is build up for the unknowns at the end of the solution step θ^{n+1}, however based on the geometry of the previous step. The changes in the position are neglected. Once the convergence is reached, the final position is computed from the old one modified by the displacement due to obtained velocity. After that, solution can proceed to the next time step.

Chapter 3

Elements

Chapter 4

Constitutive Equations

The purpose of this chapter is to present the theoretical backgroung of some handy general purpose theories and algorithms, that are provided in oofem in the form of general material base classes. They can significantly facilitate the implementation of particular material models that are based on such concepts. Typical example can be a general purpose plasticity class, that implements general stress return and stifness matrix evaluation algorithms, based on provided methods for computing yield functions and corresponding derivatives. Particular models are simply derived from the base classes, inheriting common algorithms.

4.1 Isotropic damage model

In this section, the formulation of an isotropic damage model will be described. To cover the various models based on isotropic damage concept, a base class IsotropicDamageMaterial is defined first, declaring the necessary services and providing the implementation of them, which are general. The derived classes then only implement a particular damage-evolution law.
The isotropic damage models are based on the simplifying assumption that the stiffness degradation is isotropic, i.e., stiffness moduli corresponding to different directions decrease proportionally and independently of direction of loading. Consequently, the damaged stiffness matrix is expressed as

$$\boldsymbol{D} = (1-\omega)\boldsymbol{D}_e,$$

where $\boldsymbol{D}_e$ is elastic stiffness matrix of the undamaged material and ω is the damage parameter. Initially, ω is set to zero, representing the virgin undamaged material, and the response is linear-elastic. As the material undergoes the deformation, the initiation and propagation of microdefects decreases the stiffness, which is represented by the growth of the damage parameter ω. For $\omega = 1$, the stiffness completely disappears.
In the present context, the $\boldsymbol{D}$ matrix represents the secant stiffness that relates the total strain to the total stress

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon} = (1-\omega)\boldsymbol{D}_e\boldsymbol{\varepsilon}.$$

Similarly to the theory of plasticity, a loading function f is introduced. In the damage theory, it is natural to work in the strain space and therefore the loading function is depending on the strain and on an additional parameter κ, describing the evolution of the damage. Physically, κ is a scalar measure of the largest strain level ever reached. The loading function usually has the form

$$f(\boldsymbol{\varepsilon}, \kappa) = \tilde{\varepsilon}(\boldsymbol{\varepsilon}) - \kappa,$$

where $\tilde{\varepsilon}$ is the equivalent strain, i.e., the scalar measure of the strain level. Damage can grow only if current state reaches the boundary of elastic domain ($f = 0$). This is expressed by the following loading/unloading conditions

$$f \leq 0, \quad \dot{\kappa} \geq 0, \quad \dot{\kappa} f = 0.$$

It remains to link the variable κ to the damage parameter ω. As both κ and ω grow monotonically, it is convenient to postulate an explicit evolution law

$$\omega = g(\kappa).$$

The important advantage of this explicit formulation is that the stress corresponding to the given strain can be evaluated directly, without the need to solve the nonlinear system of equations. For the given strain, the corresponding stress is computed simply by evaluating the current equivalent strain, updating the maximum previously reached equivalent strain value κ and the damage parameter and reducing the effective stress according to $\boldsymbol{\sigma} = (1-\omega)\boldsymbol{D}_e\boldsymbol{\varepsilon}$. This general framework for computing stresses and stiffness matrix is common for all material models of this type. Therefore, it is natural to introduce the base class for all isotropic-based damage models which provides the general implementation for the stress and stiffness matrix evaluation algorithms. The particular models then only provide their equivalent strain and damage evolution law definitions. The base class only declares the virtual services for computing equivalent strain and corresponding damage. The implementation of common services uses these virtual functions, but they are only declared at IsotropicDamageMaterial class level and have to be implemented by the derived classes. Together with the material model, the corresponding status has to be defined, containing all necessary history variables. For the isotropic-based damage models, the only history variable is the value of the largest strain level ever reached (κ). In addition, the corresponding damage level ω will be stored. This is not necessary because damage can be always computed from corresponding κ. The IsotropicDamageMaterialStatus class is derived from StructuralMaterialStatus class. The base class represents the base material status class for all structural statuses. At StructuralMaterialStatus level, the attributes common to all "structural analysis" material models - the strain and stress vectors (both the temporary and non-temporary) are introduced. The corresponding services for accessing, setting, initializing, and updating these attributes are provided. Therefore, only the κ and ω parameters are introduced (both the temporary and non-temporary). The corresponding services for manipulating these attributes are added and services for context initialization, update, and store/restore operations are overloaded, to handle the history parameters properly.

4.2 Multisurface plasticity driver - MPlasticMaterial class

In this section, a general multisurface plasticity theory with hardening/softening is reviewed. The presented algorithms are implemented in MPlasticMaterial class.

4.2.1 Plasticity overview

Let $\boldsymbol{\sigma}, \boldsymbol{\varepsilon}$, and $\boldsymbol{\varepsilon}^p$ be the stress, total strain, and plastic strain vectors, respectively. It is assumed that the total strain is decomposed into reversible elastic and irreversible plastic parts

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^p \tag{4.1}$$

The elastic response is characterized in terms of elastic constitutive matrix $\boldsymbol{D}$ as

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}^e = \boldsymbol{D}(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p) \tag{4.2}$$

As long as the stress remains inside the elastic domain, the deformation process is purely elastic and the plastic strain does not change. It is assumed that the elastic domain, denoted as IE is bounded by a composite yield surface. It is defined as

$$IE = \{(\boldsymbol{\sigma}, \boldsymbol{\kappa}) | f_i(\boldsymbol{\sigma}, \boldsymbol{\kappa}) < 0, \text{for all i} \in \{1, \cdots, \text{m}\}\} \tag{4.3}$$

where $f_i(\boldsymbol{\sigma}, \boldsymbol{\kappa})$ are $m \geq 1$ yield functions intersecting in a possibly non-smooth fashion. The vector $\boldsymbol{\kappa}$ contains internal variables controlling the evolution of yield surfaces (amount of hardening or softening). The evolution of plastic strain $\boldsymbol{\varepsilon}^p$ is expressed in Koiter's form. Assuming the non-associated plasticity, this reads

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_{i=1}^{m} \lambda^i \partial_{\boldsymbol{\sigma}} g_i(\boldsymbol{\sigma}, \boldsymbol{\kappa}) \tag{4.4}$$

where g_i are plastic potential functions. The λ^i are referred as plastic consistency parameters, which satisfy the following Kuhn-Tucker conditions

$$\lambda^i \geq 0,\ f_i \leq 0,\ \text{and}\ \lambda^i f_i = 0 \tag{4.5}$$

These conditions imply that in the elastic regime the yield function must remain negative and the rate of the plastic multiplier is zero (plastic strain remains constant) while in the plastic regime the yield function must be equal to zero (stress remains on the surface) and the rate of the plastic multiplier is positive. The evolution of vector of internal hardening/softening variables $\boldsymbol{\kappa}$ is expressed in terms of a general hardening/softening law of the form

$$\dot{\boldsymbol{\kappa}} = \dot{\boldsymbol{\kappa}}(\boldsymbol{\sigma}, \boldsymbol{\lambda}) \tag{4.6}$$

where $\boldsymbol{\lambda}$ is the vector of plastic consistency parameters λ_i.

4.2.2 Closest-point return algorithm

Let us assume, that at time t_n the total and plastic strain vectors and internal variables are known

$$\{\boldsymbol{\varepsilon}_n, \boldsymbol{\varepsilon}^p_n, \boldsymbol{\kappa}_n\} \text{ given at } t_n$$

By applying an implicit backward Euler difference scheme to the evolution equations (4.2 and 4.4) and making use of the initial conditions the following discrete non-linear system is obtained

$$\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{\varepsilon}_n + \Delta\boldsymbol{\varepsilon} \tag{4.7}$$

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{D}(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}) \tag{4.8}$$

$$\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \sum \lambda^i \partial_{\boldsymbol{\sigma}} g_i(\boldsymbol{\sigma}_{n+1}, \boldsymbol{\kappa}_{n+1}) \tag{4.9}$$

In addition, the discrete counterpart of the Kuhn-Tucker conditions becomes

$$f_i(\boldsymbol{\sigma}_{n+1}, \boldsymbol{\kappa}_{n+1}) = 0 \tag{4.10}$$

$$\lambda^i_{n+1} \geq 0 \tag{4.11}$$

$$\lambda^i_{n+1} f_i(\boldsymbol{\sigma}_{n+1}, \boldsymbol{\kappa}_{n+1}) = 0 \tag{4.12}$$

In the standard displacement-based finite element analysis, the strain evolution is determined by the displacement increments computed on the structural level. The basic task on the level of a material point is to evaluate the stress evolution generated by strain history. According to this, the strain driven algorithm is assumed, i.e. that the total strain $\boldsymbol{\varepsilon}_{n+1}$ is given. Then, the Kuhn-Tucker conditions determine whether a constraint is active. The set of active constraints is denoted as J_{act} and is defined as

$$J_{act} = \{\beta \in \{1, \cdots, m\} | f_\beta = 0\ \&\ \dot{f}_\beta = 0\} \tag{4.13}$$

Let's start with the definition of the residual of plastic flow

$$\boldsymbol{R}_{n+1} = -\boldsymbol{\varepsilon}^p_{n+1} + \boldsymbol{\varepsilon}^p_n + \sum_{j \in J_{act}} \lambda^j_{n+1} \partial_\sigma g_{n+1} \tag{4.14}$$

By noting that total strain $\boldsymbol{\varepsilon}_{n+1}$ is fixed during the increment we can express the plastic strain increment using (4.2) as

$$\Delta\boldsymbol{\varepsilon}^p_{n+1} = -\boldsymbol{D}\Delta\boldsymbol{\sigma}_{n+1} \tag{4.15}$$

The linearization of the plastic flow residual (4.14) yields[1]

$$\begin{aligned} &\boldsymbol{R} + \boldsymbol{D}^{-1}\Delta\boldsymbol{\sigma} + \sum \lambda \partial_{\sigma\sigma} g \Delta\boldsymbol{\sigma} + \\ &+ \sum \lambda \partial_{\sigma\kappa} g \cdot (\partial_\sigma \kappa \Delta\boldsymbol{\sigma} + \partial_\lambda \kappa \Delta\lambda) + \sum \Delta\lambda \partial_\sigma g = 0 \end{aligned} \tag{4.16}$$

[1]For brevity, the simplified notation is introduced: $f = f(\boldsymbol{\sigma}, \boldsymbol{\kappa}),\ g = g(\boldsymbol{\sigma},\ \boldsymbol{\kappa}),\ \boldsymbol{\kappa} = \boldsymbol{\kappa}(\boldsymbol{\sigma}, \lambda)$, and subscript $n+1$ is omitted.

From the previous equation, the stress increment $\Delta\boldsymbol{\sigma}$ can be expressed as

$$\Delta\boldsymbol{\sigma} = -\boldsymbol{H}^{-1}\left(\boldsymbol{R} + \sum \Delta\lambda\partial_\sigma g + \sum \lambda\partial_{\sigma\kappa} g\partial_\lambda\kappa\Delta\lambda\right) \tag{4.17}$$

where $\boldsymbol{H}$ is algorithmic moduli defined as

$$\boldsymbol{H} = \left[\boldsymbol{D}^{-1} + \sum \lambda\partial_{\sigma\sigma} g + \sum \lambda\partial_{\sigma\kappa} g\partial_\sigma\kappa\right] \tag{4.18}$$

Differentiation of active discrete consistency conditions (4.10) yields

$$\boldsymbol{f} + \partial_\sigma \boldsymbol{f}\Delta\boldsymbol{\sigma} + \partial_\kappa \boldsymbol{f}(\partial_\sigma\kappa\Delta\boldsymbol{\sigma} + \partial_\lambda\kappa\Delta\lambda) = 0 \tag{4.19}$$

Finally, by combining equations (4.17) and (4.19), one can obtain expression for incremental vector of consistency parameters $\Delta\boldsymbol{\lambda}$

$$\left[\boldsymbol{V}^T\boldsymbol{H}^{-1}\boldsymbol{U} - \partial_\kappa \boldsymbol{f}\partial_\lambda\boldsymbol{\kappa}\right]\Delta\boldsymbol{\lambda} = \boldsymbol{f} - \boldsymbol{V}^T\boldsymbol{H}^{-1}\boldsymbol{R} \tag{4.20}$$

where the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are defined as

$$\boldsymbol{U} = \left[\partial_\sigma \boldsymbol{g} + \sum \lambda\partial_{\sigma\kappa} g\partial_\lambda\kappa\right] \tag{4.21}$$

$$\boldsymbol{V} = [\partial_\sigma \boldsymbol{f} + \partial_\kappa \boldsymbol{f}\partial_\sigma\kappa] \tag{4.22}$$

Before presenting the final return mapping algorithm, the algorithm for determination of the active constrains should be discussed. A yield surface $f_{i,n+1}$ is active if $\lambda^i_{n+1} > 0$. A systematic enforcement of the discrete Kuhn-Tucker condition (4.10), which relies on the solution of return mapping algorithm, then serves as the basis for determining the active constraints. The starting point in enforcing (4.10) is to define the trial set

$$J^{trial}_{act} = \{j \in \{1, \cdots, m\} | f^{trial}_{j,n+1} > 0\} \tag{4.23}$$

where $J_{act} \subseteq J^{trial}_{act}$. Two different procedures can be adopted to determine the final set J_{act}. The conceptual procedure is as follows

- Solve the closest point projection with $J_{act} = J^{trial}_{act}$ to obtain final stresses, along with $\lambda^i_{n+1},\ i \in J^{trial}_{act}$.
- Check the sign of λ^i_{n+1}. If $\lambda^i_{n+1} < 0$, for some $i \in J^{trial}_{act}$, drop the $i-$th constrain from the active set and goto first point. Otherwise exit.

In the procedure 2, the working set J^{trial}_{act} is allowed to change within the iteration process, as follows

- Let $J^{(k)}_{act}$ be the working set at the k-th iteration. Compute increments $\Delta\lambda^{i,(k)}_{n+1},\ i \in J^{(k)}_{act}$.
- Update and check the signs of $\Delta\lambda^{i,(k)}_{n+1}$. If $\Delta\lambda^{i,(k)}_{n+1} < 0$, drop the i-th constrain from the active set $J^{(k)}_{act}$ and restart the iteration. Otherwise continue with next iteration.

If the consistency parameters $\Delta\lambda^i$ can be shown to increase monotonically within the return mapping algorithm, the the latter procedure is preferred since it leads to more efficient computer implementation.
The overall algorithm is convergent, first order accurate and unconditionally stable. The general algorithm is summarized in Tab. 4.2.2.

1. Elastic predictor

   (a) Compute Elastic predictor (assume frozen plastic flow)
   $\boldsymbol{\sigma}_{n+1}^{trial} = \boldsymbol{D}\left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_n^p\right)$
   $f_{i,n+1}^{trial} = f_i(\boldsymbol{\sigma}_{n+1}^{trial}, \boldsymbol{\kappa}_n),$ for i $\in \{1, \cdots, \mathrm{m}\}$

   (b) Check for plastic processes IF $f_{i,n+1}^{trial} \leq 0$ for all $i \in \{1, \cdots, m\}$ THEN:

   Trial state is the final state, EXIT.

   ELSE:

   $J_{act}^{(0)} = \{i \in \{1, \cdots, m\} | f_{i,n+1}^{trial} > 0\}$
   $\boldsymbol{\varepsilon}_{n+1}^{p(0)} = \boldsymbol{\varepsilon}_n^p,\ \boldsymbol{\kappa}_{n+1}^{(0)} = \boldsymbol{\kappa}_n,\ \lambda_{n+1}^{i(0)} = 0$

   ENDIF

2. Plastic Corrector

   (c) Evaluate plastic strain residual

   $\boldsymbol{\sigma}_{n+1}^{(k)} = \boldsymbol{D}\left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_{n+1}^{p(k)}\right)$
   $\boldsymbol{R}_{n+1}^{(k)} = -\boldsymbol{\varepsilon}_{n+1}^{p(k)} + \boldsymbol{\varepsilon}_n^p + \sum \lambda_{n+1}^{i(k)} \partial_\sigma g_i$

   (d) Check convergence

   $f_{i,n+1}^{(k)} = f_i(\boldsymbol{\sigma}_{n+1}^{(k)}, \boldsymbol{\kappa}_{n+1}^{(k)})$
   if $f_{i,n+1}^{(k)} < TOL$, for all $i \in J_{act}^{(k)}$ and $\|\boldsymbol{R}_{n+1}^{(k)}\| < TOL$ then EXIT

   (e) Compute consistent moduli

   $\boldsymbol{G} = \left[\boldsymbol{V}^T \boldsymbol{H}^{-1} \boldsymbol{U} - \partial_\kappa \boldsymbol{f} \partial_\lambda \boldsymbol{\kappa}\right]^{-1}$

   (f) Obtain increments to consistency parameter

   $\Delta\boldsymbol{\lambda}_{n+1}^{(k)} = \boldsymbol{G}\{\boldsymbol{f} - \boldsymbol{V}^T \boldsymbol{H}^{-1} \boldsymbol{R}\}_{n+1}^{(k)}$
   If using procedure 2 to determine active constrains, then update the active set and restart iteration if necessary

   (g) Obtain increments of plastic strains and internal variables

   $\Delta\boldsymbol{\varepsilon}_{n+1}^{p(k)} = \boldsymbol{D}^{-1}\left\{\boldsymbol{R}_{n+1}^{(k)} + \sum \Delta\lambda_{n+1}^{i(k)} \partial_\sigma g_{n+1}^{(k)} + \sum \lambda_{n+1}^{i(k)} \partial_{\sigma\kappa} g_{n+1}^{(k)} \partial_\lambda \kappa \Delta\lambda_{n+1}^{i(k)}\right\}$
   $\Delta\boldsymbol{\kappa}_{n+1}^{(k)} = \dot{\boldsymbol{\kappa}}(\boldsymbol{\sigma}^{(k)_{n+1}}, \boldsymbol{\lambda}_{n+1}^k)$

   (h) Update state variables

   $\boldsymbol{\varepsilon}_{n+1}^{p(k+1)} = \boldsymbol{\varepsilon}_{n+1}^{p(k)} + \Delta\boldsymbol{\varepsilon}_{n+1}^{p(k)}$
   $\boldsymbol{\kappa}_{n+1}^{(k+1)} = \boldsymbol{\kappa}_{n+1}^{(k)} + \Delta\boldsymbol{\kappa}_{n+1}^{(k)}$
   $\lambda_{n+1}^{i(k+1)} = \lambda_{n+1}^{i(k)} + \Delta\boldsymbol{\lambda}_{n+1}^{(k)},\ i \in J_{act}$

   (i) Set k=k+1 and goto step (b)

Table 4.1: General multisurface closest point algorithm

4.2.3 Algorithmic stiffness

Differentiation of the elastic stress-strain relations (4.8) and the discrete flow rule (4.9) yields

$$d\boldsymbol{\sigma}_{n+1} = \boldsymbol{D}\left(d\boldsymbol{\varepsilon}_{n+1} - d\boldsymbol{\varepsilon}^p_{n+1}\right) \tag{4.24}$$

$$d\boldsymbol{\varepsilon}^p_{n+1} = \sum \left(\lambda^i \partial_{\sigma\sigma} g d\boldsymbol{\sigma} + \lambda^i \partial_{\sigma\kappa} g \left(\partial_\sigma \boldsymbol{\kappa} d\boldsymbol{\sigma} + \partial_\lambda \boldsymbol{\kappa} d\lambda^i\right) + d\lambda^i \partial_\sigma g\right) \tag{4.25}$$

Combining this two equations, one obtains following relation

$$d\boldsymbol{\sigma} = \boldsymbol{\Xi}_{n+1}\left\{d\boldsymbol{\varepsilon}_{n+1} - \sum \lambda^i \partial_{\sigma\kappa} g \partial_\lambda \boldsymbol{\kappa} d\lambda^i - \sum d\lambda^i \partial_\sigma g\right\} \tag{4.26}$$

where $\boldsymbol{\Xi}_{n+1}$ is the algorithmic moduli defined as

$$\boldsymbol{\Xi}_{n+1} = \left[\boldsymbol{D}^{-1} + \sum \lambda^i \partial_{\sigma\sigma} g + \sum \lambda \partial_{\sigma\kappa} g \partial_\sigma \boldsymbol{\kappa}\right] \tag{4.27}$$

Differentiation of discrete consistency condition yields

$$\partial_\sigma f^i d\boldsymbol{\sigma} + \partial_\kappa f^i (\partial_\sigma \boldsymbol{\kappa} d\boldsymbol{\sigma} + \partial_\lambda \boldsymbol{\kappa} d\boldsymbol{\lambda}) = 0 \tag{4.28}$$

By substitution of (4.26) into (4.28) the following relation is obtained

$$d\boldsymbol{\lambda} = \boldsymbol{G}\left\{\boldsymbol{V}\boldsymbol{\Xi} d\varepsilon\right\} \tag{4.29}$$

where matrix $\boldsymbol{G}$ is defined as

$$\boldsymbol{G} = \left[\boldsymbol{V}^T \boldsymbol{\Xi} \boldsymbol{U} - \partial_\kappa \boldsymbol{f} \partial_\lambda \boldsymbol{\kappa}\right]^{-1} \tag{4.30}$$

Finally, by substitution of (4.30) into (4.26) one obtains the algorithmic elastoplastic tangent moduli

$$\frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\boldsymbol{\varepsilon}}|_{n+1} = \boldsymbol{\Xi} - \boldsymbol{\Xi}\boldsymbol{U}\left(\boldsymbol{V}\boldsymbol{\Xi}\boldsymbol{U} - [\partial_\kappa f][\partial_\lambda \boldsymbol{\kappa}]\right)\boldsymbol{V}\boldsymbol{\Xi} \tag{4.31}$$

4.2.4 Implementation of particular models

As follows from previous sections, a new plasticity based class has to provide only some model-specific services. The list of services, that should be implemented includes (for full reference, please consult documentation of MPlasticMaterial class):

- method for computing the value of yield function (computeYieldValueAt service)
- method for computing stress gradients of yield and load functions (method computeStressGradientVector)
- method for computing hardening variable gradients of yield and load functions (method computeKGradientVector)
- methods for computing gradient of hardening variables with respect to stress and plastic multipliers vectors (computeReducedHardeningVarsSigmaGradient and computeReducedHardeningVarsLamGradient methods)
- method for evaluating the increments of hardening variables due to reached state (computeStrainHardeningVarsIncrement)
- methods for computing second order derivatives of load and yield functions (computeReducedSSGradientMatrix and computeReducedSKGradientMatrix methods). Necessary only if consistent stiffness is required.

Chapter 5

Boundary conditions