CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF CIVIL ENGINEERING
DEPARTMENT OF MECHANICS

SHORT MANUAL OF THE THM MODEL FOR NUMERICAL ANALYSES OF BENTONITE MATERIALS IN ENGINEERED BARRIERS

By

Tomáš Krejčí, Jaroslav Kruis, and Tomáš Koudelka

Prague June 2022

Contents

List of Figures

List of Tables

Chapter 1

Manual summary

1.1 Introduction

This short manual is devoted to a module of SIFEL code - an open-source finite element (FE) computer code designed for Thermo-hygro-mechanical analyses of bentonite materials used in engineered barriers for nuclear waste storage. This software was developed in the scope of project No. TK01010063 supported by the Technology Agency of the Czech Republic. The Thermo-hygro-mechanical model is based on a micromechanical approach of non-isothermal water and vapor flow in a porous medium presented in [Schrefler and Lewis, 1998] and a double structure hypoplastic model for expansive clays presented in [Mašín, 2013], [Mašín and Khalili, 2016], and [Mašín, 2017]. Both models are extended, coupled, and implemented into the finite element code.

The manual is organized into three chapters. After the introduction in Chapter 1, Chapter 2 describes the mathematical background of implementing the FEM for the coupled problems in the SIFEL software package. The chapter summarizes the basic ideas of the code, which is written in C++. There is a short description of several base classes used in the code which store fundamental data used in the FEM analysis, such as topology, matrices, solvers, and internal state variables. The computer implementation is illustrated by a simple benchmark of watering a bentonite sample. The last Chapter 3 presents the program compilation and running with the description of input files.

Chapter 2

Numerical solution of coupled problems and its computer implementation

2.1 Coupled problem

2.1.1 Partially coupled approach

The coupled analysis is based on the equation of the mechanical problem written in matrix form derived from the finite element method

$$\boldsymbol{K}_u \boldsymbol{d}_u = \boldsymbol{f}_{ext}, \tag{2.1}$$

and on the equation for the transport problem also written in matrix form derived from the finite element method

$$\boldsymbol{K}_T \boldsymbol{d}_T + \boldsymbol{C}_T \dot{\boldsymbol{d}}_T = \boldsymbol{f}_T \ . \tag{2.2}$$

For example, the thermal effect on the mechanical response is usually added in to the constitutive equation (Hook's law) relating the strains $\boldsymbol{\varepsilon}$ and stresses $\boldsymbol{\sigma}$

$$\boldsymbol{\sigma} = \boldsymbol{D}_u \left(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_T \right), \tag{2.3}$$

where $\boldsymbol{D}_u$ is the stiffness matrix of the material, $\boldsymbol{\varepsilon}$ is the vector of total strains, and the vector of thermal strains, $\boldsymbol{\varepsilon}_T$, depends on the thermal expansion coefficient, α_T, and the difference of the actual temperature T, and the initial temperature, T_0,

$$\boldsymbol{\varepsilon}_T = \boldsymbol{m}^{\mathrm{T}} \alpha_T (T - T_0), \qquad \boldsymbol{m} = (1, 1, 1, 0, 0)\,. \tag{2.4}$$

After the FEM discretisation, the right-hand side of the mechanical problem is extended by temperature effect

$$\boldsymbol{K}_u \boldsymbol{d}_u = \boldsymbol{f}_{ext} \ + \boldsymbol{f}_{Tu} \tag{2.5}$$

where

$$\boldsymbol{f}_{uT} = \int_{\Omega} \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \boldsymbol{\varepsilon}_T \mathrm{d}\Omega \ . \tag{2.6}$$

The above system of Equation (2.5) represents a partially coupled problem, so-called one way coupled problem, where the mechanical problem is influenced by the heat transfer problem. For the numerical solution, it is convenient to use a staggered algorithm, in which both transport and mechanical analysis are solved simultaneously in time. The data from transport analysis are transferred only to mechanical analysis. It means the heat transfer analysis is solved first, and then in each time step, the temperatures are transferred to the mechanical part to compute thermal strains. From the numerical point of view, the rate form of Equation (2.5) is more convenient

$$\boldsymbol{K}_u \dot{\boldsymbol{d}}_u = \dot{\boldsymbol{f}}_{ext} + \dot{\boldsymbol{f}}_{uT} \tag{2.7}$$

2.1.2 Fully coupled approach

If the mechanical material properties are influenced by temperature changes or the mechanical response is non-linear, it is convenient to solve both transport and mechanical parts together in a fully coupled analysis. In such a problem, the vector with thermal strains is split into two parts

$$\boldsymbol{f}_{uT} = \int_\Omega \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \boldsymbol{\varepsilon}_T \mathrm{d}\Omega = \int_\Omega \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \alpha_T \boldsymbol{m}^{\mathrm{T}} \boldsymbol{N}_T \mathrm{d}\Omega\ \boldsymbol{d}_T - \int_\Omega \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \alpha_T \boldsymbol{m}^{\mathrm{T}} \boldsymbol{N}_T \mathrm{d}\Omega\ \boldsymbol{d}_{T0}\ , \tag{2.8}$$

where the vector $\boldsymbol{d}_{T0}$ contains initial nodal temperatures. Merging of both problem (2.5) and (2.2) together and applying of previous decomposition (2.8) lead to the system of equations for the fully coupled problem

$$\begin{pmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{uT} \\ \boldsymbol{0} & \boldsymbol{K}_{TT} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_T \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{TT} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{d}}_u \\ \dot{\boldsymbol{d}}_T \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{ext} + \boldsymbol{f}_{uT}^0 \\ \boldsymbol{f}_T \end{pmatrix} , \tag{2.9}$$

where $\boldsymbol{d}_T$ is the vector of nodal temperatures, and $\boldsymbol{f}_T$ is the vector of prescribed nodal heat fluxes and sources. The first equation in the system (2.9) expresses the equilibrium condition while the second equation in this system of equation represents the heat balance condition. The zero blocks in the heat balance equation determine the independence of the heat transfer on the mechanical problem. On the other hand, the mechanical problem is coupled with the heat transfer through coupling matrix $\boldsymbol{K}_{uT}$ and vector $\boldsymbol{f}_{uT}^0$ resulting from the first and the second part of the vector $\boldsymbol{f}_{uT}$ in Equation (2.8), respectively.

$$\boldsymbol{K}_{uT} = -\int_\Omega \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \alpha_T \boldsymbol{m}^{\mathrm{T}} \boldsymbol{N}_T \mathrm{d}\Omega \ , \quad \boldsymbol{f}_{uT}^0 = -\int_\Omega \boldsymbol{B}_u^{\mathrm{T}} \boldsymbol{D}_u \alpha_T \boldsymbol{m}^{\mathrm{T}} \boldsymbol{N}_T \mathrm{d}\Omega\ \boldsymbol{d}_{T0}\ . \tag{2.10}$$

The matrix $\boldsymbol{K}_{uu}$ is the stiffness matrix previously denoted $\boldsymbol{K}_u$, the matrix $\boldsymbol{K}_{TT}$ is the conductivity matrix $\boldsymbol{K}_T$, and the matrix $\boldsymbol{C}_{TT}$ is the capacity matrix $\boldsymbol{C}_T$, respectively.

Slightly different system of equations is obtained when using the rate form for the mechanical part (2.7)

$$\begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{TT} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_T \end{pmatrix} + \begin{pmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{uT} \\ \boldsymbol{0} & \boldsymbol{C}_{TT} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{d}}_u \\ \dot{\boldsymbol{d}}_T \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{f}}_u \\ \boldsymbol{f}_T \end{pmatrix} . \tag{2.11}$$

The numerical solution of the system of Equations (2.9) follows the v-form or d-form algorithms presented for the transport problem by relations (2.21) to (2.23).

In the case of a more complicated coupled thermo-hygro-mechanical problem, the system may have the form with non-zeros off-diagonal blocks

$$\begin{pmatrix} \boldsymbol{C}_{uu} & \boldsymbol{C}_{uT} & \boldsymbol{C}_{u\varphi} \\ \boldsymbol{C}_{Tu} & \boldsymbol{C}_{TT} & \boldsymbol{C}_{T\varphi} \\ \boldsymbol{C}_{\varphi u} & \boldsymbol{C}_{\varphi T} & \boldsymbol{C}_{\varphi\varphi} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{d}}_u \\ \dot{\boldsymbol{d}}_T \\ \dot{\boldsymbol{d}}_\varphi \end{pmatrix} + \begin{pmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{uT} & \boldsymbol{K}_{u\varphi} \\ \boldsymbol{K}_{Tu} & \boldsymbol{K}_{TT} & \boldsymbol{K}_{T\varphi} \\ \boldsymbol{K}_{\varphi u} & \boldsymbol{K}_{\varphi T} & \boldsymbol{K}_{\varphi\varphi} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_T \\ \boldsymbol{d}_\varphi \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_u \\ \boldsymbol{f}_T \\ \boldsymbol{f}_\varphi \end{pmatrix}, \tag{2.12}$$

where the subscript u denotes the displacements, the subscripts φ denotes the relative humidity and the subscript T denotes the temperature. The vectors $\boldsymbol{d}_u$, $\boldsymbol{d}_T$, and $\boldsymbol{d}_\varphi$ contain unknown nodal variables, the vectors $\boldsymbol{f}_u$, $\boldsymbol{f}_T$, and $\boldsymbol{f}_\varphi$ represent prescribed nodal forces and fluxes, the matrices $\boldsymbol{K}$ with indices stands for the stiffness, conductivity and coupling matrices and the matrices $\boldsymbol{C}$ with indices denote the capacity and coupling matrices. The vectors $\boldsymbol{f}_u$, $\boldsymbol{f}_T$, and $\boldsymbol{f}_\varphi$ are further split into three contributions. The vector $\boldsymbol{f}_u$ is the sum of vectors $\boldsymbol{f}_{uu}$, $\boldsymbol{f}_{uT}$, $\boldsymbol{f}_{u\varphi}$ representing contributions to the nodal forces from mechanical analysis, temperature changes, and humidity changes. The meaning of other components is similar in the vectors $\boldsymbol{f}_T$ and $\boldsymbol{f}_\varphi$.

The system of differential equations (2.12) can be written more compactly in the form

$$\boldsymbol{C}(\boldsymbol{d})\dot{\boldsymbol{d}} + \boldsymbol{K}(\boldsymbol{d})\boldsymbol{d} = \boldsymbol{f} \ . \tag{2.13}$$

The dependency of the stiffness, conductivity, capacity, and coupling matrices on the attained values of variables is explicitly denoted.

It has to be noted that the permanent recalculation of matrices $\boldsymbol{K}$ and $\boldsymbol{C}$ with concerning actual nodal values is very computationally demanding. In such a case, the matrix of the system of equations $\boldsymbol{C}(\boldsymbol{d}) + \Delta t \alpha_T \boldsymbol{K}(\boldsymbol{d})$ has to be always factorized, requiring additional computational time. Experiences with numerical simulation show that the modified Newton-Raphson method, which changes the system matrices only at the beginning of a new time step, is suitable for weak non-linear problems. On the other hand, the full version of the Newton-Raphson method computing matrices in each inner iteration step in each time step is the best choice for analyses with strong non-linear dependency.

2.2 Solution of system of nonlinear equations

Equation (2.13) is a system of ordinary differential equations and its time integration is based on the general trapezoidal rule [Hughes, 1987].

$$\dot{\boldsymbol{d}} = \boldsymbol{v} \tag{2.14}$$

$$\boldsymbol{d}_{n+1} = \boldsymbol{d}_n + \Delta t \boldsymbol{v}_{n+\alpha} \tag{2.15}$$

$$\boldsymbol{v}_{n+\alpha} = (1-\alpha)\boldsymbol{v}_n + \alpha \boldsymbol{v}_{n+1} \ , \tag{2.16}$$

where subscripts denote the time step. The system of Equation (2.2) has the similar form at actual time step $n+1$

$$\boldsymbol{K}\boldsymbol{d}_{n+1} + \boldsymbol{C}\dot{\boldsymbol{d}}_{n+1} = \boldsymbol{f}_{n+1} \ . \tag{2.17}$$

If time approximations (2.14) to (2.16) are taken into account, the equation (2.17) leads to another form

$$(\boldsymbol{C} + \alpha \Delta t \boldsymbol{K})\, \boldsymbol{v}_{n+1} = \boldsymbol{f}_{n+1} - \boldsymbol{K}\boldsymbol{d}_n - (1-\alpha)\Delta t \boldsymbol{K}\boldsymbol{v}_n \ . \tag{2.18}$$

The predictor-corrector method can be used for the computer implementation, where the predictor reads

$$\tilde{\boldsymbol{d}}_{n+1} = \boldsymbol{d}_n + (1-\alpha)\Delta t \boldsymbol{v}_n \tag{2.19}$$

and the corrector has the form

$$\boldsymbol{d}_{n+1} = \tilde{\boldsymbol{d}}_{n+1} + \alpha \Delta t \boldsymbol{v}_{n+1} \ . \tag{2.20}$$

With the help of the predictor and corrector, Equation (2.18) is slightly modified

$$\left(\boldsymbol{C} + \alpha\Delta t \boldsymbol{K}\right)\boldsymbol{v}_{n+1} = \boldsymbol{f}_{n+1} - \boldsymbol{K}\tilde{\boldsymbol{d}}_{n+1} \ . \tag{2.21}$$

The system (2.21) contains time derivatives of the nodal values $\boldsymbol{v}_{n+1}$. This approach is called v-form, which is not always hassle-free from the numerical point of view. Therefore additional approach, called d-form, can be used. Time derivatives of nodal values are expressed from Equation (2.20) in the form

$$\boldsymbol{v}_{n+1} = \frac{1}{\alpha\Delta t}(\boldsymbol{d}_{n+1} - \tilde{\boldsymbol{d}}_{n+1}) \tag{2.22}$$

which is reasonable for $\alpha > 0$ and $\Delta t > 0$. Substitution of expression (2.22) to the balance equation (2.21) leads to the form

$$\left(\frac{1}{\alpha\Delta t}\boldsymbol{C} + \boldsymbol{K}\right)\boldsymbol{d}_{n+1} = \boldsymbol{f}_{n+1} + \frac{1}{\alpha\Delta t}\boldsymbol{C}\tilde{\boldsymbol{d}}_{n+1} \ . \tag{2.23}$$

In case of non-linear system of Equation (2.17), where material parameters are dependent on the temperature field, the Newton-Raphson method [Bittnar and Šejnoha, 1996], [Crisfield, 1991] has to be used in every time step. For example in the v form (2.21), the trial solution $\boldsymbol{v}_{n+1,0}$ of the system of equations is used for computation of the trial nodal values $\boldsymbol{d}_{n+1,0}$ which are obtained from Equations (2.15) and (2.16). Substitution of the trial solution back to the system of Equations (2.21) with actual matrices does not generally lead to equality. An iteration loop, called the inner iteration loop, in every time step is based on residual which can be computed from the relationship

$$\begin{aligned} \boldsymbol{r}_{n+1,j} &= \boldsymbol{f}_{n+1} - \boldsymbol{K}_n\left(\boldsymbol{d}_n + \Delta t(1-\alpha)\boldsymbol{v}_n\right) \\ &- \left(\boldsymbol{C}_{n+1,j} + \Delta t\alpha\boldsymbol{K}_{n+1,j}\right)\boldsymbol{v}_{n+1,j} \ , \end{aligned} \tag{2.24}$$

where $\boldsymbol{C}_{n+1,j}$ and $\boldsymbol{K}_{n+1,j}$ denote the matrices evaluated for $\boldsymbol{d}_{n+1,j}$ and j is the index of the inner loop. Corrections of nodal time derivatives are computed from the equation

$$\left(\boldsymbol{C}_{n+1,j} + \Delta t\alpha\boldsymbol{K}_{n+1,j}\right)\Delta\boldsymbol{v}_{n+1,j+1} = \boldsymbol{r}_{n+1,j} \tag{2.25}$$

and new time derivatives are obtained

$$\boldsymbol{v}_{n+1,j+1} = \boldsymbol{v}_{n+1,j} + \Delta\boldsymbol{v}_{n+1,j+1}. \tag{2.26}$$

Another approach how to solve the nonlinear algebraic Equations (2.17) comes from the equilibrium of fluxes (computed and prescribed) in nodes, which is taken over from the mechanical problems. This strategy is based on the equation

$$\boldsymbol{f}_{int} = \boldsymbol{f}_{ext} \tag{2.27}$$

where vectors $\boldsymbol{f}_{int}$ and $\boldsymbol{f}_{ext}$ contain internal values and prescribed/computed values, respectively. Both vector depend on time t, on the vector of unknown temperature $\boldsymbol{d}$ and on derivatives of unknown temperatures with respect to time $\boldsymbol{v}$. The vector $\boldsymbol{f}_{int}$ expressed at actual time step t_{n+1} has the form

$$\boldsymbol{f}_{int}(\boldsymbol{d}_{n+1}, \boldsymbol{v}_{n+1}, t_{n+1}) \approx \boldsymbol{f}_{int}(\boldsymbol{d}_n, \boldsymbol{v}_n, t_n) + \frac{\partial \boldsymbol{f}_{int}}{\partial \boldsymbol{d}} \Delta \boldsymbol{d}_n + \frac{\partial \boldsymbol{f}_{int}}{\partial \boldsymbol{v}} \Delta \boldsymbol{v}_n \ , \tag{2.28}$$

where $\Delta \boldsymbol{d}_n$ and $\Delta \boldsymbol{v}_n$ are increments over $\Delta t = t_{n+1} - t_n$. Previous relation can be rewritten with help of notation

$$\boldsymbol{K}_n = \frac{\partial \boldsymbol{f}_{int}(t_n)}{\partial \boldsymbol{d}}, \qquad \boldsymbol{C}_n = \frac{\partial \boldsymbol{f}_{int}(t_n)}{\partial \boldsymbol{v}} \ . \tag{2.29}$$

in the new expression

$$\boldsymbol{K}_n \Delta \boldsymbol{d}_n + \boldsymbol{C}_n \Delta \boldsymbol{v}_n = \boldsymbol{f}_{int}(t_{n+1}) - \boldsymbol{f}_{int}(t_n) \ . \tag{2.30}$$

There are two sets of relations

$$\boldsymbol{d}_{n+1} = \boldsymbol{d}_n + \Delta \boldsymbol{d}_n, \qquad \boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \Delta \boldsymbol{v}_n \tag{2.31}$$

and recalled Equations (2.15) and (2.16)

$$\boldsymbol{d}_{n+1} = \boldsymbol{d}_n + \Delta t \boldsymbol{v}_{n+\alpha}, \qquad \boldsymbol{v}_{n+\alpha} = (1-\alpha)\boldsymbol{v}_n + \alpha \boldsymbol{v}_{n+1} \ . \tag{2.32}$$

After substitution of (2.31) and (2.32) into expression (2.30)

$$(\boldsymbol{C}_n + \alpha \Delta \boldsymbol{K}_n)\, \boldsymbol{v}_{n+1} = \boldsymbol{f}_{int}(t_{n+1}) - \boldsymbol{f}_{int}(t_n) + (\boldsymbol{C}_n - \Delta t(1-\alpha)\boldsymbol{K}_n)\, \boldsymbol{v}_n \ , \tag{2.33}$$

where

$$\boldsymbol{f}_{int}(t_n) = \boldsymbol{K}_n \boldsymbol{d}_n + \boldsymbol{C}_n \boldsymbol{v}_n \tag{2.34}$$

is applied. The new vector $\boldsymbol{v}_{n+1}$ is calculated from Equation (2.32), and the vector $\boldsymbol{d}_{n+1}$ is then obtained. Due to nonlinearity in material properties, the equality

$$\boldsymbol{f}_{int}(t_{n+1}) = \boldsymbol{f}_{ext}(t_{n+1}) \tag{2.35}$$

is generally not valid and the residuum is computed

$$\boldsymbol{R}_{n+1} = \boldsymbol{f}_{ext}(t_{n+1}) - \boldsymbol{f}_{int}(t_{n+1}) \tag{2.36}$$

The vector $\boldsymbol{v}_{n+1}$ corrections has to be evaluated from the relation

$$(\boldsymbol{C}_{n+1} + \alpha \Delta t \boldsymbol{K}_{n+1})\, \Delta \boldsymbol{v}_{n+1,j} = \boldsymbol{R}_{n+1,j} \ . \tag{2.37}$$

The final vector $\boldsymbol{v}_{n+1}^{\mathrm{fin}}$ is the sum of contributions from inner iteraton loop

$$\boldsymbol{v}_{n+1}^{\mathrm{fin}} = \boldsymbol{v}_{n+1} + \sum_j \Delta \boldsymbol{v}_{n+1,j} \ . \tag{2.38}$$

The final equality is reached at the and of the inner iteration process

$$\boldsymbol{f}_{int}^{\mathrm{fin}}(\boldsymbol{d}_{n+1}^{\mathrm{fin}}, \boldsymbol{v}_{n+1}^{\mathrm{fin}}, t_{n+1}) = \boldsymbol{f}_{ext}^{\mathrm{fin}}(\boldsymbol{d}_{n+1}^{\mathrm{fin}}, \boldsymbol{v}_{n+1}^{\mathrm{fin}}, t_n). \tag{2.39}$$

If the matrices $\boldsymbol{C}$ and $\boldsymbol{K}$ are updated in every inner step, the full Newton-Raphson method is used. If the matrices are updated only once after every time step, the modified Newton-Raphson method is used.

2.3 Computer Code of the THM model

Experiences with the implementation of numerical methods, material models, and tools for coupled problems and parallel computing showed several contradicting requirements, namely in commercial software. It was decided to start the development of the new open-source code SIFEL [Kruis et al., 2021]. The acronym SIFEL was derived from SImple Finite ELements. The **Thermo-Hygro-Mechanical model** for bentonite materials was built as an extension of the SIFEL code. It uses its basic features and functions. This section describes the code's philosophy, the structure, used programming techniques, and data structures. More details about this software can also be found in references [Kruis et al., 2010] and [Koudelka et al., 2010].

The following requirements were determined when programming the code:

- Portability of the code. The universities had different hardware and software equipment. Notably, the portability between different operating systems was required (Linux, Windows, HP Unix).
- Simple programming techniques. The members of the project were experts in the branch of mechanical and transport processes with solid knowledge of programming languages but were not professional programmers. Source codes should be understandable for all team members as well as for new participants.
- Speed of program execution. The programming language should be compiled (FORTRAN, C++) rather than interpreted (Java).

Comparing to FORTRAN 77 and FORTRAN 90 languages, C++ was selected as more portable and comprehensive. Moreover, fast executable C++ compilers are other benefits. Sometimes, the extensive usage of object-oriented programming techniques decreases clarity for new participants. It was concluded that C++ language would be used without most object-oriented programming features and concepts. From the object-oriented programming point of view, data abstraction and encapsulation were found to be useful concepts and understandable for all project participants. Data are joined together with essential functions, which initialize them and perform basic operations. Compared to the usual recommendations, the data was left public initially, and it can be changed to private later depending on needs and experiences [Koudelka et al., 2011].

The easy extensibility of code is probably the most crucial requirement. Another essential need is connected with code performance. These two basic requirements for the system are contradictory because the very efficient implementation of a numerical method differs significantly from the description of the method in textbooks. Therefore the orientation in the code is complicated.

The attention was rather concentrated on the suitable formulation of the problem and the correct analysis. Detailed analysis of the system of nonlinear ordinary differential Equations (2.12) reveals the similarity of particular submatrices. The stiffness and conductivity matrices (denoted by $\boldsymbol{K}$ with appropriate subscripts) generally have the form

$$\boldsymbol{K}_{ij} = \int_{\Omega} \boldsymbol{B}_i^T \boldsymbol{D}_{ij} \boldsymbol{B}_j \mathrm{d}\Omega \ , \tag{2.40}$$

where $\boldsymbol{B}_i$ and $\boldsymbol{B}_j$ denote the gradient matrices, $\boldsymbol{D}_{ij}$ denotes the matrix of stiffness or conductivity of the material and the indexes i and j substitute any of indexes u, T, p_1 or p_2. Similarly, the capacity matrices (denoted by $\boldsymbol{C}$ with appropriate indexes) have generally the form

$$\boldsymbol{C}_{ij} = \int_\Omega \boldsymbol{N}_i^T \boldsymbol{H}_{ij} \boldsymbol{N}_j \mathrm{d}\Omega \ , \tag{2.41}$$

where $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$ denote the matrices of base functions, and $\boldsymbol{H}_{ij}$ denotes the matrix of material parameters.

The part of the computer code dealing with coupled analyses is created for easy and clear extensibility. When modeling, e.g., geomechanics, additional variables must be introduced in the constitutive equations, and additional balance equations must be added to the system. In such a case, the thermo-mechanical problem (2.9) extended by the pore pressures and capacity terms results in the general form

$$\begin{pmatrix} \boldsymbol{C}_{uu} & \boldsymbol{C}_{uT} & \boldsymbol{C}_{up_1} & \boldsymbol{C}_{up_2} \\ \boldsymbol{C}_{Tu} & \boldsymbol{C}_{TT} & \boldsymbol{C}_{Tp_1} & \boldsymbol{C}_{Tp_2} \\ \boldsymbol{C}_{p_1u} & \boldsymbol{C}_{p_1T} & \boldsymbol{C}_{p_1p_1} & \boldsymbol{C}_{p_1p_2} \\ \boldsymbol{C}_{p_2u} & \boldsymbol{C}_{p_2T} & \boldsymbol{C}_{p_2p_1} & \boldsymbol{C}_{p_2p_2} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{d}}_u \\ \dot{\boldsymbol{d}}_T \\ \dot{\boldsymbol{d}}_{p_1} \\ \dot{\boldsymbol{d}}_{p_2} \end{pmatrix} +$$

$$+ \begin{pmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{uT} & \boldsymbol{K}_{up_1} & \boldsymbol{K}_{up_2} \\ \boldsymbol{K}_{Tu} & \boldsymbol{K}_{TT} & \boldsymbol{K}_{Tp_1} & \boldsymbol{K}_{Tp_2} \\ \boldsymbol{K}_{p_1u} & \boldsymbol{K}_{p_1T} & \boldsymbol{K}_{p_1p_1} & \boldsymbol{K}_{p_1p_2} \\ \boldsymbol{K}_{p_2u} & \boldsymbol{K}_{p_2T} & \boldsymbol{K}_{p_2p_1} & \boldsymbol{K}_{p_2p_2} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_T \\ \boldsymbol{d}_{p_1} \\ \boldsymbol{d}_{p_2} \end{pmatrix} =$$

$$= \begin{pmatrix} \boldsymbol{f}_u \\ \boldsymbol{f}_T \\ \boldsymbol{f}_{p_1} \\ \boldsymbol{f}_{p_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{uu} + \boldsymbol{f}_{uT} + \boldsymbol{f}_{up_1} + \boldsymbol{f}_{up_2} \\ \boldsymbol{f}_{Tu} + \boldsymbol{f}_{TT} + \boldsymbol{f}_{Tp_1} + \boldsymbol{f}_{Tp_2} \\ \boldsymbol{f}_{p_1u} + \boldsymbol{f}_{p_1T} + \boldsymbol{f}_{p_1p_1} + \boldsymbol{f}_{p_1p_2} \\ \boldsymbol{f}_{p_2u} + \boldsymbol{f}_{p_2T} + \boldsymbol{f}_{p_2p_1} + \boldsymbol{f}_{p_2p_2} \end{pmatrix} , \tag{2.42}$$

where the index u denotes the displacements, the indexes p_1 and p_2 are the pore pressures, and T represents the temperature. The vectors $\boldsymbol{d}_u$, $\boldsymbol{d}_T$, $\boldsymbol{d}_{p1}$ and $\boldsymbol{d}_{p2}$ contain unknown nodal variables. The vectors $\boldsymbol{f}_u$, $\boldsymbol{f}_T$, $\boldsymbol{f}_{p1}$, and $\boldsymbol{f}_{p2}$ represent prescribed nodal forces and fluxes. The matrices $\boldsymbol{K}$ denote the stiffness, conductivity, and the matrices $\boldsymbol{C}$ denote the capacity and coupling matrices. The vectors $\boldsymbol{f}_u$, $\boldsymbol{f}_T$, $\boldsymbol{f}_{p1}$, and $\boldsymbol{f}_{p2}$ are further split into four contributions. The right-hand side vectors f are the sum of several components, e.g., the vector $\boldsymbol{f}_u$ is the sum of vectors $\boldsymbol{f}_{uu}$, $\boldsymbol{f}_{uT}$, $\boldsymbol{f}_{up1}$, and $\boldsymbol{f}_{up2}$, which represent contributions to the nodal forces from mechanical analysis, temperature changes, and pore pressures.

The solution of the system of Equation (2.42) directly offers the instruction for efficient implementation. The implementation of the coupled hygro-thermo-mechanical problems is based on three independent modules. The first module, MEFEL, is a separate computer code for mechanical analysis that can stand alone. This module can deal with pure mechanical analyses. It assembles submatrices $\boldsymbol{K}_{uu}$,$\boldsymbol{C}_{uu}$, and subvector $\boldsymbol{f}_{uu}$. The second module, TRFEL, is an independent computer code for heat and moisture transfer, which can also be used separately. It assembles the submatrices $\boldsymbol{K}_{TT}$, $\boldsymbol{K}_{Tp_1}$, $\boldsymbol{K}_{Tp_2}$, $\boldsymbol{K}_{p_1T}$, $\boldsymbol{K}_{p_1p_1}$, $\boldsymbol{K}_{p_1p_2}$, $\boldsymbol{K}_{p_2T}$, $\boldsymbol{K}_{p_2p_1}$, $\boldsymbol{K}_{p_2p_2}$, and subvectors $\boldsymbol{f}_{TT}$, $\boldsymbol{f}_{Tp_1}$, $\boldsymbol{f}_{Tp_2}$, $\boldsymbol{f}_{p_1T}$, $\boldsymbol{f}_{p_1p_1}$, $\boldsymbol{f}_{p_1p_2}$, $\boldsymbol{f}_{p_2T}$,

$\boldsymbol{f}_{p_2p_1}$, $\boldsymbol{f}_{p_2p_2}$. The coupling between the mechanical and transport part is implemented into the third module, METR, which deals with the off-diagonal terms in the coupled problem. This module assembles the submatrices $\boldsymbol{K}_{uT}$, $\boldsymbol{K}_{up_1}$, $\boldsymbol{K}_{up_2}$, $\boldsymbol{K}_{Tu}$, $\boldsymbol{K}_{p_1u}$, $\boldsymbol{K}_{p_2u}$, and subvectors $\boldsymbol{f}_{uT}$, $\boldsymbol{f}_{up_1}$, $\boldsymbol{f}_{up_2}$, $\boldsymbol{f}_{Tu}$, $\boldsymbol{f}_{p_1u}$, $\boldsymbol{f}_{p_2u}$.

At this time, many merging software concepts can be found in the literature, which consist of combinations of the existing computer codes and the data exchanges among them. Unfortunately, they result in staggered algorithms, and they cannot attain fully coupled analysis. In the SIFEL concept, the merging of the whole other parts of the code is not proceeding, but suitable subroutines from particular parts are used. Additionally, new subroutines dealing with the coupling terms had to be implemented. For an illustration of the merging complexity, the numbers of lines of the source code are summarized. The MEFEL code contains approximately 225100 lines, the TRFEL code contains 173800 lines, and the METR code contains 50900 lines. The number of lines of source code in METR is higher than the usual amount of lines in the typical merging code. On the other hand, it enables staggered and fully-coupled analysis, and the resulting code is compiled, therefore, very fast.

The additional advantage stems from the fact that any improvement of the mechanical or transport module is automatically projected to the code for coupled problems. It is also very convenient for developers who can deal with one part of the whole system only.

The program can solve stationary and non-stationary, linear and nonlinear problems of heat and moisture transfer as well as linear and nonlinear statics, eigenvibrations, dynamics, and time-dependent problems with neglected inertial forces. Various types of finite elements can model a 2D and 3D domain. In the SIFEL program, there are bar, triangular, quadrilateral, tetrahedron, and hexahedron elements implemented. Both types of approximation functions, linear and quadratic, can be used. Other features, such as the sequential construction modeling or parallel version of the code, can be found in references [Kruis et al., 2021] and [Koudelka et al., 2011].

2.3.1 Code structure

The code is split into independent parts that deal with a single physics problem. The part dealing with mechanical analysis is denoted MEFEL; the part dealing with transport processes is denoted TRFEL. There is also part GEFEL, which contains comprehensive tools needed in connection with the finite element method. The link of the mechanical part and the transport part is implemented in an additional part METR.

Let the matrix $\boldsymbol{K}$ defined in Equation (2.12) be assumed. It can be split into submatrices separated by the lines

$$\boldsymbol{K} = \left(\begin{array}{c|ccc} \boldsymbol{K}_{uu} & \boldsymbol{K}_{uT} & \boldsymbol{K}_{up_1} & \boldsymbol{K}_{up_2} \\ \hline \boldsymbol{K}_{Tu} & \boldsymbol{K}_{TT} & \boldsymbol{K}_{Tp_1} & \boldsymbol{K}_{Tp_2} \\ \boldsymbol{K}_{p_1u} & \boldsymbol{K}_{p_1T} & \boldsymbol{K}_{p_1p_1} & \boldsymbol{K}_{p_1p_2} \\ \boldsymbol{K}_{p_2u} & \boldsymbol{K}_{p_2T} & \boldsymbol{K}_{p_2p_1} & \boldsymbol{K}_{p_2p_2} \end{array}\right) \tag{2.43}$$

The diagonal block $\boldsymbol{K}_{uu}$ is the stiffness matrix, and it represents mechanical analyses only. This submatrix is assembled in the MEFEL module. The second diagonal block

$$\begin{pmatrix} \boldsymbol{K}_{TT} & \boldsymbol{K}_{Tp_1} & \boldsymbol{K}_{Tp_2} \\ \boldsymbol{K}_{p_1T} & \boldsymbol{K}_{p_1p_1} & \boldsymbol{K}_{p_1p_2} \\ \boldsymbol{K}_{p_2T} & \boldsymbol{K}_{p_2p_1} & \boldsymbol{K}_{p_2p_2} \end{pmatrix} \tag{2.44}$$

is the conductivity matrix, and it represents the transport process, where, e.g., heat and moisture are assumed. This submatrix is assembled in the TRFEL module. Two off-diagonal submatrices

$$\begin{pmatrix} \boldsymbol{K}_{uT} & \boldsymbol{K}_{up_1} & \boldsymbol{K}_{up_2} \end{pmatrix} \tag{2.45}$$

and

$$\begin{pmatrix} \boldsymbol{K}_{Tu} \\ \boldsymbol{K}_{p_1u} \\ \boldsymbol{K}_{p_2u} \end{pmatrix} \tag{2.46}$$

describe the coupling between mechanical behavior and transport processes, and they are assembled in the METR module.

2.3.2 Structure of MEFEL, TRFEL and METR

For each module (MEFEL, TRFEL, METR), the data describing the given problem are split into five large classes.

- `probdesc` - class containing the problem description,
- `top` - class including data relating finite element mesh,
- `mat` - class including data describing materials used,
- `crsec` - class representing data for cross-sections,
- `bclc` - class containing data representing boundary conditions and loadings.

The names of classes differ for particular problems by a postfix created from the problem name abbreviation. The data of these classes are necessary almost everywhere in the code, and this led to make them global objects. Thus, each class has one instance that is a global variable. This approach reduces the number of parameters passed to functions. In addition to that, each module contains global objects connected with the system matrices and vectors of unknowns.

The class `probdesc` contains attributes describing the solved problem. There is a group of attributes describing the type of problem, quantities computed, and solver of the systems of linear equations. Also, there is an object of class `hdbcontr`, which controls storage/re-storage of time steps to/from the disk. In the case of nonlinear problems, there are also objects of classes `timecon` and `nonlinman`. The `timecon` holds data controlling time steps while the `nonlinman` contains control parameters for the Newton-Raphson or arclength methods. The `probdesc` class has data members public because they are often used for reading, and they are seldom changed.

The class `top` contains topological data connected with the mesh of elements. It includes three essential arrays of objects of classes `node`, `element`, and `edge`. The class `node` contains data intended for the node, such as coordinates, the DOFs, and code numbers of particular DOFs. The class `element` provides nodal connectivity of the given element, type of material and cross-section, code numbers, etc. Similarly, the `edge` contains data describing boundaries. The `top` also includes arrays of adjacent nodes, elements, and distances of integration points.

The array of objects of `intpoints` is the most important data member of the class mat. The class `intpoints` contains intrinsic values computed in the particular integration points such as strains, stresses, fluxes, gradients, and other quantities. There are also arrays of initial conditions for integrations points, the array of values of unknowns from coupled problems, etc. For example, in the mechanical part, the `mechmat` class contains arrays of temperature and moisture values at integration points. The `mat` class also has arrays of objects of supported material types, i.e., implemented material models. Each material type has one object per one set of material parameters.

The class `crsec` contains arrays of objects for particular cross-section types. There are also methods for retrieving basic cross-section parameters such as thickness or area.

The `bclc` class holds data about boundary conditions that are arranged in particular load cases. Several load cases can be defined in static and also in time-dependent problems. Every load case can contain several sub-load cases due to better control of the time-dependent load. The boundary conditions can be specified for the given load case at nodes, elements, edges, and surfaces. Thus, `bclc` class contains the array of objects of the `loadcase` class, in which the boundary conditions are stored, array of initial conditions, and several auxiliary data members. The `bclc` class has only several methods for data manipulation, and the `loadcase` class provides most of the functionality.

2.3.3 Data Storage

Two sets of data are needed in the case of problems solved by the finite element method. There is a set of data describing finite element mesh, i.e., node coordinates and node connectivity. The second set contains values of state and derived variables (displacements, strains, stresses, plastic strains, temperatures, heat fluxes, etc.).

Finite element mesh

Two arrays of objects describe finite element mesh. One array contains objects of the class `node`, and the second array contains objects of the class `element`. The class `node` represents a node of finite element mesh. The class definition is in Table 2.1. It contains node coordinates (line 4), the number of degrees of freedom (line 2), and the ordering of DOFs in the whole problem (line 3). The class `element` represents a finite element, and its definition is in Table 2.2. It does not take into account whether the element is one, two, or three dimensional and does not care about the element shape (triangular, quadrilateral, etc.). Particular standalone objects provide all functionality connected with the FEM with implemented FE routines. These individual elements are referred from the class `element` by `et` data member. The class `element` contains the number of nodes defining the element (line 2), the number of DOFs per element (line 3), the number of Lagrange

```
1   class node{
2       long ndofn;
3       long *cn;
4       double x,y,z;
5   };
```

Table 2.1: Class node

```
1    class element{
2        long nne;
3        long ndofe;
4        long nmult;
5        long *nodes;
6        long cne;
7        long *cn;
8        long nip;
9        long iip;
10       elemtype et;
11   };
```

Table 2.2: Class element

multipliers (if they are needed on line 4), the list of nodes (line 5), the indicator whether the code numbers are defined on the element (line 6), the list of code numbers (line 7), the list of integration points located on the element (line 8), the number of integration points defined on the element (line 8), the number of the first integration point (line 9), and the element type identifier (line 10).

2.3.4 State variables

State variables are stored in integration points. The definition of integration point in the mechanical analysis is summarized in Table 2.3. The definition of integration point in transport processes is similar, but it contains arrays of fluxes and gradients. The integration point includes the type of material model (line 2), the number of components of strain/stress tensor (line 3). In the case of inelastic problem, some auxiliary values have to be stored. For example, in the analysis based on plasticity theory, the plastic strains and plastic multipliers have to be saved. For such purposes, the array `eqother` is defined. Unfortunately, one array is not enough because equilibrated values and trial values must be stored during global equilibrium iteration. Therefore, the array `eqother` contains equilibrated quantities, while the array other contains their trial values. The class comprises the number of components of the array `eqother` (line 4), the number of components of the array other (line 5), the array of the stress components (line 6), the array of the strain components (line 7), the array of other values (line 8), the array of `eqother` values (line 9), and the array of nonlocal values (line 10).

```
class intpoints{
     mattype tm;
     long ncompstr;
     long ncompeqother;
     long ncompother;
     double *stress;
     double *strain;
     double *other;
     double *eqother;
     double *nonloc;
};
```

Table 2.3: Class intpoint

2.3.5 Data access

Access to data can be described using an example dealing with stiffness matrix assembling. The function assembling global stiffness matrix contains a loop over all finite elements in a mesh. Each element calls its function for computation of the stiffness matrix. The typical form of function which computes the stiffness matrix of a single element is in Table 2.4. The number of nodes `nne` is known for each element, see 2.2. The array of node numbers `nodes` is allocated on line No. 1. The function `give_elemnodes` of the class `top` assembles appropriate node numbers to the array `nodes` (line 2). The function `give_thickness` of the class `crsec` assembles the thicknesses to the array `t`. The function `gauss_points(gp,w,nip)` assembles the coordinates of the integration points to the array `gp` and the weights to the array `w`. `ipp` denotes the number of the first integration point on the current element (line 8). There is a loop over the number of integration points `nip`. The function `geom_matrix(gm,x,y,gp,i,jac)` assembles the strain-displacement matrix, function `matstiff(d,ipp)` of the class `mechmat` assembles the stiffness matrix of the material and the function `bdbjac(sm,gm,d,gm,jac)` of the current element computes matrix product $\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}$ (line 12). The number of the integration point is incremented on line No. 13.

2.3.6 Data Transfer

The crucial part for coupled analyses is the data transfer among all modules. In the staggered algorithm, functions `trfel_mefel()` and `mefel_trfel()` transfer state variables from TRFEL to MEFEL and MEFEL to TRFEL, respectively. In the fully coupled algorithm, the code is completed by four functions `trfel_metr()`, `metr_trfel()`, `mefel_metr()`, and `metr_mefel()` transferring data between TRFEL and METR, and between MEFEL and METR. The current SIFEL version has three types of finite element meshes using the same nodes and elements numbering in all modules. Transport and mechanical parts can have polynomial approximation functions of different degrees, where linear and quadratic functions are the most used. While, the coupling - superior part adopts approximation functions from inferior MEFEL and TRFEL parts.

There are several possibilities to transfer state variables:

```
ivector nodes(nne);
Mt->give_elemnodes (eid,nodes);
vector t(nne);
Mc->give_thickness (eid,nodes,t);
vector w(nip);
vector gp(nip);
gauss_points (gp,w,nip);
ipp=Mt->elements[eid].ipp;
for (i=0;i<nip;i++){
    geom_matrix (gm,x,y,gp,i,jac);
    Mm->matstiff (d,ipp);
    bdbjac (sm,gm,d,gm,jac);
    ipp++;
}
```

Table 2.4: Function stiffness matrix

- by nodal values, e.g., in function `trfel_mefel_by_nodes()`, where quantities are copied to nodes from the closest integration points at the particular TRFEL elements and then passed to MEFEL elements which approximate them to the MEFEL integration points;
- by nodal values, e.g., in `trfel_mefel_by_nodes_comp()`, where nodal values are computed directly at particular TRFEL element nodes and then passed to MEFEL elements which approximate them to the MEFEL integration points;
- by integration points, e. g, in function `trfel_mefel_by_aip(Mm->tnip, MTipmap)` which computes/passes coupling data from TRFEL to MEFEL. Data are taken from the auxiliary integration points in TRFEL and stored in MEFEL to the `nonmechq` array for non-mechanical quantities;
- by integration points, e. g, in the function `trfel_mefel_copyip()` which transfers TRFEL quantities to MEFEL as nonmechanical quantities required in MEFEL. In this case, the meshes must be identical in both MEFEL and TRFEL modules, and individual values are copied between corresponding integration points;

The same strategy is also used among all parts - MEFEL, TRFEL, and METR.

The universal but the most challenging strategy of the data transfer for the future work, which uses benefits of the mesh adaptivity problems, is the solution of three independent finite element meshes transferring values via the global coordinate system and finite element approximation functions.

2.3.7 Extensibility

The code extensibility can be illustrated with the help of the conductivity matrix assembling for coupled problems with many variables. The matrix for heat and moisture transfer has the form in Equation (2.44), where three unknown functions are used in the model. These unknowns are temperature T, pore pressure p_1, and pore pressure p_2. Table 2.5 shows a part of the code which computes and assembles the conductivity matrix of one finite element. `ntm` denotes the number of unknown functions. In the case of matrix (2.44), `ntm`=3. The third row in Table 2.5 represents subroutine, which computes a submatrix defined by equation(2.40). The matrix is stored in `lkm`. Appropriate row and column indexes are obtained by the subroutine `codnum` (lines 4 and 5), and they are stored in `rcn` and `ccn`. The submatrix (2.40) is added to the conductivity matrix of a finite element, which is stored in `km`. Further, the element matrix is localized into the matrix of the system of algebraic equations. This subroutine shows that extensibility is ensured, and additional state variables lead to the increase of the variable `ntm`.

```
1   for (i=0;i<ntm;i++){
2       for (j=0;j<ntm;j++){
3            conductivity_matrix (i,eid,i,j,lkm);
4            codnum (rcn,i);
5            codnum (ccn,j);
6            mat_localize (km,lkm,rcn,ccn);
7       }
8   }
```

Table 2.5: Loop for assembling of the conductivity matrix

2.4 Computer implementation of hypoplastic model and test benchmarks

Mašín in [Mašín, 2017] developed the fully-coupled model for bentonite materials determined for a material point with suction and temperature as input parameters. The implementation and coupling of this model, together with Thermo-hygro-mechanical model based on Lewis and Schrefler's approach ([Schrefler and Lewis, 1998]), was then implemented as a staggered algorithm. The transport and mechanical parts run separately with data transfer. In this concept, the transport part runs first before the mechanical part. In case of only water flow in deforming medium, the system of equations (2.42) can be modified for the partially coupled approach for transport and mechanical parts separately with the mechanical system of equation rewritten in incremental form

- *Transport part*

$$\boldsymbol{K}_{ww}\boldsymbol{d}_w + \boldsymbol{C}_{ww}\dot{\boldsymbol{d}}_w = \boldsymbol{f}_w + \boldsymbol{f}_{wu}, \tag{2.47}$$

- *Mechanical part*

$$\boldsymbol{K}_{uu}\Delta\boldsymbol{d}_u = \Delta\boldsymbol{f}_u + \Delta\boldsymbol{f}_{uw}, \tag{2.48}$$

where

$$\boldsymbol{f}_{wu} = -\boldsymbol{C}_{wu}\boldsymbol{d}_u = -\int_\Omega \boldsymbol{N}_w^{\mathrm{T}}(\alpha S_w)\boldsymbol{m}^{\mathrm{T}}\boldsymbol{B}_u \mathrm{d}\Omega\boldsymbol{d}_u = -\int_\Omega \boldsymbol{N}^{\mathrm{T}}(\alpha S_w)\Delta\boldsymbol{\varepsilon}_V \mathrm{d}\Omega. \tag{2.49}$$

Vector $\Delta\boldsymbol{\varepsilon}_V$ contains nodal increments of volumetric strains computed from the previous time step. In the presented notation, the right-hand side vector $\Delta\boldsymbol{f}_{uw}$ expresses the forces caused by changes of pore water pressure computed only in the mechanical part from pore water pressure (or suction) increments taken from the transport part

$$\Delta\boldsymbol{f}_{uw} = -\boldsymbol{K}_{uw}\Delta\boldsymbol{d}_w = -\int_\Omega \boldsymbol{B}_u^{\mathrm{T}}\boldsymbol{m}^{\mathrm{T}}(\alpha S_w)\boldsymbol{N}_w \mathrm{d}\Omega\Delta\boldsymbol{d}_w. \tag{2.50}$$

Vector $\Delta\boldsymbol{d}_w$ is the vector of pore water pressure increments. In the hypoplastic model, the vector $\Delta\boldsymbol{f}_{uw}$ is computed from the total stress definition. The vector of total stress $\boldsymbol{\sigma}^{\mathrm{tot}}$, which is defined, e.g., in [Mašín, 2017] and [Schrefler and Lewis, 1998], can be expressed in the form of vector function

$$\boldsymbol{\sigma}^{\mathrm{tot}} = \boldsymbol{g}(\boldsymbol{\varepsilon}(\boldsymbol{u}), p^w). \tag{2.51}$$

The time derivative of the stress vector has the form

$$\dot{\boldsymbol{\sigma}}^{\mathrm{tot}} = \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{\varepsilon}}\dot{\boldsymbol{\varepsilon}} + \frac{\partial\boldsymbol{g}}{\partial p^w}\dot{p}^w = \boldsymbol{D}_u\dot{\boldsymbol{\varepsilon}} + \boldsymbol{h}\dot{p}^w. \tag{2.52}$$

The stiffness matrix $\boldsymbol{D}_u$ and vector $\boldsymbol{h}$ are derived in [Mašín, 2017]. The rate of the total stress has to satisfy the equilibrium equation in the form

$$\boldsymbol{\partial}^T\left(\boldsymbol{D}_u\dot{\boldsymbol{\varepsilon}} + \boldsymbol{h}\dot{p}^w\right) + \dot{\boldsymbol{b}} = \boldsymbol{0}. \tag{2.53}$$

Recall, $\dot{\boldsymbol{b}}$ is the time derivative of the body force vector. Additionally, the hypoplastic model involves state variables given by vector $\boldsymbol{p}$ that can also be formulated in the rate form and thus generally, the stress rate can be defined by

$$\dot{\boldsymbol{\tau}} = \boldsymbol{M}\dot{\boldsymbol{\epsilon}} = \boldsymbol{\Psi}\left(\boldsymbol{\tau}(t), \Delta\boldsymbol{\epsilon}(t)\right), \tag{2.54}$$

where $\boldsymbol{\tau}$ is the generalized stress vector $\boldsymbol{\tau} = \{\boldsymbol{\sigma}, \boldsymbol{p}\}^T$, $\boldsymbol{M}$ represents the generalized stiffness matrix and $\boldsymbol{\epsilon}$ is the generalized strain vector $\boldsymbol{\epsilon} = \{\boldsymbol{\varepsilon}, p^w\}^T$ and $\boldsymbol{\Psi}$ represents the model response function on the given input of strain increment $\Delta\boldsymbol{\epsilon}$ of the actual time step and attained stress level $\boldsymbol{\tau}$. The explicit integration RKF algorithm with substepping has been selected and implemented in SIFEL. (2.54) represents the initial value problem given by the set of ordinary differential equations. These equations can be written in generic substep k at time interval $[t_n; t_{n+1}]$ formally as follows

$$\boldsymbol{\tau}_{k+1} \;=\; \boldsymbol{\tau}_k + \Delta t_k \sum_{i=1}^{s} b_i\,\boldsymbol{k}_i\left(\boldsymbol{\tau}_k, \Delta\boldsymbol{\epsilon}(t_{n+1}), \Delta t_k\right), \tag{2.55}$$

where $\boldsymbol{k}_i\left(\boldsymbol{\tau}_k, \Delta\boldsymbol{\epsilon}(t_{n+1}), \Delta t_k\right)$ represents the function $\boldsymbol{\Psi}$ evaluated for the given strain increment of the actual time step $\Delta\boldsymbol{\epsilon}(t_{n+1}) = \boldsymbol{\epsilon}(t_{n+1}) - \boldsymbol{\epsilon}(t_n)$ and attained stress levels at the prescribed points of the time interval. In Equation (2.55), dimensionless step length $\Delta t_k \in (0; 1]$ has been introduced with the following definition

$$\Delta t_k = \frac{t_{k+1} - t_k}{t_{n+1} - t_n}. \tag{2.56}$$

A detailed description of the integration by Runge-Kutta-Fehlberg methods is presented in the reference [Koudelka et al., 2017].

2.4.1 Benchmark tests

The computer implementation of the hypoplastic model in connection with Lewis and Schrefler's approach was tested on several examples and benchmarks. Suitable benchmarks can be laboratory tests presented in [Hausmannová, 2017]. These studies focus on the impact of using high hydraulic gradients on combined measurements of hydraulic conductivity and swelling pressure. The hydraulic conditions are supposed to be consistent with possible water pressures in a deep repository. Both parameters are determined in a full saturation state. Measuring these parameters in such a low-permeable bentonite material requires much time. Therefore, the high hydraulic gradients may accelerate the determination of these parameters. Experiments with the Czech bentonite 75 (B75) from Černý vrch deposit were selected for numerical simulations. The material was uniaxially compacted in the laboratory to reach the required dry density $\rho_d = 1200$ to 1750 kg/m^3. The tested samples have a diameter of 30 mm, and a height of 20 mm. The initial values of hydraulic conductivity and swelling pressure were evaluated using a saturation pore water pressure $p^w = 1$ MPa corresponding to the gradient of grad$p^w = 50$ MPa/m (hydraulic gradient 5000) [Hausmannová and Vašíček, 2014]. A unique device was used to measure the hydraulic conductivity and the swelling pressure (Figure 2.1). The setup of this device is described in detail in the reference [Hausmannová and Vašíček, 2014].

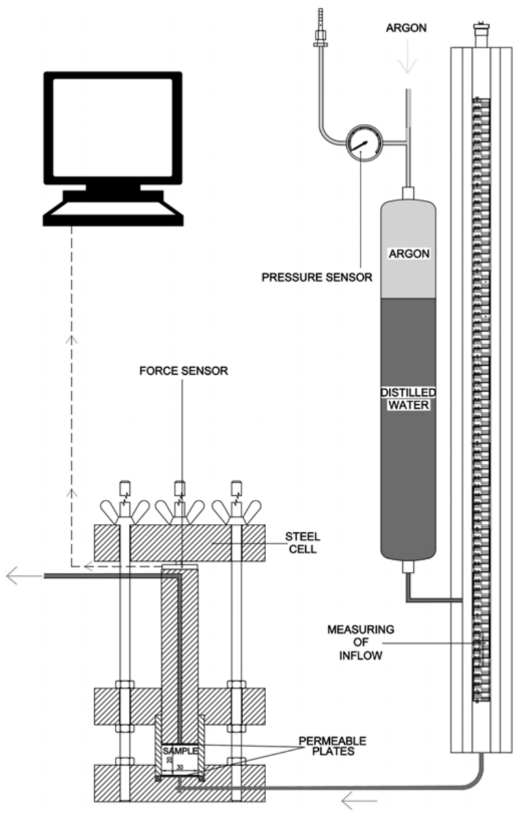

Figure 2.1: Scheme of the measuring device [Hausmannová and Vašíček, 2014].

The finite element mesh consists of 20 axisymmetric quadrilateral elements in the vertical direction. Linear approximation functions are used in the transport part and quadratic in the mechanical part. The watering process was modeled as a prescribed pore water pressure from the bottom with the values taken from the measurements. Two switching boundary conditions model the top permeable surface. For the first, the water flux is prescribed zero on the boundary until the water head reaches the closest material point, equal to zero water pressure. Then, the conditions are changed to the Dirichlet boundary condition with prescribed zero water pressure. This procedure is commonly used for free soil surface modeling. The initial pore water pressure p_0^w = -100 MPa is set for all benchmarks. The soil parameters used in the simulations are used from the recent calibration for bentonite B75 [Sun et al., 2021]. The sample is fixed to avoid its swelling, and no friction between bentonite material and the steel structure of the testing device is neglected.

Three tests with dry density ρ_d=1298 kg/m^3, ρ_d = 1498 kg/m^3, and ρ_d = 1743 kg/m^3 were used for verification and validation of coupling of mentioned material models in SIFEL computer code and setup of their parameters. A comparison of selected results for different configurations of dry density and hydraulic conductivity is presented. Figures 2.2 and 2.3 show the history of swelling stress for bentonite samples of dry densities ρ_d = 1498 kg/m^3 and ρ_d = 1743 kg/m^3, respectively. From the considerable amount of computations, the best results closed to the measurements are obtained by using of Bogacki-Shampine integration scheme [Koudelka et al., 2017] for the hypoplastic model in connection with the smoothed water retention curve [Sun et al., 2021] and for maximum time step t$_{max}$=1000 s [Scaringi et al., 2022]. It has to be mentioned that such numerical simulations are strongly non-linear, time step length-dependent, and time-consuming. Most of them took from 10 to 20 hours, despite the use of multithreading architecture via OpenMP system.

Attained levels of swelling pressure at full saturation depend only on the setup of initial dry densities. This fact corresponds to the previous experiments and hypoplastic model calibration. The swelling stress for bentonite with ρ_d = 1498 kg/m^3 is about 3 MPa, and for ρ_d = 1743 kg/m^3 is 10.5 MPa, respectively. The initial swelling pressures growth is influenced by the sample saturation rate, related to intrinsic permeability (or hydraulic conductivity). The permeability was assumed constant for all benchmarks. For better compliance with the measurements in the initial phase, the application of a relationship dependent on saturation degree can be successfully used. The coincidence between simulations and measurements is validated as relatively good. The trends of watering with loading water pressure jumps are captured well.

From the analysis of the results, it can be concluded that coupling the hypoplastic model in connection with Lewis and Schrefler's approach in a staggered scheme works well. However, the model response is primarily dependent on the hypoplastic model setup.

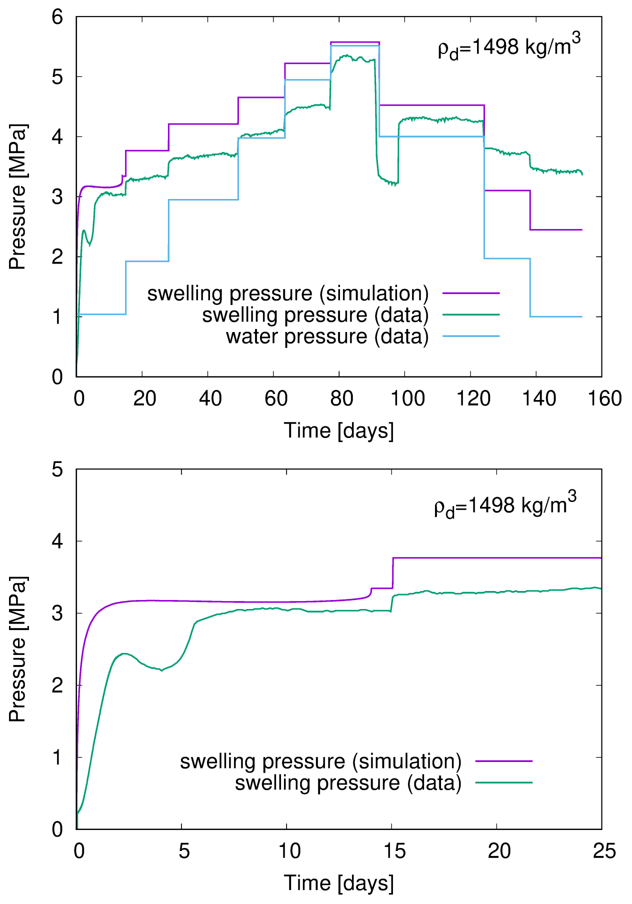

Figure 2.2: History of water pressure and swelling pressure for bentonite B75 $\rho_d = 1498$ kg/m^3 and $K^w = 2.0{\cdot}10^{-13}$ m/s [Scaringi et al., 2022] (left), and zoom of the initial phase (right).

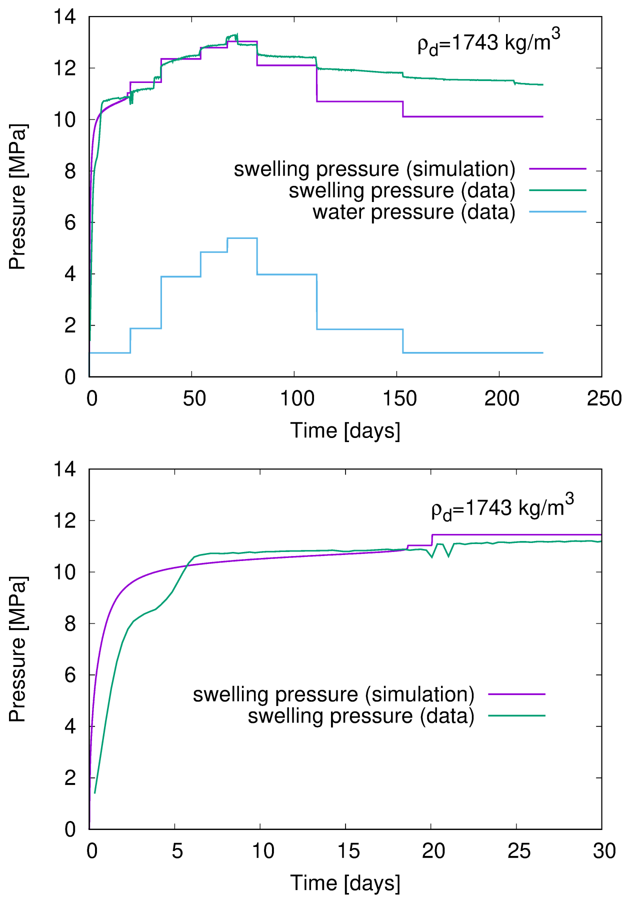

Figure 2.3: History of water pressure and swelling pressure for bentonite B75 $\rho_d = 1743$ kg/m^3 and $K^w = 1.0{\cdot}10^{-13}$ m/s [Scaringi et al., 2022], and zoom of the initial phase (right).

Chapter 3

Runing of the program and the structure of input files

3.1 Compilation and program running

Download source files of the actual version from the link

https://mech.fsv.cvut.cz/∼sifel/TACR/TK01010063/download.html

The actual version of **THM Model**, e.g., `sifel_thm_2021.zip`, then unpack in an arbitrary directory. In the directory `METR`, build binary files by

```
make
```

or with

```
make opt
```

for optimized binary files. Another option for optimized binary files and powerfull computing with threading:

```
make ompopt
```

The binary file is located in the directory `../BIN/METR/SRC/_DBG`. In case of compilation with optimization, the binary file is located in `../BIN/METR/SRC/_OPT`, and alternatively in `../BIN/METR/SRC/_OMPOPT`.

To run the computation download three packed input files and unpack into the same directory. Copy the binary file into this directory.

```
./metr bentonite_experiment_metr1_1298.in
```

runs the computer simulation for Mockup test of bentonite material with initial dry density 1298 kg/m^3.

To run the computer simulation for Mockup test of bentonite material with initial dry density 1498 kg/m^3 type

`./metr bentonite_experiment_metr1_1498.in`.

Each THM problem comprizes three files for mechanical, transport part, and coupled part. The name of the inpute file for coupled part has to be an argument when runnig the program.

3.2 Input files structure

Three input files must be created for each THM problem. It comprises one file for a mechanical problem, one for a transport problem, and a file for coupled problem, which connects them together in a staggerred procedure. The input files for mechanical (MEFEL) and trasnport parts (TRFEL) have the same structure written in a sequence:

- Problem description
- Topology
- Material models
- Boundary and initial conditions
- Outdriver

The input file for coupled part (METR) has more simple structure:

- Problem description
- Outdriver

3.2.1 METR input file

```
#
# run with ./metr bentonit_2elems_metr1_1743.in
#

## Problem description

one-phase flow in deforming porous medium for Mockup test #this is comment
10 #problem type = stagerred coupled_mech_trans
# input file names for MEFEL and TRFEL
bentonit_2elems_mefel1_1743.in -kwd=1
bentonit_2elems_trfel1_1743.in
1 #detail message priting = yes
2 #Ladyzhenskaja-Babuska-Brezzi = quadratic linear mesh
3 #type of the passing data between modules 3=data are calculated in auxiliary int.  points
1 #transported matter = mech_one-medium
```

```
3 #names of transported media = mech_water

## Time controler

1 #adaptive time increment length
0.0 #initial time
700000.0 #end time
0 #no important times
0 #constant time step
10.0 #initial time step lenght
1.0e-5 #minimal time step
1000.0 #maximum time step

## Data about solver

2 #Newton-Raphson method = newtonc
400 1.0e-6 #number of inner iterations #error
0 #clean matrices=no

## Output file name

bentonit_2elems_metr1_1743.out
```

3.2.2 MEFEL input file

```
#
# run with ./mefel bentonit_2elems_mefel1_1743.in -kwd=1
#

#
## Problem description is written with "key words"
#

Simple axisymmetric specimen on triaxial test with changed suction pressure on hypoplastic
model, coupled with one-phase flow in deforming porous medium for Mockup test #this is comment
mespr 1
problemtype mech_timedependent_prob

straincomp 1
strainpos 2
strainaver 1

stresscomp 1
stresspos 2
```

```
stressaver 1

othercomp 1
otherpos 2
otheraver 0

reactcomp 1

adaptivity 0
stochasticcalc 0
homogenization 0
noderenumber 0

time_contr_type adaptive
start_time 0.0
end_time 700000.0
num_imp_times 0
funct_type stat
const_val 10.0

dtmin 1.0
dtmax 1000.0

timetypeprin seconds

hdbackup nohdb


#solver data
nr_num_iter 300
nr_error 1.0e-10
resid_norm_type rel_react_norm
check_div off

stiffmatstor dense_matrix
stiffmat_type secant_stiff
typelinsol gauss_elim


#
## Topology
#

## Definition of nodes

13 #number of nodes
```

```
# node_id, x, y, z, num_dof, cross_sec_type=nocrsec, loc_coord_sys
1 0.00e-03 0.00e+00 0.00e+00 2 0 0
2 1.00e-03 0.00e-02 0.00e+00 2 0 0
3 0.00e-03 1.00e-02 0.00e+00 2 0 0
4 1.00e-03 1.00e-02 0.00e+00 2 0 0
5 0.00e-03 2.00e-02 0.00e+00 2 0 0
6 1.00e-03 2.00e-02 0.00e+00 2 0 0

7 5.00e-04 0.0e-02 0.00e+00 2 0 0
8 1.00e-03 5.0e-03 0.00e+00 2 0 0
9 5.00e-04 1.0e-02 0.00e+00 2 0 0
10 0.00e-03 5.0e-03 0.00e+00 2 0 0

11 1.00e-03 1.5e-02 0.00e+00 2 0 0
12 5.00e-04 2.0e-02 0.00e+00 2 0 0
13 0.00e-03 1.5e-02 0.00e+00 2 0 0


## Definition of boundary conditons at nodes = constraints


13 #number of boundary conditions
# nod_id cond_dof1 cond_dof2

1 0 0
2 0 0
3 0 1
4 0 1
5 0 0
6 0 0

7 1 0
8 0 1
9 1 1
10 0 1
11 0 1
12 1 0
13 0 1


## Definition of elements


2 #number of elements
```

```
# elem_id, elem_type=axisymqq=64, node_1, node_2, node_3, elem_code_num_flag, cross_sec_type=nocrsec,
num_mat_types, mat_type1=hypoplastusattherma_matt=421, mat_id1
1 64 1 2 4 3 7 8 9 10 0 0 1 421 1
2 64 3 4 6 5 9 11 12 13 0 0 1 421 1


#
## Definition of materials = material models
#

1 #number of different material types
# hypoplastusattherma_mat num_param_sets
421 1
# param_set_id,
1

# new parameters:
# phi, lam_star, kap_star, n_star,
25.0 0.13 0.06 1.73
# nu , ns , ls , nt , lt , m ,
0.25 0.012 -0.005 -0.07 0.0 1
# alpha_s , kappa_m, sm_star , em_star , csh , se_ref , em_ref,
0.00015 0.07 -2000.0 0.45 0.002 -2700.0 0.50
# tref , at , bt , aer , lambdap0 , p_t
294.0 0.118 -0.000154 1.0 0.7 1.0

## Aditional informations

# prescribed suction function = taken from transport part
0 #no
# prescribed temperature
1 # yes
0 # constant function
294.0

# stress integration algorithm
11 # Runge-Kutta-Fehlsberg
# RKF_type ni, err, h_min,
# RKF_type = rkf23bst
1 10000 1.0e-5 1.0e-17


## Definition of cross-section

0 # number of cross_sec_types
```

```
#
## Boundary conditions = loads
#

1 # type of load = time independent load in subloadcases
1 # number of time dependent load cases
1 # number of subloadcases


# subloadcase 1 - load will be induced by the change of suction

0 # number of loaded nodes
0 # number of loaded elements
0 # number of prescribed displacements
0 # temperature changes will not be assumed
# time function defined by constant value
0 1.0

## Definition of initial conditons at nodes |

13 # number of initial conditions at nodes
# node_id, initial_cond_type=inicond, num_of_init_values, e_0, ascan_0
1 16 2 0.64659 0.0
2 16 2 0.64659 0.0
3 16 2 0.64659 0.0
4 16 2 0.64659 0.0
5 16 2 0.64659 0.0
6 16 2 0.64659 0.0
7 16 2 0.64659 0.0
8 16 2 0.64659 0.0
9 16 2 0.64659 0.0
10 16 2 0.64659 0.0
11 16 2 0.64659 0.0
12 16 2 0.64659 0.0
13 16 2 0.64659 0.0


4 # eigenstresses will be assumed
# folowing lines defines hydrostatic pressure stress state -10.0 kPa
20 # axisymmetric strain/stress state
1 1 1 2 # indeces of general functions for each eigenstress component
2 # total number of eigenstress general functions used
0 -10.0 # the first function is constant
0 0.0 # the second function is constant
# the above setup results in assignment of stress vector sig=sig_x, sig_z, sig_r, tau_xz
= -10.0, -10.0, -10.0, 0.0 to all integration points
```

```
#
## Outdriver
#

# No text output
0

# Graphical output into GiD postprocessor files
4
# file name:
bentonit_2elems_mefel1_1743
# printing for nodes
4 1 1 # 4=periodic selection of time step, 1= print each 1st step, 1= print all load cases
1 1
0
0
0
1 1

# printing for elements
4 1 1 # 4=periodic selection of time step, 1= print each 1-st step, 1= print all load cases
1 1 0 # 1= print strains on all elements, 1= print all strain components
1 1 0 # 1= print stresses on all elements, 1= print all stress components
1 3 13 # 1 = print eqother on all elements, 3= type of selection of eqother components is
list, 12= number of list components
# id of required eqother componentsx13
12 13 14 15 16 17 18 19 20 21 22 23 24




# table text output
# number of tables for graphs
1
# file name:
bentonit_2elems_mefel1_1743.dat
# 1.  stress
16 4 1
#2=on an element, No.  of element, No.  of int.  point, ...
2 2 1 2 1 # eps_x 1
2 2 1 2 2 # eps_y 2
2 2 1 2 3 # eps_r 3
2 2 1 2 4 # gamma_xy 4
2 2 1 3 1 # sig_x 5
2 2 1 3 2 # sig_y 6
```

```
2 2 1 3 3 # sig_r 7
2 2 1 3 4 # tau_xy 8
2 2 1 8 12 # porosity e 9
2 2 1 8 13 # suction 10
2 2 1 8 14 # degree of saturation Sr 11
2 2 1 8 23 # epsv 12
2 2 1 8 24 # e_ax 13
2 2 1 8 25 # q_ax 14
2 2 1 8 21 # dtsub 15
1 2 9 # time 16
```

3.2.3 TRFEL input file

```
#
# run with ./trfel bentonit_2elems_trfel1_1743.in
#

#
## Problem description
#

Simple axisymmetric specimen on triaxial test with changed suction pressure on hypoplastic
model, coupled with one-phase flow in deforming porous medium for Mockup test #this is comment
1 #meprt
61 #nonlinear
1 2 1.0 #??
0 0 0 0 #no computation of gradients,fluxes,other,eqother
0 #no gravity
0 #adaptivity
0 #stochastic
0 #homogenization
0 #renumbering
0 #advection contribution
0 #no reaction


## Time controler

1 #adaptive
0.0 #initial time
700000 #70000.0 #57800.0 #1.0e6 #19129730.0 #end time
0 #no important times
0 #constant time step
```

```
10.0 #initial time step lenght


1.0e-5 #minimal time step
1000.0 #maximum time step

1 #type of time printing seconds

0 #no backup


#solver data

0.5 200
1.0e-10 1.0e-10

1 2 0

#information for solver storage type of conductivity and capacity matrix 3 = Dskyline, solver
type LU
3 3 3
#diagonalization of capacity matrix 1 = yes
1




#
## Topology
#


## Definition of nodes

# node_id, x, y, z, num_dof, cross_sec_type=nocrsec
13 #number of nodes
1 0.00e-03 0.00e+00 0.00e+00 1 0
2 1.00e-03 0.00e-02 0.00e+00 1 0
3 0.00e-03 1.00e-02 0.00e+00 1 0
4 1.00e-03 1.00e-02 0.00e+00 1 0
5 0.00e-03 2.00e-02 0.00e+00 1 0
6 1.00e-03 2.00e-02 0.00e+00 1 0

7 5.00e-04 0.0e-02 0.00e+00 1 0
8 1.00e-03 5.0e-03 0.00e+00 1 0
9 5.00e-04 1.0e-02 0.00e+00 1 0
10 0.00e-03 5.0e-03 0.00e+00 1 0
```

```
11 1.00e-03 1.5e-02 0.00e+00 1 0
12 5.00e-04 2.0e-02 0.00e+00 1 0
13 0.00e-03 1.5e-02 0.00e+00 1 0


## Definition of boundary conditons at nodes

13 #number of boundary conditions
# nod_id cond_dof1

1 -1
2 -1
3 1
4 1
5 1
6 1
7 0
8 0
9 0
10 0
11 0
12 0
13 0

## Definition of elements

2 #number of elements

# elem_id, elem_type=axisymlq=217, node_1, node_2, node_3, 0, cross_sec_type=2d cross_sec_no.=1,
4x (mat_type1 and mat_id1)

1 217 1 2 4 3 0 2 1 601 1
2 217 3 4 6 5 0 2 1 601 1

## Definition of materials = material models

1 #number of different material types
601 1
1 # material number
7 # model_type = lewis_schrefler for pc solver
1 # no compressibility
1.0 # alpha
2.0e11 # ks
2780.0 # rhos0
-10.0 # pw_bc
```

```
8.0e-21 # kintr0
1 # porosity type=from mefel
0 # kintr_type = constant
0 # krw_type=constant
7 # sr_type=mefel
1000.0 # mefel_units
0 # vol_strain_effect

## Definition of cross-section

1
2 1
1 1.0 1.0

#
## Boundary conditions = loads
#

1 #number of loadcases for two unknowns

# part for water pressure:

1 #number of dirichlet's b.c.
# 1st b.c.  for water pressure

# watering process from the bottom:

-100000.0e3 #initial value

2 #tablefunct
1 #piecewise linear
18 #number of table rows
0.0 -100000.0e3
100.0 -100000.0e3
7200.0 930.1e3
1732800.0 930.1e3
#7200.0 0.0
#1732800.0 0.0
1734600.0 1883.5e3
3035300.0 1883.5e3
3037100.0 3893.7e3
4695500.0 3893.7e3
4697300.0 4844.5e3
5819910.0 4844.5e3
5821710.0 5386.5e3
7073310.0 5386.5e3
```

```
7075110.0 3978.4e3
9589710.0 3978.4e3
9591510.0 1845.4e3
13219710.0 1845.4e3
13221510.0 935.2e3
19129730.0 935.2e3

0 #number of sources
1 #number of loaded elements
2 0 0 40 1 2 3 0

3 #number of nodal values for pore water pressure

#1#values for pore water pressure - ambient
2 0 0.0 0 0.0

#2#transfer coefficient for the top surface
2 0 1.0 0 1.0

#3#zero additional coefficients
2 0 0.0 0 0.0

0#number of climatic b.c.
0#nymber of climatic b.c.  2
0#number of time functions

# no boundary fluxes computation:
0


## Definition of initial conditons at nodes

# pore water pressure pw = -100MPa

1 -100000.0e3
2 -100000.0e3
3 -100000.0e3
4 -100000.0e3
5 -100000.0e3
6 -100000.0e3
7 -100000.0e3
8 -100000.0e3
9 -100000.0e3
10 -100000.0e3
11 -100000.0e3
12 -100000.0e3
```

```
13 -100000.0e3


#
## Outdriver
#

#No text output
0

# Graphical output into GiD one postprocessor file
3
# file name:
bentonit_2elems_trfel1_1743.gid
#4 1000 #printing at each 1000-th time step
4 1 #printing at each time step
1 1
1 1
1 1
1 1
1 1

0
0




4 #four graphs - output for xmgrace
bentonit_2elems_trfel1_1743.dat
7 #number of printing components
1 #printing at each time step
1 2 7 #time
#printing at node, node number, uknown printing, number of unknown component 1=pore water
pressure
1 1 1 1
1 2 1 1
1 3 1 1
1 4 1 1
1 5 1 1
1 6 1 1

7 #number of printing components
1 #printing at each time step
1 2 7 #time
#printing at node, node number, other printing, number of unknown component 1=pore water
```

```
pressure in Pa
1 1 8 1
1 2 8 1
1 3 8 1
1 4 8 1
1 5 8 1
1 6 8 1


7 #number of printing components
1 #printing at each time step
1 2 7 #time
#printing at node, node number, other printing,
1 1 8 3
1 2 8 3
1 3 8 3
1 4 8 3
1 5 8 3
1 6 8 3


2 #number of printing components
1 #printing at each time step
1 2 7 #time
#printing at node, node number, uknown printing, number of unknown component 1=pore water
pressure
1 4 1 1
```

Bibliography

[Bittnar and Šejnoha, 1996] Bittnar, Z. and Šejnoha, J. (1996). *Numerical Methods in Structural Mechanics.* USA. ASCE Press, New York.

[Crisfield, 1991] Crisfield, M. A. (1991). *Non-linear Finite Element Analysis of Solids and Structures.* UK. John Wiley & Sons Ltd, Chichester.

[Hausmannová, 2017] Hausmannová, L. (2017). *The influence of water pressure on the hydraulic conductivity and swelling pressure of Czech bentonites.* PhD thesis, Czech Technical University in Prague. In Czech.

[Hausmannová and Vašíček, 2014] Hausmannová, L. and Vašíček, R. (2014). Measuring hydraulic conductivity and swelling pressure under high hydraulic gradients. *Geological Society London Special Publications*, 400.

[Hughes, 1987] Hughes, T. J. R. (1987). *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis.* Prentice-Hall, inc. Englewood Cliffs, New Jersey.

[Koudelka et al., 2010] Koudelka, T., Krejčí, T., and Kruis, J. (2010). Moderate use of object oriented programming for scientific computing. In Topping, B. H. V., Adam, J. M., Pallarés, F. J., Bru, R., and Romero, M. L., editors, *Proceedings of the Seventh International Conference on Engineering Computational Technology*, Stirlingshire, United Kingdom. Civil-Comp Press. paper 68.

[Koudelka et al., 2011] Koudelka, T., Krejčí, T., and Kruis, J. (2011). *Modeling of Building constructions in SIFEL Environment.* CTU Reports. Czech Technical University in Prague, Czech Republic.

[Koudelka et al., 2017] Koudelka, T., Krejčí, T., and Kruis, J. (2017). Coupled hydro-mechanical model for expansive clays. *AIP Conference Proceedings*, 1863(1):290008.

[Kruis et al., 2021] Kruis, J., Koudelka, T., and Krejčí, T. (2001–2021). SIFEL package. *http://ksm.fsv.cvut.cz/∼sifel/.*

[Kruis et al., 2010] Kruis, J., Koudelka, T., and Krejčí, T. (2010). Efficient computer implementation of coupled hydro-thermo-mechanical analysis. *Mathematics and Computers in Simulation*, 80:1578–1588. DOI information: 10.1016/j.matcom.2008.11.010.

[Mašín, 2013] Mašín, D. (2013). Double structure hydromechanical coupling formalism and a model for unsaturated expansive clays. *Engineering Geology*, 165:73–88.

[Mašín, 2017] Mašín, D. (2017). Coupled thermohydromechanical double-structure model for expansive soils. *Journal of Engineering Mechanics*, 143(9).

[Mašín and Khalili, 2016] Mašín, D. and Khalili, N. (2016). Swelling phenomena and effective stress in compacted expansive clays. *Canadian Geotechnical Journal*, 53(1):134–147.

[Scaringi et al., 2022] Scaringi, G., Mašín, D., Najser, J., Sun, H., and Sun, Z. (2022). Thermo-hydro-mechanical hypoplastic modelling of bentonite buffers for nuclear waste disposal: model calibration and performance. In *In preparation for Proceedings of the 20th International Conference on Soil Mechanics and Geotechnical Engineering, Sydney 2022*.

[Schrefler and Lewis, 1998] Schrefler, B. A. and Lewis, R. W. (1998). *The Finite Element Method in the Static and Dynamic Deformation and Consolidation of Porous Media.* 2nd Edition. John Wiley & Sons.

[Sun et al., 2021] Sun, H., Scaringi, G., Mašín, D., and Najser, J. (2021). An experimental investigation on the swelling behavior of compacted b75 bentonite. *Engineering Geology*, page 106452.