

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering

Alternative solution of transport problem in the TRM2D software

Alternativní řešení transportní úlohy v softwaru TRM2D

Bachelor's Degree Project

Author: **Evgeniy Kleschenko**
Supervisor: **doc. Ing. Jan Šembera, Ph.D.**
Language advisor: **PaedDr. Eliška Rafajová**
Academic year: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Evgeniy Kleschenko
Studijní program:	Aplikace přírodních věd
Studijní obor:	Aplikovaná informatika
Název práce (česky):	Alternativní řešení transportní úlohy v softwaru TRM2D
Název práce (anglicky):	Alternative solution of transport problem in the TRM2D software

Pokyny pro vypracování:

- 1) Prostudujte vybranou literaturu o matematickém modelování v oblasti transportních úloh, zejména metodě konečných diferencí (MKD) a metodě konečných objemů (MKO).
- 2) Převezměte software TRM2D vzniklý v rámci projektu TH02030840 „Paralelizovaný reakčně-transportní model šíření kontaminace v podzemních vodách (PaReTran)“ a popište metodu řešení transportní části úlohy použitou v tomto softwaru.
- 3) Na základě znalostí MKO a MKD navrhnete a implementujete alternativní metodu řešení transportní části úlohy do softwaru TRM2D.
- 4) Proveďte základní testování nově implementované metody.

Doporučená literatura:

- 1) H. P. Langtangen, S. Linge, Finite Difference Computing with PDEs. Released under CC Attribution 4.0 license 2016.
- 2) M. Hokr, Transportní procesy [in Czech]. Lecture notes, Faculty of Mechatronics, Technical University of Liberec, 2005.
- 3) P. Štrof, et al., Final Report of the TACR project Nr. TH02030840 [in Czech]. DHI Praha, 2019.

Jméno a pracoviště vedoucího bakalářské práce:

doc. Ing. Jan Šembera, Ph.D.

Technická univerzita v Liberci, Studentská 1402/2, 461 17 Liberec 1

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2021

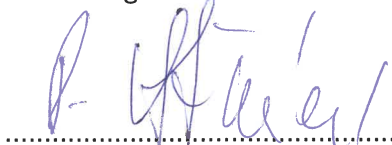
Datum odevzdání bakalářské práce: 7.7.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 21. října 2021

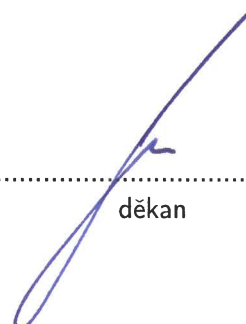


garant oboru



vedoucí katedry





děkan

Acknowledgment:

I wish to express my sincere thanks to my supervisor doc. Ing. Jan Šembera, Ph.D. for his invaluable expert guidance and show my gratitude to PaedDr. Eliška Rafajová for her language assistance. I would also like to thank my family for the unceasing support and attention.

Author's declaration:

I declare that this Bachelor's Degree Project is entirely my own work and I have listed all the used sources in the bibliography.

Prague, August 2, 2023

Evgeniy Kleschenko

Název práce:

Alternativní řešení transportní úlohy v softwaru TRM2D

Autor: Evgeniy Kleschenko

Obor: Aplikovaná informatika

Druh práce: Bakalářská práce

Vedoucí práce: doc. Ing. Jan Šembera, Ph.D., Fakulta mechatroniky, informatiky a mezioborových studií (FM), Technická univerzita v Liberci (TUL)

Abstrakt: Hlavním cílem této bakalářské práce je implementace alternativního řešení transportní úlohy v softwaru TRM2D. Za tímto účelem byly studovány matematicko-fyzikální základy transportně-reakční úlohy, metoda štěpení operátoru, a numerické metody, jako je metoda konečných diferencí a metoda konečných objemů. Implementace alternativního řešení zahrnovala studii softwaru, jeho struktury, a metod výpočtu. Po implementaci bylo nové řešení otestováno porovnáním výsledků výpočtu s analytickým řešením.

Klíčová slova: metoda konečných diferencí, metoda konečných objemů, transportní úloha

Title:

Alternative solution of transport problem in the TRM2D software

Author: Evgeniy Kleschenko

Abstract: The main goal of this bachelor project is to implement the alternative solution of the transport problem in the TRM2D software. For this purpose, the mathematical-physical basis of the transport-reactive problem, the Operator splitting method, and numerical methods such as the Finite difference method and the Finite volume method were studied. Implementing the alternative solution included the study of the software, its structure, and calculation methods. After implementation, the new solution was tested by comparing the results of the calculation with the analytical solution.

Key words: finite difference method, finite volume method, transport problem

Contents

Introduction	8
1 Transport processes in a porous medium	9
1.1 Continuous model of a porous medium	9
1.2 Fluid flow in a porous medium	9
1.2.1 Darcy's law	9
1.2.2 Balance of quantity equation	11
1.3 Transport of solutes	11
1.3.1 Advection and diffusion-dispersion processes	11
1.3.2 Advection-diffusion equation	12
1.3.3 Initial and boundary conditions	13
2 Numerical model	14
2.1 Operator Splitting Method	14
2.2 Finite Difference Method	15
2.2.1 Basic characteristics of FDM	15
2.2.2 Application of FDM to transport problem	16
2.3 Finite Volume Method	17
2.3.1 Basic characteristics of FVM	17
2.3.2 Application of the FVM to the transport problem	17
2.4 Solution of the resulting system of ordinary differential equations of the first order	19
3 TRM2D software	20
3.1 Basic description of TRM2D software	20
3.2 Description of configuration file	20
3.3 Description of program execution process	21
3.3.1 Loading of input data	21
3.3.2 Activation of transport module	21
3.3.3 Calculation of transport	22
4 Alternative solution of transport problem in TRM2D software	24
4.1 Implementation of the alternative solution	24
4.1.1 Modification of the configuration file and reading methods	25
4.1.2 Preparation of matrices for calculation	25
4.1.3 Modification of calculation method	26
4.2 Testing of implemented solution	27
4.2.1 Description of analytical solution	27

4.2.2	Formulation of testing problem	27
4.2.3	Comparison of analytical and numerical solutions	28
Conclusion		31
Appendices		33
A	Original implementation of the transport_module::calculate() method	34
B	Original version of configuration XML file	36

Introduction

The flow of the components dissolved in the groundwater is a complex process that includes a lot of factors. To describe it, the transport-reactive problem is used. This problem describes the effect of different transport and reaction processes on the concentrations of dissolved components.

The main goal of this project is to implement the alternative solution of the transport problem in the TRM2D software. This means that this project focuses on the transport part of the transport-reaction problem. To solve it separately from the reaction part, it is necessary to study the Operator splitting method that divides the transport-reactive problem into two parts. This method allows to apply the appropriate numerical methods separately to each part.

The transport part is presented by the system of differential equations for each transported component separately. To find a solution, various numerical methods can be used, such as the Finite difference method and the Finite volume method. Each method has its own advantages and disadvantages. In this project we will apply both methods to the transport part of the transport-reactive problem and compare it with the solution implemented in the TRM2D software. On the basis of the theoretical study, we will suggest our own version of the solution and implement it in the TRM2D software. The implementation of the new solution assumes the preliminary study of the structure and functionality of the software, which is necessary for the successful implementation of the new solution.

The study is divided into four main chapters: mathematical-physical description of transport processes in a porous medium, the numerical solution of the transport part of the advection-diffusion equation using different numerical methods, description of the TRM2D software, and modification of the software with subsequent testing.

Chapter 1

Transport processes in a porous medium

The first chapter is based on the Milan Hokr's textbook [1], which describes transport processes. It contains the basic mathematical-physical description of transport processes in a porous medium. In this chapter, we will introduce equations describing the flow of the solution and the transport of dissolved substances through the advection and hydrodynamic dispersion processes. These equations play an essential role in solving the transport-reaction problem.

1.1 Continuous model of a porous medium

The porous medium may be defined as a structure consisting of solid material grains or fibers (matrix) and the empty pores between them which can be filled with air or water. This project focuses on a solution-saturated porous medium. Due to the microscopic nature of pores we must introduce the continuous model of the porous medium. This model is based on considering the whole medium as a continuous material with a porosity coefficient and defining all values per volume. This approach allows us to simplify the description of transport processes in a porous medium. To use a continuous model, we need to propose the terms representative elementary volume (REV) and porosity ratio.

The term REV denotes the minimum volume required to overcome microscopic inhomogeneity. This volume must be much larger than the size of a single pore but sufficiently smaller than the entire volume of the medium. A porosity ratio is the name for a coefficient n that denotes the ratio between the volume of pores in REV and the entire REV

$$n = \frac{\text{volume of pores in REV}}{\text{REV volume}} \quad (1.1)$$

at a certain point of the medium.

1.2 Fluid flow in a porous medium

The fluid flow in the saturated porous medium is defined by two differential equations: the equation of motion and the balance of quantity equation. In our case, the equation of motion is called Darcy's law.

1.2.1 Darcy's law

Darcy's law defines the relationship between the movement of the fluid and the action force. It was derived from an experiment measuring the flow of water through an inclined tube that connects two

tanks placed at different heights. This tube is filled with a porous filter and can be interpreted as a one-dimensional porous medium. The experiment introduces two values that are essential for describing fluid flow through porous material: hydraulic head and Darcy's velocity.

The water flow Q represents the volume of water flowing through a tube per unit of time and can be calculated as

$$Q = K \cdot \frac{S \cdot (\phi_1 - \phi_2)}{L}, \quad (1.2)$$

where K is defined as the coefficient of permeability (it depends on the characteristics of the porous material and the fluid), S is a cross-sectional area of the tube, L is the tube length, and the term in brackets is the difference between the heights of the tanks. The symbol ϕ refers to the hydraulic head, the value assigned to the ends of the tube. The term hydraulic head is also known as piezometric height because it can be represented as the height of the water column pushed up by pressure at a certain point of the medium. It can be calculated as

$$\phi = z + \frac{p}{\rho g}, \quad (1.3)$$

where z is the vertical coordinate, p is pressure, ρ is density, and g is gravitational acceleration. This equation includes the potential of the gravitational field (z -position) and the potential of pressure (hydrostatic pressure of water above the given point). The term $\frac{p}{\rho g}$, which denotes the pressure expressed in units of length (as a height of a water column), is called the pressure head.

Then we will look at the flow of water more closely by introducing its local description. The term water flux q may be defined as the ratio of the volume of flowing water and the size of the area perpendicular to the direction of the flow $q = \frac{Q}{S}$. Furthermore, considering the limit in the longitudinal direction as $-\Delta\phi = \lim_{L \rightarrow 0} \frac{\phi_1 - \phi_2}{L}$, allows us to derive Darcy's law in the form

$$q = K\Delta\phi. \quad (1.4)$$

The value q is called Darcy's velocity because it is expressed in units of velocity. However, this value does not describe the macroscopic movement of any point in the water flow, but rather the amount of water flowing through any cross-sectional area of the medium.

To determine the velocity of a selected water particle or any given solute, the same model as in Darcy's experiment is used. Let us suppose a tube segment with the length L , then its volume is $V = SL$. The volume of water flowing through this segment is $V_w = n \cdot SL$, where n is the porosity coefficient. Considering the water flux q , the period of flow of this volume V_w through the cross-sectional area is

$$t = \frac{V_w}{Sq} = \frac{nL}{q}. \quad (1.5)$$

Therefore, the velocity of the flow through the tube segment of length L is equal to

$$v = \frac{q}{n}, \quad (1.6)$$

and is called the average pore water velocity.

Considering multidimensional cases, the hydraulic head and Darcy's velocity vector are functions of spatial coordinates and time, related by Darcy's law in the form

$$q = -K\Delta\phi, \quad (1.7)$$

where K is a tensor that defines hydraulic conductivity depending on the properties of the porous medium and the fluid itself.

1.2.2 Balance of quantity equation

In any selected volume, the change in water mass must be equal to the change in water mass passed across the boundary and to the change in sinks and sources, i.e.

$$\frac{\partial}{\partial t} \int_V \rho n dV = - \int_{\partial V} \rho \mathbf{q} \cdot d\mathbf{S} + \int_V P \rho dV, \quad (1.8)$$

where P is density of sinks (-) or sources (+) defined as a volume of liquid pushed into a unit volume of the porous material per unit of time. This equation can be rewritten using Gauss's theorem, after rearrangement

$$\frac{\partial(\rho n)}{\partial t} + \text{div}(\rho \mathbf{q}) = P \rho. \quad (1.9)$$

The term with time derivation denotes the change in density and porosity caused by the dependence of liquid density on concentrations of solutes or by the thermal expansion effect. In this project, to describe the transport of solutes, we will use a simplified model with constant density and porosity, therefore equation (1.9) will be reduced to

$$\text{div } \mathbf{v} = \frac{P}{n}. \quad (1.10)$$

Now, equations (1.7) and (1.10) can be used as a system of two equations with two unknown functions q , ϕ , or modified by combining them into a single equation

$$\nabla \cdot (\mathbf{K} \nabla \phi) = P \quad (1.11)$$

with one unknown function ϕ .

1.3 Transport of solutes

The transport of solutes in a porous medium occurs through many processes, e.g. advection, diffusion, dispersion, sorption, chemical reactions and radioactive decay. In this project, we will focus on three basic transport processes of a significantly different nature: advection, molecular diffusion, and mechanical dispersion.

The amount of the solute in the solution is represented by the concentration c , expressed as the mass of the solute per unit volume of solution (water).

1.3.1 Advection and diffusion-dispersion processes

Advection is defined as the transfer of a substance caused by the movement of the entire solution. The amount of transferred substance can be easily derived from the Darcy flow velocity: the mass of substance passed through a unit area per unit time is

$$\mathbf{q}_c^{adv} = c \mathbf{q}. \quad (1.12)$$

Diffusion-dispersion processes induce the movement of the solutes due to a concentration gradient, from a region of high concentration to a region of low concentration. Considering a porous environment, we will express the diffusion-dispersion processes as two processes: molecular diffusion and mechanical dispersion.

Molecular diffusion is influenced by the microscopic structure of the environment. Mechanical dispersion is caused directly by the inhomogeneity of the velocity in pores, i.e. in some places the solution moves faster and in some slower than the average pore velocity \mathbf{v} .

The intensity of molecular diffusion is independent of the flow and is given by the value of the diffusion coefficient D_m . This coefficient varies for different substances and can depend on other influences such as temperature. The diffusion in the fluid is described by Fick's law

$$\frac{\partial c}{\partial t} = D_m \Delta = D_m \sum_i \frac{\partial^2 c}{\partial x_i^2}, \quad (1.13)$$

where D_m is a scalar coefficient that indicates the isotropy of the process. In a porous medium, the diffusion is influenced by the geometry of pores described by the tensor of tortuosity \mathbf{T} . Using this tensor, the tensor of molecular diffusion can be obtained

$$\mathbf{D}_m = D_m \mathbf{T} \quad (1.14)$$

Mechanical dispersion is described by the same equation as molecular diffusion but with a different coefficient. This coefficient is called a mechanical dispersion tensor \mathbf{D}_f and depends on the flow velocity.

The combined molecular diffusion and dynamic dispersion form a total dispersion process called hydrodynamic dispersion. It is characterized by the tensor of hydrodynamic dispersion \mathbf{D}_h

$$\mathbf{D}_h = \mathbf{D}_m + \mathbf{D}_f. \quad (1.15)$$

The effect of molecular diffusion and mechanical dispersion on the total dispersion depends on the properties of the porous medium, specific boundary conditions, and the scale of observation. The character of the total dispersion is given by the Peclet number

$$\text{Pe} = \frac{vd}{D_m}, \quad (1.16)$$

where v is the average pore velocity of the solution [m/s], d is the average grain size [m], and D_m is the molecular diffusion coefficient [m²/s]. The small Peclet number ($\text{Pe} < 0.01$) corresponds to the dominant effect of molecular diffusion, while the large Peclet number ($\text{Pe} > 104$) corresponds to the dominant effect of mechanical dispersion. Molecular diffusion is usually neglected for $\text{Pe} > 20$.

1.3.2 Advection-diffusion equation

At this point, the advection-diffusion equation will be introduced. The total mass flow \mathbf{q}_c affected by advection and hydrodynamic dispersion is

$$\mathbf{q}_c = n(c\mathbf{v} + \mathbf{D}_h \nabla c). \quad (1.17)$$

Considering this equation, the balance of the solute in the solution is defined by the equation

$$\frac{\partial}{\partial t} \int_V n c dV = - \int_{\partial V} \mathbf{q}_c \cdot d\mathbf{S} + \int_V (P^+ c^* + P^- c) dV + \int_V r dV \quad \text{for any } V, \quad (1.18)$$

where P^+ and P^- express the positive and negative parts of the solution source density function (1.9). These parts represent the fact that the solution with the given concentration c^* is injected, while the solution with concentration c corresponding to the required value of the function $c(\mathbf{x}, t)$ is pumped out from a given point. The term r describes the amount of the solute created or destructed by other processes (e.g., chemical reactions).

By standard adjustment of integrals, equation (1.18) can be transformed to a differential equation

$$\frac{\partial c}{\partial t} = -\nabla \cdot (c\mathbf{v}) + \nabla \cdot (\mathbf{D}_h \nabla c) + \frac{1}{n}(P^+ c^* + P^- c) + \frac{r}{n}, \quad (1.19)$$

which is called the advection-dispersion equation. It can be modified by deriving the product in the advection term and replacing the term $\nabla \cdot \mathbf{v}$ with equation (1.10), reducing the term P^-

$$\frac{\partial c}{\partial t} = -\mathbf{v} \cdot \nabla c + \nabla \cdot (\mathbf{D}_h \nabla c) + \frac{P^+}{n}(c^* - c) + \frac{r}{n}. \quad (1.20)$$

1.3.3 Initial and boundary conditions

As concentration is a time function, it is necessary to define the initial conditions. In our case, it is the distribution of the concentrations of the solutes at the beginning of the investigated time interval

$$c(\mathbf{x}, 0) = c_0(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega, \quad (1.21)$$

where $\Omega \subset \mathbb{R}^3$ is the volume where the problem is solved. If transport is not conservative (e.g., affected by chemical reactions), the concentrations of the solutes that react with the examined solute must be defined in the same way.

Dirichlet boundary condition

The Dirichlet boundary condition specifies the concentration on a given part of the boundary $\Gamma_1 = \partial\Omega$

$$c(\mathbf{x}, t) = c_D(\mathbf{x}, t) \quad \forall \mathbf{x} \in \Gamma_1. \quad (1.22)$$

In the case of processes with the dominant effect of advection, this condition is defined on the part of the boundary where the direction of flow of the solution is inward.

Neumann boundary condition

The Neumann boundary condition applies when only the dispersion flow is specified

$$(\mathbf{D}_h \nabla c) \cdot \mathbf{v} = q_{disp} \quad \forall \mathbf{x} \in \Gamma_2. \quad (1.23)$$

This condition is used when the flow through the boundary is zero, which means that the boundary is impermeable. Another option is a case where the advection effect on the part of the boundary with outward flow is dominant, i.e., the concentrations on both sides of the boundary are equal.

Cauchy boundary condition

The Cauchy boundary condition specifies general conditions for the prescribed flow

$$(c\mathbf{v} + \mathbf{D}_h \nabla c) \cdot \mathbf{v} = q_{disp} \quad \forall \mathbf{x} \in \Gamma_3. \quad (1.24)$$

Using this condition, it is possible to derive conditions for various specific situations. For example, if there is zero flow through the boundary, the flow velocity \mathbf{v} is zero, so the condition transforms into a Neumann boundary condition.

Chapter 2

Numerical model

The second chapter is based on Milan Hokr's textbook [1], bachelor project of A. Sokolenko [3], and doctoral thesis of P. Jiranek [5]. In this chapter, we will focus on the numerical solution of the transport part of the advection-diffusion equation (1.20) formulated in the first chapter. In order to achieve this goal, two different methods will be used: the Finite Difference Method and the Finite Volume Method.

2.1 Operator Splitting Method

Before starting to solve the transport part, we need to describe general reasons for it and introduce the method that will allow us to deal and solve it separately from the reaction part.

In the first chapter, the advection-diffusion equation (1.20) was formulated as follows:

$$\frac{\partial c}{\partial t} = -\mathbf{v} \cdot \nabla c + \nabla \cdot (\mathbf{D}_h \nabla c) + \frac{P^+}{n}(c^* - c) + \frac{r}{n}. \quad (2.1)$$

This equation defines the change in concentration caused by internal processes and water inflow P^+ . Advection and diffusion terms include the spatial derivative: advection is defined by the velocity vector field and diffusion is defined by the concentration gradient. Meanwhile, the reaction term has no derivatives, which means that it evolves only in time and not in space. It will be difficult to solve such an equation on the whole at once because each part requires different mathematical methods. Due to this reason, we need to introduce the Operator Splitting Method.

The operator splitting method is used to solve complex differential equations consisting of two or more parts of different nature that require different mathematical approaches. The general idea is to rewrite the overall evolution operator as a sum of evolution operators corresponding to each part of the equation. Thus, a complex equation can be divided into sub-equations that are solved separately and then collected together. It involves the discretization of time into intervals and the sequential solution of each equation on these intervals. The results of one equation are used as the initial value for the other.

According to this method, the equation (2.1) can be rewritten using operator A which represents all operations applied to the concentration

$$\frac{\partial \mathbf{c}}{\partial t} = \mathbf{A} \mathbf{c}. \quad (2.2)$$

In our case, operator A is defined as a sum of two operators

$$\mathbf{A} = \mathbf{T} + \mathbf{R}, \quad (2.3)$$

where T represents operations in the transport part and R represents operations in the reaction part

$$\mathbf{T} \mathbf{c} = -\mathbf{v} \cdot \nabla c + \nabla \cdot (\mathbf{D}_h \nabla c) + \frac{P^+}{n}(c^* - c), \quad (2.4)$$

$$Rc = r(c). \quad (2.5)$$

Therefore, the transport-reaction problem will be split into two sub-equations:

$$\begin{aligned} \frac{\partial u(t)}{\partial t} &= Tu(t), \\ \frac{\partial v(t)}{\partial t} &= Tv(t), \end{aligned} \quad (2.6)$$

where $t \in [t_n, t_{n+1}]$ and $t_{n+1} = t_n + \Delta t$. Firstly, the transport sub-equation is solved for the time step Δt , then the reaction sub-equation is solved for the same time step Δt using the results of the transport sub-problem as an initial value. Finally, the results of the reaction sub-equation are applied as the initial value to the transport sub-equation for the next time step.

2.2 Finite Difference Method

2.2.1 Basic characteristics of FDM

The main idea of the FDM is to approximate the derivatives in a differential equation with finite differences. The medium is discretized into a mesh consisting of a finite number of points. The differences use the values of the searched function at adjacent points for the approximation of the derivatives. As a result, we obtain a mesh of points containing the approximated values of the function.

The differences are represented by special difference formulae. For the first derivative, there are three basic types of difference formulae: forward, backward, and central. The main difference between these types is the selection of points for approximation. As an example, we will approximate the spatial derivative of the concentration function c using each formula. Let us consider the mesh consisting of three points x_{n-1} , x_n , and x_{n+1} with a Δx as a distance between them.

The forward difference formula uses for the approximation the values of the function in points x_n and x_{n+1} :

$$\frac{\partial c}{\partial x} = \frac{c_{n+1} - c_n}{\Delta x} + O(\Delta x). \quad (2.7)$$

The backward difference formula uses the values of the function in points x_{n-1} and x_n :

$$\frac{\partial c}{\partial x} = \frac{c_n - c_{n-1}}{\Delta x} + O(\Delta x). \quad (2.8)$$

Together, these two types of difference formulae are called one-sided difference formulae. However, a more accurate approximation can be obtained by using the central difference formula that uses the values of the function at the points x_{n-1} and x_n :

$$\frac{\partial c}{\partial x} = \frac{c_{n+1} - c_{n-1}}{2\Delta x} + O(\Delta x^2). \quad (2.9)$$

It is also possible to formulate the central difference formula for the second derivative of the function c :

$$\frac{\partial^2 c}{\partial x^2} = \frac{c_{n+1} - 2c_n + c_{n-1}}{\Delta x^2} + O(\Delta x^2). \quad (2.10)$$

The results of this approximation will be used later in our equation.

2.2.2 Application of FDM to transport problem

After the application of the OSM to the equation (1.20), the following equation is obtained

$$\frac{\partial c}{\partial t} = -\mathbf{v} \cdot \nabla c + \nabla \cdot (\mathbf{D}_h \nabla c) + \frac{P^+}{n}(c^* - c). \quad (2.11)$$

This equation can be rewritten for the 2D problem as a sum of differential functions as follows:

$$\frac{\partial c}{\partial t} = -(v_x \frac{\partial c}{\partial x} + v_y \frac{\partial c}{\partial y}) + \mathbf{D}_h \left(\frac{\partial}{\partial x} \cdot \frac{\partial c}{\partial x} + \frac{\partial}{\partial y} \cdot \frac{\partial c}{\partial y} \right) + \frac{P^+}{n}(c^* - c). \quad (2.12)$$

After rearrangement, the equation will take the following form:

$$\frac{\partial c}{\partial t} = -v_x \frac{\partial c}{\partial x} - v_y \frac{\partial c}{\partial y} + D_{h,x} \frac{\partial^2 c}{\partial x^2} + D_{h,y} \frac{\partial^2 c}{\partial y^2} + \frac{P^+}{n}(c^* - c). \quad (2.13)$$

The terms v_x and v_y are the components of the velocity vector \mathbf{v} . They are defined as

$$\begin{aligned} v_x &= -\frac{1}{n} K_x \frac{\partial \phi}{\partial x}, \\ v_y &= -\frac{1}{n} K_y \frac{\partial \phi}{\partial y}. \end{aligned} \quad (2.14)$$

Since the evaluation of the velocity components is contained in a separate part of the TRM2D software, we will use them as they are and will not substitute them in the equation (2.11).

Furthermore, the equation (2.11) should be transformed according to the amount of different concentration components. Considering $m \in N$ concentration components the equation can be rewritten as follows

$$\frac{\partial c_I}{\partial t} = -v_x \frac{\partial c_I}{\partial x} - v_y \frac{\partial c_I}{\partial y} + D_{h,x} \frac{\partial^2 c_I}{\partial x^2} + D_{h,y} \frac{\partial^2 c_I}{\partial y^2} + \frac{P^+}{n}(c_I^* - c_I), \quad (2.15)$$

where $I \in 1, \dots, m$.

In the first chapter, advection was defined as the transfer of a substance caused by the movement of the entire solution. It means that it always depends on the orientation of the velocity vector. To approximate this process, it is appropriate to use the upwind scheme that respects the orientation of the process by considering the value of the transported concentration to be equal to the concentration in the direction opposite to the velocity vector. The forward difference formula (2.6) is used for movement in the positive direction ($v > 0$), and the backward difference formula (2.7) is used for the opposite direction ($v < 0$).

The diffusion is not dependent on the movement of the whole solution as it induces the movement of the solutes due to a concentration gradient. It allows us to use the central difference formula (2.8) for approximation. After the substitution of difference formulae, the equation (2.13) will take the following form:

$$\begin{aligned} \frac{\partial c_{I,i,j}(t)}{\partial t} &= -v_x \cdot \frac{c_{I,i,j}(t) - c_{I,i-1,j}(t)}{\Delta x} \cdot \frac{1}{2}(1 + \text{sgn}(v_x)) - v_x \cdot \frac{c_{I,i+1,j}(t) - c_{I,i,j}(t)}{\Delta x} \cdot \frac{1}{2}(1 - \text{sgn}(v_x)) \\ &\quad - v_y \cdot \frac{c_{I,i,j}(t) - c_{I,i,j-1}(t)}{\Delta y} \cdot \frac{1}{2}(1 + \text{sgn}(v_y)) - v_y \cdot \frac{c_{I,i,j+1}(t) - c_{I,i,j}(t)}{\Delta y} \cdot \frac{1}{2}(1 - \text{sgn}(v_y)) \\ &\quad + D_{h,x} \frac{c_{I,i+1,j}(t) - 2c_{I,i,j}(t) + c_{I,i-1,j}(t)}{\Delta x^2} + D_{h,y} \frac{c_{I,i,j+1}(t) - 2c_{I,i,j}(t) + c_{I,i,j-1}(t)}{\Delta y^2} + \frac{P^+}{n}(c_{I,i,j}^* - c_{I,i,j}), \end{aligned} \quad (2.16)$$

where i and j are the indices of the internal points of the mesh.

The concentrations and velocity of the flow at the boundary points are defined by the Dirichletian condition at the entire boundary

$$c_{I,i,j} = \bar{c}_{I,i,j}, \quad (2.17)$$

where I is the index of the concentration component, and i, j are the indices of the boundary points of the mesh.

2.3 Finite Volume Method

2.3.1 Basic characteristics of FVM

The FVM is based on dividing the medium into a finite number of control volumes (also called grid cells) and approximating the integral form of differential equation over each of these volumes. The variables of the given function are replaced by the average values in the control volumes. Each term of the equation could be separately approximated in a suitable way.

The terms containing a divergence are converted to surface integrals using the Gauss's theorem. These terms are then considered as fluxes through the surfaces of each finite volume. It is important to calculate the flux through the given surface in the same way for both adjacent cells. In that case, the FVM satisfies the conservation law and guarantees the conservation of the given quantity, which is considered as the advantage of the FVM over the FDM.

2.3.2 Application of the FVM to the transport problem

To apply the FVM to the transport problem, we will use the advection-diffusion equation (1.19). After deriving the transport part using the OSM and considering multiple concentration components, the following equation is obtained:

$$\frac{\partial c_I}{\partial t} = -\nabla \cdot (c_I \mathbf{v}) + \nabla \cdot (\mathbf{D}_h \nabla c_I) + \frac{1}{n} (P^+ c_I^* + P^- c_I) + \frac{r}{n}. \quad (2.18)$$

Firstly, we will divide the medium Ω into a set \mathcal{T} of finite volumes or cells. The equation (2.18) is then integrated over the control volume $K \in \mathcal{T}$

$$\int_K \frac{\partial c_I(t, x)}{\partial t} = - \int_K \text{div}(c_I \mathbf{v}) + \int_K \text{div}(\mathbf{D}_h \cdot \text{grad}(c_I) \cdot \mathbf{n}) + \int_K \frac{1}{n} (P^+ c^* + P^- c). \quad (2.19)$$

We assume that function c is continuously differentiable in space and time. Therefore, it is possible to apply the Gauss's theorem

$$\frac{d}{dt} \int_K c_I(t, x) = - \int_{\partial K} c_I \mathbf{v} \cdot \mathbf{n} + \int_{\partial K} \mathbf{D}_h \cdot \text{grad}(c_I) \cdot \mathbf{n} + \frac{1}{n} \int_K (P^+ c^* + P^- c). \quad (2.20)$$

Then we must introduce a discrete variable $\bar{c}_{I,K}$ which represents the average value of concentration of I th solute in the cell K

$$\bar{c}_{I,K}(t) = \frac{1}{m(K)} \int_K c_I(t, x) dx, \quad (2.21)$$

where $m(K)$ is the volume of cell K .

Let us consider two adjacent cells K and L . It is possible to rewrite the advection term as

$$\int_{\sigma_{KL}} c_I \mathbf{v} \cdot \mathbf{n} = f_{K|L} * \bar{c}_{I,K|L}, \quad (2.22)$$

where $f_{K|L}$ is a water flux at the boundary $\sigma_{K|L}$ between K and L. It is defined as

$$f_{K|L} = n \cdot m(\sigma_{K|L}) \cdot v_x, \quad (2.23)$$

if the cells are adjacent at the direction of x, and as

$$f_{K|L} = n \cdot m(\sigma_{K|L}) \cdot v_y, \quad (2.24)$$

if in the direction of y. The term $m(\sigma_{K|L})$ represents the area of the boundary between K and L, v_x and v_y are the elements of average pore velocity vector defined by equation (1.6), and n is the porosity coefficient.

Considering isotropic diffusion, it is possible to approximate the diffusion term in the following way

$$\int_{\sigma_{K|L}} D \cdot \text{grad}(c_I) \cdot \mathbf{n} = m(\sigma_{K|L}) \cdot d_I \cdot \frac{\bar{c}_{I,K} - \bar{c}_{I,L}}{d_{K|L}}, \quad (2.25)$$

where d_I is a diffusion coefficient related to the Ith transported concentration component, $\bar{c}_{I,K}$ and $\bar{c}_{I,L}$ are average concentrations in K and L, and $d_{K|L}$ is a distance between centers of K and L.

For the reasons described in Subsection 2.2.2, it is appropriate to apply the upwind scheme to the advection term. After the substitution of advection and diffusion terms, the equation (2.18) can be rewritten in the following way:

$$\begin{aligned} n \cdot m(K) \cdot \frac{d\bar{c}_{I,K}(t)}{dt} = & - \sum_{\sigma \in \varepsilon_K} f_\sigma \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_\sigma)) - \sum_{\sigma \in \varepsilon_K} f_\sigma \bar{c}_{I,L}(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_\sigma)) \\ & + \sum_{\sigma \in \varepsilon_K} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} + m(K) \cdot P_K^+ c_{I,K}^+(t) - m(K) \cdot P_K^- c_{I,K}(t), \end{aligned} \quad (2.26)$$

where ε_K is a whole boundary of cell K. In the advection term, if the direction of flow through the boundary σ is inward, then we consider the concentration of the adjacent cell L, if the direction of flow is outward, then the concentration in K itself. However, this equation defines the change of concentration in the internal cells only and does not consider the boundary conditions.

The initial and Dirichlet boundary conditions are defined as

$$\begin{aligned} \bar{c}_{I,K}(0) &= \frac{1}{m(K)} \int_K c_I(o, \mathbf{x}, K \in \mathcal{T}, \\ \bar{c}_{I,K}^+(0) &= \frac{1}{m(\sigma)} \int_\sigma c_I(t, \mathbf{x}, \sigma \subset \partial\Omega \end{aligned} \quad (2.27)$$

After modification of equation (2.26) using the boundary conditions (2.27), the following equation is obtained

$$\begin{aligned} n \cdot m(K) \cdot \frac{d\bar{c}_{I,K}(t)}{dt} = & - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} f_\sigma \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_\sigma)) - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} f_\sigma \bar{c}_{I,L}(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_\sigma)) \\ & - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} f_\sigma \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_\sigma)) - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} f_\sigma c_{I,K}^+(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_\sigma)) \\ & + \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} + \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} d_{\Gamma,K} d_I \cdot d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \\ & + m(K) \cdot P_K^+ c_{I,K}^+(t) - m(K) \cdot P_K^- c_{I,K}(t), \end{aligned} \quad (2.28)$$

where ε_{int} is a set of all internal boundaries of the medium, ε_{ext} is a set of all external boundaries of the medium, $d_{\Gamma,K}$ is the diffusion boundary flow coefficient representing the ratio of the external boundary area $m(\sigma)$, $\sigma \in \varepsilon_K \cap \varepsilon_{ext}$ to the distance $d_{K|\sigma}$ between the center of the cell K and the boundary σ . This equation considers boundary conditions and therefore defines the change of concentration in all cells of the grid.

2.4 Solution of the resulting system of ordinary differential equations of the first order

Since the TRM2D software uses the FVM during the simulation of groundwater flow movement, we will solve the system of ordinary differential equations derived using the FVM.

For approximation of time derivative of function $\bar{c}_{I,K}(t)$ in every cell $K \in \mathcal{T}$ of the grid we will apply the Euler's method to the system of equations (2.28).

$$\begin{aligned}
n \cdot m(K) \cdot \frac{\bar{c}_{I,K}(t + \Delta t) - \bar{c}_{I,K}(t)}{\Delta t} = & - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} f_{\sigma} \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_{\sigma})) - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} f_{\sigma} \bar{c}_{I,L}(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_{\sigma})) \\
& - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} f_{\sigma} \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_{\sigma})) - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} f_{\sigma} c_{I,K}^+(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_{\sigma})) \\
& + \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} + \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} d_{\Gamma,K} d_I \cdot d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \\
& + m(K) \cdot P_K^+ c_{I,K}^+(t) - m(K) \cdot P_K^- c_{I,K}(t),
\end{aligned} \tag{2.29}$$

where Δt is the selected time step. The ordered set of values

$$\{\bar{c}_{I,K}(t) | t \in (0, \dots, T), K \in \mathcal{T}, I \in \{1, \dots, m_I\}\} \tag{2.30}$$

is then a solution of the transport problem.

Chapter 3

TRM2D software

The third chapter contains the description of the TRM2D software and the implementation of the main objective of this bachelor project which was specified in Section 2 as an extension of the transport calculation method with the diffusion effect as an alternative solution of the transport problem.

3.1 Basic description of TRM2D software

The first subsection of the third chapter describes the TRM2D software in the context of simulating the transportation of solution components. It includes the description of the configuration file structure and methods that provide the main calculations of the transport problem.

TRM2D is a mathematical-physical software written in the language C++. It was developed for the simulation of transport and reaction processes occurring during groundwater flow movement. Its basic idea consists in combination of the PhreeqcRM geochemical library with 2D transport in the regular rectangular mesh of elements (also volumes/cells). The software is designed to support the settings of the boundary conditions in the form of the inflow and outflow of the solution volume per unit of time. It is also possible to set the changes in the boundary conditions during the calculation process. The same feature is implemented for the reaction module, i.e., it is possible to set the initial and boundary compositions of the solute and its changes during the calculation. The detailed solute compositions are stored in the Phreeqc configuration files. All these configurations are collected in the XML configuration file, which is read as input after the start of program compilation.

When the calculations are completed, the output is written in the form of XML file and could be extended with the CSV files for the transport and reaction module separately.

3.2 Description of configuration file

As mentioned in the previous subsection, the configuration file contains the definitions of the transport and reaction initial and boundary conditions. It is presented in the form of XML file. This file is stored in XML format. The format of the file implies the syntax in which the settings are written, i.e. the set of tags nested within each other in the form of tree. The most general, root tag <TRM> is divided into several child tags. Each of these child tags contains information for different purposes.

The tag <LogFile> is responsible for the log files that are used to store the internal information about software run. It specifies their format and the level of logging.

The tag `<Quantities>` specifies units of measurement for each quantity. The program uses quantityID to identify the type of data it is currently reading. It is also important to ensure that the quantities in the XML file are consistent with those in the Phreeqc configuration files.

The tag `<TransportModule>` contains the key data for the simulation of transport processes. It defines the time parameters, such as time interval, time step, and the moment of boundary conditions change; the grid parameters, such as the number of rows and columns, and volumes of the cells; initial and boundary conditions for each cell. The composition of the particular solution within the initial or boundary condition is represented by the map value defined under `<ReactionModule>` tag. The content of the `<TransportModule>` will be later corrected to implement the main project's goal.

The tag `<ReactionModule>` defines the configurations for the reaction module that is based on the Phreeqc library. It includes the name of the Phreeqc database, map values and names of the Phreeqc files containing the information about solution composition for initial and boundary conditions, and the values of physical quantities, e.g. temperature, pressure, saturation, density.

The tag `<ResultFile>` specifies the output configurations. Among the directory path and name of XML file, it is also possible to extend the output with the CSV files or separately define the frequency of the recordings from a reaction and transport modules.

3.3 Description of program execution process

3.3.1 Loading of input data

The program starts with loading of the configuration XML file in the `trm_main.cpp` file. The name of the file is set as the parameter of the `load_inputs(...)` method defined in the `trm_main` class. The method uses the TinyXML library for the interpretation of the configuration file. This library builds a Document Object Model (DOM) and provides access to its elements. The data from each tag are collected and stored in the memory.

Since this bachelor project focuses on the transport part of the TRM2D software, the most important part of the `load_inputs(...)` method relates to the reading of `<TransportModule>` tag. The transport module of the TRM2D software is implemented as a class that contains all the methods and variables needed for the transport calculation process. The instance of this class is called `trans` and is declared in the `trm_main` class.

The initialisation of the transport module starts with the setting of metadata using the method `trans.set_metadata(...)`, which reads the attributes of the `<TransportModule>` tag. The method `trans.set_grid_size(...)` reads the time settings from the `<Time>` tag. Grid parameters are loaded by the `trans.set_grid_size(...)` method from the `<GridSize>` tag.

The parameters of the cells, i.e., volume, porosity, initial and boundary conditions, are stored as children of the `<GridElements>` tag. Volume and porosity are loaded using the `init_variable(...)` method of the `trm_main` class. It is a multipurpose method that allows store the loaded data in the form of sparse/dense vectors and matrices, depending on the requirements of the user. This method compares the transport module grid parameters with the attributes of the loaded tag, and if they match, initialises the data using the `init_from_string(...)` method.

Finally, the initial and boundary conditions are loaded within the `init_components(...)` method.

3.3.2 Activation of transport module

The next and the main stage of program execution takes place in the `calculate()` method of the `trm_main` class which goes right after the `load_inputs(...)` method in the `trm_main.cpp` file. This method includes all the processes related to the calculations in the transport and reaction modules. It

activates the transport module, sets physical quantity values, initiates the reaction module and boundary conditions, and prepares all data structures required for calculation. When all preparations are done, it starts the calculation in the for cycle, which is representing the Operator splitting method. The transport module calculation occurs in the `trans.calculate(...)` method defined in `trm_trans.cpp`.

Before analysis of the `trans.calculation()` method, it is necessary to describe the activation of the transport module and the preparation of the data structures required for calculation. The transport module is activated by calling the `trans.activate()` method.

The activation starts with the calculation of temporary position matrices needed for the initialisation of the variables. Then the transport module calculates diagonal matrix `volume_diag_inverse` containing the inversed volumes of the grid cells, multiplied by the porosity coefficient. This matrix is set in the `init_volumes()` method of the `trm_trans` class based on the data read in the configuration file. After all, it calculates the vector of the hydraulic head and the matrix of grid flow for the time of boundary conditions change.

After activation of the transport module, the method `activate_boundary_conditions()` is used for initialisation of the boundary conditions for calculation. This method applies the method `trans.set_boundary_conditions(...)` with component names and corresponding conditions as parameters to set the boundary conditions in the transport module.

The last preparation before the start of calculation cycle is creating the empty vector of double type called `transport_concs`. This vector will be filled with the concentrations of the components in the `react.activate()` method and then is used as a parameter for the `trans.calculate(...)` method.

3.3.3 Calculation of transport

The calculation of transport is executed within the `trans.calculate(...)` method defined in the `trm_trans.cpp` file. In accordance with the OSM, this method is invoked for each time step. As parameters, it requires the time of the calculation, concentrations of the transport components, names of transport components, and time of boundary conditions change.

The data from the `transport_concs` vector, represented within the `trans.calculate()` method as `concentrations[]`, are assigned to the `transport_components_mat` matrix declared as a part of the transport module. The number of rows in this matrix is equal to the number of cells `ncells`, while the number of columns is equal to the number of components. All concentrations of transport components are stored in this matrix.

The main calculations are executed by the following expression on the line 33 in Appendix A

```
1 transport_components_mat += volume_diag_inverse * (in_out_flow_modif * in_comps *
2 curr_time_step + grid_flow_modif * transport_components_mat * curr_time_step);
```

To describe this expression, we will rewrite the equation (2.29) transformed to a similar form

$$\begin{aligned}
\bar{c}_{I,K}(t + \Delta t) = & \bar{c}_{I,K}(t) + \frac{1}{n \cdot m(K)} \left(- \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} f_{\sigma} \bar{c}_{I,K}(t) \cdot \frac{1}{2} (1 + \text{sgn}(f_{\sigma})) \Delta t - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} f_{\sigma} \bar{c}_{I,L}(t) \cdot \frac{1}{2} (1 - \text{sgn}(f_{\sigma})) \Delta t \right. \\
& - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} f_{\sigma} \bar{c}_{I,K}(t) \cdot \frac{1}{2} (1 + \text{sgn}(f_{\sigma})) \Delta t - \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} f_{\sigma} c_{I,K}^+(t) \cdot \frac{1}{2} (1 - \text{sgn}(f_{\sigma})) \Delta t \\
& + \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} \Delta t + \sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} d_{\Gamma,K} d_I \cdot d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \Delta t \\
& \left. + m(K) \cdot P_K^+ c_{I,K}^+(t) \Delta t - m(K) \cdot P_K^- c_{I,K}(t) \Delta t \right),
\end{aligned} \tag{3.1}$$

and compare them. The TRM2D model by default considers a grid with an impermeable external boundary. Therefore, the effect of advection through the external boundary needs to be combined with the source term.

In the expression (3.1), the term `in_out_flow_modif * in_comps * curr_time_step` represents the inflow of the concentration given by the boundary conditions. The matrix `in_out_flow_modif` is a sparse matrix that contains the data about the amount of injected solution P_K^+ . This information is defined by the user under the `<InOutFlow>` tag in the configuration XML file. The matrix `in_out_flow_modif` is element-wise multiplied by the sparse matrix `in_comps` of the same size as `transport_components_mat`. This matrix contains the concentrations of the components taken from the boundary conditions and corresponds to the $c_{I,K}^+$ in equation (3.1). When comparing the term `in_out_flow_modif * in_comps * curr_time_step` with the equation (3.1), it is equivalent to $m(K) \cdot P_K^+ c_{I,K}^+(t) \Delta t$ which in fact represents both terms $-\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} f_\sigma c_{I,K}^+(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_\sigma)) \Delta t + m(K) \cdot P_K^+ c_{I,K}^+(t) \Delta t$. Here the author of the original software made identity between boundary conditions and sources/sinks because they both bring/take water into/out of one cell from the outside. This makes the code simpler but does not limit the class of solvable problems very much.

The term `grid_flow_modif * transport_components_mat * curr_time_step` represents the advection effect through internal boundaries combined with the outflow. The matrix `grid_flow_modif` is a square matrix of size `ncellsxncells` that contains the values of water flux between each cell and the value of outflow. This matrix is calculated by the software in the `init_grid_flow(...)` method during the initialisation of the grid in the `load_inputs()` method. The `grid_flow_modif` matrix is multiplied by the `transport_components_mat`. When comparing the term `grid_flow_modif * transport_components_mat * curr_time_step` with the equation (3.1), it corresponds to $-\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} f_\sigma \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_\sigma)) \Delta t - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} f_\sigma \bar{c}_{I,L}(t) \cdot \frac{1}{2}(1 - \text{sgn}(f_\sigma)) \Delta t - \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} f_\sigma \bar{c}_{I,K}(t) \cdot \frac{1}{2}(1 + \text{sgn}(f_\sigma)) \Delta t - m(K) \cdot P_K^- c_{I,K}(t) \Delta t$.

As can be seen from the comparison, the calculation method implemented in the TRM2D software does not consider the effect of diffusion. Therefore, as an alternative solution of the transport part, we have decided to improve the TRM2D calculation method by adding the diffusion effect.

Chapter 4

Alternative solution of transport problem in TRM2D software

4.1 Implementation of the alternative solution

The Section 4.1 describes the changes made in the software in order to add the diffusion effect into calculation method. The formula of concentration change caused by diffusion only can be derived from the equation (3.1)

$$\bar{c}_{I,K}(t+\Delta t) = \bar{c}_{I,K}(t) + \frac{1}{n \cdot m(K)} \left(\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} \Delta t + \sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} d_{\Gamma,K} d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \Delta t \right). \quad (4.1)$$

Therefore, to consider the diffusion effect during the calculation, it is necessary to add two terms: the diffusion flow between the cells of the grid, i.e., through the internal boundaries $\sigma \in \varepsilon_{int}$

$$\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{K|L}} \Delta t, \quad (4.2)$$

and the diffusion flow through the external boundaries $\sigma \in \varepsilon_{ext}$

$$\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} d_{\Gamma,K} d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \Delta t. \quad (4.3)$$

The variables $\bar{c}_{I,K}(t)$ (and also $\bar{c}_{I,L}(t)$) and $\bar{c}_{I,K}^+(t)$ are already implemented in the software and represented by the matrices `transport_components_mat` and `in_comps`.

To implement the internal diffusion (4.2), we need to define two additional parameters of the grid: the sizes of all internal boundaries $\sigma \in \varepsilon_{int}$, and the distances between the centers of adjacent cells. Since the TRM2D software calculates the transport in the regular rectangular mesh of cells, these parameters can be obtained using the longitudinal and transverse sizes of each cell.

Because the advection part of the model considered impermeable boundaries, the software did not provide any option to define boundary conditions and associate them to individual edges of the mesh. We have decided to let the user specify the geometrical diffusion flow coefficient $d_{\Gamma,K}$, which represents the intensity of the inflow of the concentration by the diffusion effect.

The last value that is missing in terms (4.2) and (4.3), is the diffusion coefficient d_I related to the I th transported concentration component.

To specify these values, it is appropriate to use the same configuration XML file as for other parameters of the grid.

4.1.1 Modification of the configuration file and reading methods

The longitudinal and transverse parameters of the cells will be represented by two vectors, `deltaX` and `deltaY`. Since these values are parameters of grid cells, we created the tags `<DeltaX>` and `<DeltaY>` as children tags of the `<GridElements>` tag. The new tags contain the indexed values of the size of each column and row of the cell. Using the indices allows us to define different size values for each row and column of the grid.

To store these values in the transport module, we have declared two new variables of the component class in the `transport_module` class in the `trm_trans.h` file. The component class contains a set of variables and methods for working with matrices. It is also needed to modify the `transport_module::get_variable()` method in `trm_trans.cpp` so that it returns newly added variables.

Then we modified the `trm_main::load_inputs()` method in `trm.cpp` to load the data for the new variables. We used the method `init_variable()`, which was modified to load the data into newly added variables. Also, the condition was added to check the number of elements in `deltaX` and `deltaY`. These numbers should match the number of columns and rows defined under `<GridSize>` tag in the XML file.

By analogy with the `deltaX` and `deltaY`, we have added the variable `in_out_diff` of type component to the `transport_module` class in the `trm_trans.h` file. This variable stores the sums of diffusion boundary flow coefficients $d_{\Gamma,K}$ for each cell (specifically the values of $\sum_{\sigma \in \varepsilon_K \cap \varepsilon_{ext}} d_{\Gamma,K}$ and it is defined in the XML file under the new tag `<InOutDiff>` in the `<GridElements>` tag. This tag uses index data format that allows to define different coefficients for each cell of the grid. After modification, the method `init_variable()` is used to load the data into the new variable.

To define the diffusion coefficients related to the specific component of the solution, the new tag `<DiffusionCoefficients>` was added to the `<TransportModule>` tag. The coefficients are represented by the set of tags `<DiffusionCoefficient>` with a name of the component as an attribute. It is also possible to set the default coefficient that will be assigned to all components without a separately specified coefficient. To load these coefficients into the transport module, the new method `load_diff_coeffs(...)` was added to the `transport_module` class in the `trm_trans.cpp`. This method stores the values into new variable `diff_coeffs` of type `map<string, double>` added to the `transport_module` in the `trm_trans.h` file.

4.1.2 Preparation of matrices for calculation

The preparation of matrices for calculation was implemented within the `trm_main::calculate()` method during the activation of the transport module in the `transport_module::activate()` method.

The sizes $m(\sigma), \sigma \in \varepsilon_{int}$ of all internal boundaries are calculated in the `transport_module::init_cell_side_sizes()` method. The size of the boundary between cells K and L is derived as the arithmetic average of the adjacent boundary sizes of each cell. The size of the cell boundary σ is obtained by the division of the cell volume by the longitudinal (or transverse) side of the cell represented by `deltaX` (or `deltaY`). All these sizes are stored in the `cell_side_sizes` sparse matrix declared as part of the `transport_module` class. This matrix has `ncells` rows and `ncells` columns. By default, it is filled with zeros. Depending on the mutual position of the cells, the boundary sizes are stored in this matrix. As a result, the matrix `cell_side_sizes` contains the sizes of all boundaries between each couple of cells of the grid. Distances between the centres of adjacent cells are obtained in a similar way. The new sparse matrix `distances_inverse` is declared in the `transport_module` class. The initialisation of this matrix is done in the `transport_module::init_distances()` method, which sets the size `ncells x ncells`, fills it with zeros, and calculates the distances between cell centres as the arithmetic average of the `deltaX` (or `deltaY`) values depending on the mutual position of the cells.

According to the equation (4.2), these values are divisors, so it is appropriate to inverse them here. As a result, the matrix `distances_inverse` contains the inverse distances between all the centres of adjacent cells of the grid.

The matrix of diffusion boundary flow coefficients $\sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} d_{\Gamma,K}$ is declared in the transport module as the sparse matrix `in_out_diff_mat`. To initialise this matrix, the method `transport_module::init_in_out_diff_mat()` was created. This method sets the size `ncells x ncells`, fills the matrix with zeros, and assigns the values from the `in_out_diff` component to the diagonal of the new matrix `in_out_diff_mat`.

The matrices `cell_side_sizes` and `distances_inverse` are then combined in the sparse matrix `diff_flows_mat` declared in the transport module. The `diff_flows_mat` matrix defines the values of the diffusion flow through internal boundaries. This matrix is initialised in the new method `transport_module::init_diff_flows_mat()`. This method provides element-wise multiplication of the `cell_side_sizes` and `distances_inverse` matrices. Then it counts the sum of the elements in each row, assigns it to the temporary vector `row_sums`, and puts that vector with a negative sign on the diagonal of the `diff_flows_mat` matrix. After all, the matrix `in_out_diff_mat` is subtracted from the `diff_flows_mat` to consider the boundary conditions.

All the methods described above are executed during the activation of the transport module. However, at that moment the reactive module is not activated yet and the exact components of the transport are unknown. The initialisation of the sparse matrix `diff_coeffs_mat` containing diffusion coefficients d_I is done once after reactive module activation, in the main calculation cycle of `trm_main::calculate()`. The matrix has a size equal to the number of actual components defined in Phreeqc and increased by one. During the initialisation, we compare the component names assigned to loaded coefficients with the names of the solution components, and fill the matrix `diff_coeffs_mat` according to the actual order of components.

4.1.3 Modification of calculation method

When all required matrices are prepared, it is possible to modify the transport calculation method `transport_module::calculate(...)`.

To implement the term (4.2) representing diffusion flow through internal boundaries, the matrix `diff_flows_mat` is multiplied by the `transport_components_mat` and `diff_coeffs_mat`. The result of this multiplication is then scalar multiplied by the time step `curr_time_step`. Therefore, the term (4.2) will be implemented as

$$\sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{int}} m(\sigma) \cdot d_I \cdot \frac{\bar{c}_{I,L}(t) - \bar{c}_{I,K}(t)}{d_{KL}} \Delta t = \text{diff_flows_mat} \cdot \text{transport_components_mat} \cdot \text{diff_coeffs_mat} \cdot \text{curr_time_step}. \quad (4.4)$$

The term (4.3) representing diffusion flow through the external boundaries, will be obtained by the matrix multiplication of matrices `in_out_diff_mat`, `in_comps`, and `diff_coeffs_mat`. The resulting matrix is then scalar multiplied by the `curr_time_step`. As a result, the term (4.3) will be implemented as follows

$$\sum_{\sigma \in \mathcal{E}_K \cap \mathcal{E}_{ext}} d_{\Gamma,K} d_I \cdot (\bar{c}_{I,K}^+(t) - \bar{c}_{I,K}(t)) \Delta t = \text{in_out_flow_modif} \cdot \text{in_comps} \cdot \text{diff_coeffs_mat} \cdot \text{curr_time_step}. \quad (4.5)$$

When these terms are put together, it is possible to add them to expression on line 33 in Appendix A

```
1 transport_components_mat += volume_diag_inverse * (in_out_flow_modif * in_comps *
2 curr_time_step + grid_flow_modif * transport_components_mat * curr_time_step +
```

```

3 diff_flows_mat * transport_components_mat * diff_coeffs_mat * curr_time_step +
4 in_out_diff_mat * in_comps * diff_coeffs_mat * curr_time_step);.

```

This expression now calculates the transport of the components in the solution considering the diffusion effect.

4.2 Testing of implemented solution

Subsection 4.2 describes the testing process of the extended TRM2D software on a 1D problem. The testing is performed by comparison of the analytical solution described in the T. Havelka bachelor's project [6] and the numerical solution of the diffusion equation implemented in the TRM2D software.

4.2.1 Description of analytical solution

The analytical solution of the diffusion equation considers a model of particle propagation through a thin, infinitely long tube. This process is described by the one-dimensional equation of diffusion

$$\frac{\partial}{\partial t}\varphi(x, t) = D\frac{\partial^2}{\partial x^2}\varphi(x, t) \quad (4.6)$$

which defines the change in diffusion according to the change in time. The explicit solution of this equation is obtained in the form:

$$\varphi(x, t) = \frac{M}{2\sqrt{\pi Dt}}e^{-\frac{x^2}{4Dt}} \quad (4.7)$$

where M is the amount of substance injected at the point $x_0 = 0$ of the considered tube at time $t_0 = 0$, D is the diffusion coefficient.

4.2.2 Formulation of testing problem

The testing problem is represented by a tube of length 5 m and volume 25 m³. The porosity is set to 0.2 by default, so the volume of water in tube is equal to 5 m³. The tube is divided into the set of n cells according to the selected length Δx of cell. We consider three types of approximation: 5 cells with $\Delta x = 1$ m, 25 cells with $\Delta x = 0.2$ m, and 125 cells with $\Delta x = 0.04$ m. Each type requires different time step Δt to meet the stability conditions, for $\Delta x = 1$ m it was set to 0.1 days, for $\Delta x = 0.2$ m it was set to 0.05 days, and for $\Delta x = 0.04$ m it was set to 0.01 days.

As an initial condition, the first cell contains water with the amount of chlorine equal to 1224 mg. Therefore, concentration value in the first cell depends on the selected volume of the cell. Other cells are filled with clear water, i.e., the water without dissolved substances. All solutions are defined in the `chlorine_5.phr`, `chlorine_25.phr`, `chlorine_125.phr`, and `water.phr` files.

Since the numerical solution is represented by the average value of concentration in a cell, we will compare it with the analytical solution at the center point of the cell. Before comparing these values, it is important to set the initial time of analytical solution, because at time $t = 0$, the analytical solution is represented by the unit impulse, which has to distribute for some time to correspond with the concentration value in the first cell.

TRM2D by default considers impermeable boundary, so the conservation laws within the grid are always met. However, the analytical solution is distributed over the whole x axis, so it was needed to choose an optimal time interval, which would allow to observe the spread of the solution before it will start to be significantly influenced by the concentration outflow over the boundary. We decided to

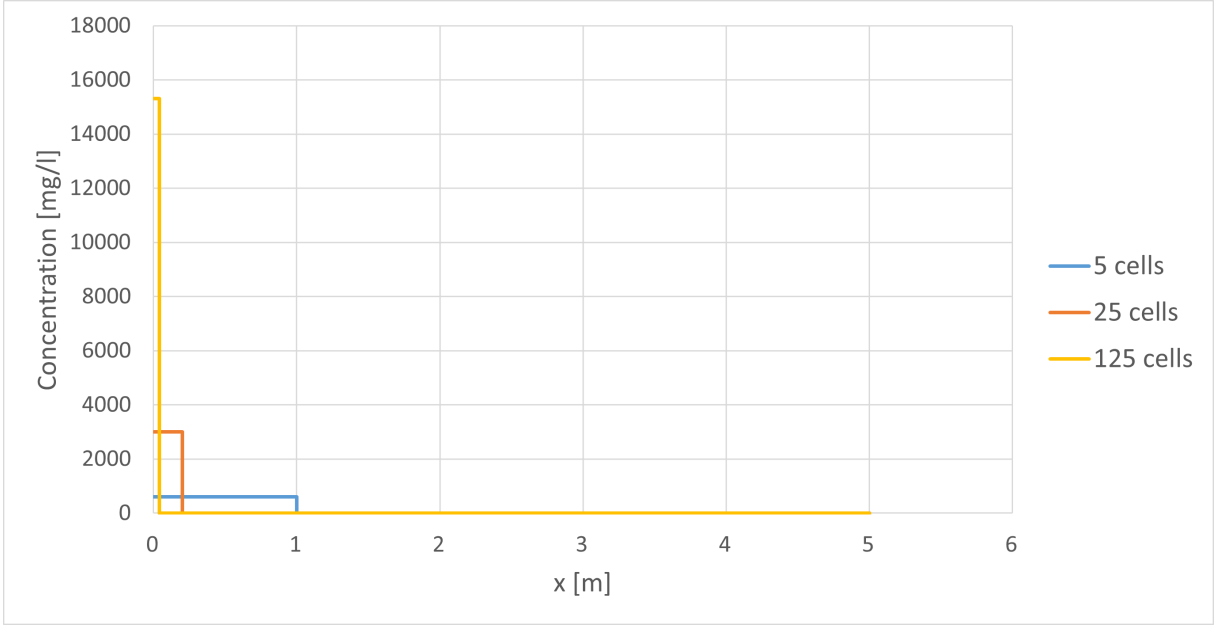


Figure 4.1: Comparison of initial conditions for 5, 25, 125 cells.

calculate the concentrations for the time interval $\langle 0, 10 \rangle$ and compare the values at the time $t = 10$ days.

The results for the chlorine concentrations are stored in the "Cl Results.csv" file in the "transport" directory. To test the calculations, for all 3 approximations we tried two different diffusion coefficients: 0.02 and 0.04.

4.2.3 Comparison of analytical and numerical solutions

When the calculations are finished, the results of each run are compared with the analytical solution. We decided to measure the difference between the analytical and numerical solutions using the l_2 -norm defined as:

$$\|\mathbf{c}_1 - \mathbf{c}_2\|_2 = \sqrt{\sum_{i=1}^N (c_{1,i} - c_{2,i})^2} \quad (4.8)$$

where \mathbf{c}_1 and \mathbf{c}_2 are the vectors of chlorine concentrations from the analytical and numerical solution respectively.

Now it is possible to collect and measure the results from the previous tests using l_2 -norm:

Δx [m]	ncells [1]	Δt [d]	l_2 [mg/l]
1	5	0.1	2,15e1
0.2	25	0.05	2,34e1
0.04	125	0.01	1,82e-1

Table 4.1: Basic numerical parameters and l_2 -norm of difference between numerical and analytical solution for $d_I = 0.02$.

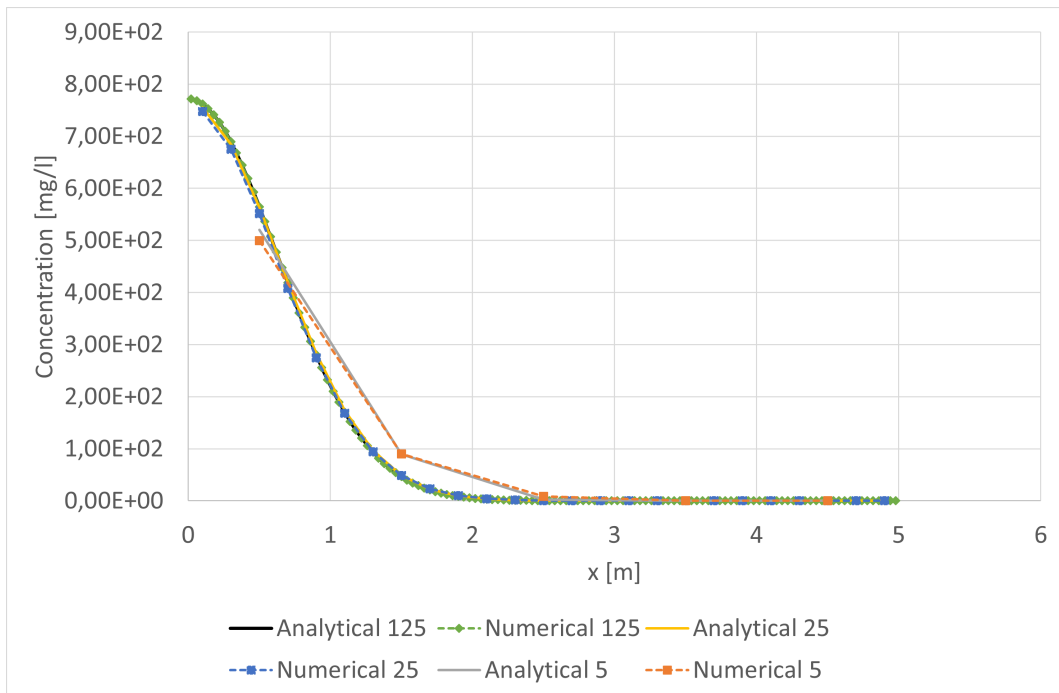


Figure 4.2: Comparison of numerical results for diffusion coefficient 0.02 using 5, 25, and 125 cells with the corresponding analytical solutions ($t = 10$ days).

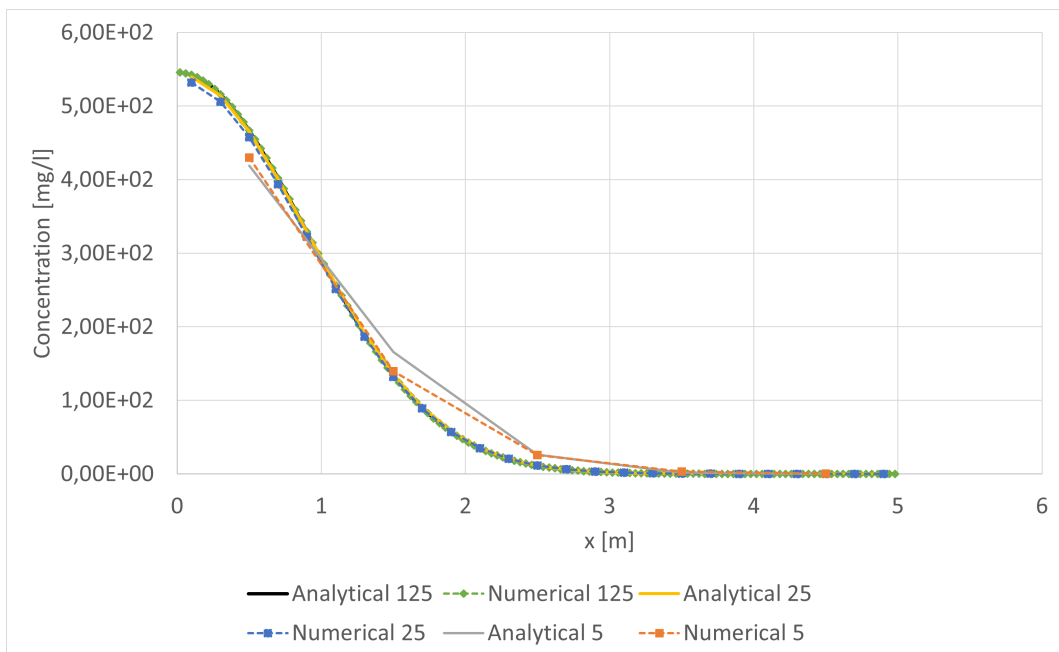


Figure 4.3: Comparison of numerical results for diffusion coefficient 0.04 using 5, 25, and 125 cells with the corresponding analytical solutions ($t = 10$ days).

Δx [m]	ncells [1]	Δt [d]	l_2 [mg/l]
1	5	0.1	2,83e1
0.2	25	0.05	1,93e1
0.04	125	0.01	1,26e-1

Table 4.2: Basic numerical parameters and l_2 -norm of difference between numerical and analytical solution for $d_I = 0.04$.

The results in the tables above express dependency between the accuracy of the results and the length of the Δx . For both diffusion coefficients, the l_2 -norm decreases with the refinement of the grid, i.e., the lower Δx is set, the more precise results are obtained.

Conclusion

The main goal of this bachelor project was to implement an alternative solution of the transport problem in the TRM2D software. For this purpose, the mathematical and physical model describing the movement of groundwater was studied. It includes the study of transport and reaction processes in the porous medium and solving of the transport-reactive problem using various numerical methods. Among the methods explained under this project were the Operator splitting method, the Finite difference method, and the Finite volume method. The most important method for this project was the FVM, because the calculation of transport in the TRM2D software is based on this method.

After study of theoretical ground, we became acquainted with the TRM2D software. We studied the structure, functionality, and main calculation methods. Working with software, we proposed an alternative method to calculate transport of the components considering the diffusion effect. The implementation of the new solution included the modification of existing methods and the creation of new ones. After the implementation of the new calculation method, all changes were listed and explained in this project. We tested the new functionality of the software by comparing the analytical solution with the numerical solution calculated by the software. As a result, we confirm that the software works correctly, and that the new solution corresponds to the analytical solution.

Bibliography

- [1] M. Hokr. *Transportní procesy* [in Czech]. Lecture notes. Faculty of Mechatronics, Technical University of Liberec, 2005.
- [2] L. Samoilov: *Operator splitting method for transport reactive problem solution*. Bachelor's degree project. FNSPE CTU in Prague, 2021.
- [3] A. Sokolenko: *Extension and testing of the algorithm for time step adaptation in the TRM2D software*. Bachelor's degree project. FNSPE CTU in Prague, 2022.
- [4] P. Jiránek: *Numerický model difúze pro model transportu látek ve spalovacím motoru*. Doctoral Thesis. Faculty of Mechatronics, Technical university of Liberec, 2003.
- [5] T. Havelka: *Modely difúze*. Bachelor's degree project. SCI MUNI in Brno, 2014.
- [6] P. Štrof et al.: *Final Report of the TACR project Nr. TH02030840* [in Czech]. DHI Prague, 2019.

Appendices

Appendix A

Original implementation of the `transport_module::calculate()` method

```
1 void transport_module::calculate(unsigned t_index, double time, vector<double>&
   concentrations, vector<string>& component_names, unsigned curr_bc_time, vector<
   double>& mapping)
2 {
3     if (has_state(DISABLED) || has_state(DISABLED_CALCULATION)) {
4         return;
5     }
6
7     unsigned col, row;
8     // initialization for t_index 1 is enough, in next steps content of the first
   column of transport_components_mat is kept
9     if (t_index == 1) {
10        for (row = 0; row < ncells; row++) {
11            transport_components_mat(row, 0) = get_component_data("
   TransportComponents", t_index - 1, "0")(row, 0);
12        }
13    }
14    unsigned c = 0;
15    for (col = 1; col < transport_components_mat.n_cols; col++) {
16        for (row = 0; row < ncells; row++) {
17            transport_components_mat(row, col) = concentrations[c];
18            c++;
19        }
20    }
21
22    // concentrations of in components taken from boundary conditions
23    arma::sp_mat in_comps(ncells, transport_components_mat.n_cols);
24    for (col = 0; col < transport_components_mat.n_cols; col++) {
25        ostringstream oss;
26        oss << col;
27        string col_str = oss.str();
28        arma::sp_mat comp_data = in_components[curr_bc_time][col_str].
   get_data_sparse();
29        for (unsigned row = 0; row < ncells; row++) {
30            in_comps(row, col) = comp_data(row, 0);
31        }
32    }
33    transport_components_mat += volume_diag_inverse * (in_out_flow_modif * in_comps
   * bc_times_steps[curr_bc_time] + grid_flow_modif * transport_components_mat *
```

```

bc_times_steps[curr_bc_time]);
34
// change mapping sent to the reaction module
35 mapping = arma::conv_to<vector<double>>::from(transport_components_mat.col(0));
36
37
// change concentration sent to the reaction module
38 c = 0;
39 for (col = 1; col < transport_components_mat.n_cols; col++) {
40     for (row = 0; row < ncells; row++) {
41         concentrations[c] = transport_components_mat(row, col);
42         c++;
43     }
44 }
45
46
if (output->is_step_for_output(t_index, time_end_id) && output->
47 get_cell_output_type() != "none") {
48     vector<string> tmp_component_names = component_names;
49     tmp_component_names.insert(tmp_component_names.begin(), 1,
get_first_component().get_name());
50
51     vector<string>::size_type ncomps = tmp_component_names.size();
52     for (vector<string>::size_type cn = 0; cn < ncomps; cn++) {
53         arma::mat tmp_comp(ncells, 1);
54         for (row = 0; row < ncells; row++) {
55             tmp_comp(row, 0) = transport_components_mat(row, cn);
56         }
57
58         component trans_comp;
59         trans_comp.copy_metadata(get_first_component());
60         ostringstream oss;
61         oss << cn;
62         trans_comp.set_id(oss.str());
63         trans_comp.set_name(tmp_component_names[cn]);
64         trans_comp.set_data(tmp_comp, true);
65         trans_comp.set_time(time);
66         add_component(trans_comp, "TransportComponents", t_index);
67     }
68 }
69 log(BASIC) << "Transport for time step ID '" << t_index << "' (time " << time <<
" ) calculated.";
70 }

```

Appendix B

Original version of configuration XML file

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <TRM xsi:schemaLocation="DHI.PASTRM C:\MyFiles\TACR\Epsilon02\TransportSim\PasTRM\
   Example5\PasTRMv10_02.xsd" xmlns="DHI.PASTRM" xmlns:xs="http://www.w3.org/2001/
   XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3
4 <LogFile filePrefix="Log\Log_" fileSuffix=".log" logDetails="
   BasicComponentMessages" />
5
6 <Quantities>
7   <None quantityID="0">1 </None>
8   <Time quantityID="1">day </Time>
9   <Length quantityID="2">m </Length>
10  <Volume quantityID="3">m3 </Volume>
11  <Flow quantityID="4">m3/d </Flow>
12  <Temperature quantityID="31">K </Temperature>
13  <Pressure quantityID="32">kPa </Pressure>
14  <Mass quantityID="33">kg </Mass>
15  <Concentration>
16    <Liquids quantityID="101">mg/l </Liquids>
17    <Phases quantityID="102">mg/kg </Phases>
18    <Gases quantityID="103">mol/l </Gases>
19  </Concentration>
20 </Quantities>
21
22 <TransportModule name="TRM" version="2020.01.13" type="console">
23   <Time quantityID="0">
24     <Init>0</Init>
25     <End>20</End>
26     <Step idTimeStep="0">1</Step>
27     <BorderCondChanges>
28       <Step idTimeStep="10">1</Step>
29     </BorderCondChanges>
30   </Time>
31   <GridSize>
32     <Nrows>40</Nrows>
33     <Ncols>1</Ncols>
34   </GridSize>
35   <GridElements>
36     <Volume typeID="denseVector" dataTypeID="double" Nrows="40" quantityID
   ="3">
37       <Data>5</Data>
38     </Volume>
```

```

39   <InOutFlow typeID="sparseVector" dataTypeID="double" Nrows="40" quantityID="4"
40   idTimeStep="0">
41     <IndexData>
42       1 -1
43       40 1
44     </IndexData>
45   </InOutFlow>
46   <InOutFlow typeID="sparseVector" dataTypeID="double" Nrows="40" quantityID="4"
47   idTimeStep="10">
48     <IndexData>
49       1 0.5
50       5 -1
51       40 0.5
52     </IndexData>
53   </InOutFlow>
54 </GridElements>
55 <GridElements2GridElements>
56 </GridElements2GridElements>
57 <InComponents idTimeStep="0">
58   <Component id="0" name="Pas2Phr" typeID="sparseMatrix" dataTypeID="double"
59   Nrows="40" quantityID="101">
60     <IndexData>
61       1 100
62     </IndexData>
63   </Component>
64 </InComponents>
65 <InComponents idTimeStep="10">
66   <Component id="0" name="Pas2Phr" typeID="sparseMatrix" dataTypeID="double"
67   Nrows="40" quantityID="101">
68     <IndexData>
69       5 50
70     </IndexData>
71   </Component>
72 </InComponents>
73 <TransportComponents idTimeStep="0" Time="0">
74   <Component id="0" name="Pas2Phr" typeID="denseMatrix" dataTypeID="double"
75   Nrows="40" quantityID="101">
76     <Data>
77       3
78     </Data>
79   </Component>
80 </TransportComponents>
81 </TransportModule>
82
83 <ReactionModule name="PhreeqCRM" version="3.3.11-12535" runReactionStep="1">
84   <Nthreads>8</Nthreads>
85   <PhreeqcDatabase directoryPath="inputs">
86     phreeqc.dat
87   </PhreeqcDatabase>
88   <OnInit writeOutput="true" errorHandlerMode="1" componentH2O="false"
89   rebalanceFraction="0.5" rebalanceByCell="true" useSolutionDensityVolume="false"
90   setPartitionUZSolids="false" unitsSolution="1" unitsPPassemblage="1"
91   unitsExchange="1" unitsSurface="1" unitsGasPhase="1" unitsSSassemblage="1"
92   unitsKinetics="1">
93     <InitCond mapValue="3" directoryPath="inputs">domain3.phr</InitCond>
94     <BorderCond mapValue="100" directoryPath="inputs" idTimeStep="0">bc_amd.phr</
95     BorderCond>

```

```

86     <BorderCond mapValue="50" directoryPath="inputs" idTimeStep="10">bc_domain2.
      phr</BorderCond>
87   </OnInit>
88   <OnActivation initFrom="MapComponent" borderCondChange="true">
89     <InitCond>
90       <MapComponent componentID="0"/>
91     </InitCond>
92     <BorderCond>
93       <IDTimeStep>0</IDTimeStep>
94       <IDTimeStep>10</IDTimeStep>
95     </BorderCond>
96   </OnActivation>
97   <RunCells timeConversion="1.1574e-5" representativeVolume="1" saturation="1"
      density="1" pressure="1" temperature="20">
98     <Run idTimeStep="0" time="0">
99       </Run>
100   </RunCells>
101   <PhreeqcOutputs>
102     <RunOutputs directoryPath="Results" prefixFilename="pichem_amd_1D"
      writeCells="all" writeSteps="all">
103       <WriteCell id="0"/>
104       <WriteStep id="0"/>
105     </RunOutputs>
106     <DumpFile></DumpFile>
107   </PhreeqcOutputs>
108 </ReactionModule>
109
110 <ResultFile directoryPath="Results" filename="results.xml" useCsvSubdirs="true">
111   <Transport writeCells="all" writeSteps="all">
112     <LoopStep>2</LoopStep>
113   </Transport>
114   <Reaction writeCells="all" writeSteps="all">
115     <LoopStep>1</LoopStep>
116   </Reaction>
117 </ResultFile>
118
119 </TRM>

```