Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Operator splitting method for transport reactive problem solution

Metoda štěpení operátoru pro řešení transportně-reakčních úloh

BACHELOR'S DEGREE PROJECT

Author:	Leonid Samoilov
Supervisor:	doc. Ing. Jan Šembera, Ph.D.
Language advisor:	Mgr. Hana Čápová
Academic year:	2020/2021

Katedra: matematiky Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Leonid Samoilov
Studijní program:	Aplikace přírodních věd
Studijní obor:	Aplikovaná informatika
Název práce (česky):	Metoda štěpení operátoru pro řešení transportně-reakčních úloh
Název práce (anglicky):	Operator splitting method for transport reactive problem solution

Pokyny pro vypracování:

1) Prostudujte vybranou literaturu o matematickém modelování v oblasti transportně reakčních úloh, zejména metodě konečných prvků, metodě štěpení operátoru a metodách pro řešení soustav obyčejných diferenciálních rovnic.
2) Převezměte software TRMgui vzniklý v rámci projektu TH02030840 „Paralelizovaný reakčně-transportní model šíření kontaminace v podzemních vodách (PaReTran)" a popište metodu řešení použitou v tomto softwaru.
3) Navrhněte vhodnou metodu kontroly časového kroku pro metodu štěpení operátoru v transportně reakční úloze realizované softwarem TRMgui.
4) Implementujte a proveďte základní testování kontroly časového kroku.

Doporučená literatura:

1) 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.
2) O. Zienkiewicz et al., The Finite Element Method: Its Basis and Fundamentals. Elsevier Butterworth- Heinemann, Oxford, 2005.
3) A. Polyanin, V. Zaitsev, Handbook of Ordinary Differential Equations: Exact Solutions, Methods, and Problems. Chapman and Hall/CRC, Boca Raton, 2017.

Jméno a pracoviště vedoucího bakalářské práce:

doc. Ing. Jan Šembera, Ph.D.
Fakulta mechatroniky, informatiky a mezioborových studií (FM), Technická univerzita v Liberci (TUL), Hálkova 6, 461 17 Liberec

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2020

Datum odevzdání bakalářské práce: 7.7.2021

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 30.10.2020

Masáková
garant oboru

[illegible]
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská
-1-

[illegible]
děkan

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.

In Prague July 6, 2020 Leonid Samoilov

Acknowledgment

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

Leonid Samoilov

Název práce:
Metoda štěpení operátoru pro řešení transportně-reakčních úloh

Autor:	Leonid Samoilov
Obor:	Aplikovaná informatika
Druh práce:	Bakalářská práce
Vedoucí práce:	doc. Ing. Jan Šembera, Ph.D. Fakulta mechatroniky, informatiky a mezioborových studií (FM), Technická univerzita v Liberci (TUL)
Konzultant:	Mgr. Hana Čápová

Abstrakt: Hlavním cílem tohoto projektu je studium a implementace metody štěpení operátoru pro řešení úloh popsaných soustavou parciálních diferenciálních rovnic, která spočívá v rozdělení transportně-reakční úlohy na transportní část a reakční část. V tomto projektu se zabýváme především analýzou metody štěpení operátoru, návrhem algoritmu automatické volby časového kroku a úpravou existujícího softwaru TRM 2D realizujícího metodu štěpení operátoru pro transportně-reakční úlohu a jeho testováním.
Klíčová slova: Numerická matematika, parciální diferenciální rovnice

Title:
Operator splitting method for transport reactive problem solution

Author: Leonid Samoilov

Abstract: The main goal of this bachelor project is to study and implement the operator splitting method to solve the problems described by the system of partial differential equations, which uses the principle of the division of the transport-reaction problem into a transport part and a reaction part. In this project we work with the analysis of the operator splitting method, we construct an automatic time step control algorithm for the transport-reaction problem, we implement this modification to the existing TRM 2D software and provide testing.
Key words: Numerical mathematics, partial differential equations

Contents

Introduction

There are various methods which solve problems such as a reactive transport problem, described by a system of the partial differential equations, but the operator splitting method (OSM) is the most widely used among them. The operator splitting method consists in dividing the problem into two parts: the transport part and the reaction part. The transport part runs in space and is described by a partial differential equation for each transported component separately. The reaction part, which runs only in time, is described by a system of ordinary differential equations, which describe chemical transformations between transported components. Such a separation allows to solve the problem in parts, using the most appropriate and at the same time the simplest methods for each part. For instance, the finite difference method can be used for the transport part, and Runge-Kutta method for the reaction part. In addition to that, use of the operator splitting method helps to save computing capacity, which leads to the opportunity to run the implemented reactive transport problem solver using personal computer. Such a solver, which is called TRM 2D (or TRM GUI) was created by the DHI company.
TRM 2D is based on the idea of the operator splitting method, which is implemented in it. Nevertheless, the discretization process of the OSM requires a method to control time step. Such a method was not the part of the original TRM 2D software since the TRM 2D calculates the problem using the constant time step.
In this project we scrupulously analyze methods, implemented in TRM 2D software, propose a time step control method for the operator splitting method discretization and implement it.
The study is divided into three main chapters: a brief description of the transport - reaction problem (1), the representation of the most common methods, which are used in the considered software (2) and the suggestion and implementation of the time step control method for the OSM inside the TRM 2D (3). The third part also includes general test presentation and the observations about accuracy of the calculated results.

Chapter 1

Physico - mathematical model of transport - reaction problem

1.1 Basic concepts

This chapter is based on the text of Milan Hokr's university study book [2] and it provides the basic mathematical - physical description of transport - reaction processes in a porous medium using a transport – reaction equation. The equation consists of parts describing solution flow, advection processes, diffusion, and hydrodynamic dispersion. It also includes the reaction part.

First of all, the terms porous medium and continuum model must be introduced. The term porous medium refers to a structure composed of grains or fibers of a solid substance (matrix) between which there are free spaces (pores) that can be filled with air or liquid. The term continuum model is meant to simplify the description of transport processes in a porous medium as a result of the approximation of the porous medium as continuous. Working with a continuum model, it is necessary to propose the term representative elementary volume (REV) and the term porosity ratio. REV is the volume large enough to cover the microscopic inhomogeneity (i.e., a size on the order of magnitude larger than the characteristic grain size of the solid substance) and small enough concerning the size of the examined area. The porosity ratio n is a value, which is equal to the proportion of pore volume to total material volume

$$n = \frac{\text{pore volume in REV}}{\text{REV volume}} \tag{1.1}$$

for REV surrounding a given point in space.

1.1.1 Darcy's law

Darcy's Law was formulated on the basis of an experiment in one-dimensional space, which describes the flow of water through a longitudinal porous filter in a tube and which can be represented as a tube filled with a porous material placed in a generally inclined position, connecting two tanks with different water levels. Describing Darcy's experiment, the following expression can be used for calculating water flow Q through the given pipe

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

where K is the permeability coefficient (depending on the properties of the porous material and the liquid), S is the cross-sectional area of the pipe, L the length of the pipe and the term in brackets is the difference in levels in both tanks. The symbol ϕ denotes the hydraulic head, the quantity assigned to the places at the ends of the pipe is defined by the relation

$$\phi = z + \frac{p}{\rho g}, \tag{1.3}$$

where z is the vertical coordinate, p is the pressure, ρ is the density and g is the gravitational acceleration. It is clear that the hydraulic head includes the potential of the gravitational field (z-position) and the pressure potential (hydro-static pressure of water above the given place). It is important to say that the term $(\frac{p}{\rho g})$ of the equation is defined as the pressure head. Next it is necessary to describe the water flow at a local point (given place), so the term water flux q was defined as the ratio of the amount of flowing water and the size of the area which is perpendicular to the flow direction $q = \frac{Q}{S}$. Considering the limit in the longitudinal direction $\nabla\phi = \frac{\phi_2 - \phi_1}{L}$, it is possible to get Darcy's law in the form

$$q = K\nabla\phi; \tag{1.4}$$

q has the dimension of speed, so it is commonly called Darcy's velocity, but this velocity can not be described as macroscopic motion of any point with speed.
To designate how fast will the selected water particle (or some dissolved substance) move through the porous medium, it is possible to use

$$v = \frac{q}{n}. \tag{1.5}$$

This velocity is called the flow velocity.

In the case of multiple dimensions (2D for TRM 2D software, which is used in this Bachelor project and 3D for the general case), the hydraulic head and the Darcy's velocity vector are functions of spatial coordinates and time. That means that Darcy's law equation for multiple dimensions can be described as follows:

$$q = \boldsymbol{K}\nabla\phi, \tag{1.6}$$

where ∇ is the gradient (for 3D: $\nabla\phi = \frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z})$ and $\mathbf{K}$. is the tensor which represents hydraulic conductivity, which depends on the properties of the porous setting and the properties of the liquid itself.
In any chosen volume (REV), the change in the weight of liquid, flowing through the chosen volume, corresponds to the weight of liquid passed across the border and to

the change in slumps and sources. These changes may be explained using the weight balance equation:

$$\frac{\partial}{\partial t}\int\limits_V \rho n \,\mathrm{d}V = -\int\limits_{\partial V} \mathbf{q}c \,\mathrm{d}\mathbf{S} + \int\limits_V \left(P^+c^* + P^-c\right) \,\mathrm{d}V + \int\limits_V r \,\mathrm{d}V, \tag{1.7}$$

where $\mathrm{d}\mathbf{S}$ can be represented as $\mathbf{n}\mathrm{d}S$, where $\mathbf{n}$ is the outer unit normal vector. Equation (1.7) can be rewritten using Gauss' theorem to

$$\frac{\partial(\rho n)}{\partial t} + \mathrm{div}(\rho\mathbf{q}) = P\rho. \tag{1.8}$$

Due to the fact that the simplified model of the solute transport equation was being used (so density and porosity are constant), equation (1.8) transforms into

$$\mathrm{div}\ \boldsymbol{v} = \frac{P}{n}. \tag{1.9}$$

Now it is possible to combine (1.6) and (1.9) equations in the system of two equations or even unite these two equations into one equation

$$\nabla \cdot (\boldsymbol{K}\nabla\phi) = P, \tag{1.10}$$

for the unknown function of the hydraulic head ϕ.

1.1.2 Balance of quantity equation. Advection. Molecular diffusion. Mechanical dispersion.

The balance equation has different forms for each type of process but in the general schematic way balance equation can be deduced as follows:

$$\frac{\text{accumulation of X in V}}{\mathrm{d}t} = \text{inflow of X to V}-\text{outflow of X from V}+\text{production of X in V} \tag{1.11}$$

where the particular parts describe different types of processes (advection, diffusion, dispersion). Converting the basic schematic description of the equation of balance (1.11) into a mathematical form (integral), for any chosen volume through which some water passes, the change in the weight of water needs to correspond to the weight of water passed across the border and to the change in sinks and sources, i.e.

$$\frac{\partial}{\partial t}\int\limits_V \rho n \,\mathrm{d}V = -\int\limits_{\partial V} \rho\mathbf{q} \,\mathrm{d}\mathbf{S} + \int\limits_V P\rho \,\mathrm{d}V, \tag{1.12}$$

where q is Darcy's velocity, ρ is the density and P is the density of sources (+) or sinks (-) expressed as the volume of water injected into the unit volume of the porous material per unit of time $[m^3/m^3/s]$. Now there is a need to examine the term of the equation (1.12) $-\int\limits_{\partial V} \rho\mathbf{q}\,\mathrm{d}\mathbf{S}$ as inflow and outflow parts of the basic schematic balance equation (1.11). The movement and transmission of substance X are described by physical processes which is called advection and diffusion - dispersion processes.

Advection means the transfer of a substance due to the movement of the whole solution. The amount of transferred substance can be easily expressed using Darcy's velocity of flow. The amount of substance passed through a unit of area per unit of time as follows

$$\boldsymbol{q_{adv}} = c \cdot \boldsymbol{q}, \tag{1.13}$$

where c is the concentration of substance and $\boldsymbol{q}$ is Darcy's velocity.

Diffusion - dispersion processes can be described as processes in which the solute moves due to the concentration gradient, from places of higher concentration to places of lower concentration. Considering porous environment, it is possible to present diffusion-dispersion processes as two processes: molecular diffusion and mechanical dispersion. Molecular diffusion is influenced by the microscopic structure of the environment and mechanical dispersion, which is caused by inhomogeneity of velocity in the pores - in some places the solution moves faster and in some slower. Molecular diffusion as well as mechanical dispersion is described by the same law, which is called Fick's law and is represented by the same equation $\frac{\partial c}{\partial t} = \mathbf{D} \triangle c = \mathbf{D} \sum\limits_i \frac{\partial^2 c}{\partial x_i^2}$ with one difference in coefficient $\mathbf{D}$. Describing molecular diffusion it is necessary to introduce molecular diffusion tensor $\boldsymbol{D}_m$, but for mechanical dispersion there is a need to introduce mechanical dispersion tensor $\boldsymbol{D}_f$. In total, diffusion - dispersion processes, consisting of molecular diffusion and mechanical dispersion are called the hydrodynamic dispersion. The hydrodynamic dispersion is characterized by a hydrodynamic dispersion tensor $\boldsymbol{D}_h$, which is defined as $\boldsymbol{D}_h = \boldsymbol{D}_m + \boldsymbol{D}_f$. So to describe diffusion - dispersion transport processes it is possible to deduce the equation

$$\boldsymbol{q_{dif-dis}} = \boldsymbol{D}_h \bigtriangledown c. \tag{1.14}$$

Now the equation for the total mass flow due to advection and hydrodynamic dispersion processes in the porous setting can be presented as the "sum" of two equations (1.13) and (1.14)

$$\mathbf{q} = nc\boldsymbol{v} + n\boldsymbol{D}_h \bigtriangledown c. \tag{1.15}$$

Returning to the balance of quantity equation (1.12), it is possible to reduce it to the form

$$\frac{\partial}{\partial t} \int\limits_V nc \, \mathrm{d}V = - \int\limits_{\partial V} \mathbf{q}c \, \mathrm{d}\mathbf{S} + \int\limits_V \left(P^+ c^* + P^- c\right) \, \mathrm{d}V + \int\limits_V r \, \mathrm{d}V, \tag{1.16}$$

where P^+ a P^- express the positive and negative parts of the source solution density function and express the fact that the solution with a given concentration c^* is injected, while the solution with concentration c corresponding to the required value of function $c(x,t)$ is pumped out from a given place and r represents the reaction part of the equation. By standard adjustment of integral equation (1.16) it is possible to get the needed differential equation of solute transport as follows

$$\frac{\partial c}{\partial t} = - \bigtriangledown \cdot (c\boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c) + \frac{1}{n}(P^+ c^* + P^- c) + \frac{r}{n}. \tag{1.17}$$

Now the equation needs to be transformed according to the amount of different particular concentration components.

To consider transport reactive problem with more concentration components, then equation (1.17) can be rewritten as

$$\frac{\partial c_i}{\partial t} = -\bigtriangledown \cdot (c_i \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_i) + \frac{1}{n}(P^+ c_i^* + P^- c_i) + \frac{r_i(c_1, \ldots, c_m)}{n}, \qquad (1.18)$$

where $i \in 1, \ldots, m,\ m \in \mathbb{N}$.

1.1.3 Calcite dissolution model in transport reaction equation

This subsection is based on Lukáš Zedek's dissertation [7] and it provides basic description of the reaction part in the transport reaction equation, which was derived in subsection 1.1.2. For clarity the example of the calcite dissolution reaction in water containing CO_2 without contact with the atmosphere is used:

- <1> $CO_2(aq) + H_2O \rightleftarrows H^+ + HCO_3^-$, constant K
- <2> $CaCO_3 + H^+ \rightleftarrows Ca^{2+} + HCO_3^-$, constants l, L
- <3> $H_2O \rightleftarrows H^+ + OH^-$, constant M.

The constant l is kinetic (refers to slower reactions, which are usually described using ODE - Ordinary differential equations), while the capital letters K, L, M denote equilibrium constants (refer to faster reactions, which are usually described using algebraic equations). To facilitate the perception of the mathematical model, symbols and designations changed according to the table

Table 1.1: Designations change

	$t = 0$	$t + t_1,\ t_1 > 0$
c_1,[$CO_2(aq)$]	$a = c_1(0)$	$a - \xi_1$
c_2,[H^+]	$b = c_2(0)$	$b + \xi_1 - \xi_2 + \xi_3$
c_3,[HCO_3^-]	$g = c_3(0)$	$g + \xi_1 + \xi_2$
c_4, [Ca^{2+}]	$d = c_4(0)$	$d + \xi_2$
c_5, [OH^-]	$e = c_5(0)$	$e + \xi_3$
c_6,[$CaCO_3$]	$f = c_6(0)$	$f - \xi_2$

The letters ξ_1, ξ_2, ξ_3 in expressions in the table 1.1 indicate the so-called extent of chemical reactions (usually measured in moles). The expressions with ξ_1, ξ_2, ξ_3 describe the concentration change of the specific specie using extent of reaction and create the system of equations. Describing equilibrium constants K, L and M it is possible to denote the following equations:

$$K = \frac{c_1^e\, c_3^e}{c_1^e} \qquad L = \frac{c_4^e\, c_3^e}{c_2^e} \qquad M = c_2^e\, c_5^e, \qquad (1.19)$$

where letters $c_1^e, c_2^e, c_3^e, c_4^e, c_5^e$ represent the values of the concentrations of the corresponding species in the chemical equilibrium. After all needed substitutions in accordance with table 1.1, the reactions transform into:

$$\begin{aligned}
&<1> \rightarrow \ \frac{c_1^e\, c_3^e}{c_1^e},\\
&<2> \rightarrow \ \frac{dc_4}{dt} = \frac{d(d+\xi_2)}{dt} = \frac{d\xi_2}{dt} = l \cdot (1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}),\\
&<3> \rightarrow \ c_2^e\, c_5^e.
\end{aligned} \tag{1.20}$$

Mathematical formulation of the calcite dissolution in the form of Differential Algebraic Equations system (DAE) looks as follows:

$$\begin{aligned}
&K = \frac{c_1^e\, c_3^e}{c_1^e} = \frac{(b+\xi_1-\xi_2+\xi_3)(g+\xi_1+\xi_2)}{(a-\xi_1)},\\
&\frac{d\xi_2}{dt} = l \cdot \left(1 - \frac{(g+\xi_1+\xi_2)\cdot(d+\xi_2)}{(b+\xi_1-\xi_2+\xi_3)\cdot L}\right),\\
&M = c_2^e\, c_5^e = (b+\xi_1-\xi_2+\xi_3)\cdot(e-\xi_3).
\end{aligned} \tag{1.21}$$

A clear disadvantage of the DAE formulation is the usage of extents of reaction instead of usual concentrations as a variable. This fact makes it almost impossible to extend the system of equations with transport, what is exactly the goal of this part of the project. That is why the DAE system (1.21) was transformed into another system of equations with new elements R_m and R_k (1.22). These elements express the contributions of equilibrium reactions to the time change of the concentration of reacting species and species, generated by reactions. R_m and R_k are unknown functions of concentrations.

$$\begin{aligned}
&K = \frac{c_2\, c_3}{c_1}\\
&M = c_2\, c_5\\
&\frac{dc_1}{dt} = (-1)\cdot R_k\\
&\frac{dc_2}{dt} = (-1)\cdot l \cdot \left(1 - \frac{c_3\, c_4}{c_2}\cdot\frac{1}{L}\right) + R_k + R_m\\
&\frac{dc_3}{dt} = l \cdot \left(1 - \frac{c_3\, c_4}{c_2}\cdot\frac{1}{L}\right) + R_k\\
&\frac{dc_4}{dt} = l \cdot \left(1 - \frac{c_3\, c_4}{c_2}\cdot\frac{1}{L}\right)\\
&\frac{dc_5}{dt} = R_m
\end{aligned} \tag{1.22}$$

In contrast with the DAE description of the calcite dissolution problem, the purely differential description presented above can easily be included in the Partial Differential Equation (PDE) transport equations. Including the transport variable changes the system (1.22) transforms into:

$$\begin{aligned}
&K = \frac{c_2\, c_3}{c_1},\\
&M = c_2\, c_5,\\
&\frac{\partial c_1}{\partial t} = L(c_1) - R_k,\\
&\frac{\partial c_2}{\partial t} = L(c_2) - l \cdot \left(1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}\right) + R_k + R_m,\\
&\frac{\partial c_3}{\partial t} = L(c_3) + l \cdot \left(1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}\right) + R_k,\\
&\frac{\partial c_4}{\partial t} = L(c_4) + l \cdot \left(1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}\right),\\
&\frac{\partial c_5}{\partial t} = L(c_5) + R_m.
\end{aligned} \tag{1.23}$$

where $L(c_i)$ is the transport term, which describes advection, diffusion and dispersion processes.

To calculate R_m and R_k the system of Ordinary Differential Equations, composed of derivations of equilibrium equations from (1.22) can be used as follows:

$$\begin{aligned}
&0 = \frac{\partial K}{\partial t} \cdot c_1^2 = \frac{\partial c_2}{\partial t} \cdot c_3\, c_1 + \frac{\partial c_3}{\partial t} \cdot c_2\, c_1 - \frac{\partial c_1}{\partial t} \cdot c_2\, c_3\\
&0 = \frac{\partial M}{\partial t} = \frac{\partial c_2}{\partial t} \cdot c_5 + \frac{\partial c_5}{\partial t} \cdot c_2.
\end{aligned} \tag{1.24}$$

Now it is possible to insert ODE equations from (1.22) into the system of obtained derivatives (1.24) and to calculate R_m and R_k corrections of concentrations according to equilibrium reactions. After all needed mathematical adjustments R_m and R_k may be represented as follows:

$$\begin{aligned}
R_m &= \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3)c_2 + c_1\, c_3 + (c_1 + c_3)c_5},\\
R_k &= -\frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3\, c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3)c_2 + c_1\, c_3 + (c_1 + c_3)c_5}.
\end{aligned} \tag{1.25}$$

Now it is possible to put R_m and R_k into the system (1.23), which creates the reactive transport system of equations (1.26) which is used in a solver - in TRM 2D software, which was mentioned above:

$$
\begin{aligned}
\frac{\partial c_1}{\partial t} &= - \bigtriangledown \cdot (c_1 \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_1) + \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right))}{(c_1 + c_3)c_2 + c_1\ c_3 + (c_1 + c_3)c_5}, \\
\frac{\partial c_2}{\partial t} &= - \bigtriangledown \cdot (c_2 \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_2) - l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right) - \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right))}{(c_1 + c_3)c_2 + c_1\ c_3 + (c_1 + c_3)c_5} + \\
&+ \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3)c_2 + c_1\ c_3 + (c_1 + c_3)c_5}, \\
\frac{\partial c_3}{\partial t} &= - \bigtriangledown \cdot (c_3 \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_3) + l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right) - \frac{c_1 \cdot (c_2 - c_3 + c_5) \cdot l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right))}{(c_1 + c_3)c_2 + c_1\ c_3 + (c_1 + c_3)c_5}, \\
\frac{\partial c_4}{\partial t} &= - \bigtriangledown \cdot (c_4 \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_4) + l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right), \\
\frac{\partial c_5}{\partial t} &= - \bigtriangledown \cdot (c_5 \boldsymbol{v}) + \bigtriangledown \cdot (\boldsymbol{D}_h \bigtriangledown c_5) + \frac{(2c_1 + c_3) \cdot c_5 \cdot l \cdot \left(1 - \frac{c_3\ c_4}{c_2} \cdot \frac{1}{L}\right)}{(c_1 + c_3)c_2 + c_1\ c_3 + (c_1 + c_3)c_5}.
\end{aligned}
\tag{1.26}
$$

Chapter 2

Operator splitting method. Finite difference method. General methods to solve the reaction part

2.1 Operator splitting method (OSM)

This part is based on [1].
The system (1.26) is constructed and it is obvious that now the main question is how to solve the system of equations of mixed type and which method would be appropriate. If two parts of equations (transport and reaction parts) are connected, then it is very difficult to propose the appropriate method, which could possibly solve the problem. But if it is possible to suggest to divide each equation from the system into two parts, then there is an opportunity to use proper methods such as Finite Difference Method for transport part and the method such as Runge-Kutta method, which is used in phreeqcRM library (a reaction module for transport simulators based on the geochemical model Phreeqc)[6].
It is very important to introduce operator splitting methods which will help to provide "dividing" of transport reaction problem with respect to the time step and initial and boundary conditions.

2.1.1 Different types of OSM

It is obvious from the previous text that the operator splitting method is based on the "separation principle". Basically different operator splitting methods separate the original equation into two or more parts over a specific time step, separately compute the solution to each part, and then combine the two separated solutions to form a solution to the original equation.
To introduce most "popular" operator splitting methods let us propose the basic

mixed type equation. It will help to to demonstrate the methods considered.

$$\frac{\partial U(t)}{\partial t} = AU(t) + BU(t) \text{ with } t \in [0, T], \qquad U(0) = U_0. \tag{2.1}$$

For example, Strang splitting (one of the most popular and widely used operator splitting methods) is based on the idea that first of all the first sub-problem for a half time step needs to be solved. Then the second sub-problem must be solved for a full-time step, and at the end it is necessary return to the first sub-problem and solve it for a half time step.

$$\begin{aligned}
\frac{\partial u^*(t)}{\partial t} &= Au(t) \text{ with } t \in [t^n, t^{n+1/2}], \qquad u(t^n) = u^n_{sp} \\
\frac{\partial v(t)}{\partial t} &= Av(t) \text{ with } t \in [t^n, t^{n+1}], \qquad v(t^n) = u^*(t^{n+1/2}) \\
\frac{\partial w(t)}{\partial t} &= Aw(t) \text{ with } t \in [t^{n+1/2}, t^{n+1}], \qquad w(t^{n+1/2}) = v(t^{n+1}),
\end{aligned} \tag{2.2}$$

where $t^{n+1/2} = t^n + 0,5\Delta t$, and the approximated split solution at the point $t = t^{n+1}$ is defined as $u^{n+1}_{sp} = w(t^{n+1})$.
Using Strang splitting it is possible to make the splitting algorithm second order accurate.

The other method is called Lie-Trotter splitting. Lie-Trotter splitting is a first order accurate splitting method which solves two sub-problems sequentially on subintervals $[t, t^{n+1}]$ where $t^{n+1/2} = t^n + 0,5\Delta t$ and $\Delta t = t_{n+1} - t_n$. Different subproblems are connected via the initial conditions as follows:

$$\begin{aligned}
\frac{\partial u(t)}{\partial t} &= Au(t) \text{ with } t \in [t^n, t^{n+1}] \text{ and } u(t^n) = u^n_{sp} \\
\frac{\partial v(t)}{\partial t} &= Bv(t) \text{ with } t \in [t^n, t^{n+1}] \text{ and } v(t^n) = u(t^{n+1}),
\end{aligned} \tag{2.3}$$

where $u^n_{sp} = U_0$. The approximated split solution at the point $t = t^{n+1}$ is defined as $u^{n+1}_{sp} = v(t^{n+1})$.

2.1.2 Application of OSM in transport-reaction problem

It is obvious from the previous section that to solve the reaction transport system of the equations (1.26) it is very helpful to use the operator splitting method.
So to describe the application of OSM to the problem, the new form of the transport reaction equation must be introduced.

$$\frac{\partial c_i}{\partial t} = \mathrm{L}(c_i) + \mathrm{R}(\mathbf{c}), \tag{2.4}$$

where $L(c_i)$ represents the transport part of the transport - reaction equation and $R(\mathbf{c})$ represents the reaction part of the same equation. It is also necessary to represent the graph below, which describes the process of evolution of the value of one concentration component over different time steps.

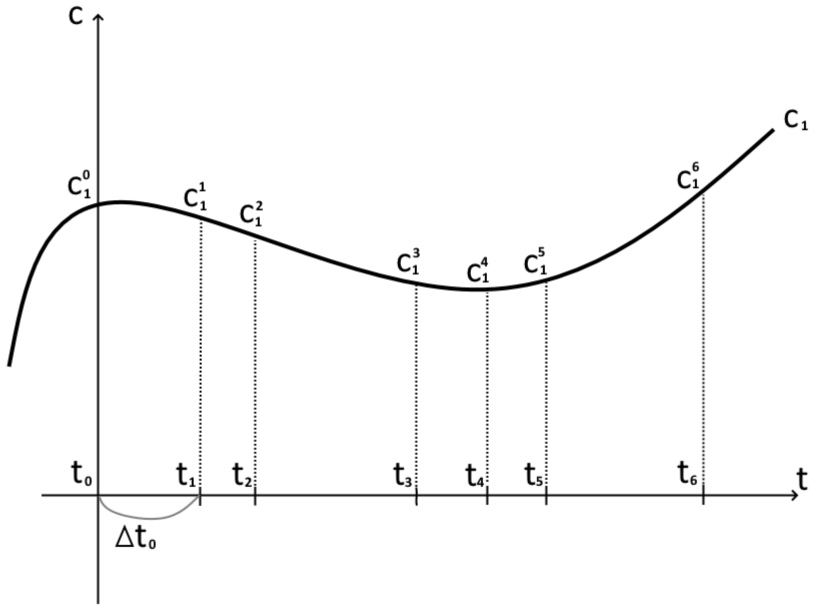

Figure 2.1: Operator splitting method

It is obvious from the graph that now it is possible to rewrite the transport reaction problem into the form of equation (2.5)

$$\frac{\partial c_i}{\partial t} \doteq \frac{c_i^{n+1} - c_i^n}{\Delta t_n} = L(c_i) + R(\mathbf{c}). \tag{2.5}$$

Then it can be transformed to

$$c_i^{n+1} = c_i^n + \Delta t_n \cdot (L(c_i) + R(\mathbf{c})) = c_i^n + \Delta t_n L(c_i) + \Delta t_n R(\mathbf{c}). \tag{2.6}$$

Finally, it is possible to divide the equation (2.4) into two parts. Each part may be solved separately, using the appropriate method.

$$\begin{cases} c_i^{(n+1)*} = c_i^n + \Delta t_n \cdot L(c_i^{(n+1)*}) \\ c_i^{n+1} = c_i^{(n+1)*} + \Delta t_n \cdot R(\mathbf{c}^{(n+1)*}) \end{cases} \tag{2.7}$$

General algorithm of the operator splitting method applied to the transport reaction problem may be formulated as follows:

1 Select a time step and initial/boundary conditions

2 Solve the first equation from the system (2.7)

3 Plug the result into the second equation from the same system (2.7) and solve it

4 Plug the results into the equation (2.5)

5 Repeat the algorithm (from the second point) for the next time step

2.2 Finite difference method

This section is based on Milan Hokr's university study book [4].
The finite difference method (FDM) consists in replacing derivatives with differences that approximate derivatives using the values of the searched function at several other points close to each. In the specific area in which we are looking for a solution, it is needed to choose a network of a finite number of points, so the result of the method is the approximated values of the function at these points.
To apply the finite difference method it is important to represent special difference formulas. Difference formulas are formulas for the approximation of derivatives using values of the specific function at network points. These formulas could be deduced using Taylor expansion of the function at adjoining points of the network.
There are three most important difference formulas for the first derivative: forward, backward and central. As an example it is possible to approximate $\frac{\partial c}{\partial t}$ using three different difference formulas, considering points $n+1$, n and $n-1$:

$$\begin{aligned}
&\text{Forward difference: } \frac{\partial c}{\partial t} = \frac{c_i^{n+1} - c_i^n}{\Delta t_n} + O(\Delta t_n) \\
&\text{Backward difference: } \frac{\partial c}{\partial t} = \frac{c_i^{n} - c_i^{n-1}}{\Delta t_{n-1}} + O(\Delta t_{n-1}) \\
&\text{Central difference: } \frac{\partial c}{\partial t} = \frac{c_i^{n+1} - c_i^{n-1}}{\Delta t_{n-1} + \Delta t_n} + O(\Delta t_{n-1} + \Delta t_n),
\end{aligned} \tag{2.8}$$

where Δt with index is the irregular time step.
By use of the operator splitting method, possibility to solve transport problem (convection - diffusion equation) separately from the reaction problem, using the finite difference method appeared. The convection equation (two-dimensional in space and one-dimensional in time) which is possible to solve using FDM may be represented in a general way as follows:

$$\frac{\partial c}{\partial t} = -v_1 \frac{\partial c}{\partial x} - v_2 \frac{\partial c}{\partial y}, \tag{2.9}$$

where x, y are space coordinates and v is the velocity, which is constant according to space.

Considering the transport problem (the convection equation) it is important to represent the explicit model which helps to simplify the approximation (and which

(from the mathematical point of view) makes it necessary to use forward difference formula on the left side of the convection equation) and the specific upwind approximation. The upwind approximation is natural for the convection equation. The physical process of the convection (in the sense of transporting a general quantity) is always associated with the movement in a certain direction and with a given orientation. The upwind approximation respects the orientation of the process - the spatial discretization is chosen asymmetrically, so for the calculation the value "against the direction of movement" is taken into account. The another advantage of the upwind approximation is better stability of the solution.
Applying the upwind approximation to (2.9) it is possible to deduce the following system:

$$\begin{aligned}
&\text{For } v_1 \geq 0, v_2 \geq 0: && \frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = -v_1 \frac{c_{i,j}^{n} - c_{i-1,j}^{n}}{\Delta x} - v_2 \frac{c_{i,j}^{n} - c_{i,j-1}^{n}}{\Delta y} + O(\Delta t, \Delta x, \Delta y)\\
&\text{For } v_1 \geq 0, v_2 < 0: && \frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = -v_1 \frac{c_{i,j}^{n} - c_{i-1,j}^{n}}{\Delta x} - v_2 \frac{c_{i,j+1}^{n} - c_{i,j}^{n}}{\Delta y} + O(\Delta t, \Delta x, \Delta y)\\
&\text{For } v_1 < 0, v_2 \geq 0: && \frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = -v_1 \frac{c_{i,j+1}^{n} - c_{i,j}^{n}}{\Delta x} - v_2 \frac{c_{i,j}^{n} - c_{i,j-1}^{n}}{\Delta y} + O(\Delta t, \Delta x, \Delta y)\\
&\text{For } v_1 < 0, v_2 < 0: && \frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = -v_1 \frac{c_{i,j+1}^{n} - c_{i,j}^{n}}{\Delta x} - v_2 \frac{c_{i,j+1}^{n} - c_{i,j}^{n}}{\Delta y} + O(\Delta t, \Delta x, \Delta y).
\end{aligned} \tag{2.10}$$

It is important to conclude that system such as (2.10) is implemented in TRM 2D software which is used in this project.

2.3 General methods to solve the reaction part

This section is based on the Jan Šembera's university study book [4].
The Phreeqc RM library, which was already mentioned as a "solver" for the reaction part of the transport reaction problem, consists of specific mathematical and numerical methods. These methods are relatively complicated to present short description, but they are based on the general mathematical problems and methods. So it is possible to introduce some of the general methods, which could possibly solve the reaction part of the transport - reaction problem.
It is important to include initial and boundary conditions into the problem, before the appropriate method was applied. Discretization and rounding errors should be also taken into account.

2.3.1 The Euler Method

The most widely known numerical method which can be used as a basis for the reaction part solver is the Euler Method.

For better understanding of the method, it is necessary to introduce the initial value problem in the form of the following equation and other important terms:

$$\begin{aligned} &\dot{c} = f(t, c), \\ &c(a) = \eta, \end{aligned} \tag{2.11}$$

where c represents the concentration, t is the time and η is the initial condition.

The Euler method is one of the discreet methods. Discreet methods are based on finding the approximate values of the function $c(x)$ in the finite number of points t_i from the interval $< a, b >$. Using such a method it is necessary to limit the choice of the point t_i to the equidistant choice:

$$t_i = a + i\Delta t, \qquad i = 0, 1, 2..., \tag{2.12}$$

where Δt is the integration step or the time step.

The Euler Method itself can be introduced in the following way. If the function f is represented as a directional field in the phase space $(t, c) \in < a, b > \times(-\infty, +\infty)$, then the graph of the solution of problem (2.11) is such a curve in this space that the directional field f is tangent to it at each point. Thus, using the Euler method, we gradually construct an angled line in phase space. The line must be such that each section between two points (t_i, c_i) a (t_{i+1}, c_{i+1}) has a direction determined by the directional field, at the initial point of the section $f(t_i, c_i)$. The point (t_0, c_0) is given by the initial condition. Gradually obtained values of c_i are considered as the approximations of the solution values at points t_i, i.e. $c(t_i)$. Now it is possible to represent the Euler Method in the following way:

$$\begin{aligned} &c_0 = \eta, \\ &c_{i+1} = c_i + \Delta t \cdot f(t_i, c_i), \qquad i = 0, ..., n-1, \end{aligned} \tag{2.13}$$

where $\Delta t = \frac{b-a}{n}$ is integration step or time step.

2.3.2 Discretization error

Describing discreet methods, it is very important to know the value or at least the estimate of the total discretization error. For the Euler method, described in the previous section, total discretization error may be represented by the following formula

$$e_i = c_i - c(t_i). \tag{2.14}$$

That formula, in general, expresses the deviation between the solution derived from the Euler method and the actual, true solution.

Total discretization error can be derived from two types of the local error. Local discretization error is the error, that is caused by the application of one step of the method. For the Euler method, the local discretization error may be represented using the following formula:

$$L(c(t);\ \Delta t) = c(t + \Delta t) - c(t) - \Delta t \cdot f(t, c(t)), \tag{2.15}$$

where the function $c(t)$ is the exact solution of the the given initial value problem. Total discretization error can be represented as a serie of local discretization errors, which were made in each integration step and error, which was made with the accordance to the fact that in each integration step we use inaccurate initial data.

It is necessary to mention that the Phreeqc RM library uses much more complex numerical methods that are constructed so that their discretization error is much lower than the one of the Euler methods. Another important remark should be added. The numerical solutions are influenced not only by the discretization error but also by the rounding error. The total rounding error can be described as an error, which was caused by the fact that the computer processor calculates the problem using the final number of decimal places. That means that the processor actually adds a little value to the result. This value is called the local rounding error.

Chapter 3

Time step control method for OSM in transport reaction problem implemented by TRM 2D

3.1 TRM 2D

3.1.1 Basic description

The TRM 2D transport-reaction model is based on the "Parallelized reaction-transport model of contamination spread in groundwater (PaReTran)" component [5]. TRM 2D is a combination of the PhreeqcRM geochemical library with 2D transport in a regular rectangular network of elements. TRM is implemented as a console application that works with settings in an XML file defined according to a specific template. In a rectangular network, the boundary and initial flow conditions can be set. These conditions can be changed during the calculation by setting the Border Conditions Changes. In the same way, boundary and initial conditions for the reaction part, defined in the Phreeqc configuration files can be set and controlled.
The basic structure of the TRM 2D configuration files can be demonstrated using the structure according to Fig. 3.1.

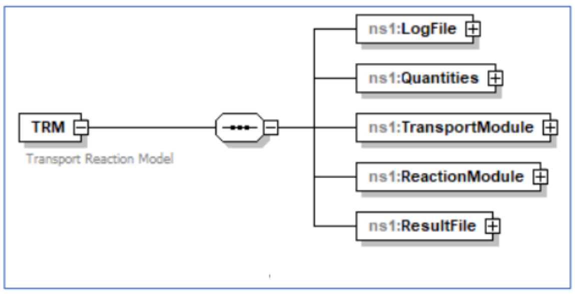

Figure 3.1: Configuration file structure. (Adopted from [5])

Within the LogFile element, the level of details of the reports and results is set to

the log file. That file is being used to store the information about software run.
The quantities element includes the units and their settings with which the TRM software works.
The element named "TransportModule" (Fig. 3.2) helps to set data on the required simulation period, the size of the rectangular grid of flow, transport of substances, information about individual transport elements, incoming components, transported and static components.

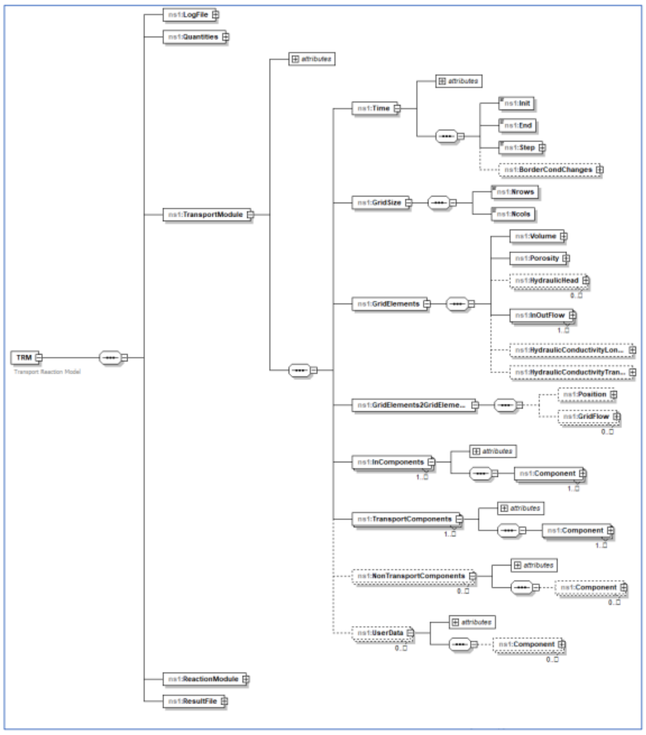

Figure 3.2: Transport Module definition. (Adopted from [5])

The ReactionModule element is used to handle a component built using the Phreeqc RM library. In this element it is necessary to set the number of threads the library will work with, initial and border conditions, the possible boundary conditions change during the simulation and other important conditions according to Fig. 3.3 and Fig. 3.4.

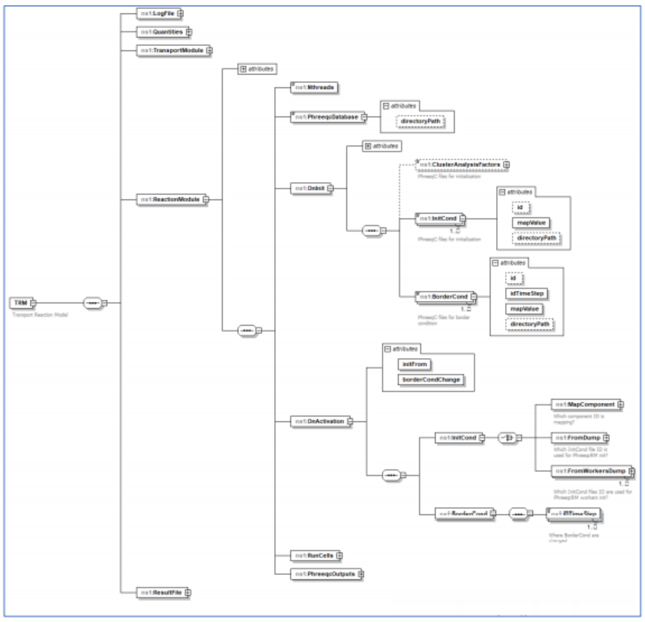

Figure 3.3: ReactionModule definition (part A). (Adopted from [5])

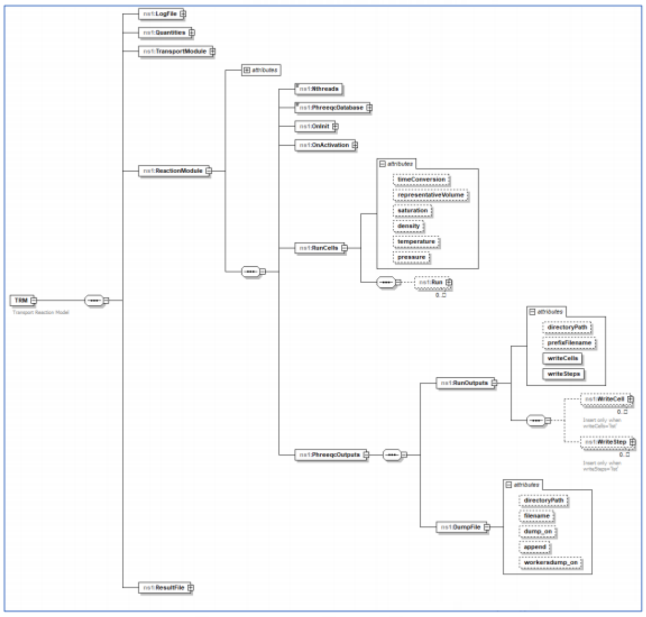

Figure 3.4: ReactionModule definition (part B). (Adopted from [5])

ResultFile is necessary to store the results. The result of the TRM program run is saved in the form of CSV files.

Now it is possible to shortly present the general algorithm of the TRM 2D software using Fig. 3.5 which illustrates gradual implementation of individual operations.

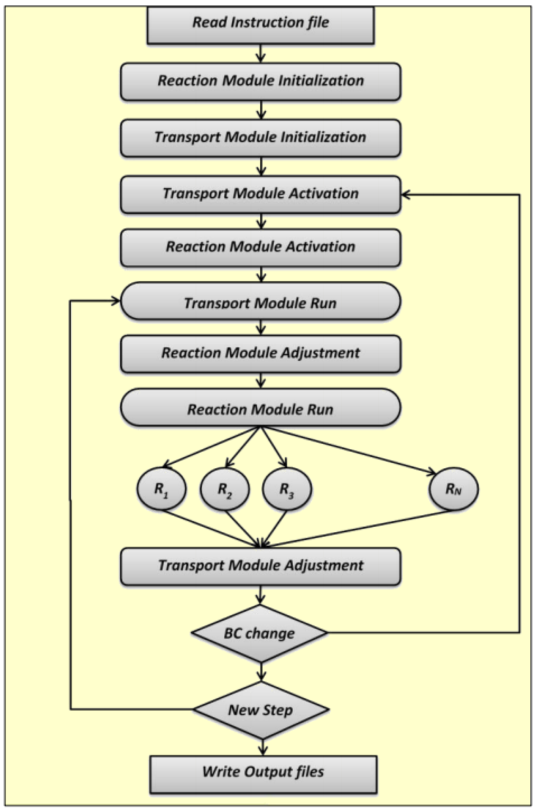
Read Instruction file
Reaction Module Initialization
Transport Module Initialization
Transport Module Activation
Reaction Module Activation
Transport Module Run
Reaction Module Adjustment
Reaction Module Run
R_1
R_2
R_3
R_N
Transport Module Adjustment
BC change
New Step
Write Output files

Figure 3.5: Description of TRM 2D software simulation run. (Adopted from [5])

It is also important to mention that the TRM 2D software includes TRM GUI, graphical interface, which is more convenient for user.
The figures from this section may be found in Appendix C.

3.1.2 Installation

It is obvious from the previous section that TRM 2D requires specific libraries and the special approach to installation and building. The original software, which was created by the DHI company, was intended to be compiled using QT. Working with the TRM 2D, we managed to install and compile it using Visual Studio 2019 on Windows. This possibility was not the part of the TRM 2D software before. The installation process among others included work with specific libraries such as Armadillo (linear algebra library) or PhreeqcRM (the geochemical reaction module). The detailed description of the TPM installation and compilation process is in Appendix A.

3.2 The time step control method

3.2.1 Basic algorithm

The main goal of this bachelor's degree project is to suggest the time step control method for the operator splitting method and implementation of that in TRM 2D software. To propose such a method, first of all, it was necessary to study the source code in detail, in search of the information about the methods, classes and variables we needed to work with. After analyzing the source code of most of the software files and comparing it with the available information about the program, we came to the conclusion that lines 19 - 23 from the following listing is the proper place to implement time step control method.

```cpp
1  unsigned step_index = 0;
2   for (t = trans.time_init + curr_time_step; t < trans.time_end + trans.time_end * NUMERIC_EPS; t = t + curr_time_step) {
3       progress_value = static_cast<unsigned>(100 * (static_cast<double>(t_index) - static_cast<double>(trans.get_time_init_id())) / static_cast<double>(time_steps_count));
4       if (trans.is_bc_change_time(t_index)) {
5           current_bc_time = t_index;
6           curr_time_step = trans.get_time_step(current_bc_time);
7           // initial conditions used only at zero time
8           if (t_index == initial_time_index) {
9               vector<double> tmp_values = arma::conv_to<vector<double>>::from(trans.get_component_data("TransportComponents", t_index, map_component_id));
10              vector_values init_cond(tmp_values);
11              react.activate(
12                  transport_concs, transport_names, init_cond, trans.get_component_data("InComponents", t_index, map_component_id), t_index, t - curr_time_step, trans.get_first_component_of_id("0"));
13              trans.init(transport_concs, transport_names, initial_time_index);
14          }
15          // transport activation follow-up, maybe make modified grid flow BC time dependant and move it to transport activation
16          trans.modify_grid_flow(t_index);
17      }
18      t_index++;
19      trans.calculate(t_index, t, transport_concs, transport_names, current_bc_time, mapping);
20      if (react.is_calculated_step(step_index)) {
21          react.calculate(t_index, t, curr_time_step, transport_concs, transport_names, mapping);
22      }
23      step_index++;
24  }
```

Listing 3.1: The part of trm.cpp file. Original version

This specific place was chosen because this is the part of the "for" cycle (lines 2 - 24) which is the part of the method `void transport_module::calculate(...)` and which

controls interaction of the transport and reaction part with time and the time steps. In the original version of the software, the time step was set by the user as constant using the specific XML file and class methods worked with an array of predefined time steps. Suggesting a method for automatic control of the time step, we added the ability to work with the variable `double curr_time_step` in the method `void calculate(...)` of the class `class transport_module`. Using this variable we will be able to implement the automatic time step change.
The main idea of the chosen automatic time step selection method may be represented using the following algorithm.

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

In the algorithm which is described above, the variable `concs` contains all concentrations in all elements during the current calculation unit run, the variable `concs_old` contains all concentrations in all elements at the beginning of the current calculation unit run and the variable `time step` contains the length of the actual time step. The condition checked in the "if" loop may be described as follows. In the first phase we verify if the geometrical distance between the results of two consecutive calculations in space, measured using the Euclidean norm, is significantly less than `eps`. Using the second "if" loop we check if the distance is bigger than `eps`. So it is obvious that the `eps` variable represents optimal value to compare results with. The value of `eps` is preset.

3.2.2 Implementation of time step control method in TRM 2D

To implement the algorithm from the previous section, we needed to construct the loops inside the cycle described in Listing 3.1.
First of all, the condition inside the "if" phase must be set. By use of the already connected Armadillo library, there is no need to access individual components of `concs` and `concs_old` using "for" cycle. Math operations such as subtraction and division are automatically preset to work with the individual components of the vector in the Armadillo library. The only problem was that the original datatype (`vector< double>`) of the `concs` and `concs_old` variables was not convenient with Armadillo, so we declared new variables `arma::rowvec tc` and `arma::rowvec tc_old`.

```
tc_old = arma::conv_to<arma::rowvec>::from(transport_concs);
tc = arma::conv_to<arma::rowvec>::from(transport_concs);
```

Listing 3.2: The declaration of new variables

These variables are intended to keep data, stored in `vector<double> concs` and `vector <double> concs_old` using Armadillo datatype `<arma::rowvec>`. Transformation from one type to another was committed using the method `arma::conv_to<arma::rowvec>:: from(...)`.
After all needed transformations, we insert the calculation result into the new variable `arma::rowvec concentrace`. Then using the preset function `(arma::norm(concentrace , 2)` we calculate the Euclidean norm and compare it with the optimal value `eps` to set the condition to the loops.

```
arma::rowvec concentrace = ((tc - tc_old) / (tc + tc_old)) * 2;
 if (arma::norm(concentrace, 2) < eps * 0.5)
 {
 ...
 }
 else if (arma::norm(concentrace, 2) > eps)
 {
 ...
 }
```

Listing 3.3: The Euclidean norm

The final version of the general algorithm is implemented as follows:

```
transport_concs_old = transport_concs;
     tc_old = arma::conv_to<arma::rowvec>::from(transport_concs);
     t_index++;
 label:
     if (t > trans.time_end)
     {
         curr_time_step = trans.time_end - t + curr_time_step;
         t = trans.time_end;
     }
     trans.calculate(t_index, t, transport_concs, transport_names, current_bc_time, mapping, curr_time_step);
     if (react.is_calculated_step(step_index)) {
         react.calculate(t_index, t, curr_time_step, transport_concs, transport_names, mapping);
     }
     tc = arma::conv_to<arma::rowvec>::from(transport_concs);
     arma::rowvec concentrace = ((tc - tc_old) / (tc + tc_old)) * 2;

     if (arma::norm(concentrace, 2) < eps * 0.5)
     {
         curr_time_step = curr_time_step * 1.1;
     }
     else if (arma::norm(concentrace, 2) > eps)
     {
         curr_time_step = curr_time_step * 0.5;
```

```
            transport_concs = transport_concs_old;
            t = t - curr_time_step;
            goto label;
        }
        step_index++;
```
Listing 3.4: The implementation of the general algorithm

All the other useful and valuable changes and additions to the source code from Listing 3.1 are described in the appendix B with extensive comments.

3.3 Tests

The task of complex and gradual testing of the changes made to the source code is of the same priority task as the task of implementation.
To test the suggested method we use the calcite dissolution problem, prescribed in the XML file. As compared outputs, CSV files with calcium concentration values were used. Using specific `<fstream>` library, we also constructed the possibility of listing the results of the software run to a text file.

```
fstream out;
    out.open("c:\\Trm\\norm.txt", fstream::out);
    .
    .
    .
     if (out.is_open())
        {
        out << arma::norm(concentrace, 2) << "/___/" <<
   curr_time_step << "/___/" << t_index << "/___/" << t << endl;
       }
    out.close();
```
Listing 3.5: Text file outputting

3.3.1 Basic tests

First of all, we need to test the impact of the intervention to the program code on the results.

It means that firstly, it is necessary to test the deviation between the original program run results (with the constant time step and constant presets) and the modified software (with the constant time step and constant presets).

Working with the calculation results it is necessary to examine the value κ, which may be called as a maximum deviation and can be represented as the maximum norm of the difference of the relative error. Using the maximum deviation from the calculations, it is possible to determine the average maximum deviation, which shows the actual deviation between the results.
Comparing the original software with the modified software, we took into account the results of five calculations. So we receive the following result: $\kappa \approx 1,95 \cdot 10^{-15}$.

Such a deviation may be caused by rounding error, which can be understood as a computing error. It is necessary to say that this deviation is absolutely acceptable.

3.3.2 Implemented method tests

Testing the implemented time step control method, we work with the `<eps>` variable, which controls loop conditions and manages the time step change. Installing different initial time steps and different values of `<eps>` we prove that the proposed method is functional and the error of the results is acceptable.
First of all we need to propose the values of the initial time steps, which are reasonable to use during tests. We choose to use constant time steps equal to 1, 0.1, 0.01 and the last value is 0.001 as follows.

Modified software with different time step results	
	κ
R_{0001}/R_1	$2.401 \cdot 10^{-4}$
R_{0001}/R_{01}	$1.817 \cdot 10^{-5}$
R_{0001}/R_{001}	$1.623 \cdot 10^{-6}$

It the table above, $R_{0001}, R_{001}, R_{01}, R_1$ denote the results of the modified software run with the constant time steps equal to 0.001, 0.01, 0.1, 1, respectively. Since we do not have the possibility to have the analytical solution of the studied problem, we choose the solution R_{0001} as the reference solution for measurement of errors of the other numerical solutions. These results in the table above express dependency between the accuracy of the results and the length of the time step. The value of $\kappa = 2.401 \cdot 10^{-4}$ corresponds to the value $\Delta t = 1$, $\kappa = 1.817 \cdot 10^{-5}$ corresponds to $\Delta t = 0.1$ and the value $\kappa = 1.623 \cdot 10^{-6}$ corresponds to the value $\Delta t = 0.01$. It means that by use of ten times bigger time step, we get roughly ten times less precise results.
It is important to mention that working with the software, we study the calculations which end up at time equal to 20 seconds. It means that to complete the calculations, the software with the constant time step needs to run through 20000, 2000, 200, 20 time steps for the time steps equal to 0.001, 0.01, 0.1, 1, respectively.
As it was mentioned above, we decided to use $\Delta t = 0.001$ as the reference solution. So to test the implemented method, we decide to compare the outputs of the modified software with constant time step equal to 0.001 and the outputs of the modified software with the initial time step equal to 1 and different values of `<eps>`. These results are represented in the following table.

Modified software with different `<eps>` results	
	κ
$R_{0001}/R_1(eps = 0.010293)$	$1.429 \cdot 10^{-6}$
$R_{0001}/R_1(eps = 0.012)$	$1.630 \cdot 10^{-6}$
$R_{0001}/R_1(eps = 0.021)$	$3.087 \cdot 10^{-6}$
$R_{0001}/R_1(eps = 0.05)$	$7.298 \cdot 10^{-6}$
$R_{0001}/R_1(eps = 0.089)$	$1.347 \cdot 10^{-5}$
$R_{0001}/R_1(eps = 0.11)$	$1.617 \cdot 10^{-5}$
$R_{0001}/R_1(eps = 0.21)$	$3.127 \cdot 10^{-5}$
$R_{0001}/R_1(eps = 0.5)$	$7.445 \cdot 10^{-5}$
$R_{0001}/R_1(eps = 0.79)$	$1.232 \cdot 10^{-4}$
$R_{0001}/R_1(eps = 0.91)$	$1.424 \cdot 10^{-4}$

Now it is possible to collect the results from the previous tests, to assemble them and to analyze them entirely. Summary of results of the main tests are presented in the following table, which shows the dependency between the preinstalled value of the `<eps>`, number of time steps and computational time.

Modified software with different `<eps>` results				
`<eps>`	κ	N	K	T
0.010293	$1.429 \cdot 10^{-6}$	2290	2297	173.749
0.012	$1.630 \cdot 10^{-6}$	2069	2076	155.117
0.021	$3.087 \cdot 10^{-6}$	1133	1139	137.930
0.05	$7.298 \cdot 10^{-6}$	505	510	63.748
0.089	$1.347 \cdot 10^{-5}$	276	280	66.573
0.11	$1.617 \cdot 10^{-5}$	235	239	54.615
0.21	$3.127 \cdot 10^{-5}$	124	127	14.052
0.5	$7.445 \cdot 10^{-5}$	55	58	6.551
0.79	$1.232 \cdot 10^{-4}$	35	36	5.295
0.91	$1.424 \cdot 10^{-4}$	31	32	4.472

In the table above, N is a number of time steps, K is a number which represents how many times the program run through the implemented cycle (the number of time steps including the ones that had to be restarted) and T is time, measured in seconds needed to make all the necessary calculations. It is obvious from the results that the adaptive time step control method changes the length of time steps as expected and the accuracy of the results that made 200 and 2000 steps is comparable to the accuracy of the results with preset constant time steps 0.1 and 0.01.

Conclusion

In this project, we became acquainted with the operator splitting method, which is exceedingly useful for the reactive transport problem and other problems described by a system of partial differential equations. We also became familiar with the TRM 2D, which was created by the DHI company as a reactive transport problem solver. Working with the software we proposed a method to control operator splitting method discretization process in accordance with the time step. The suggested algorithm of the time step control method was implemented in TRM 2D software source code and was tested. The algorithm and the results of tests are represented in this work. According to the results, we can say, that the algorithm works correctly. In addition to that, it allows us to expand the possibilities of the TRM 2D software, modifying proposed algorithm and solving some of the open problems. One of the open problems is to find a relation between the <eps> parameter and accuracy of the solution for a general problem definition. Another task for the future is the task of the suggestion of an alternative adaptive algorithm. Such an algorithm could be based on different parameters other than concentration difference between the start and the end of the time step.

Bibliography

[1] Diab. *Operator Splitting Methods.* Web page at STIMULATE European Joint Doctorates. http://www.stimulateejd.eu/content/operator-splitting-methods (2019).

[2] M. Hokr. *Transportní procesy* [in Czech]. Lecture notes. Faculty of Mechatronics, Technical University of Liberec. (2005).

[3] A. Polyanin, V. Zaitsev. *Handbook of Ordinary Differential Equations: Exact Solutions, Methods, and Problems.* Chapman and Hall/CRC. Boca Raton. (2017).

[4] J. Šembera, D. Frydrych. *Poznámky k předmětu stavba a řešení počítačových modelů* [pdf in Czech]. Faculty of Mechatronics, Technical University of Liberec. (2002).

[5] P. Štrof et al. *Final Report of the TACR project Nr. TH02030840* [in Czech]. DHI. Praha. (2019).

[6] USGS team. *PHREEQC Version 3.* Web page at USGS. https://www.usgs.gov/software/phreeqc-version-3 (2021).

[7] L. Zedek. *Modelování transportně - chemických procesů* [in Czech]. Doctoral Thesis. Faculty of Mechatronics, Technical university of Liberec. (2014).

[8] O. Zienkiewicz et al. *The Finite Element Method: Its Basis and Fundamentals.* Elsevier Butterworth - Heinemann. Oxford. (2005).

Appendix A

TRM 2D installation process decription

Algorithm is intended for Visual Studio 2019
Unzip the .zip files to the c:/Trm directory so that it contains the directories:
- armadillo-10.2.1
- phreeqcrm-3.6.2-15100
- TRMinstall

PhreeqcRM
———

In Visual Studio
Open Menu -> File -> Open -> CMake
c:/Trm/phreeqcrm-3.6.2-15100/CMakeLists.txt
Open Solution Explorer
Then file phreeqcrm-3.6.2-15100/CMakeLists.txt
Press the right mouse button
Then press Generate Cache for...
After that choose Build
And finally push Install
The results files may be found in following the directory
c:/Trm/phreeqcrm-3.6.2-15100/out/install/x64-Debug
Prepare a subdirectory "include" and a "lib" subdirectory with the PhreeqcRMd.lib library inside

Lapack
——

Put the following files to the
c:/Trm/lapack directory
- cbia.lib.blas.dyn.rel.x64.12.dll
- cbia.lib.blas.dyn.rel.x64.12.lib
- cbia.lib.lapack.dyn.rel.x64.12.dll
- cbia.lib.lapack.dyn.rel.x64.12.lib

These files may be found at cbia.lib.lapack.dyn.dbg.x64.12.zip or at https:// www.fi.muni.cz/ xsvobod2/misc/lapack/

It is also needed to download following files.

- cbia.lib.blas.dyn.rel.x64.12.dll
- cbia.lib.lapack.dyn.rel.x64.12.dll
- libifcoremd.dll
- libmmd.dll
- svml_dispmd.dll

These files may be found at www.dll-files.com

Armadillo

Edit the file c:/Trm/armadillo-10.2.1/cmake_aux/ Modules/ARMA_FindLAPACK.cmake

original part

FIND_LIBRARY(LAPACK_LIBRARY
NAMES $LAPACK_NAMES
PATHS /usr/lib64/atlas /usr/lib/atlas /usr/lib64 /usr/lib /usr/local/lib64 /usr/local/lib
)

must be replaced with

FIND_LIBRARY (LAPACK_LIBRARY
NAMES cbia.lib.lapack.dyn.rel.x64.12
PATHS c:/trm/lapack
)

And similarly in c:/Trm/armadillo-10.2.1/cmake_aux/Modules/ARMA_FindOpenBLAS.cmake

original part

find_library ($ OpenBLAS_NAME_LIBRARY
NAMES openblas libopenblas libopenblas.dll libopenblas.lib $ OpenBLAS_NAME
PATHS c:/trm/armadillo-10.2.1/examples/lib_win64 $ CMAKE_SYSTEM_LIBRARY_PATH / lib64 / lib / usr / lib64 / usr / lib / usr / local / lib64 / usr / local / lib / opt / local / lib64 / opt / local / lib / usr / lib / openblas / / usr / lib / openblas / lib / usr / local / opt / openblas / lib / opt / local / lib / openblas / opt / local / lib / openblas / lib
)

must be replaced with

find_library ($ OpenBLAS_NAME _LIBRARY
NAMES $ OpenBLAS_NAME
PATHS $ CMAKE_SYSTEM_LIBRARY_PATH / lib64 / lib / usr / lib64 / usr / lib / usr / local / lib64 / usr / local / lib / opt / local / lib64 / opt / local / lib / usr / lib / openblas / / usr / lib / openblas / lib / usr / local / opt / openblas / lib / opt / local / lib / openblas / opt / local / lib / openblas / lib

)

Then in the Visual Studio
Open Menu->File->Open->CMake
c:/Trm/armadillo-10.2.1/CMakeLists.txt
Then using Solution Explorer find armadillo-10.2.1/CMakeLists.txt file
Press the right mouse button
Then Generate Cache for ...
After that Build
And press Install
The results files can be found in the directory c:/Trm/armadillo-10.2.1/out/install/x64-Debug

To proceed with the installation process there was a need to edit some original source files. Following text shows what changes are needed to make the installation process real.

———————————————-

TRMInstall - Edit CMakeLists.txt

———————————————

add following lines to c:/Trm/TRMInstall/CMakeLists.txt

set (PHREEQCRM_PATH "c:\\Trm\\phreeqcrm-3.6.2-15100\\out\\install\\x64-Debug")
set (ARMADILLO_INCLUDE_DIRS "c: /Trm/armadillo-10.2.1/out/install/x64-Debug/include")
set (BLAS_LIBRARIES "c: /Trm/armadillo-10.2.1/examples/lib_win64/libopenblas.lib")
set (LAPACK_LIBRARIES "c: /Trm/armadillo-10.2.1/examples/lib_win64/libopenblas.lib")

instead of a backslash we use two backslash \\ or one ordinary slash /

Change the original line
target_link_libraries ($ PROJECT_NAME "$ PHREEQCRM_PATH / lib / libPhreeqcRM. $ LIBRARY_EXTENSION")

to
target_link_libraries ($ PROJECT_NAME "$ PHREEQCRM_PATH /lib/PhreeqcRMd.lib")
the letter d at the end of the library name (before .lib) is related to the x64-Debug configuration

Now change the original line
set (CMAKE_CXX_FLAGS "$ CMAKE_CXX_FLAGS -std = c ++ 14 -W -Wall -Wextra")

to
set (CMAKE_CXX_FLAGS "$ CMAKE_CXX_FLAGS")
These actions help us to remove configurations, "understandable" for gcc, but not for VS.

TRMInstall - Edit source texts
————————————

trm_module.h
———————

Add following lines
ifdef _MSC_VER
include <numeric>
undef min
undef max
endif

numeric ... iota function
ancellation of min, max macros, which conflict with min and max functions

trm_module.cpp, trm_react.cpp, trm_trans.cpp
———————————————

Replace
XMLDocument
with longer
tinyxml2 :: XMLDocument

TRM 2D installation - building process
———————

In the Visual Studio
Open->Menu->File->Open->CMake
c:/Trm/TRMInstall/CMakeLists.txt
Then open Solution Explorer
TRMInstall/CMakeLists.txt file
Press the right mouse button
Then choose Generate Cache for ...
After that Build
And push Install

TRM 2D starting process
——————

In C:/Trm/TRMinstall/out/build/x64-Debug directory must appear a trm.exe file.

Add following files from Lapack to the current directory
- cbia.lib.blas.dyn.rel.x64.12.dll
- cbia.lib.lapack.dyn.rel.x64.12.dll
- libifcoremd.dll
- libmmd.dll
- svml_dispmd.dll

and copy the c:/Trm/TRMInstall/inputs directory to the current directory
Then
- Create a log directory

- Create a results directory
- Create the results / reaction directory
- Create the results / transport directory

After that using command below and name of the chosen task, start the TRM 2D

```
trm kalcit_1d.xml
```

Appendix B

The time step control method implementation

```
unsigned step_index = 0;

   fstream out;
   out.open("c:\\Trm\\norm.txt", fstream::out);

   for (t = trans.time_init + curr_time_step; t - curr_time_step <
   trans.time_end - trans.time_end * NUMERIC_EPS; t = t +
  curr_time_step) {

        progress_value = static_cast<unsigned>(100 * (static_cast<
  double>(t_index) - static_cast<double>(trans.get_time_init_id())
  ) / static_cast<double>(time_steps_count));
        if (trans.is_bc_change_time(t_index)) {
            current_bc_time = t_index;
            curr_time_step = trans.get_time_step(current_bc_time);
            // Initial conditions used only at zero time
            if (t_index == initial_time_index) {
                vector<double> tmp_values = arma::conv_to<vector<
  double>>::from(trans.get_component_data("TransportComponents",
  t_index, map_component_id));
                vector_values init_cond(tmp_values);
                react.activate(
                    transport_concs, transport_names, init_cond,
  trans.get_component_data("InComponents", t_index,
  map_component_id),
                    t_index, t - curr_time_step, trans.
  get_first_component_of_id("0"));
                trans.init(transport_concs, transport_names,
  initial_time_index);
            }
            // Transport activation follow-up, maybe make modified
  grid flow BC time dependant and move it to transport activation
            trans.modify_grid_flow(t_index);
        }
        transport_concs_old = transport_concs;    //
  Transport_concs_old contains data from the previous run
        tc_old = arma::conv_to<arma::rowvec>::from(transport_concs)
```

```
;
      t_index++;
 label:                                                  // Restart
with shorter "curr_time_step"
      if (t > trans.time_end)
      {
          curr_time_step = trans.time_end - t + curr_time_step;
          t = trans.time_end;
      }
      trans.calculate(t_index, t, transport_concs,
transport_names, current_bc_time, mapping, curr_time_step);
      if (react.is_calculated_step(step_index)) {
          react.calculate(t_index, t, curr_time_step,
transport_concs, transport_names, mapping);
      }
      tc = arma::conv_to<arma::rowvec>::from(transport_concs);
         // Variable tc contains concentractions vector,
transformated from transport_concs
      arma::rowvec concentrace = ((tc - tc_old) / (tc + tc_old))
* 2;   // Concentrace contains the difference between new and
old concentrations elements divided by their average
      for (int i = 0; i < concentrace.n_cols; i++)
        // Cycle changes every "nan", apperared in vector because
 of division by "0", to "0"
      {
          if (isnan(concentrace(i)))
        // Checking if vector element is equal to "nan"
          {
              concentrace (i) = 0;
        // If so, change it to "0"
          }
       }
      if (out.is_open())       // If is used to writing out to
text file
      {
      out << arma::norm(concentrace, 2) << "/___/" <<
curr_time_step << "/___/" << t_index << "/___/" << t << endl;
// Writing out "norm<<curr_time_step<<t_index<<t" to text file
     }
     if (eps > 0)
     {
      if (arma::norm(concentrace, 2) < eps * 0.5)   // If norm of
 "arma::rowvec concentrace = ((tc - tc_old) / (tc + tc_old)) *
2" is smaller than half of the "eps"
      {                                            // So if norm
 of "concentrace" is really far from eps         //This helps to
 avoid the problem with repeating same step
          curr_time_step = curr_time_step * 1.1;   // Encrease "
curr_time_step"
      }
      else if (arma::norm(concentrace, 2) > eps)   // Else if
norm of "arma::rowvec concentrace = ((tc - tc_old) / (tc +
tc_old)) * 2" is bigger than "eps"
      {
          curr_time_step = curr_time_step * 0.5;   // Decrease
curr_time_step and restart calculation
```

```
            transport_concs = transport_concs_old;   // Return to variable "transport_concs" concentrations from the start of this run
            t = t - curr_time_step;                  // Return to time from the start of this run
            goto label;                              // Restart with shorter "curr_time_step"
        }
    }
        step_index++;
    }
    out.close();
```
Listing B.1: The time step control method implementation

Appendix C

TRM 2D structure description

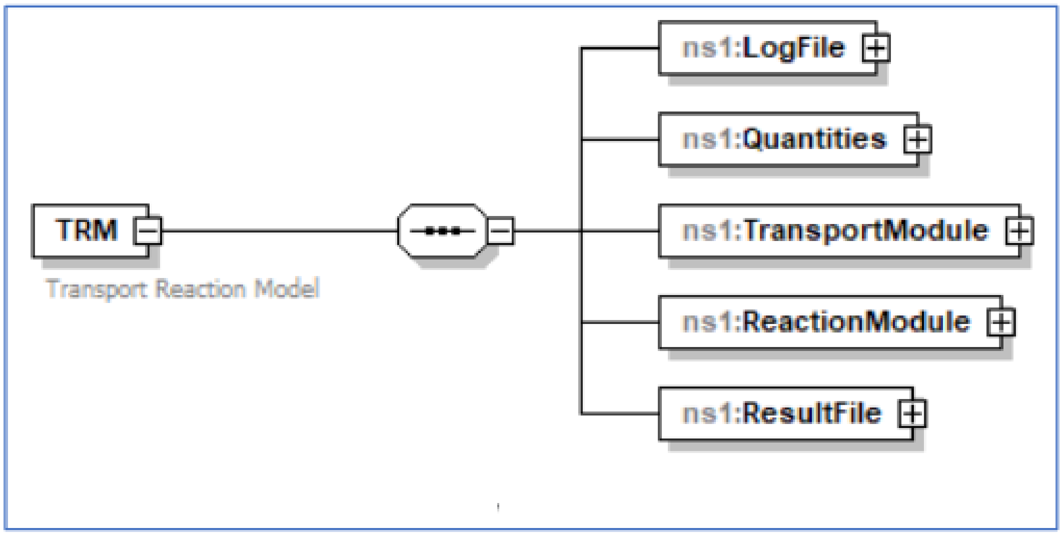

Figure C.1: Configuration file structure. (Adopted from [5])

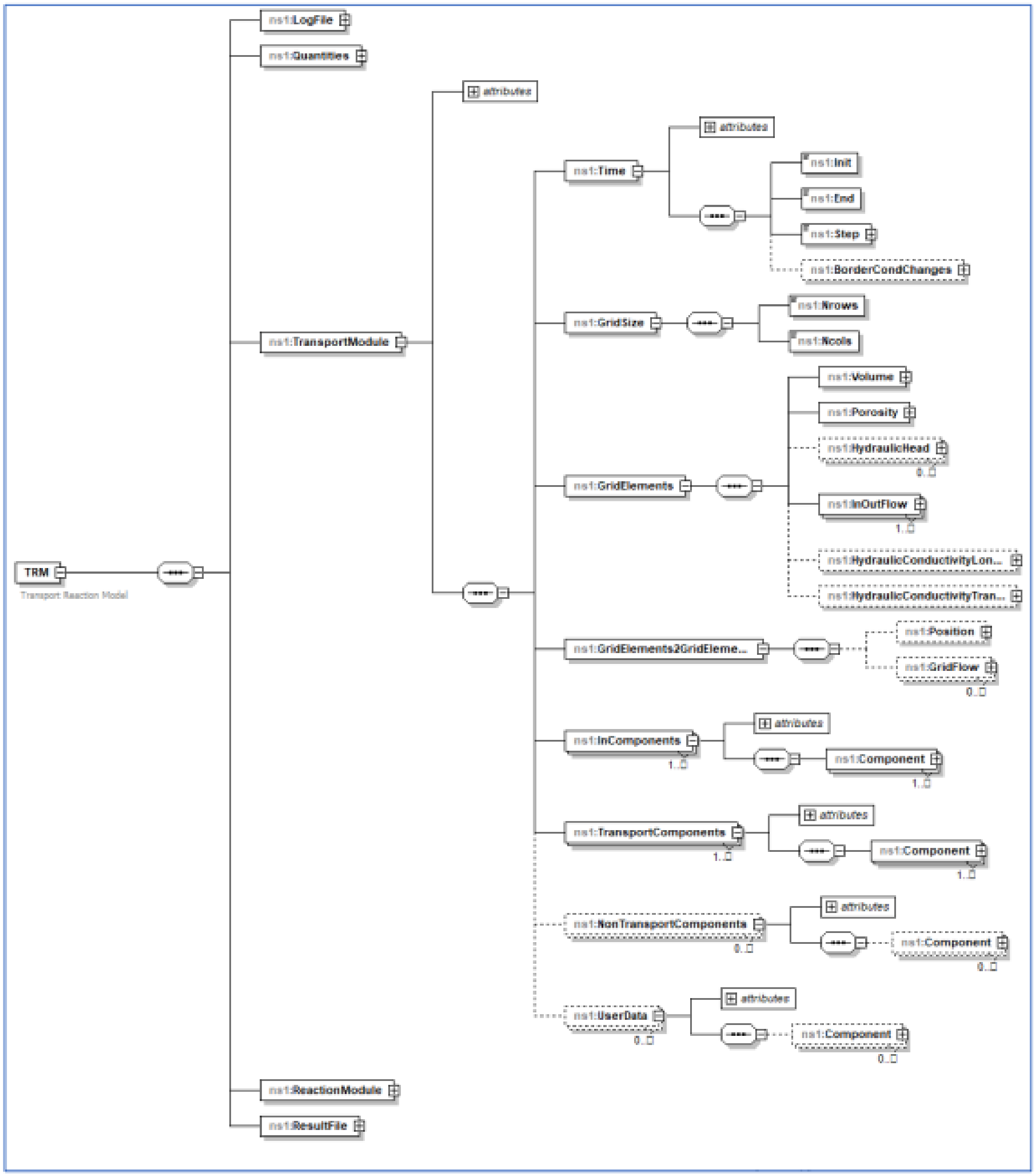

Figure C.2: Transport Module definition. (Adopted from [5])

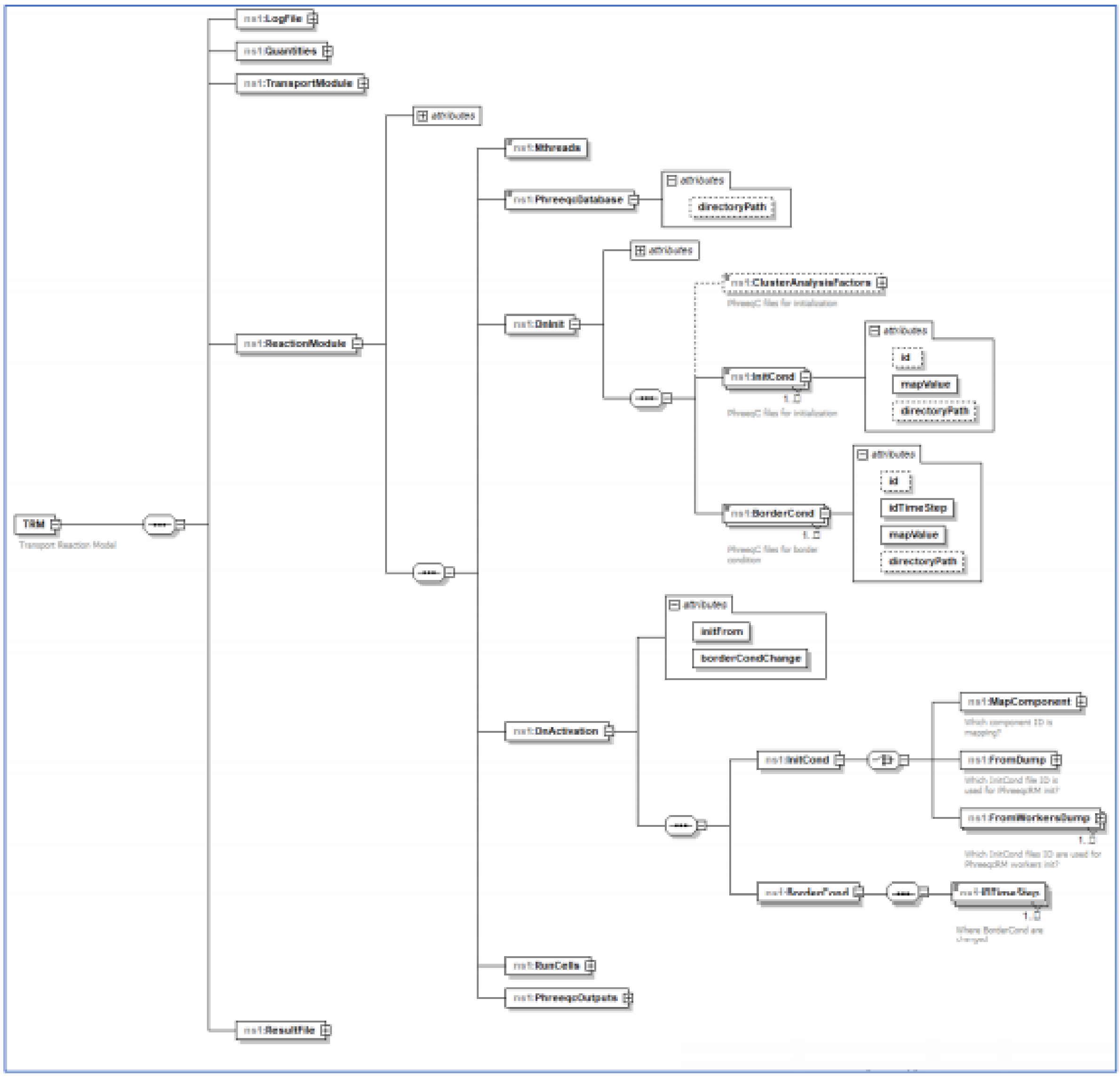

Figure C.3: ReactionModule definition (part A). (Adopted from [5])

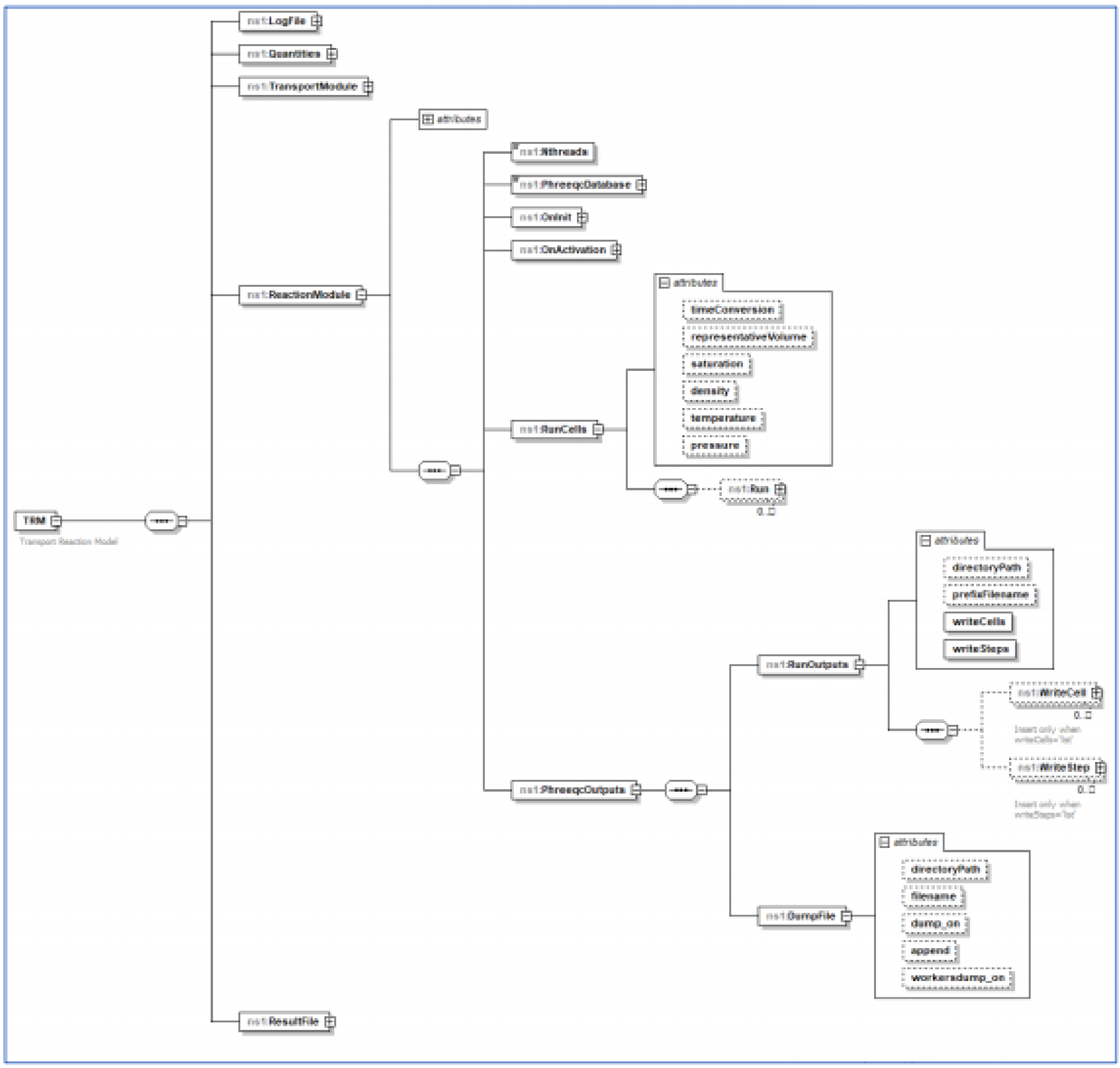

Figure C.4: ReactionModule definition (part B). (Adopted from [5])

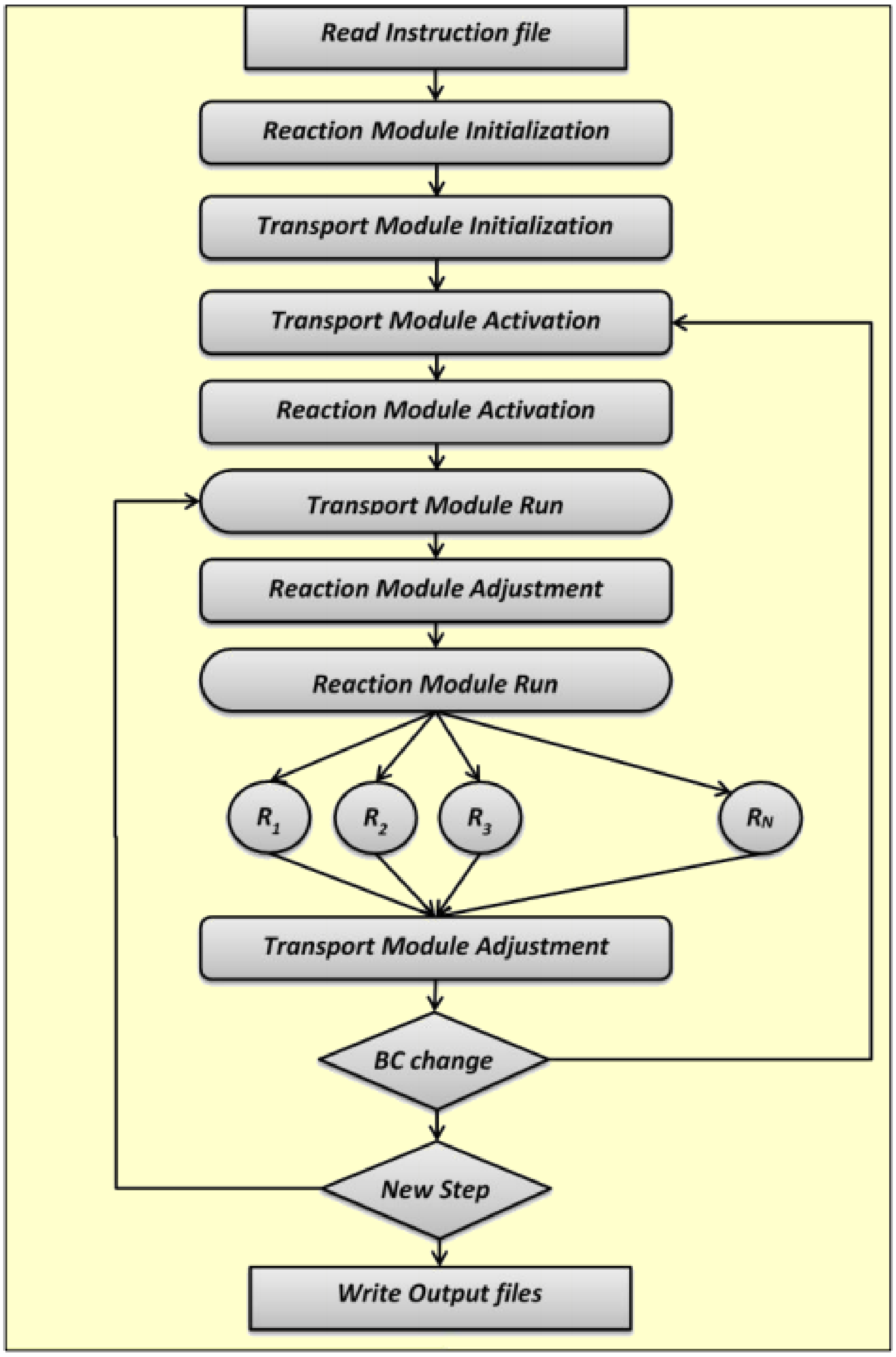
Read Instruction file
Reaction Module Initialization
Transport Module Initialization
Transport Module Activation
Reaction Module Activation
Transport Module Run
Reaction Module Adjustment
Reaction Module Run
R_1
R_2
R_3
R_N
Transport Module Adjustment
BC change
New Step
Write Output files

Figure C.5: Description of TRM 2D software simulation run. (Adopted from [5])