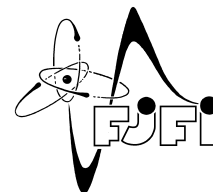




CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering



Extension and testing of the algorithm for time step adaptation in the TRM2D software

Rozšíření a otestování algoritmu pro adaptaci časového kroku v softwaru TRM2D

Bachelor's Degree Project

Author: **Anastasiia Sokolenko**
Supervisor: **doc. Ing. Jan Šembera, Ph.D.**
Language advisor: **M.A. Kevin Patrick Joseph Glanville**
Academic year: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Anastasiia Sokolenko
Studijní program:	Aplikace přírodních věd
Studijní obor:	Aplikovaná informatika
Název práce (česky):	Rozšíření a otestování algoritmu pro adaptaci časového kroku v softwaru TRM2D
Název práce (anglicky):	Extension and testing of the algorithm for time step adaptation in the TRM2D software

Pokyny pro vypracování:

- 1) Převezměte software TRM2D vzniklý v rámci projektu TH02030840 „Paralelizovaný reakčně-transportní model šíření kontaminace v podzemních vodách (PaReTran)“ a upravený v rámci řešení bakalářské práce Leonida Samoilova.
- 2) Prostudujte vybranou literaturu o metodě štěpení operátoru a adaptivním časovém kroku.
- 3) Proveďte úpravu uživatelského rozhraní tak, aby umožňovalo vstup parametrů pro algoritmus adaptace časového kroku.
- 4) Zformulujte vhodnou testovací úlohu a vypočtěte její referenční řešení. Proveďte srovnávací výpočty pro různé parametry algoritmu pro adaptaci časového kroku a jejich výsledky interpretujte.

Doporučená literatura:

- 1) L. Samoilov, Operator splitting method for transport reactive problem solution. Bakalářská práce FJFI ČVUT, Praha, 2021.
- 2) P. Štrof, et al., Final Report of the TACR project Nr. TH02030840 [in Czech]. DHI Praha, 2019.
- 3) H. Holden, K. H. Karlsen, K.-A. Lie, N. H. Risebro, Splitting Methods for Partial Differential Equations With Rough Solutions. Analysis and Matlab Programs (EMS Series of Lectures in Mathematics), European Mathematical Society, Curych, 2010.

Jméno a pracoviště vedoucího bakalářské práce:

doc. Ing. Jan Šembera, Ph.D.

Technická univerzita v Liberci, Fakulta mechatroniky, informatiky a mezioborových studií,
Ústav NTI (budova A), Hálkova 6, 461 17 Liberec

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2021

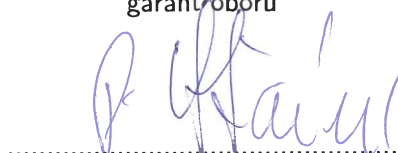
Datum odevzdání bakalářské práce: 7.7.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 21. října 2021

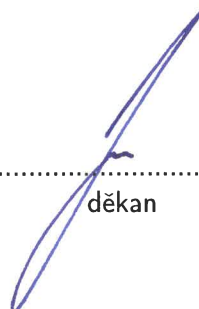


garant oboru



vedoucí katedry





děkan

Acknowledgment:

I would like to thank doc. Ing. Jan Šembera, Ph.D. for his expert guidance and express my gratitude to M.A. Kevin Patrick Joseph Glanville for his language assistance.

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, July 7, 2022

Anastasiia Sokolenko

Název práce:

Rozšíření a otestování algoritmu pro adaptaci časového kroku v softwaru TRM2D

Autor: Anastasiia Sokolenko

Obor: Aplikovaná informatika

Druh práce: Bakalářská práce

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

Abstrakt: Tato bakalářská práce zkoumá implementaci modelu popisujícího šíření kontaminace v podzemních vodách jako transportně-reakční úlohu ve formě soustavy PDR řešených pomocí metody štěpení operátoru v softwaru TRM2D, který vznikl v rámci projektu TH02030840 "PaReTran". Protože metoda štěpení operátoru v daném softwaru byla rozšířena metodou adaptivního časového kroku, hlavním cílem této práce bylo vylepšení dané metody, její otestování s různým nastavením parametrů a analýza výsledků. V rámci řešení bakalářské práce byla nalezena kombinace hodnot parametrů umožňující zachovat přesnost výpočtů s konstantním krokem a zároveň snížit počet iterací, což nyní dovoluje uživatelům dostávat poměrně přesné výsledky rychleji než při použití výpočtu s konstantním krokem.

Klíčová slova: adaptivní časový krok, metoda štěpení operátoru, transportně-reakční úloha

Title:

Extension and testing of the algorithm for time step adaptation in the TRM2D software

Author: Anastasiia Sokolenko

Abstract: This bachelor project studies the implementation of the model describing the contaminated groundwater movement as the reactive-transport problem in a form of the PDE system, solved by the operator-splitting method in the TRM2D software, developed under the project TH02030840 "PaReTran". As the operator-splitting method was later extended in this software by the adaptive time step control method, the main goal of this project lies in improving this method, testing it with different settings and analyzing the results. Eventually, under this project a combination of the parameter values was found that made it possible to keep the accuracy of the calculations with a constant time step and lower the number of iterations, which now allows the user to obtain relatively accurate results faster than with a constant time step.

Key words: adaptive time step, operator-splitting method, reactive-transport problem

Contents

Introduction	11
1 Physico-mathematical description of the groundwater flow model	13
1.1 Continuum model of porous medium	13
1.2 Transport processes in the groundwater model	14
1.2.1 Darcy's experiment, law and microscopic velocity	14
1.2.2 Mass Balance equation and transport processes	16
1.3 Mathematical formulation of the reaction part of the transport-reaction model on an example of calcite-dissolution processes	18
1.3.1 DAE formulation of the problem	18
1.3.2 ODE form of the problem	20
2 Methods used for solving the reactive-transport problem	23
2.1 Operator splitting method	23
2.1.1 Main principles and reasons for utilizing the OSM	23
2.1.2 Common types of OSM	24
2.1.3 Application of the OSM to the transport-reaction problem	26
2.1.4 Evaluation of the local error of the OSM	27
2.2 Methods used for solving the reaction part of the reactive-transport problem	28
2.2.1 The Euler method	28
2.2.2 Evaluation of the total error of the Euler method	28
2.2.3 The adaptive time step method's algorithm	30
3 TRM2D software	31
3.1 General description of the software	31
3.1.1 Reading input data	31
3.1.2 Application of the transport and reaction processes onto the input data	32
3.1.3 Implementation of the adaptive time step method	32
3.2 Adaptation of the software as a part of this bachelor project	33
3.2.1 Reading the parameters from an input file	33
3.2.2 Inputting the name of the log file for the adaptive time step	34
3.2.3 Changes in the "calculate" method	36
4 Testing of the extended TRM2D software	37
4.1 Verifying the consistency of the results between the newly altered version of the software and the previous one	37
4.2 Formulation of the testing problem	38

4.3 Optimizing the parameters	39
Conclusion	43
Bibliography	45
Apendices	45
A Original implementation of the "calculate(...)" method	47
B Original version of an input XML file	51
C TRM2D structure description	55

Introduction

In such a case where chemical reactions and dynamic processes influence the state of a system, it is typically difficult to predict the course of events happening in it, because such systems are unstable. Moreover, given the fact that there can be a change in boundary conditions with a rapid increase in concentration of chemical compounds, one has to deal with a stiff problem, where one component of a system is varying rapidly while the other one – slowly. This makes the calculation especially complicated. In this bachelor's degree project such a problem is regarded, the reactive-transport problem.

To solve it, the operator-splitting method (OSM) can be applied. It allows us to divide a complex problem into a number of simpler tasks, in other words, to solve a sequence of sub-models. There are several algorithms that enable this method. One of them is called Lie-Trotter splitting and it consists in calculating the result for the first part and using it as the initial value for the second part. It is the algorithm chosen to be implemented in the software called TRM2D, which we intend to extend under this bachelor project.

To further improve the software, a time step control method was implemented by L. Samoilov in the original software, which previously only used a constant time step. This could help better control the calculation errors. The essence of this method lies in comparing the current state of the system with the state in the previous time step. If the state of the system changes too quickly, the method shortens the time step and lengthens it in the opposite case. However, the software did not allow the user to choose the accuracy of the comparison and the degree to which the time step can be shortened or extended. Our aim in this bachelor project is to adapt the code so that the user could easily set these parameters and later test the performance of the implemented method.

The study is divided into four main chapters: a theoretical description of the reactive-transport problem (1), explanation of the methods used to solve this problem (2), description of the TRM2D software and its adaptation under this project (3) and the testing of the extended version of the TRM2D software (4).

Chapter 1

Physico-mathematical description of the groundwater flow model

This chapter serves as the theoretical part of this project and it provides the characterization of the mathematical model describing the movement of the groundwater through a porous medium, laws and transport processes taking place in it, and the mathematical formulation of chemical reactions influencing the concentration change in the model.

1.1 Continuum model of porous medium

Porous materials consist of a matrix (also known as the skeletal portion), consisting of fibers or granules, and pores between them, filled with liquid or gas. The matrix can form a continuous capillary network or it might contain pores that are not connected to the network. Furthermore, the network is usually extremely complex. Taking all of the above into consideration, it would be unnecessarily complicated to simulate the whole structure of the system to create a mathematical model of it. This is why it is important to introduce the terms porosity and permeability, which describe a porous medium, and a continuum model.

Porosity is the ratio of the volume of pores (not including the pores separated from the capillary network) to the total volume of a piece of porous material. It is usually expressed as a percentage and symbolized as n . It can be measured by filling the material with liquid of a certain volume to the point when it is not possible to absorb more of it. Therefore, essentially it describes how much liquid or gas the material can hold.

Permeability describes how easy it is for the fluid to pass through the material. It depends on the interconnectivity between pores, grain size, the diameter of capillaries, called pore throat, etc. Permeability is measured in squared meters and is commonly symbolized as k .

A continuum model is an approximation of a structure as continuous. It means that the fact that there are places in the material where concentration, density, pressure, or other properties of a fluid cannot be measured is neglected to enable the differentiation.

Dealing with the continuum model, it is necessary to introduce the term representative elementary volume (REV). REV is the smallest unit of a system large enough to correctly represent the system with its properties remaining consistent. In this example REV is a unit of a porous material that has the same porosity and permeability. Its typical dimensions can be several cm^3 to several m^3 .

1.2 Transport processes in the groundwater model

This section is based on Milan Hokr's university study book [2] and it provides the basic description of Darcy's law and the transport processes, influencing the concentration of chemical compounds dissolved in the groundwater.

1.2.1 Darcy's experiment, law and microscopic velocity

Darcy's law was deduced from the experiment conducted by Henry Darcy. In this experiment two containers known as manometers were connected by an inclined tube filled with sand or any other porous filter.

In the experiment Henry Darcy was trying to propose the law describing the dependence of the total discharge on other variables. Total discharge determines how fast the volume unit of fluid will move over a time unit and it is measured in cubic meters per second. As a result of the experiment, the total discharge was found to be defined by the following equation:

$$Q = K \cdot \frac{S \cdot (h_2 - h_1)}{L}, \quad (1.1)$$

where K is hydraulic conductivity (includes permeability and properties of fluid, measured in meters per second – m/s), S is the cross-sectional area (m^2) of the tube, L is the length of the tube (m) and h_2 and h_1 are the hydraulic heads of the two manometers.

Hydraulic head (or piezometric head) is the height that water rises in manometers relative to some arbitrary level (sea-level in experiments in natural environments, a bench top in laboratory experiments).

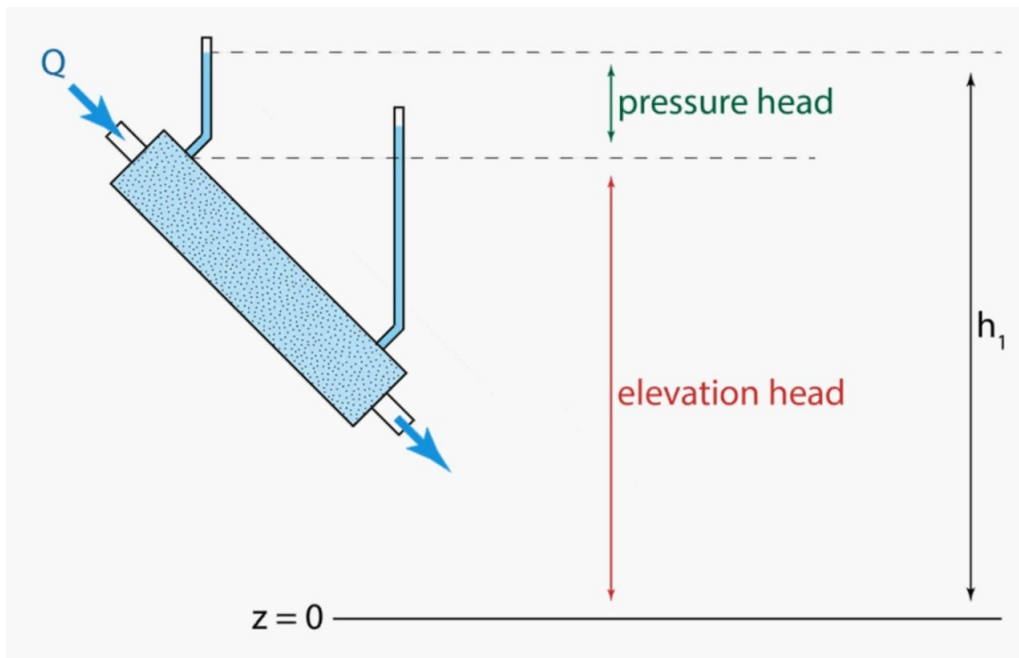


Figure 1.1: Darcy's experiment, h_1 – hydraulic head for the upper manometer, z – arbitrary level. (Adopted from [9])

Essentially a hydraulic head is the pressure expressed as a height of the water level and measured in meters. It represents a moving force in the experiment, as the fluid flows from a place with higher energy

due to higher pressure, as in the law of the communicating vessels where the water levels are known to be the same in both manometers. Hydraulic head is defined by the formula:

$$h = h_p + h_z, \quad (1.2)$$

where h_p is pressure head and h_z is elevation head, all measured in meters. The pressure head represents the hydraulic pressure of water above a vertical level, and the elevation head represents gravitational energy. Pressure head is defined by the following equation:

$$h_p = \frac{p}{\rho g}, \quad (1.3)$$

where p is fluid pressure (Pa), ρ is the density of the fluid (kg/m^3) and g is acceleration due to gravity (m/s^2). Elevation head is defined as the difference between the reference level z and the level where the manometer is attached to the tube with a porous material. Therefore, the hydraulic head formula can be rewritten as follows:

$$h = z + \frac{p}{\rho g}. \quad (1.4)$$

In practice, Darcy's law is used to determine the direction of groundwater flow and to evaluate flow rates according to the hydraulic heads in the "manometers" inserted into the ground.

For describing the movement of water in a specific place inside the porous material it is firstly needed to define the term flow density as the flow Q per unit cross sectional area of the porous medium (m/s). It is defined by the formula:

$$q = \frac{Q}{S}. \quad (1.5)$$

It can be written in another form using Darcy's law:

$$q = K \cdot \frac{\Delta h}{L}. \quad (1.6)$$

Then its limit in the longitudinal direction is considered as follows:

$$K \lim_{L \rightarrow 0} \frac{\Delta h}{L} = K \lim_{L \rightarrow 0} \frac{h(x) - h(x+L)}{L} = -\frac{dh}{dx}. \quad (1.7)$$

This way the Darcy velocity in 3D space can be deduced as follows:

$$q = -K \cdot \nabla h. \quad (1.8)$$

The Darcy velocity in 3D is a function of spatial coordinates and time. Therefore, the gradient of a hydraulic head is represented as follows:

$$\nabla h = \left(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, \frac{\partial h}{\partial z} \right). \quad (1.9)$$

And the hydraulic conductivity quotient in anisotropic material is denoted by the tensor of second order as follows:

$$K = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix}. \quad (1.10)$$

From the Darcy velocity the velocity of a particle of water or of a dissolved substance along the water flow inside the porous material can be derived.

For this first the time that the volume of liquid would take to move through a cross-sectional area of a tube is considered. The total discharge is defined by the following formula:

$$Q = \frac{V}{t}. \quad (1.11)$$

In this equation V is found by the following formula:

$$V = nSL, \quad (1.12)$$

where (SL) is equal to the volume of the area of the tube that the fluid moved through, and n is the porosity of a tube. Therefore, t from the equation (1.11) can be expressed using the equation (1.5) defining Q as follows:

$$t = \frac{nSL}{qS} = \frac{nL}{q}. \quad (1.13)$$

From this the following equation can be derived as follows:

$$v = \frac{L}{t} = \frac{q}{n}. \quad (1.14)$$

This means that the fluid particle will move along L with time t with velocity v , hence, the microscopic velocity of particles is faster than Darcy's velocity.

1.2.2 Mass Balance equation and transport processes

Mass balance equation is a formula describing the mass change of a certain chemical component in a system. Let us first consider the equation for the mass change of water written as follows:

$$\frac{\partial}{\partial t} \int_V \rho n \, dV = - \int_{\partial V} \rho q \, dS + \int_V (P\rho) \, dV. \quad (1.15)$$

In the first term of the equation, the change in mass over time is registered, where n is the porosity of a medium and ρ is the density of water.

The first term on the right-hand side of the equation describes the amount of water that crossed the boundary under the influence of transport processes, where q is the flow velocity.

The second term on the right-hand side of the equation describes the movement of water through the sink and source points. Practically, these are the points where water flows in or out through tubes from the outside of the system. Here P is the density of sources and sinks and it registers how fast the volume of the water flow enters or leaves the REV of the system and is measured in $(\text{m}^3/(\text{m}^3 \cdot \text{s}))$. It is denoted by the following formula:

$$P = \frac{Q}{V}, \quad (1.16)$$

where Q is the total discharge of the flow and V is the volume of a REV.

Similarly, the equation for the mass change of a chemical compound dissolved in the water can be indicated. It can be written as follows:

$$\frac{\partial}{\partial t} \int_V nc \, dV = - \int_{\partial V} q_c \, dS + \int_V (P^+ c^* + P^- c) \, dV + \int_V r \, dV. \quad (1.17)$$

If the Gauss divergence theorem is applied to the first term on the right-hand side of the equation and the integrals are cancelled, we get the following formula:

$$\frac{\partial c}{\partial t} = -\nabla q_c + \frac{1}{n}(P^+ c^* + P^- c) + \frac{r}{n}. \quad (1.18)$$

As can be seen, the difference between the (1.15) and (1.17) equations is that the density is replaced with the concentration of a chemical compound in a solution, in this case measured in mass of a solute over the solution's volume.

Other than that, the second term on the right-hand side of the equation is written slightly differently. Because the concentration of the solution that enters the system from the outside is different than the concentration inside the system, in the equation it is denoted by c^* . Here P^+ is the source term and P^- is the sink term.

Most importantly, there is a new term that was not present in the equation for water mass change, and it describes the change in concentration that appeared because of the chemical reactions. Chemical compounds can interact, and for this reason, they can be generated and disintegrated.

Moreover, in the first term of the right-hand side of the equation there is a mass flow denoted by q_c . It is essentially the velocity of the mass movement. This movement is caused by transport processes and is defined by the following equation:

$$q_c = n(cv + D_h \nabla c). \quad (1.19)$$

One of the transport processes included in the equation is advection. It is the transfer of a solute by the flow of a solvent. The second part of the equation represents the dispersion. D_h in this term of the equation represents the hydrodynamic dispersion tensor. It is measured in m^2/s and defined as follows:

$$D = D_m + D_f, \quad (1.20)$$

where D_m is the molecular diffusion tensor and D_f is the mechanical dispersion tensor.

Molecular diffusion is caused by the thermal motion of the molecules in the fluid, where the solvent molecules move from the areas with higher to lower concentration. It is found using the following formula:

$$D_m = D_m T, \quad (1.21)$$

where T is the tortuosity tensor, which describes the tortuosity of the canals in the medium, and D_m is the mass diffusivity, which depends on the temperature, the chemical compound in question, the size of the molecules and other factors. In a continuous fluid it is the same in all direction, but when it is multiplied by the tortuosity tensor, the velocity of the movement of molecules is different in all directions.

Mechanical dispersion, on the other hand, is caused by the difference in velocities of flow in different directions, because of the complex structure of canals in a porous medium, namely, by the interconnectivity and tortuosity of canals.

Mechanical dispersion is determined by two dispersion coefficients: longitudinal (denoted by α_L) and transverse (denoted by α_T). The transverse dispersion coefficient is usually smaller than the transverse.

Fick's law states that the velocity of molecular motion can be found by the following formula:

$$\omega = \frac{S D(c_2 - c_1)}{x}, \quad (1.22)$$

where ω is velocity measured in moles per second, S is the cross-sectional area, D is the dispersion coefficient, $(c_2 - c_1)$ is the difference in concentration between the areas of where it is going from and

to and x is the distance between the areas. If we divide both sides of the equation by S and if x tends to zero, the equation can be rewritten as follows:

$$\frac{\partial c}{\partial t} = D_m \nabla c, \quad (1.23)$$

where the left side of the equation represents the diffusion flux and ∇c is the divergence of concentration in three-dimensional space.

There are other processes that take place in the concentration change in a porous medium, such as sorption, radioactive decay, processes caused by the double porosity, etc. But it would require a much more extensive and profound description, than offered by the scope of this project.

1.3 Mathematical formulation of the reaction part of the transport-reaction model on an example of calcite-dissolution processes

The TRM2D software, described in more detail in P. Štrof's report [4], uses the PhreeqcRM library that contains the methods for transformations of geochemical reactions into the mathematical form (more on the PhreeqcRM library can be found on the USGS team's web page [5]). The methods are very complex and to demonstrate it, the following section, based on Lukáš Zedek's dissertation [6], will illustrate how difficult it is to transform only three chemical equations of calcite dissolution into the differential-algebraic and ordinary differential equations.

1.3.1 DAE formulation of the problem

The chemical equations regarded in our project are as follows:

1. $\text{CO}_2(\text{aq}) + \text{H}_2\text{O} \rightleftharpoons \text{H}^+ + \text{HCO}_3^-$,
2. $\text{CaCO}_3 + \text{H}^+ \rightleftharpoons \text{Ca}^{2+} + \text{HCO}_3^-$,
3. $\text{H}_2\text{O} \rightleftharpoons \text{H}^+ + \text{OH}^-$.

Mathematical description of chemical reactions can bring nonlinear terms into transport-reaction models. Because of that a system of nonlinear algebraic equations for each time step must be solved. To make the calculations less time-consuming, it is necessary to minimize the number of unknown variables. One of the methods to do so is to introduce a term known as the extent of a reaction. It is defined by the following equation:

$$\xi = \frac{(n_{\text{current}} - n_{\text{initial}})_i}{v_i}, \quad (1.24)$$

where n_i is the number of moles and v_i is the stoichiometric number of the i -th compound of a reaction. Stoichiometric number is the number before each compound in a chemical equation.

In Table [1.1] the initial concentrations of each chemical compound at equilibrium are represented as already known initial values, and the concentrations for the next time step are expressed using the extents of ξ_1, ξ_2, ξ_3 for each of the three chemical reactions, where the extents are the unknown variables.

	$t_0 = 0$	$t = t_0 + t_1, t_1 > 0$
$c_1, [\text{CO}_2(aq)]$	c_1^0	$c_1^0 - \xi_1$
$c_2, [\text{H}^+]$	c_2^0	$c_2^0 + \xi_1 - \xi_2 + \xi_3$
$c_3, [\text{HCO}_3^-]$	c_3^0	$c_3^0 + \xi_1 + \xi_2$
$c_4, [\text{Ca}^{2+}]$	c_4^0	$c_4^0 + \xi_2$
$c_5, [\text{OH}^-]$	c_5^0	$c_5^0 - \xi_2$

Table 1.1: Table of the concentration change with time.

It is important to notice that monitoring of the concentrations of H_2O and CaCO_3 would not be necessary because there is a large amount of them in the system, since water is a solvent of the solution and CaCO_3 is the material, which the porous medium consists of, and there will always be much more of them than any other compound.

The first and the third chemical reactions are fast and describe the simultaneous conversions of reactants to products and of product to reactants resulting in the balance of chemical compounds. The two reactions can be described by the following equations using the equilibrium constants K and M for each chemical reaction as the ratios between the concentrations of products divided by the concentrations of reactants at a state of chemical equilibrium:

$$K = \frac{c_2 c_3}{c_1}, \quad (1.25)$$

$$M = c_2 c_5. \quad (1.26)$$

The second chemical reaction is a kinetic reaction, which means that it proceeds in both directions, but the ratio between its products and reactants changes slowly. Therefore, the second reaction can be described by the following equation using the equilibrium constant L and the reaction rate constant l , which defines the direction and rate of a chemical reaction:

$$L = \frac{c_3 c_4}{c_2}, \quad (1.27)$$

$$\frac{d\xi_2}{dt} = l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L} \right). \quad (1.28)$$

The data from Table [1.1](#) can be plugged into these equations resulting in the following equations:

$$K = \frac{(c_2^0 + \xi_1 - \xi_2 + \xi_3)(c_3^0 + \xi_1 + \xi_2)}{(c_1^0 - \xi_1)},$$

$$\frac{d\xi_2}{dt} = l \cdot \left(1 - \frac{(c_3^0 + \xi_1 + \xi_2)(c_4^0 + \xi_2)}{(c_2^0 + \xi_1 - \xi_2 + \xi_3)} \cdot \frac{1}{L} \right), \quad (1.29)$$

$$M = (c_2^0 + \xi_1 - \xi_2 + \xi_3)(c_5^0 - \xi_2).$$

The disadvantage of this model is in the extents of the reactions, because it would be very difficult to combine it with the transport part of the system, since it uses concentrations of chemical compounds as unknown variables instead of the reaction extents. To solve this problem, the model was altered into a more suitable form.

1.3.2 ODE form of the problem

Another way to describe the system of chemical reactions mathematically is as a system of equations expressing the change of concentrations of the chemical compounds with time, using the equation (1.28) and the terms R_k and R_m qualifying the contribution of the first and the third chemical reactions to the concentration change of their reactants and products and found by the following formulae:

$$\begin{aligned} R_k &= \frac{d\xi_1}{dt}, \\ R_m &= \frac{d\xi_3}{dt}. \end{aligned} \tag{1.30}$$

The system of equations describing the chemical reactions mathematically can be written as follows:

$$\begin{aligned} \frac{dc_1}{dt} &= (-1) \cdot R_k, \\ \frac{dc_2}{dt} &= (-1) \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right) + R_k + R_m, \\ \frac{dc_3}{dt} &= l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right) + R_k, \\ \frac{dc_4}{dt} &= l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right), \\ \frac{dc_5}{dt} &= R_m. \end{aligned} \tag{1.31}$$

Because it is required to find the unknown functions R_k and R_m of the system, both sides of the equilibrium equations (1.25) and (1.26) can be differentiated as follows:

$$\frac{dK}{dt} = \left(\frac{dc_2}{dt} \cdot c_3 c_1 + \frac{dc_3}{dt} \cdot c_2 c_1 - \frac{dc_1}{dt} \cdot c_2 c_3 \right) : c_1^2 = 0, \tag{1.32}$$

$$\frac{dM}{dt} = \frac{dc_2}{dt} \cdot c_5 + \frac{dc_5}{dt} \cdot c_2 = 0. \tag{1.33}$$

And if we express the R_k and R_m from these two equations and plug the equations from the system of equations (1.31) into them, we can rewrite the formulae defining R_k and R_m as follows:

$$R_m = \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}, \tag{1.34}$$

$$R_k = -\frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}. \tag{1.35}$$

It is now finally possible to combine the reaction and the transport parts of the groundwater flow equations, and the system of the transport-reaction equations can be written as follows:

$$\begin{aligned}
\frac{\partial c_1}{\partial t} &= -\nabla(c_1 v) + \nabla(D_h \nabla c_1) + \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}, \\
\frac{\partial c_2}{\partial t} &= -\nabla(c_2 v) + \nabla(D_h \nabla c_2) - \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right) - \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)} + \\
&\quad + \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}, \\
\frac{\partial c_3}{\partial t} &= -\nabla(c_3 v) + \nabla(D_h \nabla c_3) + \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right) - \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}, \\
\frac{\partial c_4}{\partial t} &= -\nabla(c_4 v) + \nabla(D_h \nabla c_4) + \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right), \\
\frac{\partial c_5}{\partial t} &= -\nabla(c_5 v) + \nabla(D_h \nabla c_5) + \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3 c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3) \cdot c_2 + c_1 c_3 + c_5 \cdot (c_1 + c_3)}.
\end{aligned} \tag{1.36}$$

Seeing the number of steps and the complexity of the equations representing the system of only three chemical reactions to be able to formulate them mathematically, it becomes obvious that the amount of work that the PhreeqcRM library performs for all the geochemical processes happening in the groundwater flow makes it very profitable to use.

Chapter 2

Methods used for solving the reactive-transport problem

This chapter's purpose is to list and explain the main problems of the model implementation, calculation of the results and ways to solve these problems. It includes the description and application of the operator-splitting method and the Euler method for the reaction part, and the errors brought by these numerical methods.

2.1 Operator splitting method

This section is based on the STIMULATE European Joint Doctorates webpage's article [1] and H. Holden's series of Lectures [8].

2.1.1 Main principles and reasons for utilizing the OSM

In mathematical models the operator-splitting method (OSM) can reduce memory requirements and increase the stability range. It may also allow users to easily add increasing complexity to a model and be the only possible solution for very high dimensional problems.

The OSM is a way of dealing with a complex problem by using the divide-and-conquer approach. This method is used when sub-problems of a system differ physically and, as a result, mathematically.

For example, when the convection-diffusion equation is considered, it is natural to apply OSM, because diffusion and convection are physically completely different processes. Therefore, if the same numerical methods are used for both parts of the equation, numerical oscillation and diffusion will appear.

In another example, the system consisting of the processes of one-way wave distribution and of heat distribution are considered. If they are solved separately, using the forward Euler numerical method for the wave equation, the results would be unstable, but the leap-frog integration method would show stable results. In contrast, it would be the opposite for the heat equation. If it was needed to solve an equation consisting of both of these parts, the stable results would not be achieved using any of these numerical methods. But if the OSM is used, an appropriate technique for each part of the equation can be easily applied.

The OSM's main idea lies in writing the overall evolution operator as a sum of evolution operators for each process in the system. This way the equation is divided into a set of sub-equations, each sub-equation is solved separately with a suitable numerical method and then pieced back together in a way determined by a specific type of the OSM. The strategy for piecing them back together generally consists

in the discretization of time and calculating the results for each part of the equation sequentially on sub-intervals, using the results of one equation in the other one as an initial value.

Different types of OSM specify, in what order the sub-equations will be solved and how the problem will be split. Now two of the many types of OSMs will be introduced to better understand the general differences between the types of OSMs.

2.1.2 Common types of OSM

There are several types of OSM, but the three most simplest types are Lie-Trotter splitting method, Strang splitting method and Additive splitting, and to better understand the principle of OSM methods it is appropriate to explain it using the simpler ones. The types will be explained on one of the simplest mixed-type problems, i.e. initial value Cauchy problem described by the following equation:

$$\frac{\partial U(t)}{\partial t} = AU(t) + BU(t), \quad (2.1)$$

with $t \in [0, T]$, $U(0) = U_0$.

2.1.2.1 Lie-Trotter splitting method

Lie-Trotter splitting is a first order splitting method where the first sub-equation is solved first using the result from the previous time step as an initial value, then the second sub-equation with the result from the first sub-equation as an initial value is solved.

The initial value Cauchy problem is split with the Lie-Trotter method into two sub-equations as follows:

$$\begin{aligned} \frac{\partial u(t)}{\partial t} &= Au(t), \\ \frac{\partial v(t)}{\partial t} &= Bv(t), \end{aligned} \quad (2.2)$$

where $t \in [t^n, t^{n+1}]$ and $t^{n+1} = t^n + \Delta t$.

To find a primitive function of such equations, the method of separation of variables can be used. Hence, we get the following equations of an exponential form:

$$\begin{aligned} u(t) &= e^{\Delta t A} u(t), \\ v(t) &= e^{\Delta t B} v(t). \end{aligned} \quad (2.3)$$

The algorithm for the Lie-Trotter splitting method using the equations (2.3) is defined as follows:

1. $u(t_n + \Delta t) = e^{\Delta t A} u(t_n)$ with $u(t_n) = u_0(t_n)$,
2. $v(t_n + \Delta t) = e^{\Delta t B} v(t_n)$ with $v(t_n) = u(t_n + \Delta t)$,
3. $u_0(t_n + \Delta t) = v(t_n + \Delta t)$,
4. if $T < (n + 1)\Delta t$, go to step 1, otherwise stop.

The algorithm in words can be written as follows:

1. After splitting the equation into two parts with the A and B operators denoted by $u(t)$ and $v(t)$ respectively, solve the first equation for a full time step with an initial value of $u_0(t_n)$ for the first time step or the results from the previous time steps for others.
2. Solve the second equation using the result of the first step as an initial value for the full time step.
3. Repeat the steps 1-2, but using the result of the second equation as an initial value for the first equation.
4. Repeat the algorithm until the end of the given time interval with each iteration corresponding to one time step.

2.1.2.2 Strang splitting method

Strang splitting method (also known as the leap-frog method) is a more advanced and more popular OSM than the previous one. In the Lie-Trotter method there appears to be a problem of which sub-equation will be chosen first. In the Strang splitting method this problem is minimized by splitting the calculation in the middle of a time step. The calculation starts with the first sub-equation for the first half of a time step, then the second one is solved for a full time step and only after that the calculation of the first sub-problem is continued for the second half of a time step. This way both sub-problems are solved more symmetrically. This is why the Strang splitting method is considered more accurate as it is a second order splitting method.

The initial value Cauchy problem is split with Lie-Trotter method into two sub-equations as follows:

$$\begin{aligned}
\frac{\partial u(t)}{\partial t} &= Au(t) \text{ with } t \in [t^n, t^{n+1/2}] \text{ and } u(t^n) = u_0^n, \\
\frac{\partial v(t)}{\partial t} &= Bv(t) \text{ with } t \in [t^n, t^{n+1}] \text{ and } v(t^n) = u(t^{n+1/2}), \\
\frac{\partial \omega(t)}{\partial t} &= B\omega(t) \text{ with } t \in [t^{n+1/2}, t^{n+1}] \text{ and } \omega(t^{n+1/2}) = v(t^{n+1}).
\end{aligned} \tag{2.4}$$

The strategy of the Strang splitting method lies in solving the first sub-equation for the first half of the time step, then solving the second sub-equation for the full-time step with the previous results as an initial value and then turning back to the first sub-equation, solving it with a result from the second sub-equation as an initial condition. The algorithm for it is defined using the equations (2.4) as follows:

1. $u(t_n + \Delta t/2) = e^{(\Delta t/2)A} u(t_n)$ with $u(t_n) = u_0(t_n)$,
2. $v(t_n + \Delta t) = e^{\Delta t B} v(t_n)$ with $v(t_n) = u(t_n + \Delta t/2)$,
3. $\omega(t_n + \Delta t/2) = e^{(\Delta t/2)A} \omega(t_n)$ with $\omega(t_n + \Delta t/2) = v(t_n + \Delta t)$,
4. $u_0(t_n + \Delta t) = \omega(t_n + \Delta t)$,
5. if $T < (n + 1)\Delta t$, go to step 1, otherwise stop.

The algorithm in words can be written as follows:

1. Solve the first sub-problem for the first half of a time step with an initial value of $u_0(t_n)$ for the first time step or the results from the previous time steps for other stages of calculation.

2. Solve the second sub-problem for a full time step using the result of the first sub-problem solved for the first half of a time step as an initial value.
3. Solve the first sub-problem for the second half of a time step using the result of the second sub-problem.
4. Repeat steps 1-3, but using the result of the third step of the algorithm as an initial value for the first equation in the first step of the algorithm.
5. Repeat the algorithm until the end of the given time interval with each iteration corresponding to one time step.

In the TRM2D software, the Lie-Trotter splitting method was applied. Even though it is less precise than other operator-splitting methods including the Strang splitting method described above, it is easier to implement and takes less time to calculate the results.

2.1.3 Application of the OSM to the transport-reaction problem

In Subsection [1.2.2](#) of the first chapter of this bachelor project, the groundwater flow equation was defined as

$$\frac{\partial c}{\partial t} = -\nabla \cdot (cv) + \nabla \cdot (D_h \nabla c) + \frac{r}{n}, \quad (2.5)$$

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

$$\begin{aligned} \frac{\partial \mathbf{c}(t, \mathbf{x})}{\partial t} = & -\frac{\partial}{\partial x}(v_x(\mathbf{x}) \cdot \mathbf{c}(t, \mathbf{x})) - \frac{\partial}{\partial y}(v_y(\mathbf{x}) \cdot \mathbf{c}(t, \mathbf{x})) + \frac{\partial}{\partial x}((D_x(\mathbf{x}) \cdot \frac{\partial \mathbf{c}(t, \mathbf{x})}{\partial x}) + \\ & + \frac{\partial}{\partial y}((D_y(\mathbf{x}) \cdot \frac{\partial \mathbf{c}(t, \mathbf{x})}{\partial y}) + r(\mathbf{c}(t, \mathbf{x})), \end{aligned} \quad (2.6)$$

where the form of the last term of the equation was described in Section [1.3](#) of this project. Here $\mathbf{c}(t, \mathbf{x})$ is the vector of concentration functions of different chemical compounds present in the solution, and \mathbf{x} is the vector of space dimensions.

As can be seen from the system of equations ([1.36](#)), the reaction part of the equations has no differential terms unlike the transport part. Therefore, it is difficult to solve the equations, because the methods solving the differential equations are different than the ones solving non-differential equations. For this reason, the operator splitting method can be applied.

All the operations applied to the vector of concentrations in the transport-reaction equation are included into the operator A, and it can be rewritten the following way as the sum of the operators T and R, including the operations in the transport and reaction parts of the equation:

$$\begin{aligned} T\mathbf{c} = & -\frac{\partial}{\partial x}(v_x(\mathbf{x}) \cdot \mathbf{c}(t, \mathbf{x})) - \frac{\partial}{\partial y}(v_y(\mathbf{x}) \cdot \mathbf{c}(t, \mathbf{x})) + \frac{\partial}{\partial x}((D_x(\mathbf{x}) \cdot \frac{\partial \mathbf{c}(t, \mathbf{x})}{\partial x}) + \\ & + \frac{\partial}{\partial y}((D_y(\mathbf{x}) \cdot \frac{\partial \mathbf{c}(t, \mathbf{x})}{\partial y})), \end{aligned} \quad (2.7)$$

$$R\mathbf{c} = r(\mathbf{c}(t, \mathbf{x})). \quad (2.8)$$

The reactive transport equation can be written as follows:

$$\frac{\partial \mathbf{c}}{\partial t} = A\mathbf{c}, \quad (2.9)$$

where $A = T + R$ and

$$\begin{aligned}\frac{\partial \mathbf{u}(t)}{\partial t} &= T\mathbf{u}(t), \\ \frac{\partial \mathbf{v}(t)}{\partial t} &= R\mathbf{v}(t).\end{aligned}\tag{2.10}$$

To find a primitive function, the method of separation of variables can be used. Hence, if we suppose both T and R to be linear operators, we get the equations of an exponential form prior to the application of OSM:

$$\Phi_h(\mathbf{c}) = e^{A\Delta t} \mathbf{c},\tag{2.11}$$

and after the application of the OSM

$$\Psi_h(\mathbf{c}) = e^{R\Delta t} e^{T\Delta t} \mathbf{c},\tag{2.12}$$

where h is Δt .

Because it was decided to use the Lie-Trotter splitting method in the TRM2D, the transport sub-problem of the groundwater flow equation is solved first for the full time step, then the reaction sub-problem is solved for the full time step using the results of the transport sub-problem as an initial value. Finally, the result of the reaction sub-problem is applied to the transport sub-problem as an initial value during the next time step.

2.1.4 Evaluation of the local error of the OSM

This subsection is based on M. Leok's lecture [10].

Although it was stated above that the A operator can be rewritten as a sum of T and R operators, the equation is not always valid. To demonstrate it let us find the difference between the equations (2.11) and (2.12), i.e. between the solution of the transport-reaction equation before and after the application of the OSM.

$$\tau = \Phi_h(\mathbf{c}) - \Psi_h(\mathbf{c}).\tag{2.13}$$

By expanding the exponential functions as the Taylor series, the equation takes the following form:

$$\tau(h) = (\mathbb{1} + hA + h^2A^2 + \dots) \cdot \mathbf{c} - (\mathbb{1} + hR + h^2R^2 + \dots) \cdot (\mathbb{1} + hT + h^2T^2 + \dots) \cdot \mathbf{c}.\tag{2.14}$$

By expanding and simplifying the brackets the equation will take the following form:

$$\tau(h) = h^2 \left(\frac{1}{2} (R + T)^2 - \left(RT + \frac{1}{2} R^2 + \frac{1}{2} T^2 \right) \right) \cdot \mathbf{c} + O(h^3).\tag{2.15}$$

Following that the local error of the operator-splitting method will take the following form:

$$\tau(h) = \frac{h^2}{2} (RT - TR) \cdot \mathbf{c} + O(h^3).\tag{2.16}$$

The local error of the OSM is generally of the second order in time step h for linear operators T and R . In the case of commutative operators T and R the local error can be of the third order in h . In the case of reactive transport problem, T and R do not commute and moreover R is not generally linear. That is why this analysis is only an inspiration for our application of the OSM.

2.2 Methods used for solving the reaction part of the reactive-transport problem

This section is based on J. Šembera's Lecture notes [3].

As previously mentioned in Section 1.3, the PhreeqcRM module includes plenty of complicated mathematical and numerical methods for solving the reaction part of the reactive-transport problem. It is beyond the scope of this bachelor's degree project to list and explain all of these methods, but it is important to understand their general principle and problems that can occur while using them, which is why the Euler method, one of the simplest methods for solving such problems, will be explained in this section, even though it is less precise.

2.2.1 The Euler method

The PhreeqcRM library gives us information regarding the change of concentration at a certain time and place for each chemical compound as a system of first order ODEs listed in the system of equations (1.31). The system of functions on their right-hand sides can be interpreted as a vector of functions $f : \langle a, b \rangle \times \langle 0, +\infty \rangle^5 \rightarrow \mathbb{R}_0^{+5}$, which can be represented as a directional field in phase space.

The reaction problem is essentially an initial value problem, because it consists of a system of ODEs with an initial condition. Moreover, an exact solution for this system cannot be found, and for this reason, the Euler method is used.

This initial value problem can be written as follows:

$$\begin{aligned}\frac{dc}{dt} &= f(t, c), \\ c(a) &= \xi,\end{aligned}\tag{2.17}$$

where f is a vector of ODE functions, a is the beginning of a time interval and ξ is the initial value of concentration.

The Euler method is one of the simplest discrete numerical methods which can solve such a problem. Its main idea is in dividing a continuous time interval $\langle a, b \rangle$ into a finite number n of usually equidistant time steps $t_i = a + i\Delta t, i = 0, 1, \dots, n$, and then finding an approximation of the exact solution as a jagged line. The method starts with finding a fragment of a tangent to a directional field at a certain place of a phase space defined by an initial condition for a certain time step. Then it is continued by finding a fragment of a tangent to a directional field for a current time step at a place where the previous fragment ended using its value as the initial condition. Therefore, the system of equations (2.17) can be rewritten as follows using the Euler method:

$$\begin{aligned}c_{i+1} &= c_i + (t_{i+1} - t_i) \cdot f(t_i, c_i) \\ &= c_i + \Delta t \cdot f(t_i, c_i), \\ c_0 &= \xi, \\ \Delta t &= \frac{b - a}{n},\end{aligned}\tag{2.18}$$

where Δt is an evaluation of the length of an equidistant time step.

2.2.2 Evaluation of the total error of the Euler method

Because the Euler method finds a discretized approximation of a problem, it is obviously important to estimate its global discretization error. The global discretization error of the method is generally

described as the accumulated local discretization errors for each time step and defined by the following formula:

$$e_i = c_i - c(t_i), \quad (2.19)$$

where c_i is the approximation calculated with the Euler method and $c(t_i)$ is the exact solution.

A local discretization error is the difference between the numerical approximation and the exact solution of the problem for each time step. It is defined by the following formula:

$$L(c(t_{i+1}), \Delta t) = c_i + \Delta t \cdot f(t, c(t)) - c(t_{i+1}). \quad (2.20)$$

The total error also includes the deviation due to possibly inaccurate initial values in each time step.

To minimize this deviation it is natural to make a calculation for a fine-scale discretization. However, if it is done, the number of calculations grows dramatically. For this reason, it is also important to take into consideration a rounding error. It is caused by the rounding of the results of calculations made by the computer processor, because it is only able to calculate the final number of decimal places. This is why with the growing number of calculations the accumulated rounding error might become significant. Moreover, the growing number of calculations will prolong the calculation time. However, as shown in Figure 2.1 the optimal time step length can be found with a minimal total error.

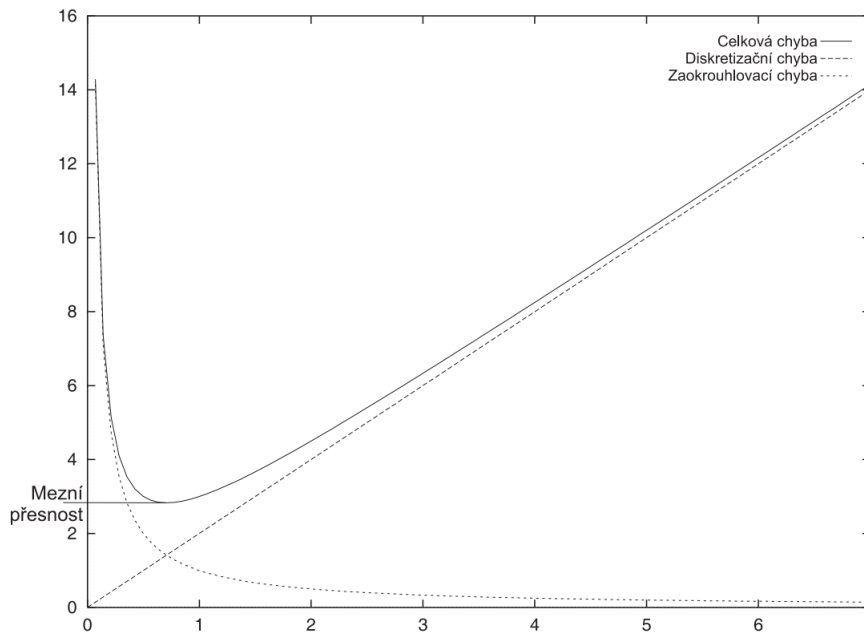


Figure 2.1: Relation between the discretization and the rounding error, with the time step length lying on the x-axis and the size of total error of numerical computation – on the y-axis. (Adopted from [3])

It is also important to notice that for certain periods of time the constant time step length might be too large because the values are changing too rapidly and it is possible to miss some features of the solution and it is necessary to study this part of the graph more slowly. On the other hand, if the time step is smaller than necessary for a relatively flat region of the solution graph, some time might be wasted calculating the results which could be predicted from a directional field for a longer time step. This is why the adaptive time step control method could be profitable for minimizing these problems.

2.2.3 The adaptive time step method's algorithm

This subsection is based on L. Samoilov's bachelor's degree project [7]. In this subsection the algorithm of the adaptive time step control method, implemented in the TRM2D software, will be explained.

The main idea of this method lies in comparing the results of the calculations for the previous time step with the those for the current time step. For this the Euclidean norm of these two results is evaluated and if it is significantly larger than the *eps* parameter, the time step is increased by 10%. Otherwise, it is halved and the calculation is conducted once again for a new length of a time step.

The algorithm of this method may be represented as shown in Figure 2.2.

```
concsold = concs
transport calculation ()
reaction calculation ()
if  $\|(\textit{concs} - \textit{concs}_{old}) / ((\textit{concs} + \textit{concs}_{old}) / 2)\| > 0.5 \cdot \textit{eps}$  then
    timestep = 1.1 · timestep
else if  $\|(\textit{concs} - \textit{concs}_{old}) / ((\textit{concs} + \textit{concs}_{old}) / 2)\| < \textit{eps}$  then
    timestep = 0.5 · timestep
    restart calculation
else
    Proceed with the same time step
end if
```

Figure 2.2: The algorithm of the adaptive time step method. (Adopted from [7])

Chapter 3

TRM2D software

This chapter's purpose is to describe how the TRM2D software works, the changes that were made to the software as a part of this bachelor project and the reasons for these changes.

3.1 General description of the software

TRM2D is software written in C++, which serves for simulating the groundwater flow movement, including the chemical and transport processes, described in previous chapters. The following subsections will describe the steps and details of the implementation of the mathematical model of the transport-reaction problem.

3.1.1 Reading input data

The basic structure of the TRM2D software is shown in Appendix [C](#)

The process starts with interpreting data from an input XML file. One of the version of such an XML file can be seen in Appendix [B](#). An input XML file for this project is divided into several child elements of the root tag `<TRM>` on line 2. The first child tag determines the information regarding the output files, i.e. the filenames, the directory, file suffix and other log details. The second one determines the units of different physical quantities, mentioned later in this input file under other tags. The units for them are set as an index number written as an attribute of a tag corresponding to units listed in the `<Quantities>` tag. The next two tags include the information regarding the transport and reaction modules. The last tag in the XML input file on lines 91-98 describes how the output information will be written, e.g. how often it shall be recorded (e.g. every 1000 of loop steps) from a reaction and transport modules.

The reaction module tag, starting on line 66, includes the data regarding the boundary condition changes, i.e. if they should be applied, how the PhreeqcRM library will be used for the reaction part of the problem, which chemical processes will be employed, the values of physical quantities, e.g. temperature, pressure, saturation, density, etc.

The most important part of the XML input file for this bachelor project is under the transport module tag, starting on line 22. Firstly, the space characteristics are determined in it, such as the number of rows and columns of a space grid, where the solution will flow in and out of the system, which transport components are present in the simulation and most importantly the time data under the `<Time>` tag on line 23. It includes data regarding the initial and end values of time and the length of time steps in the beginning of the simulation. The length of time steps can be changed under the `<BorderCondChanges>` tag, where according to an index of a time step the length of it is set as a child-element of each child tag called `<Step>`.

The data from the XML input file can be interpreted with the library known as TinyXML. This is an open-source XML parser which then builds a Document Object Model (DOM) from a file and then is read and loaded from it by the `load_inputs(...)` method, partially illustrated in Source Code 3.4, of the `trm_main` class responsible for the run of the program in the `trm_main.cpp` file. This method determines the way the data from the XML file will be saved and the attributes that will hold the data. Later the data are saved to a log file with the results of a calculation with the `save_inputs(...)` method of the same class.

The part that is the most important to understand in the TRM2D software is the `load_time(...)` method of the `trm_module` class intended for setting a general manner of how the transport and reaction parts of the system will work and is a parent to the `transport_module` and `reaction_module` classes. The `load_time(...)` method reads the data from the `<Time>` tag in the XML file, which can be seen in Appendix B, regarding the initial and end values of time and the unit of time and saves it with the `set_time(...)` method. Then it reads the value of the initial time step index and the length of the initial time step length and then saves it into the `bc_times_steps[...]` map where an index serves as a key and the length of a step is written under the corresponding index. Other data written into this map are read from the `<BorderCondChanges>` tag later.

3.1.2 Application of the transport and reaction processes onto the input data

After the input data are loaded the calculation starts in the `calculate(...)` method of the `trm_main` class. This method, illustrated in Appendix A, first activates the transport part, sets the physical quantity values, initiates the reaction part and the boundary conditions, sets other input data and declares local data structures. Following that the calculation itself starts as a `for` loop declaring the `t` loop variable with an initial value of time as a starting point, the end value of time as a threshold marking the end of the for cycle and the `curr_time_step` local variable as an increment. In this cycle the `t_index` local variable, set to zero in the beginning and incremented in the end of the loop, is used to check if it is currently the time index where the boundary conditions are changed, and if so it carries out the instructions for the potential change of the time step length and boundary conditions according to a specific time index. Following that it is checked if the current time value exceeds the end value of time and if so it subtracts their difference from the current time step, so the calculation is run exactly until the end of the given time interval. Then the transport module is calculated and if the current time index requires a calculation in the reaction module it is conducted.

During the run of the program the events are logged into the log XML and CSV files (according to the settings put in the methods of classes defined in the `trm_inout.*` files) containing data regarding the concentrations of each chemical compound changing over time, values of other physical quantities, lengths of time steps, etc.

3.1.3 Implementation of the adaptive time step method

This subsection briefly describes the adaptive time step control method, whose algorithm was explained in Subsection 2.2.3 of the previous chapter, implemented in the TRM2D software. It was placed into the `calculation(...)` method mentioned in the previous subsection and illustrated in Appendix A.

Firstly, concentrations of the chemical compounds calculated in the previous time step are saved to the locally declared vector `transport_concs_old` on line 57 and transformed into a row vector `tc_old` on line 58. Then inside the fragment of the code starting with a labeled statement the last current time step is evaluated on lines 62-66 in case it gets changed because of the time step control

method. Following that the calculations of the transport and reaction parts are run, and the concentration vector is changed after the calculation is transformed into the row vector `tc` on lines 67-72. It is followed by the calculation of the `concentrace` row vector containing the difference between the old and the new concentrations divided by their average and changes NaN to zero on lines 73-80, appearing if the vector elements are divided by zero. And finally, the adaptive time step algorithm is implemented on lines 87-99 as an `if` statement and in case the current time step needs to be changed and the data must be recalculated for a new time step, the `goto` statement returns us to the beginning of the fragment of code starting with a labeled statement.

It is important to notice that originally the parameters used in the algorithm were declared as local variables of the method and fixed.

3.2 Adaptation of the software as a part of this bachelor project

The main aim of this bachelor project was testing of the time step control method for different values of the three parameters and finding their optional settings for the method. However, these parameters were fixed in the source code of the original TRM2D software, requiring the user to recompile the source code each time after changing them, which would hinder the process of the parameter optimization. This is why the adaptation of the software was needed for the testing. In the following subsections the changes made to the software will be listed and explained.

3.2.1 Reading the parameters from an input file

It was decided to enable the software to read the values of the parameters from an input XML file. For this the software itself needed to be adapted for a correct interpretation of the data.

As was established earlier, the data regarding the time settings are read from the `<Time>` tag of an XML input file with a `load_time(...)` method. So, it was chosen to be the most appropriate to alter the source code and XML file in these two places. Since it is information regarding the time step length, the parameters were added into the `<Step>` tag. The altered `<Step>` tag of the XML file can be seen in the Source Code [3.1](#).

```
1      -<Step idTimeStep="0" adaptivestep="yes">
2          <InitialStep>1</InitialStep>
3          <Eps>0.01</Eps>
4          <Par1>0.5</Par1>
5          <Par2>1.3</Par2>
6      </Step>
```

Source Code 3.1: An altered version of the `<Step>` tag of an input XML file.

Understandably, the parts of the code reading the input data are in order of the corresponding data in the XML file. So, the new piece of code was added after the `set_time(...)` method and the part reading the initial time index. The `"adaptivestep"` attribute allows the user to switch from an adaptive time step to the constant more easily.

```

1  double time_step;
2  if (output->get_attr(step_zero_elem, "adaptivestep", false) == "") {
3      time_step = output->get_elem_double(step_zero_elem);
4  } else {
5      time_step = output->get_child_elem_double(step_zero_elem,
6          ↪ "InitialStep");
7      if(output->get_attr(step_zero_elem, "adaptivestep", false) == "yes"){
8          set_adaptive_step(output->get_child_elem_double(step_zero_elem,
9              ↪ "Eps"), output->get_child_elem_double(step_zero_elem, "Par1"),
10             ↪ output->get_child_elem_double(step_zero_elem, "Par2"));
11     }
12 }
13 bc_times_steps.clear();
14 bc_times_steps[zero_time_id] = time_step;

```

Source Code 3.2: An altered version of the `load_time(...)` method.

The initial length of a time step can be read from both versions of an input file, from the old version for a constant time step, and the new one which includes data regarding the time step control parameters. To determine which version of an input file is used, the `if` statement checks if there is the "adaptivestep" attribute in the `<Step>` tag. If not, the `time_step` local variable saves the value with the `get_elem_double(...)` method from the TinyXML library. Otherwise, the `time_step` local variable saves the value with the `get_child_elem_double(...)` method by reading the value of the `<InitialStep>` child tag. Following that the `if` statement checks if the "adaptivestep" attribute is equal to "yes". if so, the `set_adaptive_step(...)` method, similar to the preexisting `set_time(...)` method of the `trm_module` class, is called. It initializes the attributes of the `trm_module` class according to the parameters. The implementation of this method is show in Source Code [3.3](#).

```

1  void trm_module::set_adaptive_step(double eps, double par1, double par2) {
2      adaptive_time_step = true;
3      adapt_eps = eps;
4      adapt_par1 = par1;
5      adapt_par2 = par2;
6  }

```

Source Code 3.3: The implementation of the `set_adaptive_step(...)` method.

3.2.2 Inputting the name of the log file for the adaptive time step

Another factor that could hinder the testing process is the fixed setting of a log filename in the source code, because the file would always get rewritten over the last version of it. However, sometimes it is needed to save an old version before running the software again to be able to compare the results later, or to change its name or a directory. The file contains the norms of the relative deviation vector of the

compared concentrations from the previous and current time steps, current lengths of time steps, time indices and time values for each iteration.

```

1  for (XMLElement *child = root_elem->FirstChildElement(); child != nullptr;
    ↪  child = child->NextSiblingElement()) {
2      string tag_name = child->Name();
3      if (tag_name == "LogFile") {
4          log.set_values(
5              out.get_attr(child, "filePrefix"), out.get_attr(child,
    ↪  "fileSuffix"), out.get_attr(child, "logDetails"));
6          log.start_logging();
7          log(BASIC) << "Input data loaded from the XML file '" +
    ↪  input_file + "'.";
8      }
9      if (tag_name == "EpsFile") {
10         bool required = true;
11         if (out.get_attr(child, "requireOutput") == "no") {
12             required = false;
13         }
14         if (required) {
15             log.set_eps_logfile(required, out.get_attr(child,
    ↪  "fileName"));
16         }
17     }
18     ...
19 }

```

Source Code 3.4: A part of the `load_inputs(...)` method of the TRM2D source code.

Like any other tag from the XML input file, the newly added `<EpsFile />` tag is read in the `load_inputs(...)` method after the `<LogFile />` tag. To leave the source code compatible with older versions of the input files again, if the tag is not found, nothing happens and the information does not need to be read. If it is found, then the `"requireOutput"` attribute is read, and if it shows that the output file logging data regarding the adaptive time step does not need to be created, then the next tag is read. Otherwise, other attributes are read and saved as attributes of the `log_file` class with the `set_eps_logfile(...)` method of the same class.

The XML input file and the source code were altered in a similar way described in the previous subsection analogically to the `<LogFile />` tag and the way it was read.

```

1  <LogFile logDetails="BasicComponentMessages" fileSuffix=".log"
    ↪  filePrefix="./Log/Log_"/>
2  <EpsFile requireOutput="yes" fileName="norm.txt"/>

```

Source Code 3.5: The `<LogFile />` and the `<EpsFile />` tag from an input XML file.

3.2.3 Changes in the "calculate" method

After the input data are read from an input file, it needs to be used in an appropriate place. As was mentioned in Subsection 3.1.3, the parameters are initialized and used in the `calculate(...)` method in the original code. Because the parameters were read from an XML file, the local variables holding their values can be replaced with the attributes used for saving the input data. Moreover, the `if` statement now checks if the `adaptive_time_step` attribute is true instead of checking if the `eps` parameter is not equal to zero.

```
1  if (trans.adaptive_time_step) {
2      if (arma::norm(concentrace, 2) < trans.adapt_eps * 0.5)
3          {
4              curr_time_step = curr_time_step * trans.adapt_par1;
5          }
6      else if (arma::norm(concentrace, 2) > trans.adapt_eps)
7          {
8              curr_time_step = curr_time_step * trans.adapt_par2;
9              transport_concs = transport_concs_old;
10             t = t - curr_time_step;
11             goto label;
12         }
13 }
```

Source Code 3.6: An altered part of the `load_time(...)` method.

To incorporate the input data regarding the log file of the adaptive time step method, the `if` statement now checks the `require_eps_output` attribute in the corresponding fragment of the source code and, if the output is required, uses the output filename saved in the `eps_output_filename` attribute.

```
1  fstream out;
2  if (log.require_eps_output) {
3      out.open(log.eps_output_filename, fstream::out);
4  }
```

Source Code 3.7: An altered part of the `calculate(...)` method using data regarding a log file.

Chapter 4

Testing of the extended TRM2D software

In this chapter the tests ran on the new version of the TRM2D software will be described and the results of these tests will be interpreted.

4.1 Verifying the consistency of the results between the newly altered version of the software and the previous one

To be able to continue the further testing and optimization of the parameters, it is important to ensure that the version of the TRM2D software extended under this bachelor project works the same way as the previous version with the parameters fixed in the source code. For this purpose, the old version was compiled three times, resulting in obtaining three different executable files corresponding to three values of the fixed `eps` parameter: 0.91, 0.5 and 0.010293. These files were then run 5 times each. Following that the new version of the software was run 5 times with three different XML input files setting the `eps` parameter to the same three values. The testing problem and the values of the `eps` parameter were from L. Samoilov's bachelor's degree project [7].

The results for the calcium concentrations, written to the "Ca_Results.csv" in the "transport" directory, were compared. For all 5 runs of both software versions and the `eps` parameter set to 0.91 and 0.5 the results appeared to be exactly the same. However, they were not for the 0.010293 value of the `eps` parameter. For this reason it was decided to compare the maximal deviation of the calcium concentrations for the last time step between each run of the software. The maximal deviation is defined as the maximal norm of the relative deviation vector and found by the following formula:

$$\kappa = \|\text{rel}(c_1, c_2)\|_{\max}, \quad (4.1)$$

where c_1 and c_2 are the vectors of calcium concentrations from the first and second solution respectively, and $\text{rel}(x, y)$ is the vector of the relative deviation of vectors x and y denoted by

$$[\text{rel}(x, y)]_i = 2 \cdot \frac{|x_i - y_i|}{|x_i + y_i|}. \quad (4.2)$$

It was established that κ was approximately equal to $8.31 \cdot 10^{-12}$, which can be considered tolerable and might be explained by the rounding error, because the number of iterations for this `eps` value reaches 2500. Therefore, we can conclude that the two versions of the software show equivalent results and the changes made to the software under this project do not affect the results of the calculations in it.

4.2 Formulation of the testing problem

In this section the testing problem for optimizing the parameters of the TRM2D software will be described.

It was decided to test the software on a 2D problem with the system consisting of 4 columns and 10 rows with each grid cell of volume 5 m^3 . The porosity was set to 0.2 by default, so, the volume of the water in each grid cell stands at 1 m^3 or 1000 l. The water flow was set to have a direction from the top left corner to the bottom right. The above data were set in the XML input file.

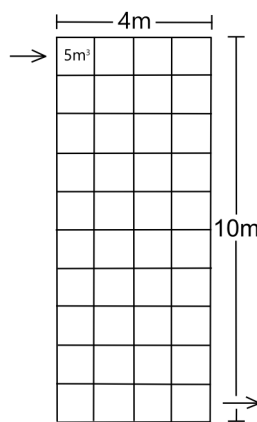


Figure 4.1: Visual representation of the testing problem.

The amount of calcium in water in equilibrium with calcite was set to 6.54 mg/l and carbon to 1.95 mg/l as an initial condition. The above data were set in the "komain3.phr" file used by the PhreeqcRM library.

The concentration of calcium in the solution flowing from the top left corner was set to 0.04987 mg/l. The flowing-in solution was also set to have higher pH and a significantly lower concentration of carbon. The data regarding the flowing-in solution were set in the "kc_amd.phr".

Because it was needed to calculate the reference solution for a very fine scale, it was necessary to choose an optimal time interval, which would allow us to observe the spread of the solution entering the system while not requiring too much memory space and calculation time. For this reason different states of the system with a corresponding time value are illustrated in Figures [4.2](#).

6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00

0 days

4,97E-02	7,21E-02	1,20E+00	4,73E+00
5,66E-02	3,36E-01	2,52E+00	5,60E+00
3,79E-01	1,59E+00	4,34E+00	6,20E+00
1,84E+00	3,73E+00	5,71E+00	6,44E+00
4,04E+00	5,47E+00	6,32E+00	6,51E+00
5,66E+00	6,26E+00	6,49E+00	6,52E+00
6,33E+00	6,48E+00	6,52E+00	6,52E+00
6,50E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00
6,52E+00	6,52E+00	6,52E+00	6,52E+00

10 days



Figure 4.2: Stages of solution distribution for different time values. The side ratios are distorted due to the visualisation method. See actual ratios in Figure [4.1](#).

As can be seen in the figures, the earliest stage where the spread "wave" can already be observed is for the stage corresponding to 20 days. Therefore, the testing problem was chosen to be calculated for the time interval $< 0, 20 >$ days.

4.3 Optimizing the parameters

In this section the process of identifying the optimal parameters for the time step control method will be described. The strategy for it may lie in improving the accuracy of the results, while the time steps would have to be as short as possible and, thereby, the computational time would be longer. However, the strategy may also lie in focusing on the reduction of the number of time steps and, consequently, reducing the computational time while keeping the accuracy at a satisfactory level. In this project it was decided to follow the second strategy, where the accuracy is supposed to be comparable to the accuracy for calculation with a constant time step.

The process of testing will consist of consecutively finding the most appropriate values for each parameter (by comparing the results with the reference data) and using them for finding the optimal value for the next parameter.

Because we were unable to find an analytical solution for this problem, it was decided to calculate a solution for the shortest time step possible and use it as reference data for the further testing. It was empirically determined that the shortest time step possible and reasonable to calculate with was 0.0001 days.

To start the optimization of the parameters, it was decided to find an optimal value of the eps parameter first with temporary default values of the par1 and par2 parameters set to 1.5 and 0.5 and an initial time step equal to 1 day. The values that the eps parameter was going to be set to were from the "norm.txt" log file, registering the Euclidean norms of the concentrations of the current and previous time steps after the calculation for a constant time step. It was suggested that because the Euclidean norm is compared to the eps parameter, the results might be of similar precision to the calculation for a certain constant time step if the eps parameter is equal or similar to the minimal Euclidean norm registered in

the "norm.txt" file logged after the calculation for this length of a time step. For this purpose, the results for different lengths of a constant time step were calculated and put into Table 4.1.

Δt [days]	0.1	0.01	0.001
$N[1]$	200	2,000	20,000
$\kappa[1]$	$3.12 \cdot 10^{-2}$	$3.05 \cdot 10^{-3}$	$2.76 \cdot 10^{-4}$
$\epsilon[1]$	0.113765	0.011282	0.001127

Table 4.1: Results for different lengths (Δt) of a constant time step, where N – number of iterations, κ – maximal norm of the relative deviation vector, ϵ – minimal Euclidean norms.

The minimal Euclidean norms for each calculation from Table 4.1 were used as the values of the eps parameter with the initial length of a time step equal to 1 day and the results of these calculations were put into Table 4.2.

eps [1]	0.113765	0.011282	0.001127
$N[1]$	469	4,537	40,580
$\kappa[1]$	$2.68 \cdot 10^{-2}$	$2.64 \cdot 10^{-3}$	$2.38 \cdot 10^{-4}$

Table 4.2: Results for different values of the eps parameter corresponding to the time step lengths.

As can be seen from Table 4.2 the number of iterations for an adaptive time step is approximately twice as large as for a corresponding constant time step calculation, but κ is slightly smaller. To make the calculation more effective there were attempts to find a more appropriate value for the eps parameter, similar to the one used before, resulting in a smaller number of iterations and the maximal deviation still comparable to that of a constant time step calculation. In the following tables the results of such attempts are listed.

eps [1]	0.19	0.25	0.4
$N[1]$	280	213	137
$\kappa[1]$	$4.5 \cdot 10^{-2}$	$6.05 \cdot 10^{-2}$	$9.44 \cdot 10^{-2}$

Table 4.3: Results of calculations with maximal deviation similar to that of a constant time-step calculation with the length of a time step 0.1 days.

eps [1]	0.019	0.025	0.04
$N[1]$	2500	2113	1224
$\kappa[1]$	$4.81 \cdot 10^{-3}$	$5.81 \cdot 10^{-3}$	$9.94 \cdot 10^{-3}$

Table 4.4: Results for calculations with maximal deviation similar to that of a constant time-step calculation with the length of a time step 0.01 days.

Having observed the results of the calculations, it was decided that to obtain similar maximal deviation from the adaptive time-step method and smaller number of iterations, it is necessary to set the value for the eps parameter ranging between 0.19 and 0.4 or 0.019 and 0.4 to obtain more precise results. Therefore, the maximal deviations for the adaptive time-step method are 10 times more precise for 10 times smaller value of the eps parameter. The value of eps that is corresponding well enough to the constant time steps 0.1, 0.01, 0.001 can be considered $0.27 \cdot 10^{-x}$ where $x \in \{0, 1, 2\}$, because κ and N of

the calculations with this value were found to be the most compromise. The results of the calculations for different values of x are listed in Table 4.5.

eps [1]	0.27	0.027	0.0027
$N[1]$	163	1,953	18,743
$\kappa[1]$	$7.71 \cdot 10^{-2}$	$6.29 \cdot 10^{-3}$	$5.91 \cdot 10^{-4}$

Table 4.5: Results for calculations with different values of the eps parameter.

After fixing the testing values of eps, the next step was to find optimal values for the other two parameters: par1 determining by how many percent a time step will be prolonged and par2 determining by how many percent a time step will be shortened. To get a general idea of how the two parameters would affect the results of the calculations faster, 0.27 value of the eps parameter was used first. In the following tables the results of such calculations with different values of the two parameters are shown.

par2 [1]	0.1	0.3	0.4	0.45	0.5	0.55	0.6
$N[1]$	164	168	172	171	163	174	162
$\kappa[1]$	$2.59 \cdot 10^{-1}$	$1.69 \cdot 10^{-1}$	$1.01 \cdot 10^{-1}$	$6.2 \cdot 10^{-2}$	$3.91 \cdot 10^{-2}$	$8.91 \cdot 10^{-2}$	$1.65 \cdot 10^{-1}$

Table 4.6: Calculations for the eps parameter set to 0.27 and par1 to 1.5 by default.

From the data shown in Table 4.6 it was deduced that the optimal value of the par2 parameter ranges between 0.45 and 0.55. Moreover, the results for 0.1 and 0.3 were inconsistent between different runs.

While testing the software with different values of par1 it was determined that they did not affect the results at all for the eps equal to 0.27 and slightly for eps equal to 0.027. Therefore, it was decided to calculate the results with different values of par1 with the value of eps set to 0.0027.

par1 [1]	2	1.7	1.5	1.4
$N[1]$	15,007	15,971	18,743	17,354
$\kappa[1]$	$8.46 \cdot 10^{-4}$	$7.51 \cdot 10^{-4}$	$5.91 \cdot 10^{-4}$	$6.56 \cdot 10^{-4}$

Table 4.7: Calculations for the eps parameter set to 0.0027 and par2 to 0.5 by default.

From the data shown in Table 4.7 it is evident that κ still does not change significantly, but the 1.5 value gives the smallest κ and the value 2 gives the smallest number of iterations.

Because it was difficult to determine which value of par1 is the most appropriate to use, the results were calculated for different values of par1 and optimal values of par2 for eps equal to 0.27.

par2 [1]	0.45			0.5			0.55		
par1 [1]	2	1.7	1.4	2	1.7	1.4	2	1.7	1.4
$N[1]$	155	159	178	163			202	162	182
$\kappa[1]$	$6.98 \cdot 10^{-2}$	$6.42 \cdot 10^{-2}$	$6.09 \cdot 10^{-2}$	$3.91 \cdot 10^{-2}$			$1.05 \cdot 10^{-1}$	$9.17 \cdot 10^{-2}$	$8.78 \cdot 10^{-2}$

Table 4.8: Calculations for the eps parameter set to 0.27.

From the data in Table 4.8 it is evident that the value of par2 with the smallest κ is 0.5 and it also gives one of the smallest N . Therefore, it can be suggested that the most appropriate value of par2 is equal to 0.5. Moreover, for par2 equal to 0.45 and 0.55 κ is slightly smaller for par1 equal to 1.4.

In Table 4.9 there are results for eps equal to 0.0027, because the larger eps did not show any significant differences between the results for different values of par1, and for par2 equal to 0.5.

par1 [1]	1.3	1.45	1.6
$N[1]$	17,958	17,084	16,375
$\kappa[1]$	$3.18 \cdot 10^{-4}$	$3.42 \cdot 10^{-4}$	$3.66 \cdot 10^{-4}$

Table 4.9: Calculations for the eps parameter set to 0.0027 and par2 to 0.5.

From Table 4.9 it seems evident that even for such a small eps value the difference between κ for different par1 parameters is not significant. Therefore, the most appropriate value of par1 for this testing problem can be considered 1.6, because it gives the smallest N .

To sum up, various values of the eps parameter corresponding to certain orders of precision were found during the testing and the combination of par1 equal to 1.6 and par2 equal to 0.5 can be considered the most suitable for this testing problem. Finally, the combination of the three chosen values for the parameters gives the results with slightly larger κ and quite smaller N than for the corresponding constant time steps.

Conclusion

This bachelor's degree project addresses the difficulties related to solving the reactive-transport problem with the operator-splitting method (OSM) and how it can be extended by the adaptive time step control method. For this purpose, the theoretical basis behind the mathematical model describing the movement of the contaminated groundwater through a porous material and its application in the TRM2D software were studied. Among the needed theoretical grounds were the occurring transport and reaction processes, principles for building the PDE system describing the chemical reactions, and numerical methods solving them. Among the methods explained under this project were the Euler method, the OSM and the adaptive time step method. Particular attention was paid to the OSM, because its main idea determines the operating principle of the TRM2D software, which lies in dividing it into transport and reaction modules and then calculating the results by calling the modules' methods consecutively for each time step.

For a possible improvement of the software's performance, the adaptive time step control method was later implemented. One of the main goals of this bachelor's degree project was the extension of this method in the TRM2D software. After the implemented changes were listed and explained in this project, it was needed to test if they did not affect badly the performance of the software, which was later confirmed.

Another important goal of this project was to analyze the impact of the different values of the parameters on the performance of the TRM2D software. As a result, we managed to find a combination of the parameter values showing the results with accuracy similar to accuracy of the results calculated with a constant time step, but requiring less computational time.

Bibliography

- [1] Diab: *Operator Splitting Methods*, Web page at STIMULATE European Joint Doctorates, 2019, accessed 6.7.2022. <http://www.stimulate-ejd.eu/content/operator-splitting-methods>
- [2] M. Hokr: *Transportní procesy* [in Czech], Lecture notes. Faculty of Mechatronics, Technical University of Liberec, 2005.
- [3] J. Šembera, J. Maryška: *Poznámky k předmětu Aplikovaná Matematika* [in Czech], Lecture notes. Faculty of Mechatronics, Technical University of Liberec, 2005.
- [4] P. Štrof et al.: *Final Report of the TACR project Nr. TH02030840* [in Czech]. DHI Prague, 2019.
- [5] USGS team: *PHREEQC Version 3*, Web page at USGS, 2021, accessed 6.7.2022. <https://www.usgs.gov/software/phreeqc-version-3>
- [6] L. Zedek: *Modelování transportně-chemických procesů* [in Czech], Doctoral thesis. Faculty of Mechatronics, Technical University of Liberec, 2014.
- [7] L. Samoilov: *Operator splitting method for transport reactive problem solution*, Bachelor's degree project. FNSPE CTU in Prague, 2021.
- [8] H. Holden, K. H. Karlsen, K.-A. Lie, N. H. Risebro: *Splitting Methods for Partial Differential Equations With Rough Solutions. Analysis and Matlab Programs*, EMS Series of Lectures in Mathematics. European Mathematical Society, Curych, 2010.
- [9] M. Kirk: *Earth in Action - basic aspects of groundwater flow*, youtube video, accessed 6.7.2022. <https://www.youtube.com/watch?v=mb8clQdvrvo>
- [10] M. Leok: *Splitting methods*, youtube video, accessed 6.7.2022. https://www.youtube.com/watch?v=yiebrv_RbMU

Appendix A

Original implementation of the "calculate(...)" method

```
1 void trm_main::calculate()
2 {
3     trans.remove_results();
4     react.remove_results();
5
6     trans.activate();
7     vector<double> tmp_porosity = trans.get_component_data("Porosity", "",
8     ↪ -1);
9     react.set_porosity(tmp_porosity);
10    react.set_components_for_factors(&trans.porosity, &trans.volume,
11    ↪ &trans.conductivity_long, &trans.conductivity_trans);
12    react.init();
13    activate_boundary_conditions();
14
15    log(BASIC) << "Calculation of time steps loop started.";
16    vector<double> transport_concs;
17    vector<double> transport_concs_old;
18    arma::rowvec tc;
19    arma::rowvec tc_old;
20    double eps = 0;
21    //////////////////////////////////////
22    // mapping component Pas2Phr to be used in reaction part to calculate
23    ↪ factors
24    vector<double> mapping;
25    vector<string> transport_names;
26    current_bc_time = 0;
27    unsigned initial_time_index = trans.get_time_init_id();
28    unsigned t_index = initial_time_index;
29    string map_component_id = react.get_map_component_id();
30
31    double curr_time_step = trans.get_time_step(trans.time_init_id);
```

```

29  double t;
30  unsigned time_steps_count = 0;
31
32  for (t = trans.time_init + curr_time_step; t < trans.time_end +
33  ↪ trans.time_end * NUMERIC_EPS; t = t + curr_time_step) {
34      time_steps_count++;
35  }
36  unsigned step_index = 0;
37
38  fstream out;
39  out.open("c:\\Trm\\norm.txt", fstream::out);
40
41  for (t = trans.time_init + curr_time_step; t - curr_time_step <
42  ↪ trans.time_end - trans.time_end * NUMERIC_EPS; t = t + curr_time_step)
43  {
44      progress_value = static_cast<unsigned>(100 *
45      ↪ (static_cast<double>(t_index) -
46      ↪ static_cast<double>(trans.get_time_init_id())) /
47      ↪ static_cast<double>(time_steps_count));
48      if (trans.is_bc_change_time(t_index)) {
49          current_bc_time = t_index;
50          curr_time_step = trans.get_time_step(current_bc_time);
51          // initial conditions used only at zero time
52          if (t_index == initial_time_index) {
53              vector<double> tmp_values =
54              ↪ arma::conv_to<vector<double>>::from(
55              ↪ trans.get_component_data("TransportComponents", t_index,
56              ↪ map_component_id));
57              vector_values init_cond(tmp_values);
58              react.activate(
59                  transport_concs, transport_names, init_cond,
60                  ↪ trans.get_component_data("InComponents", t_index,
61                  ↪ map_component_id),
62                  t_index, t - curr_time_step,
63                  ↪ trans.get_first_component_of_id("0"));
64              trans.init(transport_concs, transport_names,
65              ↪ initial_time_index);
66          }
67          trans.modify_grid_flow(t_index);
68      }
69      transport_concs_old = transport_concs;
70      tc_old = arma::conv_to<arma::rowvec>::from(transport_concs);
71      t_index++;
72
73      label:
74      if (t > trans.time_end)

```



```

63     {
64         curr_time_step = trans.time_end - t + curr_time_step;
65         t = trans.time_end;
66     }
67     trans.calculate(t_index, t, transport_concs, transport_names,
68         ↪ current_bc_time, mapping, curr_time_step);
69     if (react.is_calculated_step(step_index)) {
70         react.calculate(t_index, t, curr_time_step, transport_concs,
71             ↪ transport_names, mapping);
72     }
73
74     tc = arma::conv_to<arma::rowvec>::from(transport_concs);
75     arma::rowvec concentrace = ((tc - tc_old) / (tc + tc_old)) * 2;
76     for (int i = 0; i < concentrace.n_cols; i++)
77     {
78         if (isnan(concentrace(i)))
79         {
80             concentrace(i) = 0;
81         }
82     }
83
84     if (out.is_open())
85     {
86         out << arma::norm(concentrace, 2) << "/___/" << curr_time_step <<
87             ↪ "/___/" << t_index << "/___/" << t << endl;
88     }
89
90     if (eps > 0) {
91         if (arma::norm(concentrace, 2) < trans.adapt_eps * 0.5)
92         {
93             curr_time_step = curr_time_step * 1.1;
94         }
95         else if (arma::norm(concentrace, 2) > trans.adapt_eps)
96         {
97             curr_time_step = curr_time_step * 0.5;
98             transport_concs = transport_concs_old;
99             t = t - curr_time_step;
100             goto label;
101         }
102     }
103     step_index++;
104 }
105
106 out.close();

```

```
106     log(BASIC) << "Calculation of time steps loop ended.";
107     react.close();
108 }
```

Appendix B

Original version of an input 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">2</Step>
29        <Step idTimeStep="20">13</Step>
```

```

30     </BorderCondChanges>
31 </Time>
32 <GridSize>
33     <Nrows>40</Nrows>
34     <Ncols>1</Ncols>
35 </GridSize>
36 <GridElements>
37 <Volume typeID="denseVector" dataTypeID="double" Nrows="40"
   ↪ quantityID="3">
38     <Data>5</Data>
39 </Volume>
40     <InOutFlow typeID="sparseVector" dataTypeID="double"
   ↪ Nrows="40" quantityID="4" idTimeStep="0">
41         <IndexData>
42             1         0.5
43             5         -1
44             40        0.5
45         </IndexData>
46     </InOutFlow>
47 </GridElements>
48 <GridElements2GridElements>
49 </GridElements2GridElements>
50 <InComponents idTimeStep="0">
51     <Component id="0" name="Pas2Phr" typeID="sparseMatrix"
   ↪ dataTypeID="double" Nrows="40" quantityID="101">
52         <IndexData>
53             5         50
54         </IndexData>
55     </Component>
56 </InComponents>
57 <TransportComponents idTimeStep="0" Time="0">
58     <Component id="0" name="Pas2Phr" typeID="denseMatrix"
   ↪ dataTypeID="double" Nrows="40" quantityID="101">
59         <Data>
60             3
61         </Data>
62     </Component>
63 </TransportComponents>
64 </TransportModule>
65
66 <ReactionModule name="PhreeqCRM" version="3.3.11-12535"
   ↪ runReactionStep="1">
67     <Nthreads>8</Nthreads>
68     <PhreeqcDatabase directoryPath="inputs">
69         phreeqc.dat
70     </PhreeqcDatabase>

```

```

71     <OnInit writeOutput="true" errorHandlerMode="1"
      ↪ componentH2O="false" rebalanceFraction="0.5"
      ↪ rebalanceByCell="true" useSolutionDensityVolume="false"
      ↪ setPartitionUZSolids="false" unitsSolution="1"
      ↪ unitsPPassemblage="1" unitsExchange="1" unitsSurface="1"
      ↪ unitsGasPhase="1" unitsSSassemblage="1" unitsKinetics="1">
72         <InitCond mapValue="3"
      ↪ directoryPath="inputs">komain3.phr</InitCond>
73         <BorderCond mapValue="50" directoryPath="inputs"
      ↪ idTimeStep="0">kc_amd.phr</BorderCond>
74     </OnInit>
75     <OnActivation initFrom="MapComponent" borderCondChange="true">
76         <InitCond>
77             <MapComponent componentID="0"/>
78         </InitCond>
79     <BorderCond>
80         <IDTimeStep>0</IDTimeStep>
81 </BorderCond>
82 </OnActivation>
83 <RunCells timeConversion="1.1574e-5" representativeVolume="1"
      ↪ saturation="1" density="1" pressure="1" temperature="20">
84     <Run idTimeStep="0" time="0">
85         </Run>
86 </RunCells>
87 <PhreeqcOutputs>
88     <RunOutputs directoryPath="Results" prefixFilename="pichem_amd_1D"
      ↪ writeCells="all" writeSteps="all">
89         <WriteCell id="0"/>
90         <WriteStep id="0"/>
91     </RunOutputs>
92     <DumpFile></DumpFile>
93 </PhreeqcOutputs>
94 </ReactionModule>
95
96 <ResultFile directoryPath="Results" filename="results.xml"
      ↪ useCsvSubdirs="true">
97     <Transport writeCells="all" writeSteps="all">
98         <LoopStep>1</LoopStep>
99     </Transport>
100    <Reaction writeCells="all" writeSteps="all">
101        <LoopStep>1</LoopStep>
102    </Reaction>
103 </ResultFile>
104
105 </TRM>

```


Appendix C

TRM2D structure description

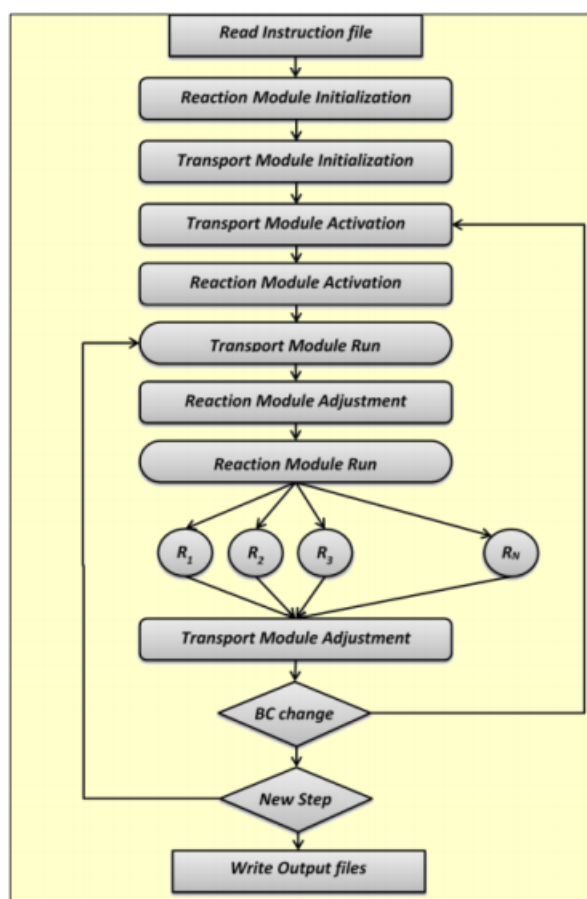


Figure C.1: Basic structure of the TRM2D software. (Adopted from [4])