Modelling of Data Uncertainties on Hybrid Computers

GRS - 392

Gesellschaft für Anlagen-
und Reaktorsicherheit
(GRS) gGmbH

Modelling of Data Uncertainties on Hybrid Computers

Anke Schneider (ed.)

June 2016

Remark:

This report was prepared under contract No. 02 E 11062A with the German Federal Ministry for Economic Affairs and Energy (BMWi).

The work was conducted by the Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH.

The authors are responsible for the content of this report.

GRS - 392
ISBN 978-3-944161-73-0

Keywords:

Density-driven Flow, Modelling, Uncertainty, Fracture Flow, Final Repository

Acknowledgement

This report summarizes the work in the joint project “Modellierung von Daten-ungewissheiten auf hybriden Rechnern“ (H-DuR) of the project partners from the University of Frankfurt, the University of Jena and the GRS. The project was funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) under the contract no’s 02 E 11062C, 02 E 11062B and 02 E 11062A.

We would like to thank all our colleagues who contributed to this final report and to the successful conclusion of the project, the members of the working groups of Prof. Sabine Attinger (University of Jena) and Prof. Gabriel Wittum (University of Frankfurt) as well as the collegues involved at the Steinbeis Research Center 936. Finally, we would like to express our thanks to Peter Frolkovič (Slovak University of Technology) as well as the not diretly involved collegues of the Repository Safety Research Division of GRS for their consultation.

Various colleagues have contributed to this report. These were

Chapter 2:	Michael Hoffer, Michael Lampe, Arne Nägel, Sebastian Reiter, Andreas Vogel, Gabriel Wittum
Chapter 3:	Dmitrij Logashenko, Arne Nägel, Martin Rupp, Andreas Vogel, Gabriel Wittum
Chapter 4:	Sabine Attinger, Katharina Roß
Chapters 1, 5, 6:	Anne Gehrke, Klaus-Peter Kröhn, Anke Schneider, Hong Zhao

Abstract

The codes d³f and r³t are well established for modelling density-driven flow and nuclide transport in the far field of repositories for hazardous material in deep geological formations. They are applicable in porous media as well as in fractured rock or mudstone, for modelling salt- and heat transport as well as a free groundwater surface.

Development of the basic framework of d³f and r³t had begun more than 20 years ago. Since that time significant advancements took place in the requirements for safety assessment as well as for computer hardware development. The period of safety assessment for a repository of high-level radioactive waste was extended to 1 million years, and the complexity of the models is steadily growing. Concurrently, the demands on accuracy increase. Additionally, model and parameter uncertainties become more and more important for an increased understanding of prediction reliability. All this leads to a growing demand for computational power that requires a considerable software speed-up. An effective way to achieve this is the use of modern, hybride computer architectures which requires basically the set-up of new data structures and a corresponding code revision but offers a potential speed-up by several orders of magnitude.

The original codes d³f and r³t were applications of the software platform UG /BAS 94/ whose development had begun in the early nineteennineties. However, UG had recently been advanced to the C++ based, substantially revised version UG4 /VOG 13/. To benefit also in the future from state-of-the-art numerical algorithms and to use hybrid computer architectures, the codes d³f and r³t were transferred to this new code platform. Making use of the fact that coupling between different sets of equations is natively supported in UG4, d³f and r³t were combined to one conjoint code d³f++.

A direct estimation of uncertainties for complex groundwater flow models with the help of Monte Carlo simulations will not be possible in the near future because of the related high computational effort. Therefore handling uncertainties was paid special attention here, and particular models were developed.

The VRL based graphical user interface was advanced and adapted to the new code developments and the user demands. Based on Java, it allows a visual as well as a script based controlling and was extended by an integrated visualization tool. The output of files in the vtk-format allows the use of modern postprocessors. The preprocessor ProMesh for the data input, creation of model geometries and grid generation was also extended and improved thereby facilitating the application of d³f++ considerably.

Finally, the newly developed code d³f++ underwent a series of tests. It was successfully applied to several large complex models in crystalline as well as in sedimentary rock.

Zusammenfassung

Die Programme d³f und r³t wurden mit dem Ziel entwickelt, die dichtebeeinflusste Grundwasserströmung und den Transport von Nukliden und anderen Schadstoffen im Fernfeld von Endlagern in tiefen geologischen Formationen modellieren zu können. Sie sind sowohl in porösen Medien als auch in Kluftgesteinen, zur Modellierung des Salz- und Wärmetransportes sowie in Gebieten mit freier Grundwasseroberfläche einsetzbar.

Die Entwicklung der Grundstruktur der Programme d³f und r³t liegt bereits mehr als 20 Jahre zurück. Seit den 90er Jahren haben sich sowohl die Anforderungen an eine Langzeitsicherheitsanalyse als auch die Rechentechnik erheblich weiterentwickelt. Der Nachweiszeitraum für die Sicherheit eines Endlagers für stark wärmeentwickelnde radioaktive Abfälle wurde auf eine Million Jahre ausgedehnt, und die Modelle werden immer komplexer. Zusätzlich steigen die Ansprüche an die Genauigkeit und Zuverlässigkeit von Modellen. Hinzu kommt die wachsende Bedeutung der Berücksichtigung von Modell- und Parameterungewissheiten zur Erhöhung der Vorhersagezuverlässigkeit. All dies bedeutet ein wesentliches Anwachsen des Rechenaufwandes, der es notwendig macht, die Software erheblich zu beschleunigen und moderne, hybride Rechnerarchitekturen effizient zu nutzen. Dies erforderte den Aufbau neuer Datenstrukturen und damit eine grundlegende Überarbeitung der Simulationsprogramme, eröffnet jedoch Beschleunigungsmöglichkeiten um mehrere Größenordnungen.

Die Programme d³f und r³t stellten Anwendungen der Software-Plattform UG /BAS 94/ dar. Diese Plattform wurde innerhalb der letzten Jahre zu einer auf C++ basierenden, substantiell überarbeiteten Version UG4 weiterentwickelt /VOG 13/. Um weiterhin numerische Verfahren auf dem neuesten Stand der Wissenschaft und hybride Rechnerstrukturen nutzen zu können, war es notwendig, auch die Funktionalitäten von d³f und r³t auf der Basis von UG4 neu zu implementieren. Dabei wurde die in der Struktur von UG4 angelegte Möglichkeit genutzt, die Lösung verschiedener Differentialgleichungssysteme miteinander zu koppeln. So entstand aus d³f und r³t der gekoppelte Code d³f++.

Eine direkte Abschätzung von Datenungewissheiten mit Hilfe von Monte-Carlo-Simulationen wird für komplexe Grundwassermodelle wegen des hohen Rechenaufwandes in naher Zukunft nicht möglich sein. Deshalb wurde ein neues Modell entwickelt, das Ungewissheiten direkt mit in das Differentialgleichungssystem integriert.

Die VRL-basierte Benutzeroberfläche wurde weiterentwickelt und an die Neuentwicklungen und den Bedarf der Benutzer angepasst. Auf Java-Basis erlaubt sie sowohl eine visuelle als auch eine textbasierte Programmierung. Sie wurde außerdem um eine integrierte Visualisierungsmöglichkeit erweitert. Die zusätzliche Ausgabe von Files im VTK-Format ermöglicht die Nutzung moderner Postprozessoren. Auch der Präprozessor ProMesh zum Einlesen von Daten und zum Aufbau von Modellgeometrien sowie zur Gittergenerierung wurde erweitert und verbessert.

Die Neuentwicklungen wurden umfangreichen Tests unterzogen. Der neue Code d^{3}f++ wurde erfolgreich auf große, komplexe Modellgebiete sowohl im Kristallin als auch im Sedimentgestein angewendet.

Table of contents

1 Introduction

In Germany, radioactive waste is to be disposed in deep geological formations. Long term safety assessment for a repository requires a comprehensive system understanding and qualified high-performance tools. These tools have to be able to describe all relevant processes concerning nuclide transport through the host rock or the overlying geological formations, respectively.

To meet the needs of modeling groundwater flow and nuclide transport, in the period from October 1994 to August 1998 under the identification numbers 02 C 0254 6 (GSF) and 02 C 0465 0 (GRS) and from October 1 1998 to December 2003 under the identification number 02 E 9148 2 the computer codes d^3f (distributed density-driven flow) and r^3t (radio-nuclides, reaction, retardation, and transport) were developed /FEI 99/, /FEI 04/. Afterwards, these codes were substantially advanced and continuously adapted to the state-of-the-art of science and technology. ("E-DuR", 02 E 10336, /SCH 12/, „A-DuR", 02E10558, /SCH 13/, and "ESTRAL 02E10518, /NOS 12/". They were applied and qualified in different projects such as WEIMAR (02 E 11072A), URSEL (02 E 10750), KOLLORADO (02 E 10669), ISIBEL (02 E 10719) and QUADER (02 E 11213). All these works were funded by the Federal Ministry of Education and Research (BMBF) and by the Federal Ministry of Economics and Technology (BMWi), respectively. By means of these two computer codes it became feasible to simulate density driven flow and pollutant transport in porous and fractured media, including heat transport as well as free surface flow. They enable to handle large models with complex hydrogeological structures within reasonable processing times.

With an increasing degree of approximation towards the real, three-dimensional geological and hydrogeological conditions including all relevant processes, the modelling becomes more and more complex. According to the German safety case requirements for heat-generating radioactive waste an assessment period of one million years has to be regarded. The demands for accuracy and grid resolution are growing, and model and parameter uncertainties have to be taken into account. This implies a substantially increase of computational effort and leads easily to inadmissibly long computing times. Therefore, the most advanced hardware and cutting edge numerical solvers have to be used at all times.

The codes d^{3}f and r^{3}t were based on version 3 of the UG Toolbox, developed at the Frankfurt University /BAS 94/. In the H-DuR project, they were adapted to the substantially updated, C++-based version UG4 /VOG 13/. During this process the codes were combined to one conjoint code named d^{3}f++.

State-of-the art computer codes have not only to run on massively parallel computers, they also have to make use of modern multicore and hybrid computer structures. Each processor consists of multiple cores that are accessing at the same, hierarchically structured main memory, and, moreover, the cache memory may be organized in a much more complex way. In many cases processors of this type are supplemented by very specialized processors like GPUs and Cell processors. The efficient use of these modern computer architectures relies on appropriate data structures. Additionally, the solvers have to be adapted to these new structures. That means conventional computer codes need basically to be restructured to be able to employ these modern computer types. In this report the porting of d^{3}f++ to multicore and hybrid computer systems is described as well as the enhancement of its multigrid solvers.

At every site investigated some of the the rock properties as well as flow and transport parameters remain partially unknown. These uncertainties can be taken into account by stochastic modeling. But in the case of regional groundwater models it can be anticipated that Monte Carlo simulations will not be applicable in the medium-term future because of their high computational costs. Therefore special attention is paid to the adaption and application of a new stochastic approach resulting in the replacement of the salt concentration by a so called “filtered probability density function“ directly in the differential equation system. This leads, on the other hand, to higher dimensional equation systems requiring new numerical solvers.

Finally, a modern computer code needs an integrated user interface including comfortable pre- and postprocessing tools that enable the user to set-up regional, complex-structured models, reduce input data errors and ease code handling.

Members of the joint project

Goethe Center for Scientific Computing (G-CSC), University of Frankfurt
Kettenhofweg 139, 60325 Frankfurt
Prof. Gabriel Wittum
Ingo Heppner
Christian Poliwoda
Dr. Michael Lampe
Dr. Arne Nägel
Dr. Sebastian Reiter
Martin Rupp
Dr. Sabine Stichel
Dr. Andreas Vogel

Institute for Geosciences, University of Jena, Burgweg 11, 07749 Jena
Prof. Sabine Attinger
Dr. Jude Musuuza
Dr. Katharina Ross

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH
Repository Safety Research Division, Th.-Heuss-Str. 4, 38122 Braunschweig
Anke Schneider
Dr. Judith Flügge
Anne Gehrke
Dr. Klaus-Peter Kröhn
Dr. Hong Zhao

and as RD-Contractor of GRS

Steinbeis-Forschungszentrum „Technische Simulation”
Bussardweg 6, 75446 Wiernsheim-Iptingen
Dr. Michael Heisig
Dr. Dmitrij Logashenko
Michael Hoffer

2 Design of UG4

2.1 General

The codes d^{3}f (distributed density-driven flow, /FEI 99/) and r^{3}t (radionuclides, reaction, retardation, and transport, /FEI 04/) enable the simulation of density driven flow and pollutant transport in porous media. Both codes are based on the simulation toolbox UG (unstructured grids, /BAS 94/, /BAS 97/), a software framework for the numerical solution of coupled systems of partial differential equations. The code basis UG has been created during the early 1990s and continuously enhanced subsequently. Using a modular software design and focusing on geometric and algebraic multigrid solvers – asymptotically optimal solvers for large sparse systems of equations – this approach has been successfully applied to the field of flow and transport in porous media.

While a lot of concepts in the overall design of the software layout in the UG library have proven to be successful, some external requirements to efficient numerical software have changed in the past decades. In order to adapt the flow and transport simulations to this needs, the renewed code basis UG4 for the simulation of coupled partial differential equations has been developed /VOG 13/. The new implementation is grounded on an object-oriented software design and written in C++. It is strongly influenced by the predecessor version transferring all concepts that have proven to be useful. However, some design aspects have been redesigned and are described subsequently.

In the original design of the UG library applications such as d^{3}f and r^{3}t have been built as separate applications on top of the core libraries. While this design enabled the reusage of core components like solvers and discretizations, the coupling and thereby simultaneous simulation of the flow field and species transport was not natively supported. In contrast, a transport simulation had to be carried out after an entire computation for the flow equations writing the flow field to hard disk and read in afterwards. This prevented the direct coupling. In the renewed code version UG4 coupling between different sets of equations is natively supported and the standard way to implement discretizations for physical systems. Grounded on this coupling mechanism the codes d^{3}f and r^{3}t have been transferred to the new code basis and composed to one coupled code d^{3}f++.

To allow for such a tight coupling of subsystems while still maintaining a modular software layout, different discretizations are implemented as separate plugins in the new code. On start-up UG4 calls the initialization routine of each plugin and passes a reference to the central registry object (cf. /VOG 13/). Each plugin then registers its functionality (functions, classes, and methods) at this registry. After all plugins are initialized, the registry thus contains information on all functions and classes in all available plugins, together with extensive information on parameter- and return-types. Since plugins are built on a common core functionality and mainly implement methods that operate on this core functionality, the interaction between different plugins is straight forward given in this setup.

Frontends like UG4's scripting system (cf. /VOG 13/), ProMesh (/REI 14/) or the Visual Reflection Language VRL (cf. /HOF 13/) query the registry for available functionality and build script bindings or visual representations for the different classes and algorithms.

The registry thus provides a reflection mechanism for UG4's functionality which not only allows to build flexible frontends but also to couple functionality from different plugins which transform a common set of core functionality. An overview over the different modules of UG4 and their interplay is depicted in Fig. 1.

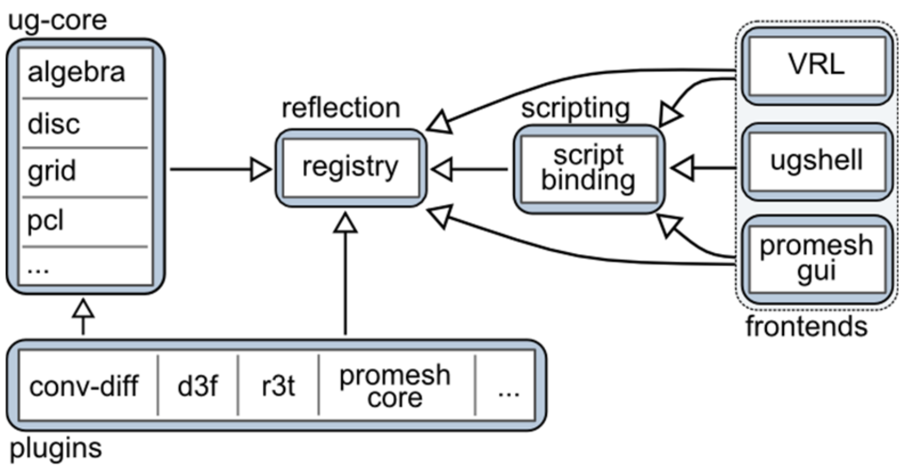


Fig. 2.1 UG4 software layout. Arrows point in the direction of dependencies. Shown are the core libraries, plugins, the reflection system, the script binding and various frontends

Larger and more refined simulations are nowadays carried out on modern parallel computer clusters that easily reach hundred thousands of computing cores and can be accelerated using a hybrid architecture using GPUs. In order to use these growing computing resources a code must be efficiently scalable to these architectures. Within UG4 a strong focus is given to highly efficient multigrid methods that have proven to scale close to optimal up to hundred thousands of cores /REI 13/.

User interfaces are important to handle numerous input specifications and control the simulations. This topic is addressed by the new software basis in manifold ways. First, a common basis for the control flow through a registry function provides the possibility to dock both graphical user interfaces as well as script based interfaces to the simulation codes. Second, the grid format ugx is now used as a standard for all simulations and computational domains can be created, adapted, optimized and partitioned into physical subsets using the preprocess mesh-generation software ProMesh. Third, for the output of the created data an interface to the established VTK framework has been created.

2.2 Coupling of different systems of equations

The discretization interfaces in UG4 have been realized with a strong focus on the possibility to couple different physical systems if required. To this end, all different modules fulfil the specification of standardized modules that can be combined. In the following the relevant techniques and implementations are described that enable the coupling of the d^{3}f and r^{3}t modules on the new code basis. Further details on the implementation can be found in /VOG 13/, /VOG 14/.

2.2.1 Systems of partial differential equations

During the modelling process for flow and transport in porous media a set of partial differential equations for a set of unknown functions arises. In the most general description this set of equations can be formally described as follows.

Definition (General system of differential equations)

Let $\Omega \subset \mathbb{R}^d$ be a physical domain and denote by $\boldsymbol{u} := (u_1, \dots, u_N)$ the set of the unknown functions $u_i : \Omega \mapsto \mathbb{R}^{n_i},\ n_i \in \mathbb{N}, (i = 1, \dots, N)$. Provided functionals $\mathcal{A}_i(\boldsymbol{u}), \mathcal{M}_i(\boldsymbol{u}), \mathcal{b}_i, (i = 1, \dots, N)$ a general system of time-dependent partial differential equations is given by

$$\begin{cases} Find\ \boldsymbol{u} := (u_1, \dots, u_N), such\ that \\ \partial_t \mathcal{M}_1(\boldsymbol{u}) + \mathcal{A}_1(\boldsymbol{u}) = \mathcal{b}_1,\ \ in\, \Omega, \\ \qquad\vdots \\ \partial_t \mathcal{M}_N(\boldsymbol{u}) + \mathcal{A}_N(\boldsymbol{u}) = \mathcal{b}_N,\ \ in\, \Omega, \end{cases} \tag{2.1}$$

and additional boundary conditions.

In UG4 every physical system is modelled by one set of differential equations. These problem descriptions are intended to be closed systems. However, coupling of several systems will be allowed by an appropriate specification of the data specified for the user parameters.

In the following the most important physical systems are listed that have been implemented for the transfer of d^{3}f and r^{3}t to the new code basis.

2.2.1.1 Haline Flow

The transport of dissolved salt in ground water flow in porous media is modelled by two nonlinear, coupled, time-dependent differential equations for the brine mass fraction and the pressure.

Definition (Haline Flow)
Let $\Omega \subset \mathbb{R}^d$ be a physical domain and let the brine mass fraction $\omega\colon \Omega \mapsto \mathbb{R}\ [-]$ and the pressure $p\colon \Omega \mapsto \mathbb{R}\ [\,Pa\,]$ be the unknown functions. The system of equations for haline flow is given by

$$\left\{\begin{aligned} &\qquad\qquad Find\ (\omega, p), such\ that \\ \partial_t(\phi\rho) &+ \nabla\cdot(\rho\boldsymbol{q}) = q,\ \ in\ \Omega, \\ \partial_t(\phi\rho\omega) &+ \nabla\cdot(\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega) = q_s,\ \ in\ \Omega, \\ \boldsymbol{q} &= -\frac{\mathbf{k}}{\mu}(\nabla p - \rho\boldsymbol{g}). \end{aligned}\right. \tag{2.2}$$

The physical parameters are given by:

- ϕ [-]: the porosity
- $\rho \equiv \rho(\omega)$ $[\,kg\ m^{-3}]$: the fluid density
- $\mu \equiv \mu(\omega)$ $[\,kg\ m^{-1}\ s^{-1}]$: the fluid viscosity
- $\mathbf{D}_{\text{disp}} \equiv \mathbf{D}_{\text{disp}}(\mathbf{q})$ $[\,m^2 s^{-1}\,]$: the mechanical dispersion tensor
- $\mathrm{D_m} \equiv \mathrm{D_m}$ $[\,m^2 s^{-1}\,]$: the molecular diffusion coefficient
- $\mathbf{T}$ $[\,-\,]$: the tortuosity tensor
- $\mathbf{D} \equiv \phi\, D_m\, \mathbf{T} + \mathbf{D}_{\text{disp}}(\mathbf{q})$ $[m^2 s^{-1}]$: the hydrodynamic dispersion tensor
- $\mathbf{k}$ $[\,m^2\,]$: the permeability tensor
- $\mathbf{g}$ $[\,m\ s^{-2}\,]$: the gravity vector

For the dispersion tensor the Bear-Scheidegger-Modell can be used, given by:

$$\boldsymbol{D}_{disp}(\boldsymbol{q}) = \alpha_L\boldsymbol{I} + (\alpha_L - \alpha_T)\,\frac{\boldsymbol{q}\cdot\boldsymbol{q}^T}{||\boldsymbol{q}||}, \tag{2.3}$$

with the parameters:

- α_L [m]: the longitudinal dispersion length
- α_T [m]: the transverse dispersion length

The equations for the haline flow fit into the general framework by setting $\boldsymbol{u} := (\omega, p)$ and defining

$$\begin{aligned}
&\mathcal{A}_p(\boldsymbol{u}) := \nabla \cdot (\rho \boldsymbol{q}),\\
&\mathcal{A}_\omega(\boldsymbol{u}) := \nabla \cdot (\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega),\\
&\mathcal{M}_p(\boldsymbol{u}) := \phi\rho,\\
&\mathcal{M}_\omega(\boldsymbol{u}) := \phi\rho\omega.
\end{aligned} \tag{2.4}$$

2.2.1.2 Thermohaline Flow

The transport of dissolved salt in ground water flow in porous media taking into account temperature effects is modelled by three nonlinear, coupled, time-dependent differential equations for the brine mass fraction, the pressure and the temperature.

Definition (Thermohaline Flow)
Let $\Omega \subset \mathbb{R}^d$ be a physical domain and let the brine mass fraction $\omega: \Omega \mapsto \mathbb{R}\ [-]$, the pressure $p: \Omega \mapsto \mathbb{R}\ [Pa]$ and the temperature $\theta: \Omega \mapsto \mathbb{R}\ [K]$ be the unknown functions. The system of equations for thermohaline flow is given by

$$\left\{\begin{array}{r}
Find\ (\omega, p, \theta), such\ that\\
\partial_t(\phi\rho) + \nabla\cdot(\rho\boldsymbol{q}) = q,\ \ in\ \Omega,\\
\partial_t(\phi\rho\omega) + \nabla\cdot(\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega) = q_s,\ \ in\ \Omega,\\
\partial_t\left(\left(\phi\rho C_f + (1-\phi)\rho_s C_s\right)\theta\right) + \nabla\cdot\left(\rho C_f\boldsymbol{q}\theta - \Lambda\nabla\theta\right) = 0,\ \ in\ \Omega,\\
\boldsymbol{q} = -\frac{\boldsymbol{k}}{\mu}(\nabla p - \rho\boldsymbol{g}).
\end{array}\right. \tag{2.5}$$

The physical parameters as far as not described in Section 2.2.1.1 are given by:

- Λ $[W\ m^{-1}\ K^{-1}]$: the thermal conductivity
- C_s $[J\ kg^{-1}\ K^{-1}]$: the heat capacity of the solid / rock
- C_f $[J\ kg^{-1}\ K^{-1}]$: the heat capacity of the fluid
- ρ_s $[kg\ m^{-3}]$: the rock density

The equations for the thermohaline flow fit into the general framework by setting $\boldsymbol{u} := (\omega, p, \theta)$ and defining

$$\begin{aligned}
&\mathcal{A}_p(\boldsymbol{u}) := \nabla \cdot (\rho \boldsymbol{q}),\\
&\mathcal{A}_\omega(\boldsymbol{u}) := \nabla \cdot (\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega),
\end{aligned} \tag{2.6}$$

$$\mathcal{A}_\theta(\boldsymbol{u}) := \nabla \cdot \left(\rho C_f \boldsymbol{q}\theta - \Lambda \nabla \theta\right),$$
$$\mathcal{M}_p(\boldsymbol{u}) := \phi\rho,$$
$$\mathcal{M}_\omega(\boldsymbol{u}) := \phi\rho\omega,$$
$$\mathcal{M}_\theta(\boldsymbol{u}) := \left(\phi\rho C_f + (1-\phi)\rho_s C_s\right)\theta.$$

2.2.1.3 Pressure-driven Flow

The flow due to pressure effects if modelled by one partial differential equation for the pressure.

Definition (Pressure-driven Flow)

Let $\Omega \subset \mathbb{R}^d$ be a physical domain and let the pressure $p: \Omega \mapsto \mathbb{R}\ [\,Pa\,]$ be the unknown function. The system of equations for pressure-driven flow is given by

$$\begin{cases} Find\ (p), such\ that \\ \nabla \cdot (\rho \boldsymbol{q}) = \ q, \ \ in\ \Omega, \\ \boldsymbol{q} = \ -\dfrac{\mathbf{k}}{\mu}\,(\nabla p - \rho \boldsymbol{g}). \end{cases} \tag{2.7}$$

The equations for the pressure-driven flow fit into the general framework by setting $\boldsymbol{u} := (p)$ and defining

$$\begin{aligned} &\mathcal{A}_p(\boldsymbol{u}) := \nabla \cdot (\rho \boldsymbol{q}), \\ &\mathcal{M}_p(\boldsymbol{u}) := 0. \end{aligned} \tag{2.8}$$

2.2.1.4 Prescribed Flow

If the flow field is user-specified as a known function of space and time, this is used as a very simple system of physics.

Definition (Prescribed Flow)

Let $\boldsymbol{\Omega} \subset \mathbb{R}^{\boldsymbol{d}}$ be a physical domain. For the prescribed flow system the Darcy velocity is specified as a function

$$\left\{ \boldsymbol{q} \equiv \boldsymbol{q}(\boldsymbol{x}, t), \ \ in\ \Omega, \right. \tag{2.9}$$

2.2.1.5 Transport equations

The transport of radionuclides for the r^{3}t functionality is modelled using a physical system of convection-diffusion type. The velocity field used for the convection is given by the Darcy velocity from the flow equations.

Definition (Transport equation)

Let $\boldsymbol{\Omega} \subset \mathbb{R}^{\boldsymbol{d}}$ be a physical domain and let the radionuclide concentration $\boldsymbol{c_i}: \boldsymbol{\Omega} \mapsto \mathbb{R}$ $[\boldsymbol{mol\, m^3}]$ be the unknown function. The system of equations for the transport is given by

$$\left\{ \begin{array}{c} Find\ (c_i), such\ that \\ \partial_t(\phi R_i c_i) + \nabla \cdot (c_i \boldsymbol{q} - \boldsymbol{D}\nabla c_i) + \phi R_i \lambda_i c_i = q_i,\ \ in\ \Omega. \end{array} \right. \tag{2.10}$$

The physical parameters are given by:

- ϕ $[-]$: the porosity
- ρ_r $[\,kg\, m^{-3}]$: the rock density
- $K_d^{(i)}$ $[\,m^3\, kg^{-1}]$: the distribution coefficient
- $R_i = 1 + \frac{1-\phi}{\phi} \rho_r K_d^{(i)}$ $[\,-\,]$: the retardation factor
- $\mathbf{D}$ $[\,m^2 s^{-1}\,]$: the diffusion-dispersion tensor
- $\mathbf{q}$ $[\,m\, s^{-1}\,]$: the darcy velocity
- $T_{1/2}^{(i)}$ $[\,s\,]$: the half-life
- $\lambda_{\mathrm{i}} = \frac{\ln 2}{T_{1/2}^{(i)}}$ $[\,s^{-1}\,]$: the decay constant

The source term q_i includes those radionuclides k that decay into radionuclide i,

$$q_i = \phi \sum_k R_k \lambda_k \, c_k\,, \tag{2.11}$$

with:

- λ_k: the decay constant of radionuclide k
- c_k: the concentration of radionuclide k

The equations for the transport fit into the general framework by setting $\boldsymbol{u} := (c_i)$ and defining

$$\mathcal{A}_c(\boldsymbol{u}) := \nabla \cdot (c_i \boldsymbol{q} - \boldsymbol{D}\,\boldsymbol{\nabla} c_i) + \phi \lambda_i R_i c_i, \\ \mathcal{M}_c(\boldsymbol{u}) := \phi R_i c_i. \tag{2.12}$$

2.2.2 Discretization of the equations

The partial differential equations describing the flow and the transport in porous media can be derived using Reynold's transport theorem and the fact that certain quantities obey a conservation law, i.e., within a closed volume and in the absence of sinks or sources the overall quantity within the volume is conserved over time. In order to reflect the conservation law on the discrete level, finite volume methods are used that ensure the balance equation for discrete so-called control volumes.

The discretization of the domain is obtained by a partition into a finite element grid.

Definition (Grid)

Let $\boldsymbol{\Omega} \subset \mathbb{R}^{\boldsymbol{d}}$ be a physical domain. A set of disjoint sets $\boldsymbol{\Omega_h} = \{\boldsymbol{K_1}, \dots, \boldsymbol{K_N}\}, \boldsymbol{K_i} \subset \boldsymbol{\Omega}$, is called a grid (or mesh), if it forms partition of $\boldsymbol{\Omega}$, i.e.,

$$\Omega = \bigcup_{K \in \Omega_h} K, \quad K \cap K' = \emptyset \ \ for\ all\ \ K \neq K'. \tag{2.13}$$

The functions that must be computed (e.g., brine mass fraction, pressure, solute density) are represented using this grid in order to define an appropriate discrete function space.

Definition (Discrete function spaces)

Let $\boldsymbol{\Omega_h} = \{\boldsymbol{K_1}, \dots, \boldsymbol{K_N}\}, \boldsymbol{K_i} \subset \boldsymbol{\Omega}$ be a grid for the domain $\boldsymbol{\Omega} \subset \mathbb{R}^{\boldsymbol{d}}$. Let $\boldsymbol{P_k}$ be the space of all polynomials up to order k, then the discrete function space for a single unknown function is given by

$$U_h = \{u_h \in C(\Omega); u_h|_K \in P_k, for\ all\ K \in \Omega_h\}. \tag{2.14}$$

A discrete function can be represented using shape functions $\boldsymbol{\Phi_i}: \Omega_h \mapsto \mathbb{R}$ by an additive composition,

$$u_h(\boldsymbol{x}) = \sum_i u_{h,i}\, \boldsymbol{\Phi_i}(\boldsymbol{x}), \quad u_{h,i} \in \mathbb{R}. \tag{2.15}$$

The elementwise space is fixed using polynomial spaces of lagrange type, i.e. for an appropriate set of points on the element $\boldsymbol{a_i} \in K$, the basis functions fulfil the requirement $\boldsymbol{\phi_j}(\boldsymbol{a_i}) = \delta_{ij}$. The elementwise space is denoted by U_K, the global space is denoted by U_h.

2.2.3 Element-local considerations

The general idea for the coupling is to restrict the process to an elementwise consideration. During the assembling process all grid elements are looped and the coupling is performed on the element.

2.2.3.1 Trial spaces on elements

Let U_K be the element function space and U_h the global function space. Denote by $|U_K|$ and $|U_h|$ the number of degrees of freedom on the spaces. On each grid element define the mapping between element-local and global unknown numbering by

$$g(K,j) \colon \{1, \dots, |U_K|\} \ni j \mapsto i \in \{1, \dots, |U_h|\} \tag{2.16}$$

Using this mapping the local solution can be extracted from the global space via

$$u_{K,j} = u_{h,\mathrm{g}(K,j)}, \quad 1 \le j \le |U_K|. \tag{2.17}$$

2.2.3.2 Defect equation on elements

The global defect

$$d_h \colon \mathbb{R}^{|U_h|} \simeq U_h \ni u_h \mapsto d_h(u_h) \in \mathbb{R}^{|V_h|} \tag{2.18}$$

is decomposed into a set of local contributions

$$d_h(u_h) = \sum_{K \in \Omega_h} d_h^K(u_h), \qquad d_h^K(u_h) \in \mathbb{R}^{|V_h|}. \tag{2.19}$$

It is assumed that the elementwise contribution only depends on the element-local unknowns and thus can write

$$d_K \colon \mathbb{R}^{|U_K|} \simeq U_K \ni u_K \mapsto d_K(u_K) \in \mathbb{R}^{|V_K|}, \tag{2.20}$$

by setting

$$d_{K,j} = d^K_{h,\mathrm{g}(K,j)}, \quad 1 \le j \le |U_K|. \tag{2.21}$$

Example (Flux discretization using finite volumes)

The flow term of a conservation equation can be discretized as

$$\begin{aligned} d_{\mathcal{B}_h,i}(u_h) &= \sum_{B\in\mathcal{B}_h} \int_{\partial B} F(u_h)\cdot n\,\chi_i = \sum_{K\in\Omega_h}\sum_{B\in\mathcal{B}_h}\int_{\partial B\cap K} F(u_h)\cdot n\,\chi_i \\ &=: \sum_{K\in\Omega_h} d^K_{\mathcal{B}_h,i}. \end{aligned} \tag{2.22}$$

2.2.3.3 Jacobian on elements

In the same way the global Jacobian

$$J_h\colon \mathbb{R}^{|U_h|} \simeq U_h \ni u_h \mapsto J_h(u_h) \in \mathbb{R}^{|V_h|\times|U_h|} \tag{2.23}$$

is decomposed into elementwise contributions

$$J_h(u_h) = \sum_{K\in\Omega_h} J^K_h(u_h), \qquad J^K_h(u_h) \in \mathbb{R}^{|V_h|\times|U_h|}\,. \tag{2.24}$$

Given the local solution this can be written as

$$J_K\colon \mathbb{R}^{|U_K|} \simeq U_K \ni u_K \mapsto J_K(u_K) \in \mathbb{R}^{|V_K|\times|U_K|}, \tag{2.25}$$

by setting

$$J_{K,ij} = J^K_{h,\mathrm{g}(K,i)\,\mathrm{g}(K,j)}, \quad 1 \le i \le |V_K|, \qquad 1 \le j \le |U_K|. \tag{2.26}$$

Example (Flux discretization using finite volumes)

The flow term of a conservation equation can be discretized as

$$\begin{aligned} J_{\mathcal{B}_h,ij} &= \sum_{B\in\mathcal{B}_h} \int_{\partial B} F(\phi_j)\cdot n\,\chi_i = \sum_{K\in\Omega_h}\sum_{B\in\mathcal{B}_h}\int_{\partial B\cap K} F(\phi_j)\cdot n\,\chi_i \\ &= \sum_{K\in\Omega_h} J^K_{\mathcal{B}_h,ij} \end{aligned} \tag{2.27}$$

2.2.3.4 Element based evaluation

It is thus sufficient to restrict the computation of the defect and jacobian to elements. In the equations of the physical systems there are, however, still integrals that must be approximated numerically. This is achieved via numerical quadrature rules that give the final element-local formulation and show that user data and other numerically provided information must be given for point evaluation on an element basis.

Example (Flux discretization using finite volumes)

The local defect for the convection diffusion equation

$$-\nabla \cdot (D\nabla u) + ru = 0, \quad \text{in } \Omega, \tag{2.28}$$

is discretized using a reference element mapping $T_K\colon \mathbb{R}^{d'} \supset \widehat{K} \ni x \mapsto x = T_K(x) \in \mathbb{R}^d$, the corresponding jacobian of the transformation $J_K(\hat{x}) := \frac{\partial T_K(\xi)}{\partial \xi}\Big|_{\xi=\hat{x}}$ and an appropriate quadrature rule $\{\omega_k^{\partial B}; \hat{x}_k^{\partial B}\}_{k=1,\dots,n_{ip}^{\partial \widehat{B}}}$ (weights and points) using the following evaluations:

$$\begin{aligned}
d_{K,i}(u_h) &= -\int_{\partial B_i \cap K} D(x)\nabla_x u_h(x) \cdot n(x)\chi_i(x)dS(x) \\
&\quad + \int_{B_i \cap K} r(x)u_h(x)\chi_i(x)dx \\
&= -\int_{\partial \widehat{B}_i \cap \widehat{K}} D\big(T_F(x)\big)J_K^{-T}(\hat{x})\nabla_x \hat{u}_h(x) \cdot n(x)\sqrt{\det g_{\partial B_i \cap K}(\hat{x})}\, d \\
&\quad + \int_{\widehat{B}_i \cap \widehat{K}} r\big(T_K(x)\big)\hat{u}_h(x)\, \big|\det J_{B_i \cap K}(\hat{x})\big| dx \\
&\approx -\sum_{k=1}^{n_{ip}^{\partial \widehat{B}}} D\left(T_F\big(x_k^{\partial B}\big)\right)J_K^{-T}\big(\hat{x}_k^{\partial B}\big)\nabla_x \hat{u}_h\big(x_k^{\partial B}\big) \cdot n\big(x_k^{\partial B}\big)\sqrt{\det g_{\partial B_i \cap K}\big(\hat{x}_k^{\partial B}\big)}\, \omega_k^{\partial B} \\
&\quad + \sum_{l=0}^{n_{ip}^{\widehat{B}}} r(T_K(x_l^B))\, \hat{u}_h(x_l^B)\ \ |\det J_{B_i \cap K}(\hat{x}_l^B)|\, \omega_l^B
\end{aligned}$$

Example (Gradients and values of shape functions)

For the local evaluation of the discrete solutions the shape functions must be evaluated. This is done on an element basis. Using the reference element transformation $T_K\colon \mathbb{R}^{d'} \supset \widehat{K} \ni x \mapsto x = T_K(x) \in \mathbb{R}^d$ this evaluation can be reduced to evaluations on the reference elements.

Value of discrete function

$$\Phi_{\mathrm{i}}(x) = \phi_i\left(T_K^{-1}(x)\right) = \Phi_i^{\mathrm{ref}}(\hat{x}) \tag{2.29}$$

Gradient

$$\nabla_x\, \phi_i|_x = J_K^{-T}(\hat{x})\, \nabla_{\hat{x}}\, \Phi_i^{\mathrm{ref}}|_{\hat{x}} \tag{2.30}$$

2.2.4 Coupling of discretizations

The previous section has shown how the discrete computation of the solutions can be reduced to an elementwise consideration. In particular the solutions as well as the defect and jacobian have been written in an element fashion and the required computation were based on the integration points of the finite volume scheme. This is the basis for the coupling mechanism that is described in the following considerations. All mentioned physical systems have been implemented using this approach and thus are available for simultaneous computation of coupled systems and can be used as building blocks to construct combined models.

2.2.4.1 Systems

It is assumed that the overall solution can be partitioned into several parts that form a reasonable subdivision of the solution space. For example, every physical system of partial differential equations (e.g., haline flow, thermohaline flow, transport) can be considered as a single system and the associated unknown functions are a useful splitting of the overall function space. Now, the solution space on every element is partitioned.

Definition (System function spaces)
Let $\boldsymbol{U_K}$ be the function space of the whole system on a grid element $\boldsymbol{K} \in \boldsymbol{\Omega_h}$. A partitioning into $\boldsymbol{n_{sys}} \in \mathbb{N}$ system spaces $\boldsymbol{U_{s,K}}, \boldsymbol{1 \leq s \leq n_{sys}}$, is a partitioning of the solution space such that it can be written as

$$U_K = U_{1,K} \times \ldots \times U_{n_{\mathrm{sys}},K}, \tag{2.31}$$

with $|U_K| = \sum_{s=1}^{n_{\mathrm{sys}}} |U_{s,K}|$. By this splitting of the function space also the element-local solution $u_K \in \mathbb{R}^{|U_K|}$ is splitted into the solution parts $u_{s,K} \in \mathbb{R}^{|U_{s,K}|}$ associated with the subsystem s and using the formal mapping

$$\mathrm{g}_s(K,i)\colon \{1,\dots,|U_{s,K}|\} \ni i \mapsto j \in \{1,\dots,|U_K|\} \tag{2.32}$$

this can be written as association

$$u_{K,\mathrm{g}_s(K,i)} = u_{s,K,i}, \quad 1 \le i \le |U_{s,K}|. \tag{2.33}$$

2.2.4.2 Data imports

Due to the splitting of the function spaces into several subspaces the overall set of equations can be partitioned into subsets of equations that are associated with the system components. However, this relates to a set of unrelated physical systems. In order to allow the coupling of the different physics a mechanism must be introduced that takes care of this coupling. Therefore, it is assumed that the equations for a physical system explicitly depend on their associated solutions only. The coupling with other physical systems is achieved via the data that must be specified for the equations. This data is allowed to depend on other solutions from the overall problem.

Definition (Import)

Let $\boldsymbol{K} \in \boldsymbol{\Omega_h}$ be a grid element and let the solution space $\boldsymbol{U_K}$ be partitioned into system solutions $\{\boldsymbol{U_{1,K}},\dots,\boldsymbol{U_{n_{sys},K}}\}, \boldsymbol{1 \le s \le n_{sys}}$. A data import is a functionality that allows the position based read of some C++-Type data $\boldsymbol{D} \simeq \mathbb{R}^{\boldsymbol{m}}$ and may depend on the solution of the entire problem, i.e.,

$$\begin{aligned} &\mathcal{I}\colon \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{|U_K|} \mapsto \mathrm{D}, \\ &x, t, u_K \mapsto \mathcal{I}(x,t,u_K). \end{aligned} \tag{2.34}$$

Using the imports of the restriction, that every system defect is allowed to depend on its own solutions only, the defect can now be formalized as

$$d_{s,K} \equiv d_{s,K}(u_{s,K}, \mathcal{I}_{s,1},\dots,\mathcal{I}_{s,n_{\mathcal{I}_s}}), \ n_{\mathcal{I}_s} \in \mathbb{N} \text{ (number of imports)}, \tag{2.35}$$

and the coupling with other solution components is realized via the imports.

Example (Imports for convection diffusion type)

Let c be a unknown density of a solute. The finite volume discretization forms a system with the associated subsolution c. Different types of imports appear:

The reaction rate r_r is an import of type $\mathbb{R}$ and used as

$$d_{c,K,i}(u_{c,K}) = \int_{B_i} r_r c = \sum_{ip} w_{\mathrm{ip}}^{B_i}\, r_r(x_{\mathrm{ip}}) c(x_{\mathrm{ip}}). \tag{2.36}$$

The velocity $\boldsymbol{v}$ is an import of type $\mathbb{R}^d$ and used as

$$d_{c,K,i}(u_{c,K}) = \int_{\partial B_i} c\, \boldsymbol{v} \cdot n = \sum_{ip} w_{\mathrm{ip}}^{\partial B_i}\, c(x_{\mathrm{ip}})\, \boldsymbol{v}(x_{\mathrm{ip}}) \cdot n(x_{\mathrm{ip}}). \tag{2.37}$$

The diffusion $\boldsymbol{D}$ is an import of type $\mathbb{R}^{d\times d}$ and used as

$$d_{c,K,i}(u_{c,K}) = -\int_{\partial B_i} \boldsymbol{D} \nabla c \cdot n = -\sum_{ip} w_{\mathrm{ip}}^{\partial B_i}\, \boldsymbol{D}(x_{\mathrm{ip}}) \nabla c(x_{\mathrm{ip}}) \cdot n(x_{\mathrm{ip}}). \tag{2.38}$$

2.2.4.3 Computation of Jacobian for coupled systems

Given that every system defect depends on the system solutions only and other solutions are incorporated via imports, the computation of the jacobian can be automated. This is due to the fact that the chain rule can be employed to first compute the linearization of the defect with respect to the imports and then compute the derivative of the import with respect to the solution components. Formally, give a defect as

$$d_{s,K}: \mathbb{R}^{|U_{s,K}|} \ni u_{s,K} \mapsto d_{s,K}(u_{s,K}, \mathcal{I}_{s,1}, \ldots, \mathcal{I}_{s,n_{\mathcal{I}_S}}) \in \mathbb{R}^{|V_{s,K}|}, \tag{2.39}$$

the entries of the jacobian are given by

$$J_{ss,K}(u_K) = \left.\frac{\partial d_{s,K}}{\partial u_{s,K}}\right|_{u_K} + \sum_{k=1}^{n_{\mathcal{I}S}} \left.\frac{\partial d_{s,K}}{\partial \mathcal{I}_{s,k}}\right|_{u_K} \cdot \left.\frac{\partial \mathcal{I}_{s,k}}{\partial u_{s,K}}\right|_{u_K} \tag{2.40}$$

for the diagonal part (i.e., the dependency w.r.t. the system unknowns) and

$$J_{st,K}(u_K) = \sum_{k=1}^{n_{\mathcal{I}_S}} \left.\frac{\partial d_{s,K}}{\partial \mathcal{I}_{s,k}}\right|_{u_K} \cdot \left.\frac{\partial \mathcal{I}_{s,k}}{\partial u_{t,K}}\right|_{u_K} \tag{2.41}$$

for the dependencies w.r.t. other solution components which are introduced via the data imports. This shows the general idea to implement the physical systems: First, the computation of the defect w.r.t. its own unknowns must be implemented. Second, the linearization of the defect w.r.t. the data imports must be available. In addition the data import must provide the dependency of the import w.r.t. the solution components that it depends on.

Example (Linearization of defect for convection diffusion type)

Let c be an unknown density of a solute. The finite volume discretization forms a system with the associated subsolution c. The required derivations are:

For the reaction rate term $r_r c$ one computes

$$\mathbb{R} \ni \frac{\partial d_{c,K,i}}{\partial u_{c,K,j}} = \int_{B_i} r_r \phi_j = \sum_{ip} w_{\mathrm{ip}}^{B_i} r_r|_{\mathrm{ip}}\, \phi_j|_{\mathrm{ip}}\,, \tag{2.42}$$

$$\mathbb{R} \ni \left.\frac{\partial d_{c,K,i}}{\partial r_r}\right|_{\mathrm{ip}} = w_{\mathrm{ip}}^{B_i} c|_{\mathrm{ip}}\,. \tag{2.43}$$

For the convection term $\nabla \cdot (vc)$ one computes

$$\mathbb{R} \ni \frac{\partial d_{c,K,i}}{\partial u_{c,K,j}} = \int_{\partial B_i} \phi_j\, v \cdot n = \sum_{ip} w_{\mathrm{ip}}^{\partial B_i}\, \phi_j|_{\mathrm{ip}}\, v|_{\mathrm{ip}}\, \cdot n|_{\mathrm{ip}}\,, \tag{2.44}$$

$$\mathbb{R}^d \ni \left.\frac{\partial d_{c,K,i}}{\partial v}\right|_{\mathrm{ip}} = w_{\mathrm{ip}}^{\partial B_i}\, c|_{\mathrm{ip}}\, n|_{\mathrm{ip}}\,. \tag{2.45}$$

For the diffusion term $-\nabla \cdot (D\nabla c)$ one computes

$$\mathbb{R} \ni \frac{\partial d_{c,K,i}}{\partial u_{c,K,j}} = -\int_{\partial B_i} D\nabla\phi_j \cdot n = -\sum_{ip} w_{\mathrm{ip}}^{\partial B_i}\, D|_{\mathrm{ip}}\, \nabla\phi_j|_{\mathrm{ip}}\, \cdot n|_{\mathrm{ip}}\,, \tag{2.46}$$

$$\mathbb{R}^{d\times d} \ni \left.\frac{\partial d_{c,K,i}}{\partial D}\right|_{\mathrm{ip}} = -w_{\mathrm{ip}}^{\partial B_i}\, n|_{\mathrm{ip}}\, \nabla c^T|_{\mathrm{ip}}\,. \tag{2.47}$$

2.2.4.4 Computation of user data

It remains to specify how the data imports are filled with data. This is accomplished by connecting an import to a user data item. This can be done at runtime of the program and allows for the flexibility to couple together several systems. In order to be as general as possible the minimum requirement for the user data interface is chosen as follows.

Definition (User data)

Let $K \in \Omega_h$ be a grid element and let the solution space U_K be the entire elemtentlocal function space. A user data is a functionality that allows the position based evaluation of some C++-Type data $D \simeq \mathbb{R}^m$ and the derivates w.r.t. to the solution U_K, i.e.,

$$\mathcal{D}: \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{|U_K|} \mapsto \mathrm{D} \tag{2.48}$$

$$x, t, u_K \mapsto \mathcal{D}(x, t, u_K).$$

Several kinds of user data have been implemented and can be categorized as follows.

Constant data

The data is given as constant function.

Position- and time-dependent data

The data is provided as a function of space and time

System-dependent data

The data is computed by a physical system. In this case the data will depend on the solution components that are associated with the system and the derivate w.r.t. the functions must be implemented. An important example of such a system-dependent data export is the computation of the Darcy velocity that is known to the haline/thermohaline system and can be imported into the convection equation for the solute transport.

Data linker

Even more flexibility in the coupling of user data is gained via the implementation of a so-called data linker. These objects take some user data as input and combine existing data to these new user data. The derivative of the new data is automatically available using the chain rule and the known dependencies of the combined data. Using such kind of data combination a broad variety of couplings can easily be realized with small implementation effort or even just by connection on a script level. This gives a large flexibility for the future generalization for other kind of data dependencies. Formally, one can define this setup as a functionality

$$\begin{aligned} &\mathcal{D}: \mathcal{D}_1 \times \ldots \times \mathcal{D}_{n_\mathcal{D}} \mapsto \mathrm{D}, \\ &\mathcal{D}_1, \ldots, \mathcal{D}_{n_\mathcal{D}} \mapsto \mathrm{D}(\mathcal{D}_1, \ldots, \mathcal{D}_{n_\mathcal{D}}). \end{aligned} \tag{2.49}$$

and thus compute the derivatives via

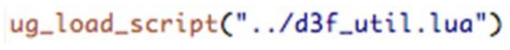
$$\left.\frac{\partial \mathcal{D}}{\partial u_{t,K}}\right|_{u_K} = \sum_{k=1}^{n_\mathcal{D}} \left.\frac{\partial \mathcal{D}}{\partial \mathcal{D}_k}\right|_{u_K} \cdot \left.\frac{\partial \mathcal{D}_k}{\partial u_{t,K}}\right|_{u_K}. \tag{2.50}$$

2.3 Script interface

One way to control the execution of the code is via user scripts. For the new code basis the scripting language LUA[1] has been chosen in order to specify the required user data and the solver control parameters. Thereby one script file is used to specify an entire problem.

2.3.1 General introduction

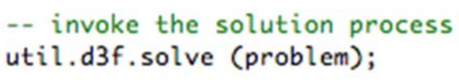
Every script starts by including the necessary utility for density driven flow problems. This is accomplished using the code line shown in Fig. 2.2.



```
ug_load_script("../d3f_util.lua")
```

Fig. 2.2 Loading the utility for density driven flow problems

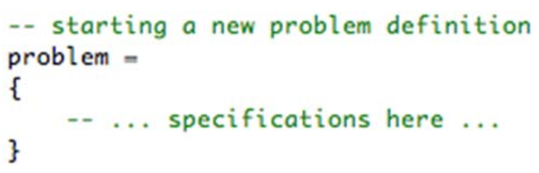
The utility script contains a central function that starts and controls the whole solution process. It is called passing a LUA-table that specifies the problem and the solvers. This line is shown in Fig. 2.3.



```
-- invoke the solution process
util.d3f.solve (problem);
```

Fig. 2.3 Starting the computation: “problem” is the LUA-table with the model specification and the solver setup

A LUA-table is created opening a pair of brackets. The most simple (empty) specification of the problem is as demonstrated in Fig. 2.4.

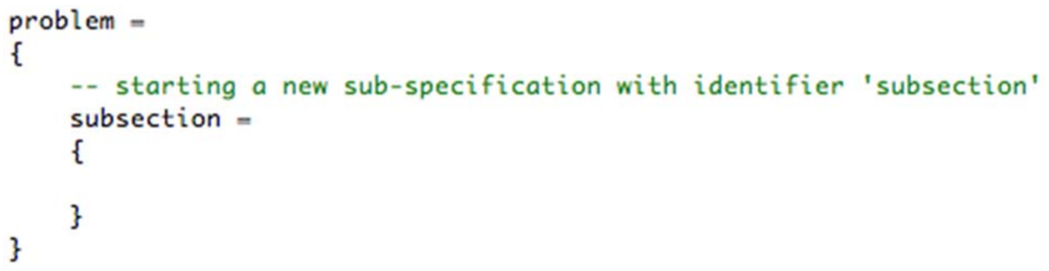


```
-- starting a new problem definition
problem =
{
    -- ... specifications here ...
}
```

Fig. 2.4 Starting a new problem specification using bracket-notation

[1] LUA project site, http://www.lua.org

Inside the problem table the specifications are grouped with respect to their content. This not only allows a better overview but is also used in order to activate or deactivate an entire problem part, e. g. the transport part can be turned on or off by using or not using this subgroup and is independent from the flow section. The basic sub-specification syntax is presented in



```
problem =
{
    -- starting a new sub-specification with identifier 'subsection'
    subsection =
    {

    }
}
```

Fig. 2.5 Starting a new sub-specification using bracket-notation

The available sub-specifiers are shown in Tab. 2.1.

Tab. 2.1 Available sub-specifier for the d^{3}f++ utility

Specifier	Specification for …
domain	… the grid, world dimension and refinements
flow	… the flow problem to be solved
transport	… the transport problem to be solved
time	… the time control
output	… the output / balancing options

2.3.2 The domain and grid specification

The physical domain is provided as a file in ugx-format. Such grids are created using the software tool ProMesh and provide a geometric description of the domain, the grid as well as a partitioning of the grid into distinguished subsets that can be used to set different types of equations or boundary conditions for different subsets. One, two and three dimensional grids are supported. Starting from a coarser grid the number of refinements is controlled by the numRef specification. An example for the domain entry is shown in Fig. 2.6.

```
problem =
{
    -- The domain specific setup
    domain =
    {
        dim = 3,                        -- world dimension
        grid = "grids/file.ugx",        -- grid file
        numRefs = 3,                    -- refinements
        numPreRefs = 1,                 -- (parallel) ref. before dist.
    },

    -- ... further specifications
}
```

Fig. 2.6 Entry for the domain specific setup

2.3.3 The flow module (integrated d^{3}f)

On the given grid a flow field can be computed, controlled by the specifications in the flow section. The flow field can be used in the transport section in order to compute simultaneously the transport of radionuclides.

There are several types of flow that can be computed and they require different type of user data input. In any case the entry “type” must be specified. This item allows to choose between the different physics. The variables or functions to be computed are specified by the item “cmp”. An example is shown in Fig. 2.7.

```
-- The flow setup
flow =
{
    type = "haline",        -- system type
    cmp = {"c", "p"},       -- names of components
```

Fig. 2.7 Starting a specification for density driven flow

The available types are listed in Tab. 2.2. All types characterize a physical system as described in Section 2.2.1 and can thus be coupled with other systems. This is used e.g. to use the Darcy velocity as user data for the import of the velocity in the transport section.

Tab. 2.2 Physical systems for the flow section and examples for the component specification dependent on the chosen type

type	description	cmp
prescribed	user defined flow field	-
pressure-driven	pressure dependent flow field	{ “p” }
haline	density driven flow	{ “w”, “p” }
thermohaline	thermohaline flow	{ “w”, “p”, “T” }

In the following for every type the required data input is presented.

2.3.3.1 Prescribed flow

The prescribed flow equation (see Section 2.2.1.4) is given by

$$\boldsymbol{q} \equiv \boldsymbol{q}(\boldsymbol{x}, t), \quad in\ \Omega \tag{2.51}$$

and requires the specification of the Darcy velocity as a known function. This can be given in two ways. First, the velocity can be specified as a constant field by simply passing the vector as shown in Fig. 2.8.

```
-- The flow setup
flow =
{
    type = "prescribed",
    value = {1, 0, 0},        -- [ m s^{-1} ]
},
```

Fig. 2.8 A constant flow field specification

Alternatively, the flow field can be specified as a function of spatial coordinates and time. To this end one can specify a LUA-function (see Fig. 2.9) before and then pass this function as a flow field specification (see Fig. 2.10).

```
function FlowField(x,y,z,t)
    return x*t, -y*t, 0
end
```

Fig. 2.9 A LUA-function specifying a flow field

```
-- The flow setup
flow =
{
    type = "prescribed",
    value = FlowField,       -- [ m s^{-1} ]
},
```

Fig. 2.10 Passing the Lua-function as data value

Alternatively, the user function can be specified inline as shown in Fig. 2.11.

```
-- The flow setup
flow =
{
    type = "prescribed",
    value = function (x,y,z,t) return x*t, -y*t, 0 end
},
```

Fig. 2.11 Inline version of the user function specification

2.3.3.2 Pressure-driven flow

The pressure-driven flow (see Section 2.2.1.3) is computed as follows:

$$\begin{cases} Find\ (p), such\ that \\ \nabla \cdot (\rho \boldsymbol{q}) = \ q,\ \ in\ \Omega, \\ \boldsymbol{q} = -\dfrac{\mathbf{k}}{\mu}\left(\nabla p - \rho \boldsymbol{g}\right). \end{cases} \tag{2.52}$$

The parameters can be set as shown in Fig. 2.12.

```
-- The pressure-driven-flow setup
flow =
{
    type = "pressure-driven",
    cmp = {"p"},

    density = 1025,            -- [ kg m^{-3} ]
    permeability = 1.0e-17,    -- [ m^2 ]
    viscosity = 1.5e-3,        -- [ kg / (m s) ]
    gravity = 0,               -- [ m / s^2 ]
}
```

Fig. 2.12 Example for the pressure-driven setup

2.3.3.3 Haline flow

The haline flow (see Section 2.2.1.1) is computed by

$$\begin{cases} \qquad\qquad Find\ (\omega, p), such\ that \\ \\ \partial_t(\phi\rho) \ + \nabla\cdot(\rho\boldsymbol{q}) = \ q,\ \ in\ \Omega, \\ \partial_t(\phi\rho\omega) + \nabla\cdot(\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega) = \ q_s,\ \ in\ \Omega, \\ \boldsymbol{q} = -\dfrac{\mathbf{k}}{\mu}(\nabla p - \rho\boldsymbol{g}). \end{cases} \tag{2.53}$$

The parameters are specified as shown in Fig. 2.13.

```
-- The density-driven-flow setup
flow =
{
    type = "haline",
    cmp = {"c", "p"},

    gravity = -9.81,            -- [ m s^{-2}]
    density =
    {   "linear",               -- ["const", "linear", "ideal"]
        min = 1000,             -- [ kg m^{-3} ]
        max = 1025              -- [ kg m^{-3} ]
    },

    viscosity =
    {   "linear",               -- ["const", "linear", "real"]
        min = 1e-3,             -- [ kg m^{-3} ]
        max = 1.5e-3,           -- [ kg m^{-3} ]
        brine_max = 0.0357
    },

    diffusion       = 18.8571e-6,   -- [ m^2 s^{-1} ]
    alphaL          = 0,            -- [ m ]
    alphaT          = 0,            -- [ m ]

    upwind      = "partial",    -- no, partial, full
    boussinesq  = false,        -- true, false

    porosity        =  0.35,        -- [ - ]
    permeability    =  1.019368e-9, -- [ m^2 ]
}
```

Fig. 2.13 Example for the density-driven flow setup

2.3.3.4 Thermohaline flow

The thermohaline flow (see Section 2.2.1.2) is computed using the equation

$$\begin{cases} \text{Find } (\omega, p, \theta), \text{such that} \\ \partial_t(\phi\rho) \ + \nabla \cdot (\rho \boldsymbol{q}) = \ q, \ \ in\ \Omega, \\ \partial_t(\phi\rho\omega) + \nabla \cdot (\rho\omega\boldsymbol{q} - \rho \mathbf{D}\nabla\omega) = \ q_s, \ \ in\ \Omega, \\ \partial_t\left((\phi\rho C_f + (1-\phi)\rho_s C_s)\theta\right) + \nabla \cdot \left(\rho C_f \boldsymbol{q}\theta - \Lambda\nabla\theta\right) = \ 0, \ \ in\ \Omega, \\ \boldsymbol{q} = -\frac{\mathbf{k}}{\mu}\left(\nabla p - \rho \boldsymbol{g}\right). \end{cases}$$

The parameter can be set as shown in Fig. 2.14.

```
-- The density-driven-flow setup
flow =
{
    type = "thermohaline",
    cmp = {"c", "p", "T"},

    gravity = -9.81,              -- [ m s^{-2}]
    density =
    {   "Oldenburg/Pruess",       -- ["const", "Oldenburg/Pruess"]
        rho_ref = 1e3,
        rho_s = 2.650e3,
        ref_rho_pw = 800,
        ref_rho_pb = 1000,
        alpha_w = 1.45e-3,
        alpha_b = 8.13e-4,
        theta_w = 523.15,
        theta_b = 563.15,
    },

    viscosity =
    {   "const",                  -- ["const", "linear", "real"]
        min = 1e-3,               -- [ kg m^{-3} ]
    },

    diffusion       = 1e-8,       -- [ m^2 s^{-1} ]
    alphaL          = 0,          -- [ m ]
    alphaT          = 0,          -- [ m ]

    thermalConductivity = 1.8,        -- [ W m^{-1} s^{-1} ]
    heatCapacityFluid = 4.184e3,      -- [ J kg^{-1} K^{-1} ]
    heatCapacitySolid = 1e3,          -- [ J kg^{-1} K^{-1} ]
    rockDensity = 2.650e3,            -- [ kg m^{-3} ]
    rho_ref = 1e3,

    upwind      = "partial",      -- no, partial, full
    boussinesq  = false,          -- true, false

    porosity        =  0.1,           -- [ 1 ]
    permeability    =  5e-12,         -- [ m^2 ]
}
```

Fig. 2.14 Example for the thermohaline density-driven flow setup

2.3.4 The reaction-transport module (integrated r^{3}t)

The transport equation uses the Darcy velocity $\boldsymbol{q}$ provided by the flow section in order to simulate the transport of radionuclides or other substancies. Thereby, effects as sorption or radioactive decay have to be regarded. The parameters have to be specified in the transport section as shown in Fig. 2.15.

```
-- The transport setup
transport =
{
    rockdensity = 2500,  -- [ kg / m^3 ]

    {   cmp = "U-236",

        --  ... further data ...
    }
}
```

Fig. 2.15 Starting the transport problem section

As detailed in Section 2.2.1.5 the equation for a radionuclide is given by

$$\begin{cases} \text{Find } (c_i), \text{such that} \\ \partial_t(\phi R_i c_i) + \nabla \cdot (c_i \boldsymbol{q} - \boldsymbol{D}\, \boldsymbol{\nabla} c_i) + \phi \lambda_i R_i c_i = q_i, \ \ in\ \Omega. \end{cases} \tag{2.54}$$

For every radionuclide the parameters are thus specified as shown in Fig. 2.16

```
{   cmp = "Pu-244",
    diffusion = 1e-9,         -- [ m^2 / s ]
    Kd = 0.0,                 -- [ m^3 / kg ]
    thalf = 80e6 * YearInSeconds,
    into = "U-236",
    initial = StartPu244,
},
```

Fig. 2.16 Adding a radionuclide to the transport problem

2.3.5 Start values, boundary conditions and subset data

In general the presented user data can be specified not only by a constant value. For most of the data the input can also be a LUA-function. They can be specified by a predefined function or inline analogously as shown in Fig. 2.9 to Fig. 2.11.

For all variables start values have to be specified in the initial section. For every component a constant value as well as any user-defined function can be used. Fig. 2.17 provides an illustration.

```
flow =
{
    -- .. other info ...

    initial =
    {
        { cmp = "c", value = 0.0 },
        { cmp = "p", value = function (x,y,t) return -10055.25 * y end },
    },
}
```

Fig. 2.17 Start value specification

Boundary conditions are specified in the boundary section. Fig. 2.18 illustrates the setup.

```
flow =
{
    -- .. other info ...

    boundary =
    {
        { cmp = "c", type = "level", bnd = "Inflow", value = 0.0 },
        { cmp = "c", type = "level", bnd = "Sea", value = 1.0 },
        { cmp = "p", type = "level", bnd = "Sea", value = "HydroPressure" },

        { cmp = "p", type = "flux", bnd = "Inflow", inner = "Medium", value = 3.3e-2 }
    }
},
```

Fig. 2.18 Boundary condition specification

In addition to these global definitions, parameters also may be specified on certain subsets. Usually e. g. the permeability varies over different parts of the domain and must be set therefore for each subset. This can be accomplished in two ways. First the user can simply open a new bracket within a section and use the subset keyword to restrict the subsequent specifications to this subset only. This is shown in Fig. 2.19.

```
flow =
{
    -- ... other info ...

    porosity        =  0.35,            -- [ 1 ]
    permeability    =  1.019368e-9,

    {
        subset          = {"Fracture"},
        porosity        =  0.7,      -- [ 1 ]
        permeability    =  1.019368e-6,
    },
}
```

Fig. 2.19 Using different specifications on subsets

Second, a data table can be used to give the parameters in a table format. An example is shown in Fig. 2.20.

```
flow =
{
    -- ... other info ...

    datatable =
    {
        {   "subset",       "porosity",     "permeability"  },
        {   "Medium",       0.3,            1.019368e-9     },
        {   "Fracture",     0.7,            1.019368e-6     },
    },
}
```

Fig. 2.20 Data table format for user data specification

2.3.6 Solver setup and time control

A special section in the LUA script is reserved for the solver settings. The nonlinear solver needs the parameters shown in Fig. 2.21.

```
solver =
{
    type = "newton",
    lineSearch = {                  -- ["standard", "none"]
        type = "standard",
        maxSteps        = 30,       -- maximum number of line search steps
        lambdaStart     = 1,        -- start value for scaling parameter
        lambdaReduce    = 0.5,      -- reduction factor for scaling parameter
        acceptBest      = true,     -- check for best solution if true
        checkAll        = false     -- check all maxSteps steps if true
    },

    convCheck = {
        type        = "standard",
        iterations  = 100,          -- number of iterations
        absolute    = 5e-8,         -- absolut value of defact to be reached; usually 1e-6 - 1e-9
        reduction   = 1e-20,        -- reduction factor of defect to be reached; usually 1e-6 - 1e-7
        verbose     = true          -- print convergence rates if true
    },
```

Fig. 2.21 Specification of the newton solver setup

The linear solver can be controlled as presented in Fig. 2.22.

```
linSolver =
{
    type = "bicgstab",          -- linear solver type ["bicgstab", "cg", "linear"]
    precond =
    {
        type        = "gmg",    -- preconditioner ["gmg", "ilu", "ilut", "jac", "gs", "sgs"]
        smoother    = "ilu",    -- gmg-smoother ["ilu", "ilut", "jac", "gs", "sgs"]
        cycle       = "V",      -- gmg-cycle ["V", "F", "W"]
        preSmooth   = 3,        -- number presmoothing steps
        postSmooth  = 3,        -- number postsmoothing steps
        rap         = false,    -- comutes RAP-product instead of assembling if true
        baseLevel   = 0,        -- gmg - baselevel

    },
    convCheck = {
        type        = "standard",
        iterations  = 30,       -- number of iterations
        absolute    = 0.5e-10,  -- absolut value of defact to be reached;
        reduction   = 1e-3,     -- reduction factor of defect to be reached;
        verbose     = true,     -- print convergence rates if true
    }
}
```

Fig. 2.22 Specification of the linear solver setup

For the control of the time stepping the section “time” is used. This is shown in Fig. 2.23.

```
time =
{
    control = "prescribed",
    start   = 0.0,      -- [s]  start time point
    stop    = 200,      -- [s]  end time point
    dt      = 10,       -- [s]  initial time step
    dtmin   = 0.01,     -- [s]  minimal time step
    dtmax   = 10,       -- [s]  maximal time step
    dtred   = 0.1,      -- [1]  reduction factor for time step
    tol     = 1e-2,

},
```

Fig. 2.23 Specification of the time control

2.4 Graphical user interface

2.4.1 Introduction

The new code structure as well as the enhanced functionality of d³f++ necessitates the development of a new, integrated graphical user interface.

The first problem in applying a groundwater code is the set-up of of the hydrogeological structure that may be arbitrarily complicated and based on huge amounts of data provided in various different formats. To support the user in this stage of work the ProMesh tool was created that allows compiling of hydrological layers to a regional, hydrogeological model, designing fractures or other structures and that also includes a grid generator (see chapter 2.4.3).

Using a comprehensive code like d³f++ requires an in-depth knowledge of its functionality as well as a good understanding of mathematical modelling and numerical methods. The complete functionality of the code may be accessed with the help of the LUA scripts described in chapter 0. Operating with scripts offers a high flexibility. To make the application of d³f++ more convenient and clear and to help the user in avoiding errors, the graphical user interface is created, based on the “Virtual Reflection Library” (VRL, cf. /HOF 13/, see chapter 2.4.2).

2.4.2 VRL-Studio

2.4.2.1 Introduction

VRL-Studio is an interactive visual programming environment for controlling complex simulation workflows. It can be easily extended with the help of a capable plugin architecture. To use VRL-Studio efficiently, several plugins have been developed to provide interactive access to UG4 and d^{3}f++ based applications.

As mentioned in chapter 2.1 VRL-Studio plugins for UG4 have access to the UG registry and use it to provide interactive user interfaces for registered UG functionality.

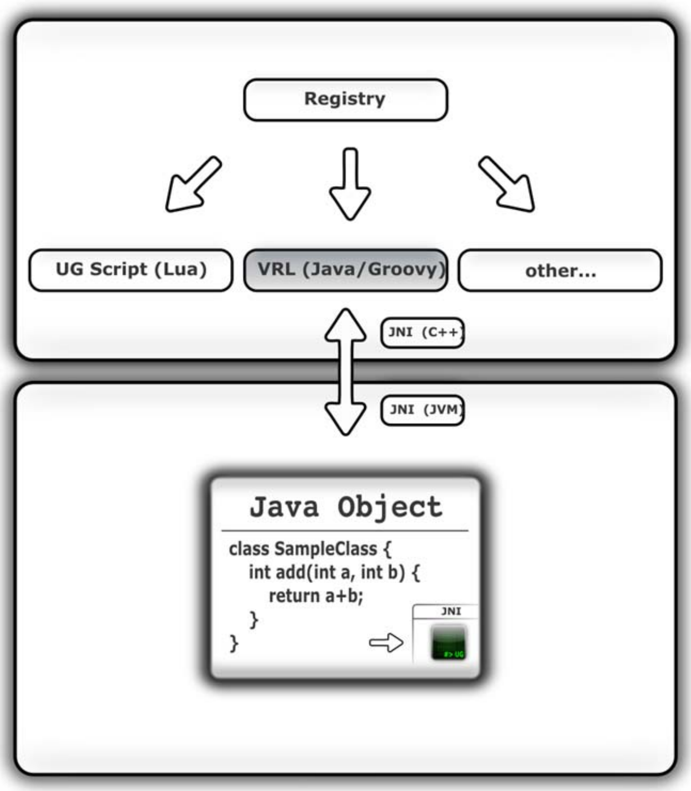


Fig. 2.24 VRL-Studio and UG4 Registry

2.4.2.2 Server-Client Communication

Interactive access to the simulation tool chain is a huge improvement. However, problems may arise if the simulation is computationally too expensive. Therefore, the UG plugin for VRL contains a server-client infrastructure for remotely controlling simulations. This type of remote computation retains most interactive aspects of the simulation, i.e., the user gets notified of intermediate results, as well as errors that may have occurred during the simulation.

In the current implementation the server-client communication is based on the XML-RPC protocol and does not require additional infrastructure.

2.4.2.3 Console Application Support

To run VRL-Studio workflows in a text based, non-graphical environment, VRL-Studio projects can be exported as console applications. This enables the user to run applications remotely, if interactive server-client execution is not applicable.

Therefore, VRL-Studio converts visually defined workflows into a code representation that can be compiled as regular Java library (JAR-File). To further improve the user experience, VRL-Studio also bundles all necessary plugins and a start script for UNIX based operating systems and Windows that can be used to run the workflow.

Exported projects can be easily transferred to a different computer. The user does not have to manually install plugins. Plugins are installed on the target system on first usage. To maintain compatibility, each exported console app comes with its own set of plugins and configuration files.

To execute a console app called console-app.zip on UNIX/Linux the following commands can be used:

```
#> unzip console-app.zip && sh console-app/run.sh
```

Console apps can be used in combination with SCP and SSH to execute them on remote computers.

2.4.2.4 LUA Support for VRL-Studio

For simulations that cannot be executed interactively, e.g. if they are running in batch-mode on high performance computers, it is important for VRL-Studio to have full scripting support.

Therefore, the UG plugin for VRL provides a LUA executor that can be fully integrated into visual VRL-Studio workflows and projects. Scripts that are designed for being used on high performance computers can be executed with no or little modifications.

2.4.2.5 Integrated LUA Editor

Introduction

The scripting language LUA is used in many projects based on the scientific simulation system UG4. LUA is tightly integrated into UG4 via a customized LUA-based shell environment. The LUA-based shell environment is interfaced with the C++ implementation of all UG4 classes and functions, and allows modelling UG4 control flows/work flows from within a flexible and untyped language.

The LUA scripts are also commonly used to define simulation parameters and constraints, as well as to implement and integrate domain specific algorithms with the capabilities of the UG4 environment. While integrated development environments (IDEs) like Eclipse, IntelliJ or Netbeans offer support for writing LUA scripts, the available libraries lack specific support for the UG4 shell environment and/or are too tightly bound to a given IDE in order to be easily ported to the VRL-Studio IDE.

The presented LUA auto-completion extension for the VRL-Studio IDE solves this for the UG4 user, and offers a viable auto-completion and LUA scripting support for UG4 development within VRL-Studio. The feature is realized as a VRL plugin. VRL plugins are loaded by the VRL-Studio IDE at start-up and provide seamless extensions to existing IDE functionality and other plugins for d^{3}f++, UG and data visualisation. The following language features of LUA are supported in the current implementation:

- UG shell-specific functions, classes and their respective documentation
- Global and local variables from the current and all included LUA scripts
- Functions, parameters and Doxygen[2]-style comments from current and all included LUA scripts
- Nested hash tables and arrays
- LUA (pseudo) class definitions

The auto-completion feature is context-sensitive. Proposed completions are valid within the variable scope at the cursor position.

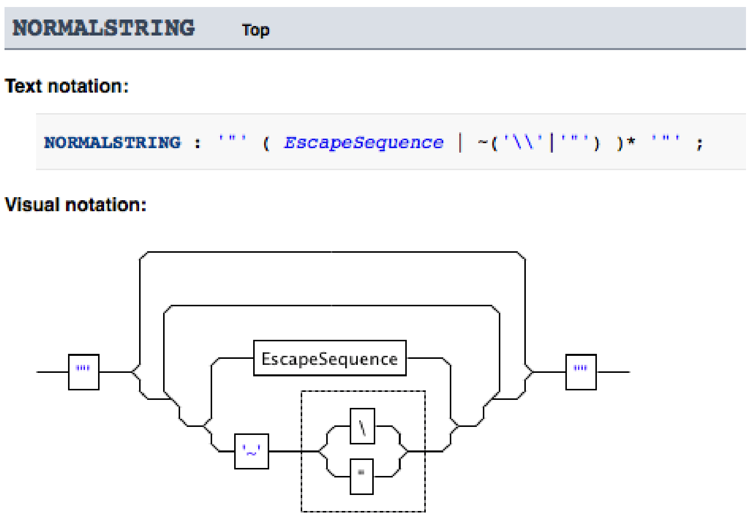
Component Architecture

The LUA auto-completion feature is based on the ANTLR[3] 4.x parser library (Another Tool for Language Recognition). The used parser definition fully supports the LUA language version 5.2.

The Java code of the LUA parser is generated as part of the build of the LUA auto-completion plugin by the ANTLR plugin. Fig. 2.25 shows a sample ANTLR definition for a string (“NORMALSTRING”) in test and visual notation:

[2] Doxygen web site, http://www.stack.nl/~dimitri/doxygen

[3] /ANTLR/ ANTLR project main page, http://www.antlr.org



NORMALSTRING Top

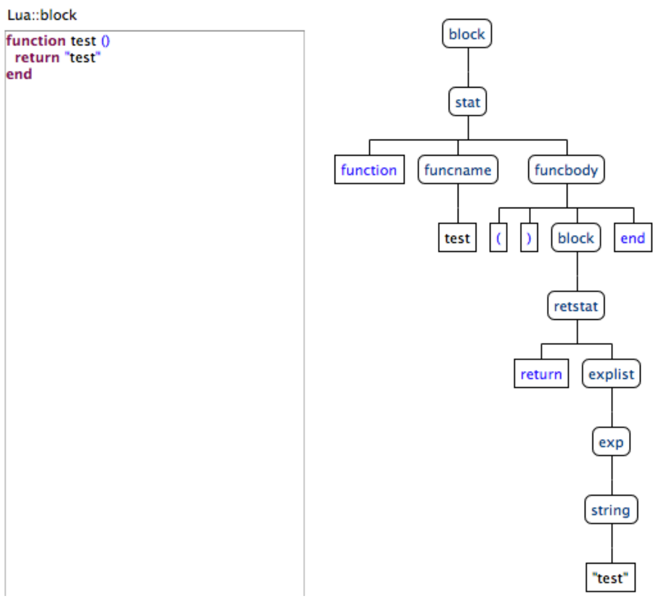
Text notation:

```
NORMALSTRING : '"' ( EscapeSequence | ~('\\'|'"') )* '"' ;
```

Visual notation:

Fig. 2.25 Lua Grammar Visualisation

At run-time, the parser code builds an abstract syntax tree from arbitrary LUA scripts. The parser can partially handle scripts that contain syntax errors. Fig. 2.26 details a sample syntax tree generated by ANTLR for a simple function definition.



Lua::block

```
function test ()
  return "test"
end
```

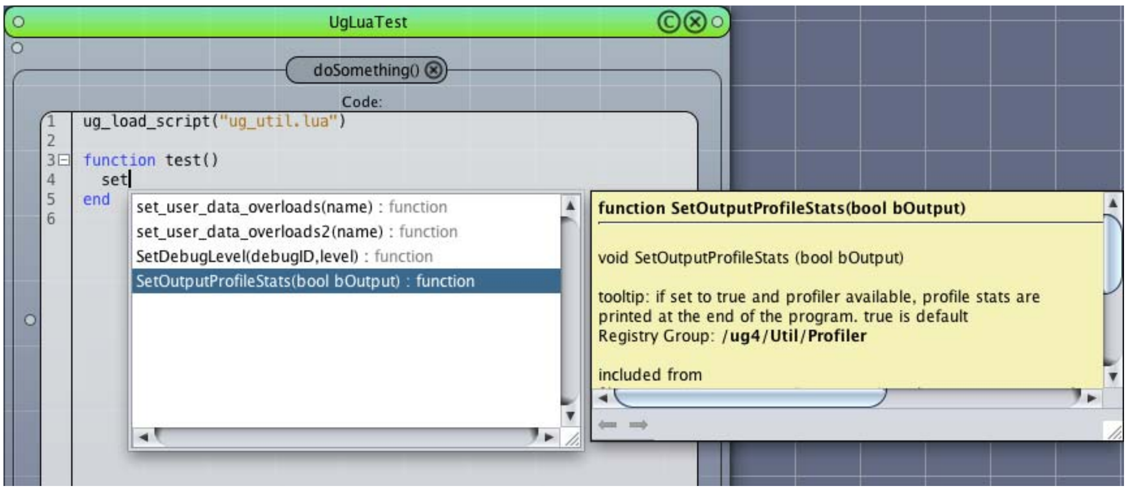
Fig. 2.26 Parse Tree (Lua Grammar)

The ANTLR syntax tree is then analysed by two separate components, the LuaAuto-Complete library and the UG4LuaAutoComplete library, like UG-specific functions and classes. Using the current cursor position within the document and the current text under the cursor position, those two components generate a set of proposals. The first library generates generic completion proposals for pure LUA code, the second adds UG shell specific proposals. This allows the LuaAutoComplete library to be used separately for pure LUA scripts that are not executed within the UG shell environment. The generated proposals take the current cursor position into consideration. Global and local variable scopes, scopes local to function and loop declarations are taken into account.

The run-time data type of the generated proposal objects is not specific for a given IDE. This allows the libraries to be reused in other IDEs like Eclipse or IntelliJ by mapping the generic library type to the specific IDE type. In the provided implementation a mapping between library and IDE types is performed for the VRL IDE. Here, the target data types are the data types used by the RSyntaxTextArea library, which provides a rich Swing-based text editor with syntax highlighting support.

UG shell-specific functions and classes with documentation

The UG shell provides several thousands of functions and hundreds of classes to a LUA script executed within its context. The underlying C++ implementation of the functions and classes often has extensive user documentation. This documentation as well as the function and class signatures are used by the LUA support to compute completion proposals. The function and class signatures are included from a custom "UGCompleter" file format provided during the build process of the UG4 environment. As it is possible that different UG shell versions are used between projects, specific "UGCompleter" files can be configured.



```
ug_load_script("ug_util.lua")

function test()
  set
end
```

```
set_user_data_overloads(name) : function
set_user_data_overloads2(name) : function
SetDebugLevel(debugID,level) : function
SetOutputProfileStats(bool bOutput) : function
```

function SetOutputProfileStats(bool bOutput)

void SetOutputProfileStats (bool bOutput)

tooltip: if set to true and profiler available, profile stats are printed at the end of the program. true is default
Registry Group: **/ug4/Util/Profiler**

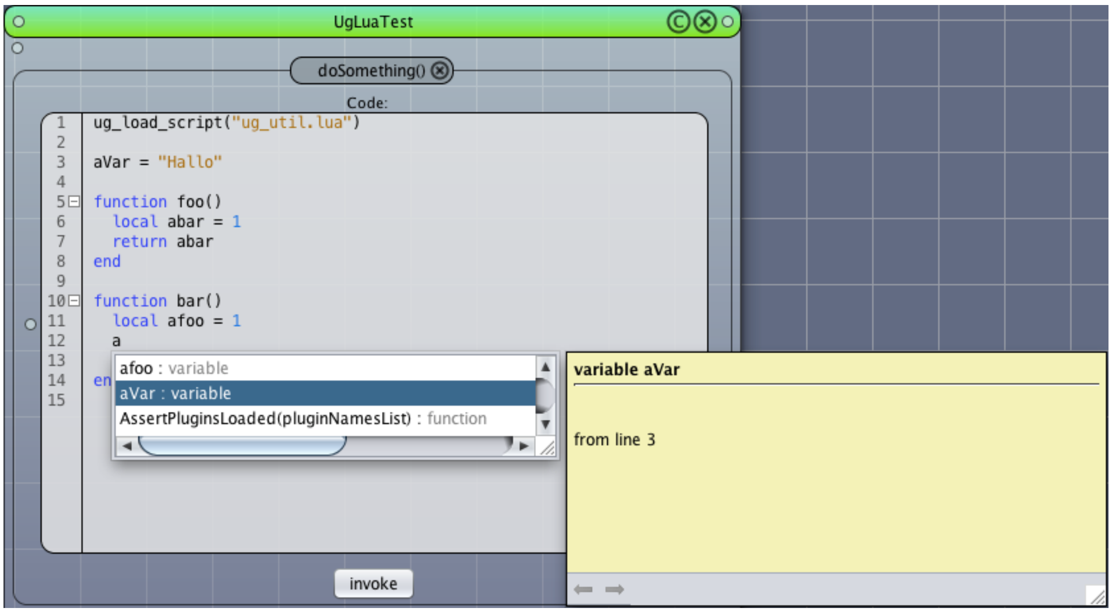
included from

Fig. 2.27 Auto Completion for UG scripts

Fig. 2.27 shows a completion proposal for the function SetOutputProfileStats, a UG4 shell function. In the text editor in the left part of the screenshot the curser position is at the end of the word ‘set’. A small popup window below the cursor lists set of possible completions for the word ‘set’. To the right of the selected list item is a descriptive pop-up window with details for the selected function SetOutputProfileStats.

Global and local variables from the current and included LUA scripts

The LUA support provides completion proposals for global and local variables and properly recognizes variable scopes, e.g. local variables only valid within a function or loop declaration.



```
1  ug_load_script("ug_util.lua")
2
3  aVar = "Hallo"
4
5  function foo()
6    local abar = 1
7    return abar
8  end
9
10 function bar()
11   local afoo = 1
12   a
13
14 en
15
```

```
afoo : variable
aVar : variable
AssertPluginsLoaded(pluginNamesList) : function
```

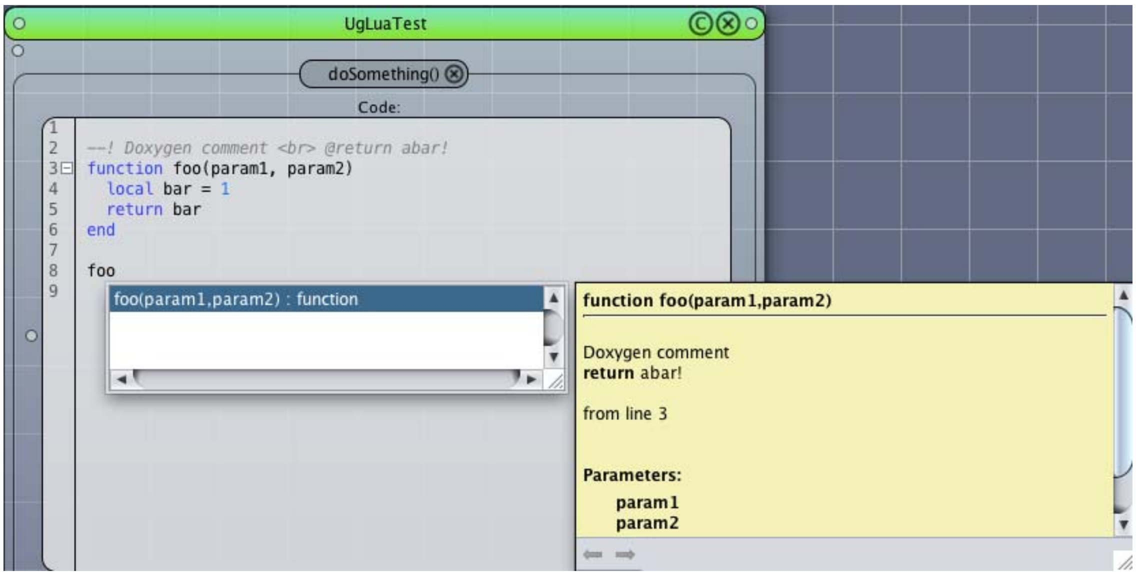
```
variable aVar

from line 3
```

Fig. 2.28 Auto Completion for Variables

In Fig. 2.28 the code defines two functions foo, bar with local variables ‘abar’ and ‘afoo’, as well as a global variable ‘aVar’. For line 12 the LUA support proposes only the variables ‘afoo’ and ‘aVar’, and properly ignores the variable ‘abar’ which is not visible within the scope of the function ‘bar()’. The descriptive window also provides the line number of the source code location used to generate the proposal.

Functions, parameters comments from current and included LUA scripts

Functions, their parameters and comments (with leading ‘—!’ statement) from the current script and all included scripts are provided as completion proposals. In the following sample a function ‘foo’ is defined with two parameters and a Doxygen parameter. The parameter list and the comment content are provided by the LUA support.



```
--! Doxygen comment <br> @return abar!
function foo(param1, param2)
  local bar = 1
  return bar
end

foo
```

```
foo(param1,param2) : function
```

function foo(param1,param2)

Doxygen comment
return abar!

from line 3

Parameters:
param1
param2

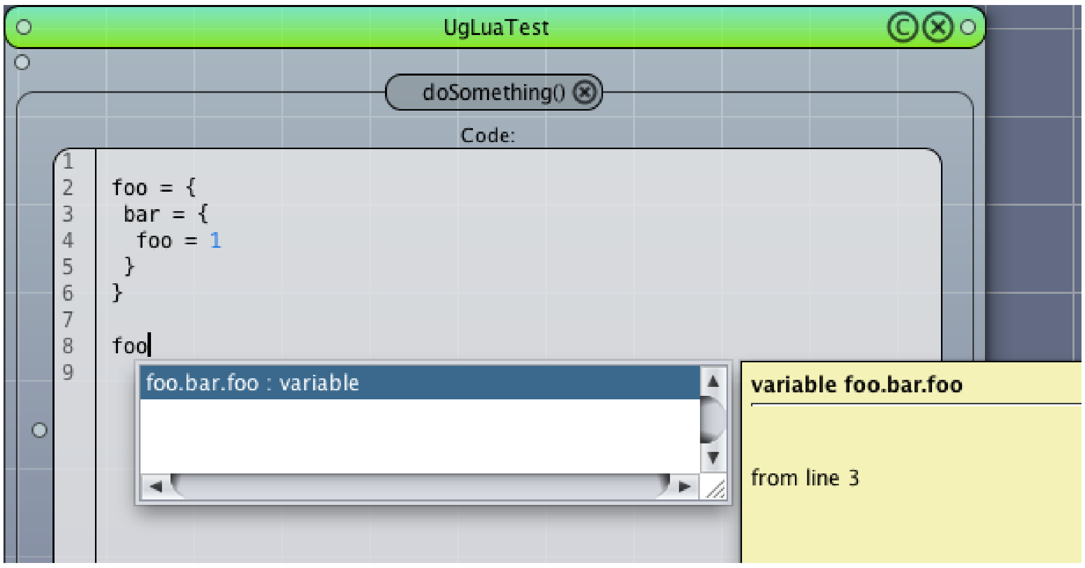
Fig. 2.29 Auto Completion for LUA Functions

The parameters param1 and param2 in Fig. 2.29 can directly be taken from the proposal list, the Doxygen comment is provided via the descriptive window to the right of the list.

Nested hash tables and arrays

A practical LUA language feature is its unverbose support for generic nested hash tables and arrays. The LUA support for hash tables recognizes defined literal keys and detects the full path to a nested variable declaration for completion proposals.

In Fig. 2.30 a hash table ‘foo’ is defined, with a nested set of keys ‘bar’ and ‘foo’. The LUA auto-completion proposes then completion for ‘foo.bar.foo’.



```
foo = {
 bar = {
  foo = 1
 }
}

foo
```

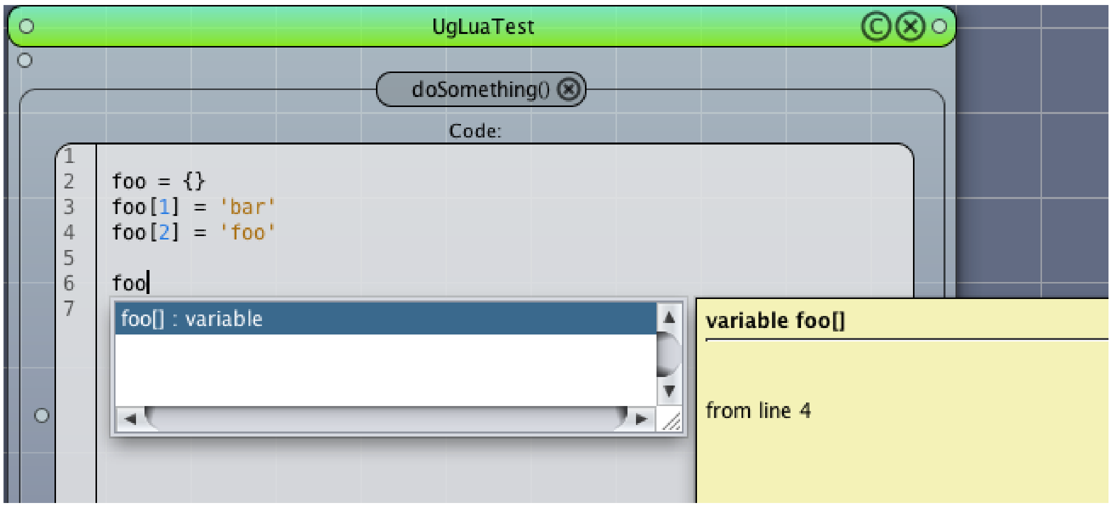
```
foo.bar.foo : variable
```

variable foo.bar.foo

from line 3

Fig. 2.30 Nested Tables

Arrays are supported too, when a variable is recognized as array proposals will be followed by a bracket like in the screen shot in Fig. 2.31.



```
foo = {}
foo[1] = 'bar'
foo[2] = 'foo'

foo
```

```
foo[] : variable
```

variable foo[]

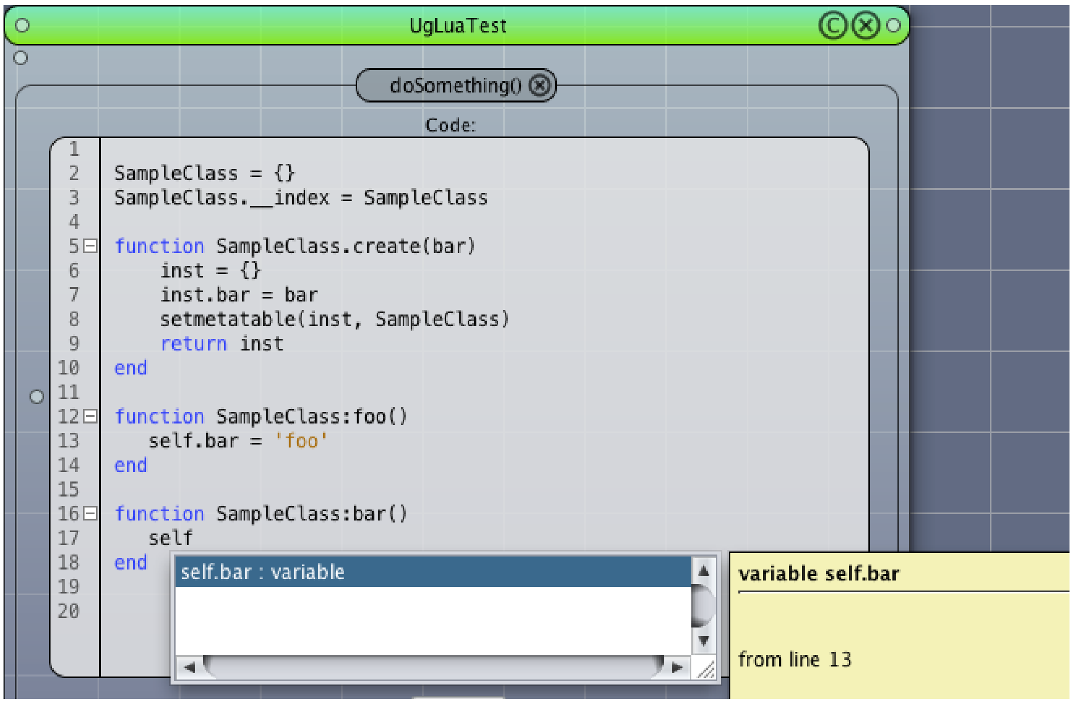
from line 4

Fig. 2.31 Array Completion

LUA class definitions

While there is no full class semantic defined in the LUA language specification, class-style semantics can be emulated by a mix of hash tables definitions and function pointers. A common way to define a class in LUA is depicted in the next screen shot, together with a sample proposal pop-up generated when listing the available class methods.

Support for the ‘self’ keyword is implemented. Within the scope of a member function of an existing class, all variables implicitly declared by a leading ‘self’ in other member functions are proposed by the auto completion feature. An example is shown in Fig. 2.32, where the member variable ‘self.bar’ declared in line 7 and used in line 13 is proposed within the block of the new member function ‘bar()’.



```
 1
 2  SampleClass = {}
 3  SampleClass.__index = SampleClass
 4
 5  function SampleClass.create(bar)
 6      inst = {}
 7      inst.bar = bar
 8      setmetatable(inst, SampleClass)
 9      return inst
10  end
11
12  function SampleClass:foo()
13     self.bar = 'foo'
14  end
15
16  function SampleClass:bar()
17     self
18  end
19
20
```

self.bar : variable

variable self.bar

from line 13

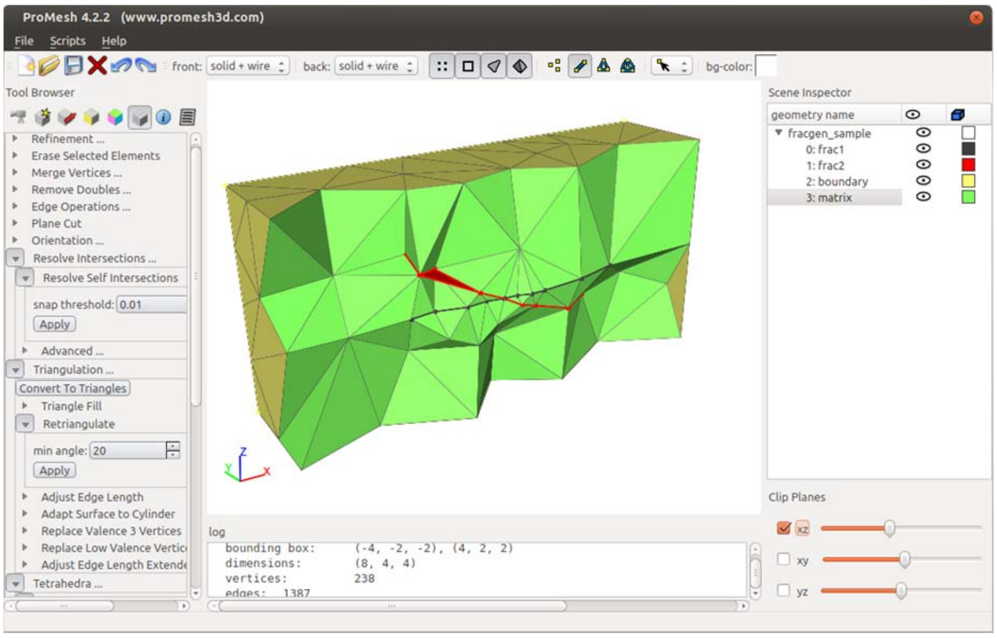
Fig. 2.32 Auto Completion for Self Keyword

2.4.3 ProMesh

A crucial aspect in simulations of groundwater flow is the accurate representation of the physical domain in which a problem is considered. Special features of such domains, like the extension and shape of different soil layers or the nature of possibly present fracture networks, can have a severe impact on the obtained flow patterns. Therefore a faithful reconstruction of those features in the computational domain is required.

Simulations of groundwater flow are not restricted to specific domains or even to specific scales. A toolchain that allows for the generation of the underlying computational grids thus shouldn't impose any unnecessary restrictions either. Instead such a tool should feature a set of specialized algorithms together with a broad set of more general meshing tools, which allow for the preparation, visualization, and manipulation of a broad range of grids through a common user interface. This high degree of flexibility and accessibility plays a key role in the design, implementation, and evolution of the cross platform meshing software ProMesh (cf. Fig. 2.33, /REI 14/).

Originally, ProMesh was developed as a stand-alone meshing software. However, with the development of UG4 and its powerful reflection mechanism (cf. /VOG 13/), the idea arose that a tighter coupling of the meshing functionality of ProMesh with the simulation aspects of UG4 could allow for the realization of even more complex simulation setups. Especially since UG4's reflection mechanism would then allow users to define integrated meshing and simulation procedures in one common script or graphical user interface.



```
bounding box:       (-4, -2, -2), (4, 2, 2)
dimensions:         (8, 4, 4)
vertices:           238
edges:  1387
```

Fig. 2.33 The ProMesh user interface

The implementation of ProMesh thus was split into a stand-alone graphical user interface (GUI), which is maintained to allow for manual editing, and a separate UG4 plugin, in which ProMesh's data-structures and algorithms are implemented and exposed to UG4's registry module. This registry serves as a C++ runtime reflection system which allows other programs to query for, to instantiate and to execute registered data-structures and algorithms. This reflection system is not only used by the ProMesh-GUI to automatically generate tool-representations for available meshing algorithms in its graphical user interface (cf. /VOG 13/), it is also used to expose ProMesh's meshing functionality to UG4's scripting system and to the Java based Visual Reflection Library (VRL, cf. /HOF 13/). This drastically broadens the scope of application in which ProMesh may be used. For example, one can now prepare a grid for simulation by embedding fully automated meshing steps in a simulation setup using the VRL or UG4 scripts. The potential of this interplay with regards to a unified and accessible meshing solution will be considered in the remainder of this section.

First, the general concepts behind ProMesh are reconsidered in Section 2.4.3.1. The interplay between UG4, ProMesh, and VRL is then examined in Section 2.4.3.2. Finally, examples that demonstrate the potential of the given approach with regards to a unified and accessible meshing toolset are given in Section 2.4.3.3.

2.4.3.1 Concepts

A central aspect of ProMesh is the visualization and manipulation of one-, two-, and three-dimensional grids consisting of edges, triangular and quadrilateral elements as well as tetrahedra, hexahedra, prisms and pyramids. Furthermore, ProMesh allows for the partitioning of a grid into subsets (or parts), which may then be used to associate different material properties or boundary conditions with the different parts of the grid during a simulation run. Those subsets are preserved even if topological or geometric changes are performed to the underlying grid.

Below, the basic concepts used by ProMesh will be given in more detail. Those concepts build a common ground on which the different meshing algorithms operate. Building on a set of such well defined concepts has the advantage that different algorithms can easily be combined to define more complex meshing methods (cf. /REI 14/).

Grids and Grid-Elements

Algorithms in ProMesh operate on grids consisting of vertices, edges, faces and volume-elements. The name grid-element refers to any of these. For a given grid G the set of vertices of G is denoted by N_G, the set of edges by E_G, the set of faces by F_G, and the set of volume elements by V_G. The grid G itself is then defined as the union of those element sets: $G \coloneqq \{N_G \cup E_G \cup F_G \cup V_G\}$.

With each vertex $n \in N_G$ a point in $\mathbb{R}^d$ is associated through the mapping

$$p: N_G \to \mathbb{R}^d. \tag{2.55}$$

Each grid-element $e \in G$ is defined by its set of corner vertices $n_1, n_2, \dots, n_m, m \geq 1$ through the mapping

$$[\cdot,\cdot, \dots ,\cdot]: (N_G)^m \to G. \tag{2.56}$$

Let $\Phi: G \to \mathbb{R}^d$ be an embedding of G into $\mathbb{R}^d$ such that:

1. For each vertex $n \in N_G$: $\Phi(n) \coloneqq p(n)$.

2. For each edge $e \in E_G, e = [n_1, n_2]$: $\exists \phi: \overline{\Phi(e)} \to [0,1]$, where ϕ is a homeomorphism and $\partial\Phi(e) = \{\Phi(n_1), \Phi(n_2)\}$.

3. For each face $f \in F_G: \exists \phi: \overline{\Phi(f)} \to \{x \in \mathbb{R}^2 \mid \|x\| \leq 1\}$, where ϕ is a homeomorphism and $\exists\, n_1, \dots, n_m \in N_G, e_1, \dots, e_m \in E_G: \partial\Phi(f) = \bigcup_{i=1}^{m} \Phi(n_i) \cup \bigcup_{i=1}^{m} \Phi(e_i)$.

4. For each volume-element v$\in V_G: \exists \phi: \overline{\Phi(v)} \to \{x \in \mathbb{R}^3 \mid \|x\| \leq 1\}$, where ϕ is a homeomorphism and $\exists n_1, \dots, n_m \in N_G, e_1, \dots, e_l \in E_G, f_1, \dots, f_k \in F_G$:

$\partial\Phi(f) = \bigcup_{i=1}^{m} \Phi(n_i) \cup \bigcup_{i=1}^{m} \Phi(e_i) \cup \bigcup_{i=1}^{m} \Phi(f_i)$.

For each $e \in G, \Phi(\mathrm{e})$ is then called the geometric representation of e.

A grid G is called **consistent** if the following conditions hold for all elements $e_1, e_2 \in G, e_1 \neq e_2$:

1. $\Phi(e_1) \cap \Phi(e_2) = \emptyset$, and

2. if $\dim(e_1) = dim(e_2)$:

   - $\overline{\Phi(e_1)} \cap \overline{\Phi(e_2)} = \emptyset$, or

   - $\exists e \in G, \dim(e) < \dim(e_1)$: $\overline{\Phi(e)} = \overline{\Phi(e_1)} \cap \overline{\Phi(e_2)}$.

For elements $e_1, e_2 \in G$ the following notation is used:

$$e_1 \in e_2 \Leftrightarrow \dim(e_1) < \dim(e_2)\, and\ \Phi(e_1) \subset \overline{\Phi(e_2)}. \qquad (2.57)$$

If G is consistent and $e_1 \in e_2$ holds, then e_1 is called an associated element of e_2 and vice versa. If furthermore $\dim(e_1) = \dim(e_2) - 1$ holds, then e_1 is called a side of e_2. If two elements share a common associated element, the elements are called neighboured elements.

Element selections

Element selections provide a central facility through which users can interact with ProMesh. Each element is therefore considered to be either selected or deselected. The set of selected elements will be denoted by $S_G \coloneqq \{e \in G\ |e \text{ is selected}\}$.

Elements can be assigned to S_G by user input, e.g. mouse gestures in the graphical user interface, or algorithmically. A vast variety of such selection algorithms exists. Such algorithms, for example, allow for the automated selection of neighboured elements of selected ones, or for the selection of elements based on special geometric or topological properties.

Many meshing algorithms require the user to specify the set of elements of a grid, on which the algorithm shall operate, e.g., for adaptive or anisotropic refinement, retriangulation, assignment of subsets (see below), and many other purposes. Through the concept of element selections, ProMesh provides an unified approach to specify those elements independent of the actual meshing algorithm that shall be applied. On the other hand, new meshing algorithms which build upon the concept of selections can easily be integrated into ProMesh's toolchain and user interface.

Subdomains (Subsets)

The partitioning of a domain into different subdomains can be very useful to allow for the definition of different parameter sets, discretization methods, or boundary conditions on those subdomains. ProMesh features so called subsets, which allow for an analogous partitioning of a grid. To this end, an index $i \in \{-1\} \cup \mathbb{N}_0$ is associated with each element through the mapping $sub_G: G \to \{-1\} \cup \mathbb{N}_0$:

$$sub_G(e) := \begin{cases} i \in \mathbb{N}_0, & e \text{ is assigned to subset } i, \\ -1 & e \text{ is not assigned to any subset.} \end{cases} \quad (2.58)$$

Subsets are typically defined by the user by assigning all elements of the current selection to a given subset. This has the advantage that the whole set of selection tools available in ProMesh can be used to efficiently and comfortably define subsets. During algorithms like refinement and remeshing, subsets are automatically preserved.

Fig. 2.34 illustrates how a subset can be defined on a given mesh using element selections.

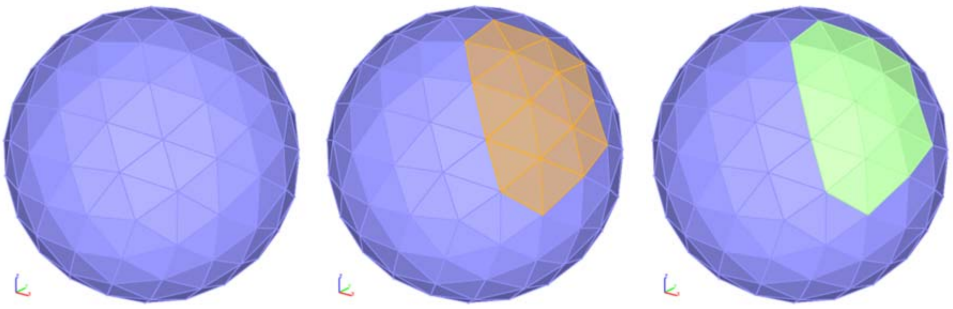


Fig. 2.34 Raw mesh (left), selected elements (middle), and new subset (right)

Tools and Scripting

Each algorithm that defines an operation that transforms a mesh, the current selection, or the subset structure, is accessible through a tool-dialog in the ProMesh-GUI. Using UG4's reflection mechanism, those tool-dialogs are automatically generated from the function signatures of the different registered meshing algorithms.

Furthermore it is possible to write scripts that call different algorithms on a given mesh. Those scripts can define import parameters using a special syntax in the comments section of each script. For each such script ProMesh then generates an additional tool-dialog through which the script can be executed. Those script-tools integrate seamlessly with the existing predefined tools, thus allowing for easy extendibility of ProMesh (cf. Fig. 2.35 and Fig. 2.36).

2.4.3.2 Integration of ProMesh into UG4 and VRL

While ProMesh already features an easy to use graphical user interface, it is often desirable to perform some of the required meshing steps as a part of a simulation run. Especially when the mesh properties have to be varied between different runs, e.g., for parameter estimation runs or uncertainty quantification, automated meshing can become a crucial step in the simulation setup.

To this end all meshing functionality previously only available in ProMesh has been transferred to a ProMesh-plugin for UG4. The ProMesh application thus now only contains code related to the graphical user interface as well as code that creates the tool-dialogs from available meshing algorithms. The meshing algorithms themselves are now provided by the ProMesh-plugin and are accessible through UG4's reflection mechanism. This setup has numerous advantages:

1. All meshing algorithms registered at UG4's registry are immediately available in UG4's scripting environment and can thus be used in script based simulation setups.

2. Users of ProMesh's graphical user interface can now use UG4's scripting facility to define more complex meshing algorithms by combining the already available ones in custom scripts. Those scripts are then available in the graphical user interface as additional tools.

3. All algorithms are also available in the VRL and can be used directly through visual representations as well as through UG4's scripting environment. They can thus easily be embedded in visual simulation setups. Furthermore scripts that define more involved meshing procedures for ProMesh or UG4 can be applied in between other meshing steps thanks to the new UG4-script integration in the VRL.

4. Using automated test scripts instead of manually executing algorithms in the graphical user interface for debugging purposes allows for a more rapid development of new meshing-algorithms.

One key feature for reusability and code reduction in this setup is the availability of a common scripting language which can be used by the ProMesh-GUI, the UG4 shell interpreter and the VRL. This allows for the implementation of utility methods in common scripts. The overall workflow and the objects on which those utility scripts operate are still defined by the respective application. This new approach is possible thanks to two key aspects. One is the already known reflection mechanism implemented in UG4 through the registry module. The other integral part is the availability of the script interpreter as a registered object itself. The interpreter can thus be queried and used through the already established registry binding for a given language. The interpreter features facilities to set and get parameters and object-references and to load and run scripts. Instances of this interpreter can then be used by the ProMesh-GUI, the UG4 shell interpreter and the VRL to load and run common meshing scripts with the provided mesh instance and custom parameters.

This new setup obviously allows for a very tight integration of meshing and simulation on many different levels. It reduces the implementation overhead tremendously, since script-bindings, tool-dialogs, and visual representations in the VRL are all created automatically from UG4's reflection mechanism. At the same time a highly specialized and intuitive graphical user interface is still available (ProMesh-GUI) and allows for the preparation of complex grids that require manual adjustment.

The approach presented above is an important step towards a work environment with clearly defined and coherent user interfaces and drastically improves the interoperability of the different tools involved.

2.4.3.3 Fractured domain meshing example with ProMesh, UG4, and VRL

To demonstrate the different meshing approaches, a sample domain shall be meshed using different frontends. The domain will contain two intersecting low dimensional fractures surrounded by a tetrahedral net representing the matrix. First the manual construction method using ProMesh's user interface is described. Then a script will be specified which automates the process, and finally, the whole meshing procedure will be specified using the visual programming language VRL. All three approaches are equivalent and result in the same grid (cf. Fig. 2.38), which demonstrates the high flexibility achieved through the new implementation of the ProMesh meshing functionality as a UG4 plugin.

Meshing with ProMesh-GUI

When manually meshing a domain with the ProMesh-GUI (cf. Fig. 2.33), a user executes a sequence of algorithms which create or transform the element structure of the currently active mesh. As depicted in Fig. 2.33 (left), those algorithms are represented by graphical tools in the Tool-Browser. By specifying the parameters of each tool and pressing the 'apply' button, a user can execute one tool after the other. The exact sequence in which the tools have to be applied to the current mesh in order to generate the desired grid is the same as in the scripting example in Fig. 2.35.

Meshing with ProMesh-Scripts

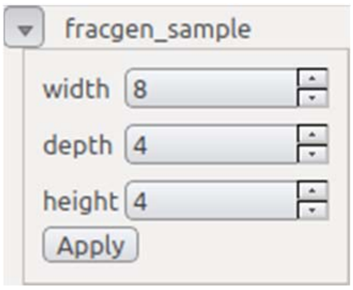
As detailed above, all meshing functionality available in the ProMesh-GUI is also available in the ProMesh/UG4 scripting environment. A mechanism was implemented in the ProMesh-GUI that automatically searches for meshing scripts in predefined folders. Furthermore, ProMesh-GUI features tools to generate new scripts easily (Menu-Scripts-NewScript). For each script a tool representation is created in the ProMesh-GUI, which allows users to specify custom parameters and to execute the script on the currently selected mesh (cf. Fig. 2.36 for the tool to the script from Fig. 2.35).

```
-- pm-declare-name: fracgen_sample
-- pm-declare-input: w | width | double | val = 8; min = 0
-- pm-declare-input: d | depth | double | val = 4; min = 0
-- pm-declare-input: h | height | double | val = 4; min = 0
CreatePlane(mesh, MakeVec(-2, 1, 0), MakeVec(2, 1, 0),
            MakeVec(-2, -1, 0), MakeVec(2, -1, 0), 0, true)
CreatePlane(mesh, MakeVec(-2, 1, 0), MakeVec(2, 1, 0),
            MakeVec(-2, -1, 0), MakeVec(2, -1, 0), 1, true)
RotateAroundCenter(mesh, MakeVec(0.4, 0.4, 0))
Move(mesh, MakeVec(0, 0.25, 0))
SelectAll(mesh)
ResolveSelfIntersections(mesh, 0.01)
Retriangulate(mesh, 20)
CreateBox(mesh, MakeVec(-0.5 * w, -0.5 * d, -0.5 * h),
          MakeVec(0.5 * w, 0.5 * d, 0.5 * h), 2, false)
Tetrahedralize(mesh, 10, false, false, false, true, 0)
```

Fig. 2.35 Mesh generation with ProMesh-Script. The variables mesh, w, d, h are provided by the calling application or script

The first lines contain comments describing the variables required by the script. Those variables have to be defined in the calling script interpreter. When a script is loaded into ProMesh-GUI, the script is automatically parsed for such comments and a tool-representation is created through which users can supply the required parameters. Those parameters are then set in the used script interpreter before actually executing the script itself.
All further lines contain code that operates on the provided mesh object. Most of those methods are provided by the UG4 ProMesh-plugin. It is of course also possible to mix in functions defined by other plugins, as well.

Instead of executing a script in the ProMesh-GUI, one could of course also run the script using ugshell or VRL by means of the UG4 script interpreter, after providing values for the required variables.


fracgen_sample

width 8
depth 4
height 4

Fig. 2.36 Tool representation generated by the ProMesh graphical user interface for the script from Fig. 2.35

Meshing with ProMesh-VRL

In the VRL, all available meshing algorithms provided by UG4's registry can be accessed through their visual representations. By creating a chain of such visual representations, one can describe complex meshing algorithms. Fig. 2.37 shows a setup that describes the exact same meshing procedure as specified in the script from Fig. 2.35.

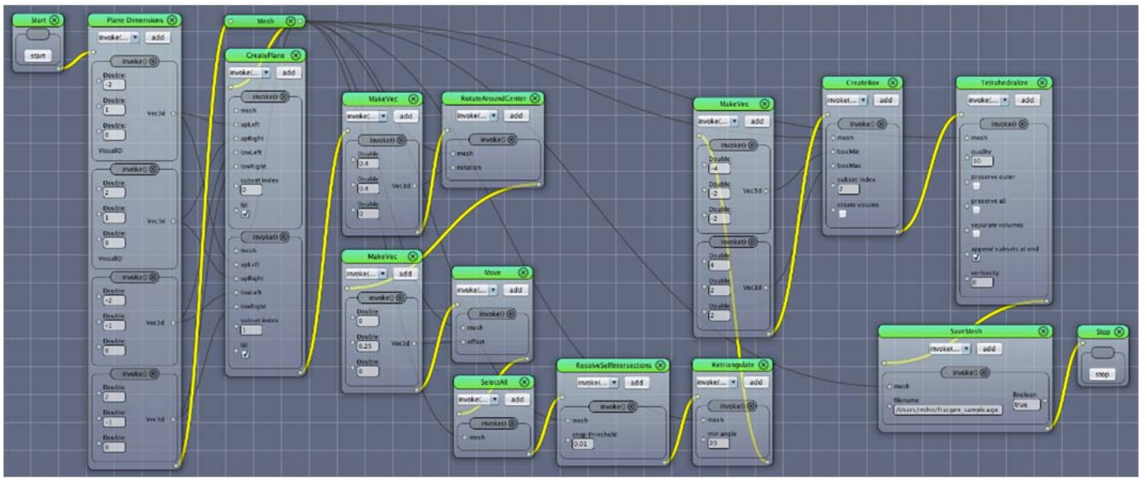


Fig. 2.37 VRL meshing sample

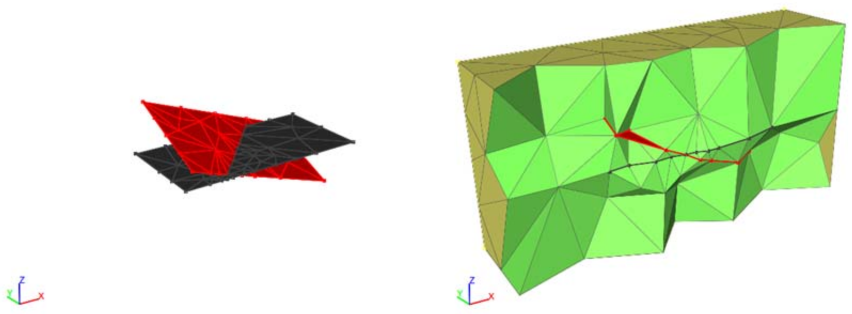


Fig. 2.38 Constrained Delaunay triangulation/tetrahedrization of intersecting fractures (left) and surrounding matrix (right). Grid generated with the meshing script from Fig. 2.35

2.4.4 Data output and visualisation

2.4.4.1 Overview

Visualising simulation data is an essential part of the simulation workflow and often necessary for developing an understanding of the simulation results. Therefore, it is important to provide capable visualisation tools within the application tool chain.

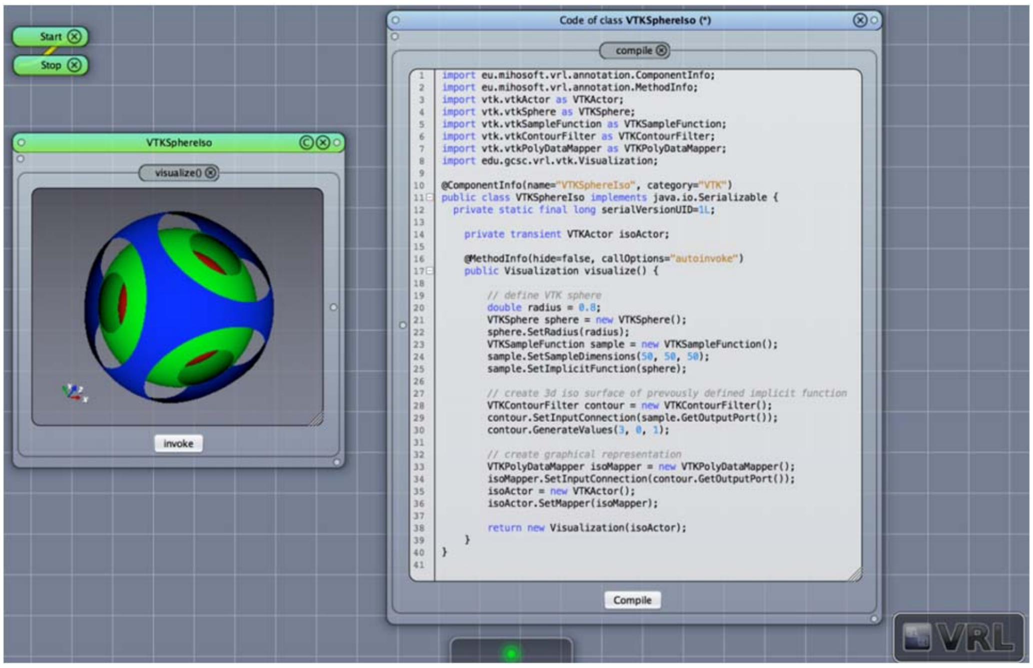
For UG based applications and d^{3}f++, several visualisation frameworks, such as VTK and JFreeChart have been integrated.

2.4.4.2 VTK

VTK is a powerful framework for interactive 2d and 3d visualisations. It provides a flexible API that supports several different programming languages /VTK 06/. Among others VTK provides components for complex surface rendering and volume rendering. Simulation results from UG4 based applications such as d^{3}f++ can be saved as VTK compatible output. This enables a broad variety of options for visualising simulation results.

2.4.4.3 VTK Plugin for d^{3}f++ based VRL-Studio Projects

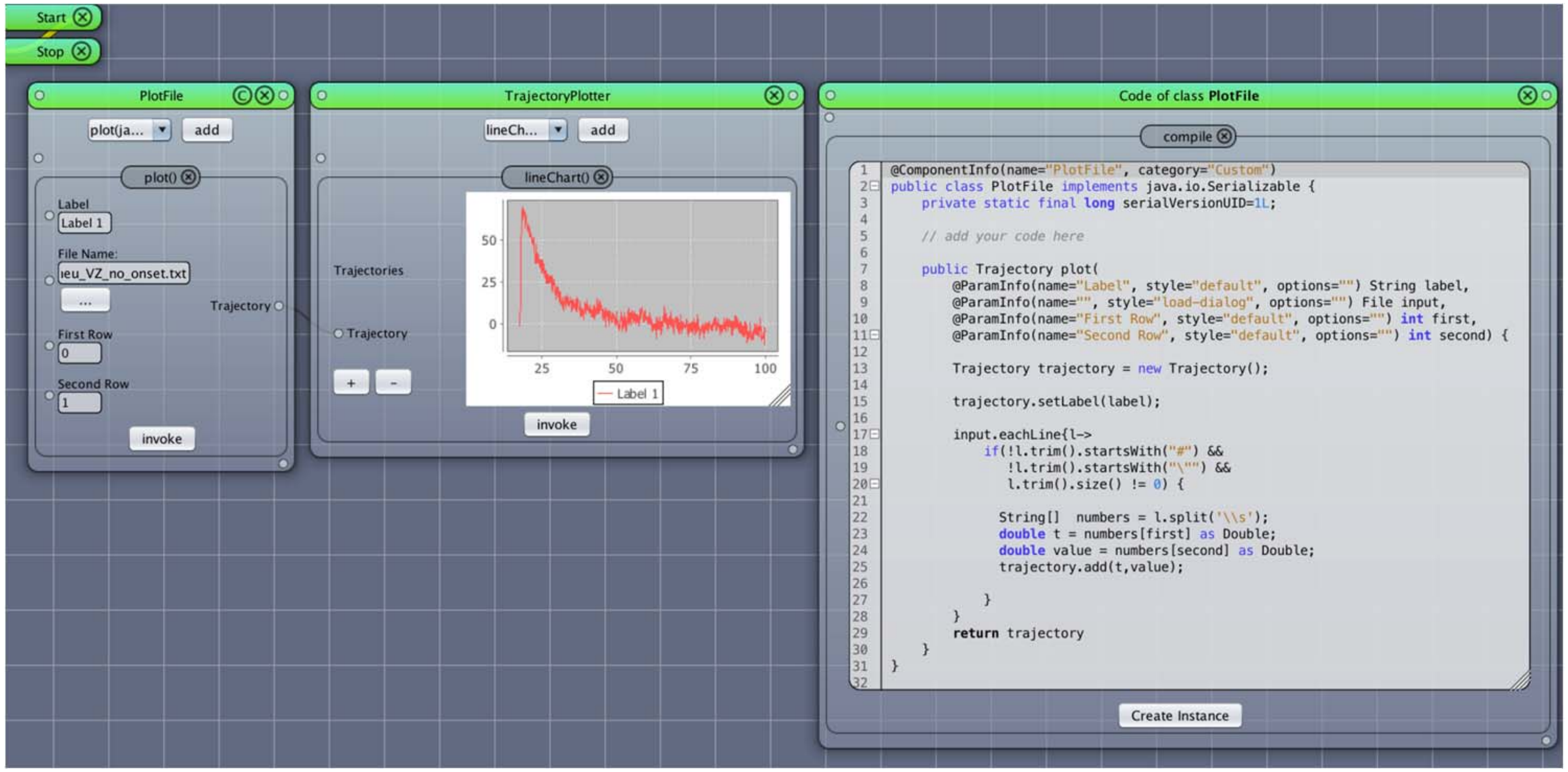
For VRL-Studio, a plugin has been developed that enables direct interaction with VTK visualisations. In previous projects the plugin mainly provided predefined visualisation components. In addition to this, the direct access to the VTK API via the VRL-Studio IDE has been improved. It allows for fully customised visualisation components that are specifically designed for the problem at hand. Fig. 2.39 shows a custom VTK visualisation that has been developed inside a VRL-Studio project.



```
import eu.mihosoft.vrl.annotation.ComponentInfo;
import eu.mihosoft.vrl.annotation.MethodInfo;
import vtk.vtkActor as VTKActor;
import vtk.vtkSphere as VTKSphere;
import vtk.vtkSampleFunction as VTKSampleFunction;
import vtk.vtkContourFilter as VTKContourFilter;
import vtk.vtkPolyDataMapper as VTKPolyDataMapper;
import edu.gcsc.vrl.vtk.Visualization;

@ComponentInfo(name="VTKSphereIso", category="VTK")
public class VTKSphereIso implements java.io.Serializable {
  private static final long serialVersionUID=1L;

    private transient VTKActor isoActor;

    @MethodInfo(hide=false, callOptions="autoinvoke")
    public Visualization visualize() {

        // define VTK sphere
        double radius = 0.8;
        VTKSphere sphere = new VTKSphere();
        sphere.SetRadius(radius);
        VTKSampleFunction sample = new VTKSampleFunction();
        sample.SetSampleDimensions(50, 50, 50);
        sample.SetImplicitFunction(sphere);

        // create 3d iso surface of prevously defined implicit function
        VTKContourFilter contour = new VTKContourFilter();
        contour.SetInputConnection(sample.GetOutputPort());
        contour.GenerateValues(3, 0, 1);

        // create graphical representation
        VTKPolyDataMapper isoMapper = new VTKPolyDataMapper();
        isoMapper.SetInputConnection(contour.GetOutputPort());
        isoActor = new VTKActor();
        isoActor.SetMapper(isoMapper);

        return new Visualization(isoActor);
    }
}
```

Fig. 2.39 Custom VTK Component, JFreeChart

JFreeChart is a Java based charting library. It supports various chart types, such as line chart, bar chats and histograms. For d^{3}f++ based applications a simplified API is integrated that can be directly used inside VRL-Studio projects. 2d charts can be generated with the JFreeChart plugin for VRL-Studio. Fig. 2.40 shows a custom 2d line chart visualisation.



Label: Label 1
File Name: ieu_VZ_no_onset.txt
First Row: 0
Second Row: 1

```
@ComponentInfo(name="PlotFile", category="Custom")
public class PlotFile implements java.io.Serializable {
    private static final long serialVersionUID=1L;

    // add your code here

    public Trajectory plot(
        @ParamInfo(name="Label", style="default", options="") String label,
        @ParamInfo(name="", style="load-dialog", options="") File input,
        @ParamInfo(name="First Row", style="default", options="") int first,
        @ParamInfo(name="Second Row", style="default", options="") int second) {

        Trajectory trajectory = new Trajectory();

        trajectory.setLabel(label);

        input.eachLine{l->
            if(!l.trim().startsWith("#") &&
               !l.trim().startsWith("\"") &&
               l.trim().size() != 0) {

                String[]  numbers = l.split('\\s');
                double t = numbers[first] as Double;
                double value = numbers[second] as Double;
                trajectory.add(t,value);

            }
        }
        return trajectory
    }
}
```

Fig. 2.40 2d Chart based on simplified JFreeChart API

2.5 Summary

In order to adapt the flow and transport simulations to the growing requirements of modern efficient numerical software, the renewed code basis UG4 for the simulation of coupled partial differential equations has been developed /VOG 13/. The new implementation is grounded on an object-oriented software design and written in C++.

To be able to participate in the current and future enhancements and numerical advances the UG-applications d^{3}f and r^{3}t had to be transformed to this new software platform. Benfitting the fact that coupling between different sets of equations is natively supportet by UG4, both codes were coupled in this process to the new code d^{3}f++. This allows the simultaneous simulation of density-driven groundwater flow and pollutant transport.

UG4 applications are controlled by scripting or a graphical user interface. Therefore, LUA-scripting and the Visual Reflection Language VRL had to be adapted to the needs of d^{3}f++. Additionally, d^{3}f++ profits of the UG4 pre-processor ProMesh that enables the user to set-up model geometries based on different types of data input and to generate the computational grid. ProMesh was also enhanced by several features that are helpful in the buildup of hydrogeological models in the new grid format ugx that is now used as a standard for all simulations.

For the output of the simulation results an interface to the well established VTK framework has been created, offering the possibility to use e. g. PARAVIEW or VISIT for visualisation.

3 Solvers

3.1 Multicore architectures and GPU

3.1.1 Multicore CPUs

Until recently, Moore's Law (rather an extrapolation from the past than a natural law) predicted that the number of transistors incorporated in a chip and the performance achieved therewith doubles about every two years. Nowadays, only the part about the biannual doubling of transistors per chip still holds, while the performance increase of individual processing units manufactured with new semiconductor technology is becoming smaller and smaller. The reason is simply that the clock speeds of current processors are hitting hard physical limits. Consequently, as performance can no longer be increased with higher clock frequencies, the industries' solution is to put more and more individual processing units (CPU cores) on a single chip. Such a multicore processor appears to a computer's operating system basically as a collection of multiple single CPUs, something long supported by all major server operating systems in use today. However, programs cannot use these additional CPU cores automatically, they will have to be rewritten and parallelized to benefit from multiple cores. This is in contrast to the old picture where new CPUs always ran faster than the previous models and software directly profited from this without any modification to the code. Another point to consider is the divergence of CPU speed and memory bandwidth. This gap has been widening all the time because increases in memory bandwidth could never keep up with the much faster increases in CPU clock speed. Now, with multicore processors including up to 16 cores and more to come, all sharing a single bus to the memory subsystem, things are getting worse and worse. Even codes that have grown up on traditional supercomputers like d^{3}f++ and therefore should already be capable of using multicore CPUs efficiently will have to be modified to be better adapted to these recent trends in microprocessor technology.

3.1.2 GPUs

A second new trend are "boosters", coprocessor cards that are put in computers alongside the CPUs and that are intended to accelerate certain parts of a traditional computer program that are offloaded to them. These boosters mostly stem from the graphics processing units (GPUs) of graphics cards. Trying to use them for non-graphical purposes became known as “general purpose computation on GPUs” (GPGPU). Early GPUs had their functionality mostly hardwired on the chip and they could only be programmed by presenting the data as textures and the operations to be applied as OpenGL drawing commands which made the whole subject rather esoteric. But after GPUs turned from hardwired to programmable, graphic card manufacturers saw a new market in GPGPU and by now are offering software development kits to target their chips directly in C-like programming languages. But GPUs differ considerably from CPUs. Their primary design is still targeted on graphics. Most of the chip’s silicon is devoted to processing units of which several (16, e.g.) are grouped together with a single control unit in a “streaming processor”. These are grouped again in larger units called “streaming multiprocessors” (192 cores, e.g.), and finally enough of them to get several thousand cores altogether are put on a single chip. Each streaming multiprocessor has some very fast local memory and a texture cache. They are all connected to the GPU’s main memory (several GB) via some high speed interconnect. A memory cache between the GPU’s main memory and the streaming multiprocessors may or may not be present. The general idea behind such an architecture is stream or throughput computing. Basically, all processing units are executing the same instructions but on different data. To alleviate the effect of insufficient memory bandwidth (of course it’s the same with GPUs as with CPUs) a ratio of 20−40 arithmetical instructions per memory access (high computational intensity) is advised. The control units also allows for multiplexing several threads to one streaming processor, with the idea of always having a thread ready to go in case the current one is waiting for hundreds of clock cycles on a memory access. The ideal GPU program has tens of thousands of threads active at any time.

3.1.3 Benefits of Multicore CPUs and GPUs

Multicore CPUs are not only the industry’s sole answer to the partial invalidation of Moore’s Law, there are also amenities for end-users. It’s now possible to put 20 or more CPU cores in a single chassis at pretty low costs. So far, d^3f dealt with multicore CPUs the same way as the operating system does: use every core as a single CPU (with a single core). Parallelization is then done via MPI which is already available in d^3f and r^3t since its beginnings. This however does not take into account that now all the cores in a multicore CPU share the same memory interface and its available bandwidth. Other programming models, like using OpenMP for the parallelization on a single multicore CPU, may be more adequate to tackle this problem. Another general idea is to reduce the number of memory accesses in favour of more computational instructions, e.g. by not assembling matrices for the solution of systems of linear equations but to use discretization stencils in connection with regular grids inside the linear solvers. These ideas are discussed in the sections below.

The primary appeal of boosters and GPUs are their theoretical performance data: over one TFlop/s in double precision arithmetic, an aggregated memory bandwidth of about ten times of what is available to a multicore CPU (due to optimized circuit board design not possible in a server with many memory sockets, and a low number of unsocketed memory chips directly soldered on the board, allowing for a much wider 368–512 bits memory bus compared to the 64 bits for multicore CPUs), and a power consumption that results in at least five times the performance of a multicore CPU at the same wattage. The main challenge is of course to reprogram and/or reformulate existing code and algorithms so that they fit in the GPU's stream-processing model described above, otherwise GPUs will not deliver anything close to their advertised performance data. Ideas to this end are similar to the ones mentioned for multicore CPUs above (namely increase data reuse and computational intensity, lower number of memory accesses) and will also be discussed in the following sections. An extra point for GPUs is the overhead due to data offload from the computer's main memory to the GPU before a GPU-boosted section can run, and the transfer back from the GPU to the computer's main memory afterwards.

As a general note it might be added that programs like d^{3}f++ do have a relatively high number of memory accesses compared to computational instructions and that no programming trick can ever change this. It is not expected that any adapted version of d^{3}f++ will ever come close to the theoretical peak performance of both multicore CPUs and GPUs. It's still reasonable to tune d^{3}f++ to get the most of current technology. And despite GPUs have the same problem with memory bandwidth that CPUs have, their overall higher aggregate memory bandwidth makes their use for d^{3}f++ still promising.

3.2 Improving the performance in d^{3}f++ using accelerators

The existing implementation provides a very high level of flexibility with respect to various aspects: It supports, e.g., unstructured grids with heterogeneous element types in arbitrary dimensions. It not only supports plain density-driven, but also thermal flow. Moreover, the discretisation is flexible enough to deal with different type of discretizations, boundary conditions, upwinding schemes etc.

Many of these features are of vital important for the end-user. However, this high level of flexibility is afflicted with *branchings* (if-then-else structures) in the machine code. Thus, it is inherently conflicting with the streaming properties of hardware architectures outlined in the introduction. As a consequence, the optimization of the existing code base of d^{3}f++ for a novel hardware paradigm is a challenging task. Seeking for optimality requires defining a trade-off between intrusions in the code and hardware-optimal execution.

To that end two different strategies were pursued: The first one, Strategy 1, accelerates the code by replacing the sub-routines of the linear algebra module (matrix-vector multiplications etc) only, while the second one, Strategy 2, aims at an optimization of the full code. Both approaches as well as the corresponding results are described in closer detail below.

3.2.1 Strategy 1: General Purpose Acceleration for Unstructured Grids

The routines of the linear algebra module (access to matrix elements, matrix-vector multiplications, solvers, etc) are separated from the rest of the code (grid-geometry and discretization and solvers) by a well-separated interface. Replacing this module only thus allows for a comparatively non-intrusive way of code-optimization.

In addition to the existing default CPU-algebra, now a separate GPU-algebra module extends UG4. Therein, for all operations on the GPU, specialized code is supplied. For the most prominent languages, CUDA and OpenCL, this is achieved by a common interface.

The lifecycle of objects in this module is as follows: Initially, all matrices and vectors are created and reside in the memory of the CPU. After the discretization has assembled these objects, they can be shipped (copied) to the GPU. These copy operations are expensive w.r.t time. Thus, in order to minimize the number of copy operations, additional state flags indicate, if the objects reside on CPU and GPU, CPU only, or GPU only.

In a test for strategy 1, test both CPU and GPU implementation are compared. Poisson's equation was solved on the unit square in 2d with Dirichlet boundary conditions. The solver was a linear iteration with Jacobi preconditioner, reducing the error by 12 orders of magnitude. The test was performed on a workstation (2x Intel Xeon, 32 GB memory, 8 cores per CPU; 2x NVIDIA® Tesla® GPU Accelerators for GPU boosting, each equipped with 5GB memory), with double with accuracy for floating operations. Tab. 3.2 shows an acceleration by a factor of about 10.

Tab. 3.1 Wall clock times for the solution with CPU and GPU implementation respectively

1/h	256	512	1024	2048	4096
DoFs	**66,049**	**263,169**	**1,050,625**	**4,198,401**	**16,785,409**
T_{CPU} [s]	0.270	2.730	19.460	178.330	1,249.100
T_{GPU} [s]	0.090	0.390	2.340	16.710	128.990
Acceleration factor	**3**	**7**	**8.32**	**10,.7**	**9.68**
CG Steps	195	383	753	1,487	2,957

3.2.2 Strategy 2: Optimized Implementation for Structured Grids

The approach in the previous section focused on optimizing the linear algebra only. The goal of Strategy 2 is to exploit the capabilities of the GPGPUs to a maximum extent. The whole infrastructure (linear algebra, discretization, solvers) was re-written and tailored for streaming-type state-of-the-art architectures. The result is a UG4 plugin for Just In Time compilation for Structured Grids (JITSG). This plugin realizes two strategies of vital importance for high performance on GPGPU systems:

3.2.2.1 Just in time (JIT) compilation

JIT is a strategy to generate machine code on the computational device once it is needed. In UG4, all model components, such as nonlinear density models, coefficients, boundary conditions etc. are typically formulated as functions in small LUA scripts. LUA is an interpreted programming language. This means, that the code is not compiled (translated) into an executable, but is executed in a step-by-step fashion by an interpreter. In a typical process, scripts are read at run time and execute a sequence of commands for C++ functions and objects. When these C++ objects have been pre-compiled, the loss of performance is typically within acceptable bounds. However, a severe problem occurs, if the member functions of the C++ object depend on LUA code themselves: Then, in each call of such function, the LUA interpreter is executed to perform some computation. Since calling the interpreter is slower and since addresses need to be resolved indirectly using pointers to LUA functions, this typically slows down the code by orders of magnitude and should be avoided by all means.

On a system with a single CPU as execution unit, this is typically not a problem, as LUA code can be transformed in C++ code manually. In a JIT framework, this can be done automatically; the implemented approach supports C++, Cuda and OpenCL. As an additional trick, the assembly of the matrix and the defect occurs just-in-time. In the classical d^{3}f framework, matrix and vector entries are stored in memory explicitly. In the new approach, they are only given implicitly by functions that are compiled just-in-time. Thus, the evaluation of a matrix entry the defect at a particular physical grid point corresponds to the call of a function.

3.2.2.2 Structured grids

The second component that is crucial for high performance is the use of structured (i.e., logically rectangular) grids. As all elements of the grid and all control volume for the finite volume method have essentially the same shape, this reduces the occurrence of branching and is suitable for the streaming architecture on accelerators. All entities such as elements, vertices, and matrix entries can be accessed by a triple (I,j,k) characterizing its position. References to neighbours are made by incrementing and decrementing the corresponding component accordingly. Since the size of matrices and vectors for each thread on the accelerator is known a-priori, there is also no need for dynamic memory allocation.

3.2.2.3 Numerical Results

The test problem for Strategy 2 is a convection diffusion equation in steady state

$$- \triangle u + \nabla[\bar{v}u] + r(x,y)u = 0 \text{ in } \Omega = (0,1)^2 \tag{3.1}$$

with $u(x,y) = \frac{y(1-y)}{2}$ for $\mathrm{x} = 0$, and $u(x,y) = 0$ in any other case on the boundary $\partial\Omega$.

Tab. 3.2 shows that the GPU implementation yields accelerations up to a factor of 40. However, since the accelerator features a smaller memory, the problem size is limited. Going beyond would require additional accelerators. Again the configuration is a CPU system with 2x8cores of the Intel Xeon with 32 GB RAM vs. a GPU system with 2 Tesla K 20 (2496 CUDA cores, 5GB memory each).

Tab. 3.2 Wall clock times for the solution with of a convection diffusion problem using JITSG (CPU vs. GPU implementation respectively)

1/h	256	512	1024	2048	4096	8192	16384
DoFs	**66,049**	**263,169**	**1,050,625**	**4,198,401**	**16,785,409**	**67,125,249**	**268,468,225**
T_{CPU} [s]	0.103	0.331	1.203	4.842	19.344	77.710	317.179
T_{GPU} [s]	0.016	0.023	0.047	0.138	0.488	1.899	---
Acceleration factor	**6.44**	**14.39**	**25.6**	**35.09**	**39.64**	**40.92**	---
Lack of data (---) indicates that for the corresponding run, the machine ran out of memory.							

3.3 Multigrid solvers

In /REI 13/, /HEP 13/, /VOG 13/ a massively parallel geometric multigrid approach implemented in the software framework UG4 has been presented. This approach has shown very good weak scaling properties up to hundred thousands of computing cores on largest computer clusters for the laplacian equation. Further details on the implementation and algorithmical aspects can be found in above references and in /REI 14/, /VOG 14/.

This parallel geometric multigrid approach has been adapted for problems of density driven flow type. In the following the analysis of the parallel efficiency and speedup gained with this approach for haline flow problems is presented.

For the benchmark problem the solver shows very good scaling properties on massively parallel systems.The benchmark problem focuses on the following haline flow problem.

$$\begin{cases} \text{Find } (\omega, p), \text{such that} \\ \partial_t(\phi\rho) + \nabla \cdot (\rho \boldsymbol{q}) = q, \ \ in\, \Omega, \\ \partial_t(\phi\rho\omega) + \nabla \cdot (\rho\omega\boldsymbol{q} - \rho\boldsymbol{D}\nabla\omega) = q_s, \ \ in\, \Omega, \\ \boldsymbol{q} = -\dfrac{\mathbf{k}}{\mu}(\nabla p - \rho\boldsymbol{g}). \end{cases} \tag{3.2}$$

Tab. 3.3 Physical parameter for the scaling benchmark problem

symbol	quantity	unit	value
ϕ	porosity	–	0.1
$\mathrm{D_m}$	mol. diffusion	$m^2 s^{-1}$	3.565e-6
$\mathbf{D}_{\mathrm{disp}}$	mech. dispersion	$m^2 s^{-1}$	0
$\mathbf{k}$	permeability	m^2	4.845e-13
$\mathbf{g}$	gravity	$m\, s^{-2}$	- 9.81
ρ	density	$kg\, m^{-3}$	$1000 + 200\,\omega$
μ	viscosity	$kg\, m^{-1}\, s^{-1}$	1e-3

The physical parameters are chosen as listed in Tab. 2.1.

The equations are considered on a 2m x 1m domain and the boundary conditions are chosen as shown in Fig. 2.6.

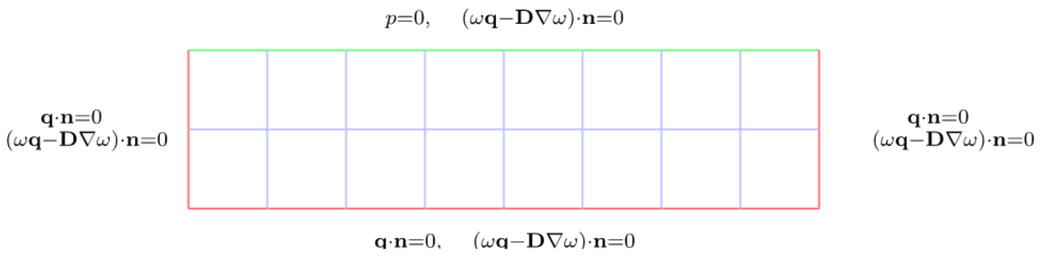

$p=0, \quad (\omega\mathbf{q}-\mathbf{D}\nabla\omega)\cdot\mathbf{n}=0$
$\mathbf{q}\cdot\mathbf{n}=0$
$(\omega\mathbf{q}-\mathbf{D}\nabla\omega)\cdot\mathbf{n}=0$
$\mathbf{q}\cdot\mathbf{n}=0$
$(\omega\mathbf{q}-\mathbf{D}\nabla\omega)\cdot\mathbf{n}=0$
$\mathbf{q}\cdot\mathbf{n}=0, \quad (\omega\mathbf{q}-\mathbf{D}\nabla\omega)\cdot\mathbf{n}=0$

Fig. 3.1 Domain and boundary conditions for the parallel scaling problem

As initial condition hydrostatic pressure is set and for the brine mass fraction a linear function from 1 at the top of the domain to 0 at the bottom is. Since the fluid with higher density is situated in the upper part of the model a downward flow is expected.

In order to test the parallel multigrid solver the first time-step and the first newton linearization is considered. The solver inverts the Jacobian matrix using multiplicative multigrid, V-cycle, ILU smoother, two pre- and postsmoothing steps and a LU factorization as base solver. The system of equations is solved until an absolute size of the residuum of 1e-9 is reached in the L2-norm. Thus, the number of iterations is not fixed a-priori and constant iteration counts indicate a robustness of the solver.

The results of the scaling study are shown in Tab. 3.4. Up to 131,072 computing cores are used. With every increase of the number of processes the solved problem has also been increased by one more grid refinement such that at the largest process number the problem has about 8.6 billion degrees of freedom.

Focussing on the algorithmic aspect of the multigrid iteration the results are very satisfactory since the number of iterations needed to achieve the prescribed accuracy remains constant over the whole range of problem and process numbers. In addition the consumed wallclock time to solve the entire problem – including programm startup, grid loading and refinement, problem setup, matrix assembling and matrix inversion – is presented. The achieved parallel efficiency of about 70% is very satisfactory.

In Tab. 3.5 a closer look at separate code phases and its performances is shown for the weak scaling study. Since the assembling process is inherently parallel a perfect parallel efficiency of 100% is observed as expected. The time and efficiency for the solver initialization and the solver execution show good results. A parallel efficiency of over 80 % at 131 thousand computing cores is achieved. The timings are graphically shown in Fig. 3.3. In Fig. 3.2 the ideal speedup is compared to the gained speedup for the different execution phases. A close to optimal speedup is observed.

Tab. 3.4 Weak scaling results
Level: Grid refinements, DoFs: degrees of freedom, Iterations: number of multigrid iterations, Time: whole program run, Efficiency: parallel efficiency

Processes	Level	DoFs	Iterations	Time [s]	Efficiency [%]
32	8	2’102’274	11	37.96	-
128	9	8’398’850	11	38.15	99.5
512	10	33’574’914	11	39.37	96.4
2’048	11	134’258’690	11	40.18	94.5
8’192	12	536’952’834	11	41.11	92.3
32’768	13	2’147’647’490	10	48.45	78.3
131’072	14	8’590’262’274	10	53.37	71.1

Tab. 3.5 Weak scaling: Times and efficiency for code phases

Processes	Time [s] Assemble	Eff. [%] Assemble	Time [s] Setup	Eff. [%] Setup	Time [s] Solve	Eff. [%] Solve
32	6.15	-	4.93	-	8.62	-
128	6.16	99.8	4.86	101.4	8.70	99.1
512	6.11	100.7	4.97	99.2	9.31	92.6
2’048	6.18	99.5	5.09	96.9	9.45	91.2
8’192	6.13	100.3	5.03	98.0	9.96	86.6
32’768	6.17	99.6	6.22	79.3	10.84	79.6
131’072	6.10	100.7	5.99	82.3	10.66	80.9

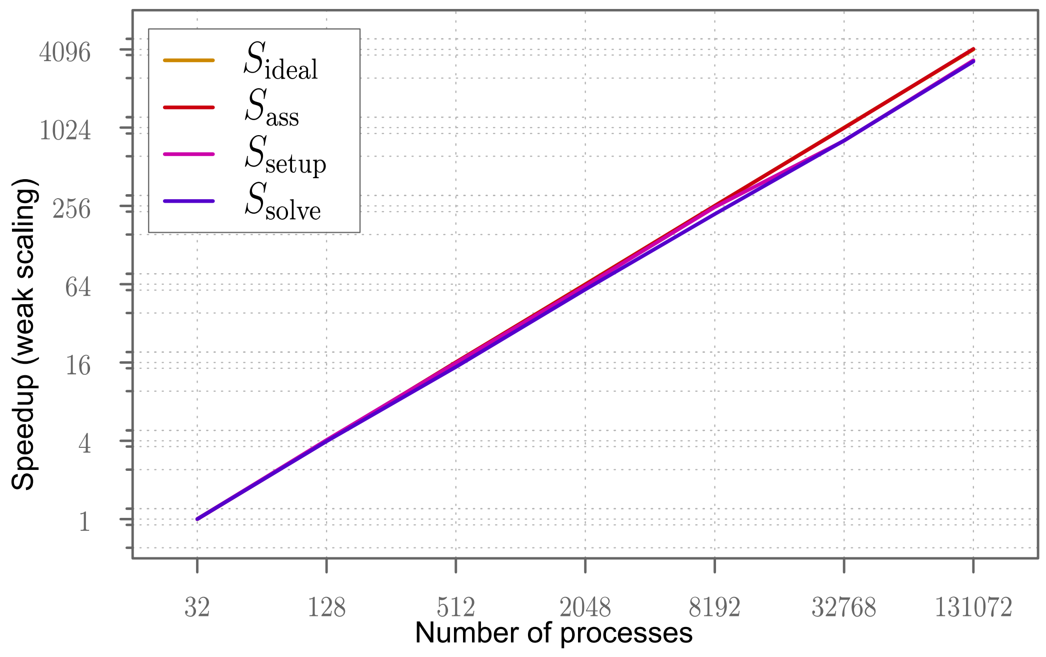


Fig. 3.2 Measured speedup for the parallel scaling problem

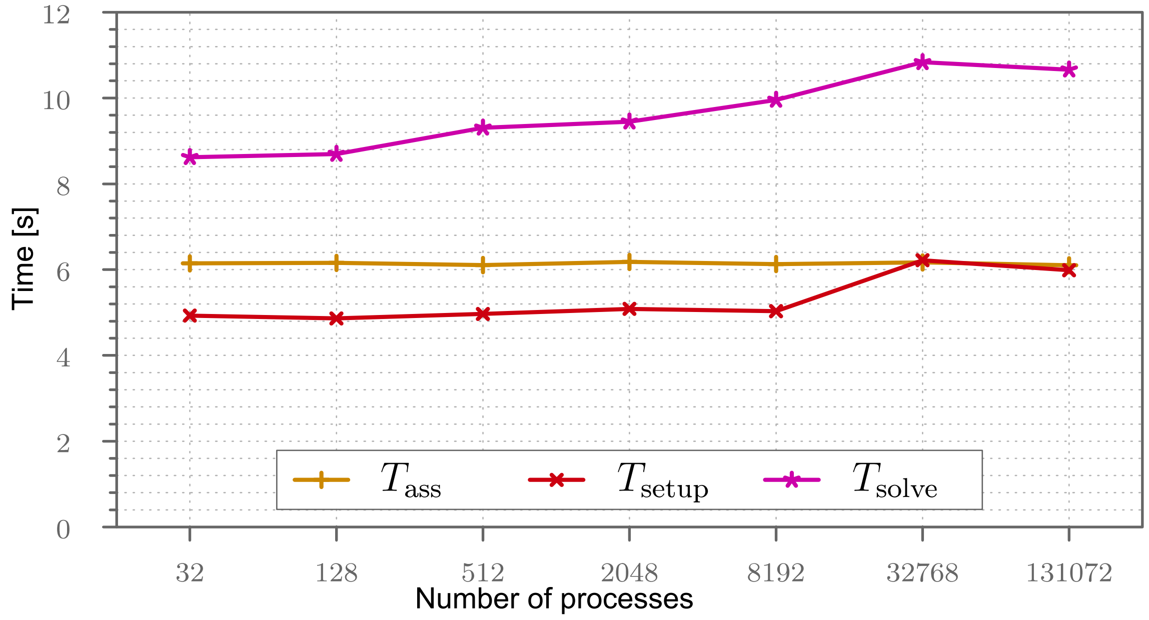


Fig. 3.3 Measured timings for the parallel scaling problem

3.4 Solvers for nonlinear transient problems

The demand for fast solvers for this time-dependent, non-linear process is obvious: In each single time step, a non-linear equation must be solved. In a classic setup, this is typically achieved by some fixed-point iteration. Since a fully coupled Newton iteration is regarded being very demanding with respect to both discretization and solvers, often variants of Picard or Newton iterations are preferred. These are investigated in Section 3.4.2. As an alternative to the fixed point iteration, one can consider linear implicit iterations. These are discussed in Section 3.4.3. Parts of this section have also been published in /NAE 15/.

3.4.1 Preliminaries

The governing equations in this section are the continuity equations for fluid and salt mass /BEA 91/, /HOL 98/:

$$\partial_t(\Phi\rho) + \nabla \cdot [\rho\mathbf{q}] = \rho Q \tag{3.3}$$

$$\partial_t(\Phi\rho\omega) + \nabla \cdot [\omega\rho\mathbf{q} - \rho\mathbb{D}\nabla\omega] = \rho Q \tag{3.4}$$

The system is closed by constitutive equations, e.g., for the Darcy velocity

$$\mathbf{q} = -\frac{K}{\mu}(\nabla p - \rho\mathbf{g}) \tag{3.5}$$

as well as for the permeability K, viscosity μ etc. For the sake of simplicity boundary conditions are not considered explicitly.

The goal is to introduce solvers for problems (3.3) and (3.4). These are based on fixed-point iterations. In an abstract setting, a solution $u = (p, \omega)^T$ has to be found for

$$\mathcal{F}_p(p, \omega) = 0, \tag{3.6}$$

$$\mathcal{F}_\omega(p, \omega) = 0. \tag{3.7}$$

For the purpose of illustration and motivation, e.g., /JOH 06/, considers $\mathcal{F}$ given by (3.6) and (3.7). Linearising at $u_0 = (p_0, \omega_0)^T$ the Newton method determines a search direction $(\delta p, \delta\omega)^T$ as the solution of

$$\nabla \cdot \left[-\frac{K}{\mu} \nabla \delta \mathrm{p} + \mathbf{q}_0, \delta\omega \right] = -\mathcal{F}_{p,0} \tag{3.8}$$

$$\partial_t(\Phi\delta\omega) + \nabla \cdot \left[-\omega_0 \frac{K}{\mu_0} \nabla\delta p + (\omega_0 \mathbf{q}_0, + \mathbf{q}_0)\delta\omega - \mathbf{D}_0 \nabla \delta\omega \right] = -\mathcal{F}_{\omega,0} \tag{3.9}$$

Here, all quantities with the subscript 0 are evaluated at the linearization point u_0. In particular

$$\mathbf{q}_0 := -\frac{K}{\mu}(\nabla p_0 - \rho_0 \mathbf{g}), \quad \mathbf{q}_0' := \frac{K}{\mu} \rho'_0 \mathbf{g}, \quad \rho'_0 = \rho'(\omega_0) \tag{3.10}$$

are the Darcy velocity, its derivative w.r.t. ω, and the derivative of ρ in the linearization point respectively. For the sake of simplicity, derivatives of the dispersion tensor $\mathbf{D}_0$ and the viscosity μ_0 have been neglected.

Eqs. (3.8) and (3.9) allow deducing the following facts for this system: First, the problem is elliptic w.r.t. p and the parabolic w.r.t. ω. Second, if $\omega_0 = \mathrm{const}$, the variables decouple, since the dependence on δp in (3.9) may be eliminated by means of (3.8). In this case, one can first solve for $\delta\omega$ and then, in a next step for δp. Note that although this assumption is unrealistic, it may be fulfilled in parts of the computational domain, e.g. /NAE 15/.

3.4.2 Nonlinear Solvers I: Classic Newton-type schemes

The aforementioned system is discretized in space and time. For times t_n let $\mathbf{u}_h^{(n)} = (\mathbf{p}_h^{(n)}, \omega_h^{(n)})^T$ denote the vector with coefficients w.r.t. the space discretisation. Given $\mathbf{u}_h^{(n)}$ assume that the step $t_n \to t_{n+1} := t_n + \tau$ is performed using an implicit Euler method. This yields a non-linear equation for $\mathbf{u}_h^{(n+1)}$ at time t_{n+1}:

$$\mathbf{F}_h(\mathbf{u}_h^{(n+1)}) = \mathbf{L}_h(\mathbf{u}_h^{(n+1)}) + \mathbf{E}_h(\mathbf{u}_h^{(n)}) = 0 \tag{3.11}$$

The term $\mathbf{E}_h(\mathbf{u}_h^{(n)})$ summarizes all explicit dependencies on the solution $\mathbf{u}_h^{(n)}$ at the old time, whereas $\mathbf{L}_h(\mathbf{u}_h^{(n+1)})$ summarizes the implicit, non-linear dependencies on the solution $\mathbf{u}_h^{(n+1)}$. As a result of the time discretisation may be written $\mathbf{L}_h = \mathbf{M}_h + \tau\mathbf{A}_h$.

Rewriting (3.11) componentwise yields

$$\mathbf{F}_{p,h}(\mathbf{p}_h^{(n+1)}, \omega_h^{(n+1)}) = 0, \tag{3.12}$$

$$\mathbf{F}_{p,h}(\mathbf{p}_h^{(n+1)}, \omega_h^{(n+1)}) = 0. \tag{3.13}$$

This must be solved by some fixed-point iteration. Various strategies exist; in this study, three different iterative approaches are focussed:

- Early works, e.g., /PUT 95/, highlighted the benefits of a *partial Newton* method. These approximate the Jacobian and consider only the self couplings for each unknown component. This strategy is also employed, e.g., in FEFLOW /DIE 98/, /DIE 09/. These works describe a predictor-corrector with an explicit predictor and an implicit corrector. The scheme is also suitable for thermohaline flow and features a time stepping strategy and error estimates.

- A related class of solvers are *iterative coupling* strategies. These provide a natural way to couple different modules and can be considered as variants of operator splitting technique. This class has widely been applied, e.g., to multiphase flow /LAC 01/, /LU 09/, or geomechanics /KIM 11a/, /KIM 11b/, /MIK 13/, /MIK 14/. Based on a Picard iteration a similar (partially explicit) strategy is pursued in MODFLOW /LAN 06/, /LAN 08/.

- However, *fully coupled Newton* iterations have also been applied successfully to both density driven /JOH 02/, /LAN 05/ and thermohaline flow /GRI 10/ based on the d^3f software /FEI 99/, /JOH 04/.

To look for the solution at a fixed time t_{n+1}, the time superscript index $n+1$ is dropped. Instead the iteration index is introduced as a subscript , i.e., $\mathbf{u}_{h,k} = (\mathbf{p}_{h,k}, \omega_{h,k})^T$.

3.4.2.1 Full coupling: Newton method

The standard approach is to employ a Newton method to the fully coupled system. Defining

$$\mathbf{u}_{h,k+1} = \mathbf{u}_{h,k} + \delta\mathbf{u}_h \tag{3.14}$$

The objective is to find a root of the linearised defect equation
$\mathbf{F}_h(\mathbf{u}_{h,k+1}) \approx \mathbf{F}_h(\mathbf{u}_{h,k}) + J_h(\mathbf{u}_{h,k})\delta\mathbf{u}_{h,k} = 0$, i.e.,

$$J_h\delta\mathbf{u}_{\mathbf{h},\mathbf{k}} = \begin{pmatrix} J_h^{pp} & J_h^{pw} \\ J_h^{wp} & J_h^{ww} \end{pmatrix}\begin{pmatrix} \delta\mathbf{p}_h \\ \delta\omega_h \end{pmatrix} = -\mathbf{F}_h(\mathbf{u}_{h,k}). \tag{3.15}$$

Here, $\mathbf{F}_h(\mathbf{u}_{h,k})$ from (3.11) is the nonlinear defect of the current iterate, and $J_h = J_h(\mathbf{u}_{h,k})$ is the Jacobian. The vector $\delta\mathbf{u}_\mathbf{h}$ is the resulting correction and search direction respectively. Typically, a line search strategy is employed for a globalisation of the method.

3.4.2.2 Approximate Coupling: Partial Newton

Modifying (3.15) slightly one can approximate J_h by its diagonal /PUT 95/, /DIE 98/:

$$\tilde{J}_h\delta\mathbf{u}_{\mathbf{h},\mathbf{k}} = \begin{pmatrix} J_h^{pp} & 0 \\ 0 & J_h^{ww} \end{pmatrix}\begin{pmatrix} \delta\mathbf{p}_h \\ \delta\omega_h \end{pmatrix} = -\mathbf{F}_h(\mathbf{u}_{h,k}). \tag{3.16}$$

This strategy is also referred to as *partial Newton method* /PUT 95/, /DIE 09/. Note that solving (3.16) is much easier than solving the fully coupled system (3.15): Since the matrices J_h^{pp} and J_h^{pp} correspond to discretisations of a Poisson-type problem and convection-diffusion equation respectively, good preconditioners are available. As a downside of this advantage, the method will, in general, only provide linear convergence.

One should stress that (3.16) can be viewed as a single Newton step applied to the system

$$\mathbf{F}_{p,h}(\mathbf{p}_{k+1}, \omega_k) = 0, \tag{3.17}$$

$$\mathbf{F}_{\omega,h}(\mathbf{p}_k, \omega_{k+1}) = 0. \tag{3.18}$$

This corresponds to an inexact nonlinear Jacobi iteration, where a potential line-search strategy provides a suitable damping factor.

3.4.2.3 Iterative Coupling: Nonlinear Gauss-Seidel

As a last alternative, a strategy is studied also referred to as *iterative coupling* in the context of different equations /LAC 01/, /LU 09/, /KIM 11a/, /MIK 13/. Starting from (3.1.2), it is stated as a non-linear Gauss-Seidel type iteration /RHE 98/ here:

$$\mathbf{F}_{\omega,h}(\mathbf{p}_k, \omega_{k+1}) = 0, \tag{3.19}$$

$$\mathbf{F}_{p,h}(\mathbf{p}_{k+1}, \omega_{k+1}) = 0. \tag{3.20}$$

Again, both equations are treated and solved independently. In contrast to (3.1.2), each substep employs the latest update that is available. Following /ACK 04/, it is solved for ω first and then for p. In this ω, p-ordering, a new distribution of salt is firstly computed as a result of an unmodified pressure distribution. Then, in the next step, a suitable pressure is determined. This can be viewed as a projection of the solution into the space where the conservation of fluid mass holds.

Like the partial Newton method from the previous subsection, this approach will at best provide linear convergence. The method is in particular attractive, when flow and transport equation are discretised and solved in different code modules. By adding an additional outer loop, the problem can be solved without any changes to the algorithmic design.

3.4.3 Nonlinear Solvers II: Linear Implicit schemes

The algorithmic description in Section 3.4.2 followed the classical approach solving a nonlinear problem in each time step. In the following, it shall be suggested an alternative that comes from the theory of ordinary differential equations /DEU 90/, /DEU 02/. Its major advantages are that (i) it requires to solve a single system of equations only, and (ii) can elegantly be combined with error estimates and adaptive time-stepping.

The first step is to reformulate (3.3), (3.4): For $u = (\omega, \mathrm{p})$, let

$$\mathcal{Q}(\mathrm{u}) \overset{\text{def}}{=} \begin{pmatrix} \Phi\rho(\omega) \\ \Phi\rho\omega \end{pmatrix}, \qquad \mathcal{F}(\mathrm{u}) \overset{\text{def}}{=} \begin{pmatrix} \omega\rho Q - \nabla \cdot [\omega\rho\mathbf{q} - \rho\mathbb{D}\nabla\omega] \\ \rho Q - \nabla \cdot [\rho\mathbf{q}] \end{pmatrix} \tag{3.21}$$

Then (3.3), (3.4) may be restated as

$$\frac{\partial \mathcal{Q}(u)}{\partial t} = \mathcal{F}(\mathrm{u}) \tag{3.22}$$

The next step is to bring this into a Cauchy problem in quasi-linear form. Assuming that $\rho = \rho(\omega)$ is differentiable the left hand side can be evaluated component-wise:

$$\partial_t(\Phi\rho) = \Phi\,\frac{\partial\rho}{\partial\omega}\frac{\partial\omega}{\partial \mathrm{t}} \tag{3.23}$$

$$\partial_t(\Phi\rho\omega) = \;\Phi\left(\varrho + \,\omega\frac{\partial\rho}{\partial\omega}\right)\frac{\partial\omega}{\partial \mathrm{t}} \qquad (3.24$$

or equivalently:

$$\frac{\partial \mathcal{Q}(u)}{\partial t} = \mathcal{B}(u)\frac{\partial u}{\partial t} \overset{\text{def}}{=} \begin{pmatrix} \Phi\left(\varrho(\omega) + \,\omega\frac{\partial\rho}{\partial\omega}\right) & 0 \\ \Phi\,\frac{\partial\rho}{\partial\omega} & 0 \end{pmatrix}\begin{pmatrix} \frac{\partial\omega}{\partial \mathrm{t}} \\ \frac{\partial \mathrm{p}}{\partial \mathrm{t}} \end{pmatrix} \tag{3.25}$$

This stresses that (3.3), (3.4) is in essence a differential-algebraic equation (DAE): The continuity of the salt mass is described as a transient process (w.r.t. w), while the continuity of the fluid mass is a static process w.r.t. p.

The linear-implicit Euler is now defined as follows: For an arbitrary linear operator $\mathcal{J}$, one may subtract the product $\mathcal{J}\mathrm{u}$ on both sides:

$$\mathcal{B}(\mathrm{u})\frac{\partial u}{\partial t} - \mathcal{J}\mathrm{u} = \mathcal{F}(\mathrm{u}) - \mathcal{J}\mathrm{u} \tag{3.26}$$

The purpose of this is to remove stiff components from the right hand side. Evaluating the modified right hand side at u_t yields the linear-implicit Euler:

$$(\mathcal{B}(\mathrm{u}_t) - \tau\mathcal{J})\mathrm{u}_{t+\tau} = \tau\,\mathcal{F}(u_t) + (\mathcal{B}(\mathrm{u}_t) - \tau\,\mathcal{J})\mathrm{u}_t \tag{3.27}$$

As pointed out in /DEU 90/, one suitable choice for J

$$\mathcal{J} = \frac{\partial}{\partial \mathrm{u}}\left(\mathcal{F}(\mathrm{u}) - \mathcal{B}(u)\frac{\partial u}{\partial t}\right)_{|u=u_0} \tag{3.28}$$

or a computationally feasible approximation thereof.

For practical purposes, one can use

$$(\mathcal{B}(\mathrm{u}_t) - \tau\mathcal{J}) = \frac{\partial}{\partial \mathrm{u}}(\tau\,\mathcal{F}(\mathrm{u}) - \mathcal{Q}(\mathrm{u}) + \mathcal{Q}(\mathrm{u}_t))_{|u=u_t} \tag{3.29}$$

i.e., the linearization in the previous point.

3.4.4 Numerical Experiments

Section 3.4.2 introduced three different solvers. These were compared and evaluated with respect to performance. As a benchmark the Elder problem is used. This features a highly dynamic velocity field in the beginning, which then stabilizes for larger times. Since effects of the linear solver should be avoided, a coarse spatial mesh with 4420 degrees of freedom (1024 elements) is used. The tests are conducted for the full non-linear equations (3.3)-(3.5) using *ug4* /VOG 13/.

The first test investigates the convergence of the two decoupling nonlinear solvers. In the first time step, the partial Newton and the iterative coupling achieve a reduction of the residual by 0.5×10^{-6} in 26 and 21 steps respectively.

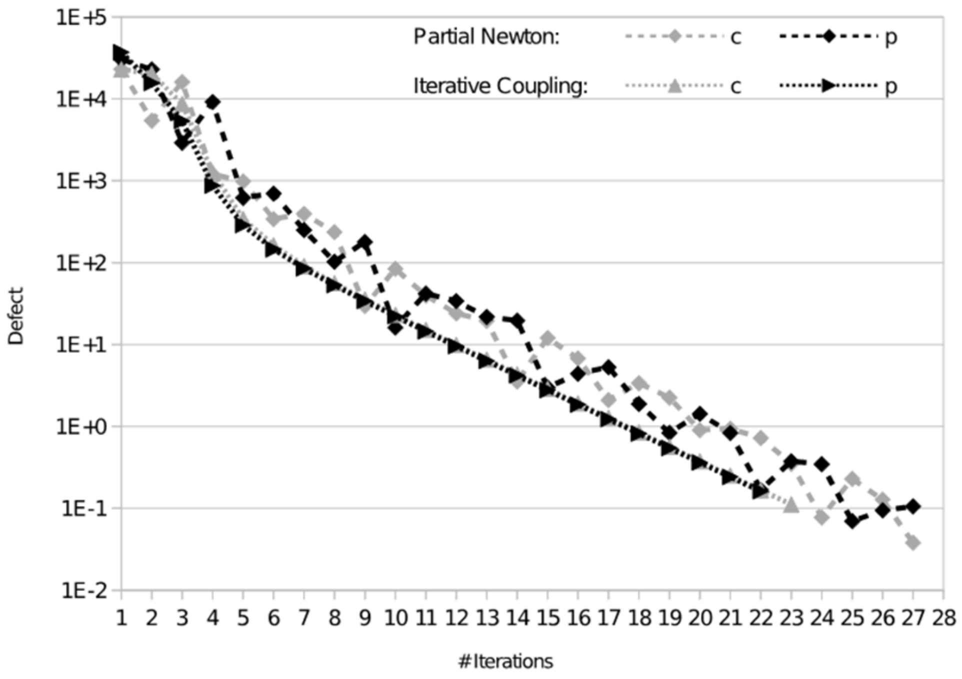


Fig. 3.4 Defect reduction of Partial Newton (diamond) and Iterative Coupling (triangle) for computing t=τ=0.025a in the first time step

Fig. 3.4 visualizes details about the reduction of the nonlinear defects

$$d_{p,k}^{nl} := \| \mathbf{F}_{p,h,k} \|_2, \qquad d_{\omega,k}^{nl} = \| \mathbf{F}_{p,h,k} \|_2 \tag{3.30}$$

of the k-th iterate. The vectors $\mathbf{F}_{\alpha,h,k}$ are defined by (3.6)-(3.7) for both components $\alpha \in \{p, \omega\}$.

Both methods converge linearly with similar rates of convergence. However, it is observed that $d_{p,k}^{nl}$ and $d_{\omega,k}^{nl}$ behave differently: While they resemble each other in the order of magnitude for the iterative coupling, they differ substantially for the partial Newton. In the latter case, an oscillating behavior can be observed.

Fig. 3.5 provides a history of the nonlinear iteration steps required for each single time step over a complete simulation run of 5 years. As expected, the full Newton method performs best. Large time steps are permitted ($\tau = 0.1a$), at the same time, only a constant number of 4 iterations per time step, i.e., a total of 80 iterations is required.

Both decoupling iterations require a smaller time step $\tau = 0.025a$. In the comparison for a full simulation run, however, they behave differently: The partial Newton requires 25-

30 iterations per time step, resulting in a total of 6,384 iterations. Although the iterative coupling also starts with $\sim$ 25 iterations in the first steps, the number of required iterations gradually decreases to 8-10 iterations per time step. This leads to a total of 1,925 iterations

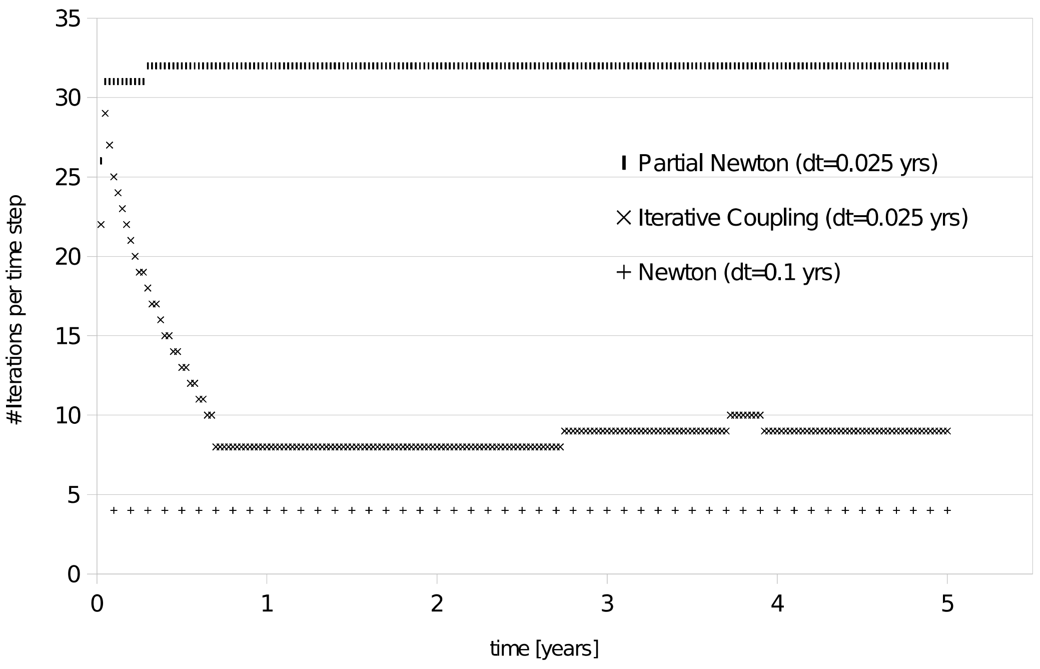


Fig. 3.5 Iterations per time step for a full simulation run: Newton (time step $\tau = 0.1a$), Partial Newton ($\tau = 0.025a$), and Iterative Coupling ($\tau = 0.025a$)

The previous analysis was based on fixed step sizes. As a next step, the three algorithms are tested with an adaptive time stepping strategy. The problem was the fully non-linear Elder problem with three levels of refinement. Coincidence of the solutions was tested with the Euclidean norm for the salt mass fraction. The tolerance for guiding the time step was set to 0.01. If the solution process failed, the time step was bisected. The error was controlled by an extrapolation technique /DEU 90/, /DEU 02/. Results are shown in Fig. 3.6. The result is the same as for the previous test: The Full Newton outperforms the two other approaches from Section 3.4.2.

Finally, this can also be employed for the linear-implicit schemes presented in Section 3.4.3. As shown in Fig. 3.7, the method allows for time steps that are similar to the Full Newton. However, since only a single linear system is solved per step, the total number of iterations is reduced significantly by a factor of 4.

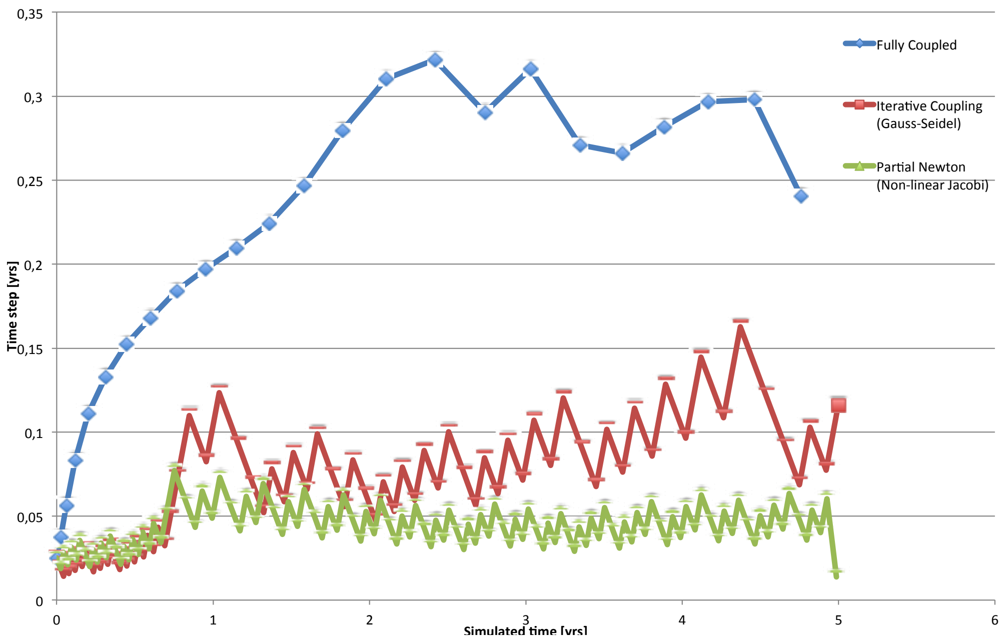


Fig. 3.6 Time steps within an adaptive time-stepping strategy for the algorithms from Section 3.4.2: Partial Newton, Iterative Coupling, and Full Newton

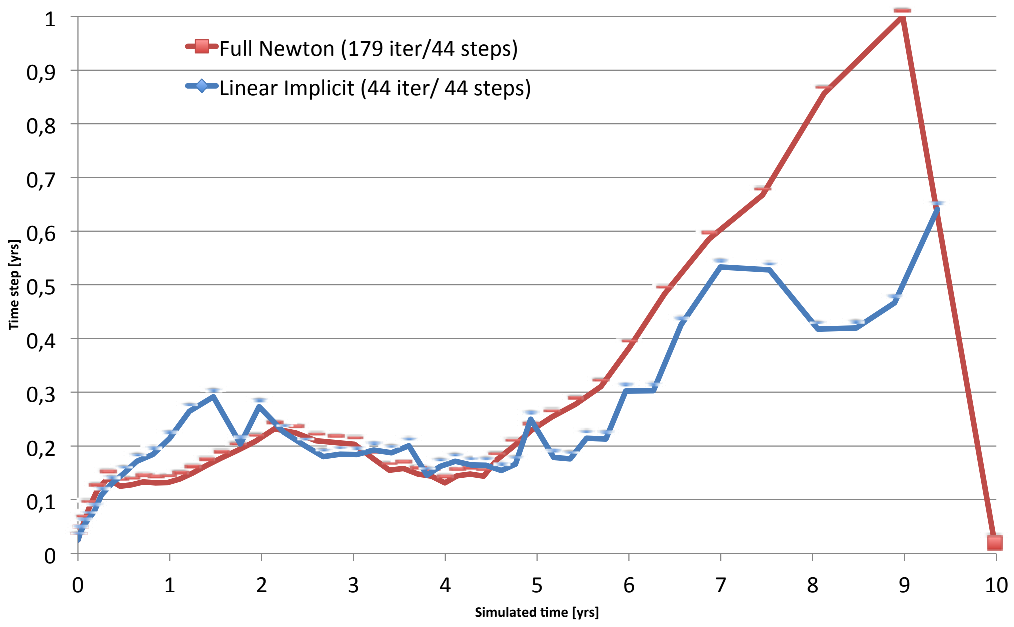


Fig. 3.7 Time steps within an adaptive time-stepping strategy: Full Newton (Section 3.4.2) vs. Linear Implicit scheme (Section 3.4.3)

3.4.5 Conclusion

This chapter compared different types of non-linear solvers. The first class of solvers are fixed point iterations introduced in Section 3.4.2. According to this data, the iterative coupling should be preferred over a partial Newton. In particular, if the alterations in the velocity field are small, i.e., the corresponding initial guess for the nonlinear scheme is sufficiently good, the Gauss-Seidel-style arrangement of the iterative coupling seems to be more appropriate. However, as expected, the Full Newton method outperforms both previously mentioned methods. The reason is that decoupling iterations produce iterates that oscillate around the fixed point: For the iterative coupling, for example, an erroneous velocity in the transport equation (3.19) tends to overshoot the concentration, which is then corrected in the next step (3.20) by the flow equation. Similar results have previously been reported, e.g., for fluid structure interactions /HEI 04/, /MAT 06/.

A potent alternative to Newton-type fixed point iterations are linear-implicit schemes introduced in Section 3.4.3. These schemes allow to linearize the problem only once, and thus, when compared to the Full Newton, the number of linear iterations is reduced considerably. Moreover, since the error is controlled for the solution (and not for the residual, as it is typically done for Newton approaches), this can elegantly be combined with adaptivity. Although this study focused on adaptivity of the time step, the same also holds true for the error of the spatial discretisation.

3.5 Higher Dimensional Problems

3.5.1 Introduction: Transport equations for probability density

In the traditional formulation of transport problems in porous media, the concentration is regarded as completely determined by the convective velocity, the molecular diffusion as well as by initial and boundary conditions. However, the stochastic structure of these media motivates a probabilistic consideration of the distribution of the concentration (see chapter 4): At every point $\boldsymbol{x} \in \boldsymbol{\Omega} \subset \mathbb{R}^d$, concentrations $c_1, \dots, c_N$ of the dissolved substances are described by a probability density function $f: \mathbb{R}_+ \times \boldsymbol{\Omega} \times \mathbb{R}_+^N \to \mathbb{R}_+$, so that $f(t, \boldsymbol{x}, c_1, \dots, c_N)$ is the probability that at time t and point $\boldsymbol{x}$, the concentrations attain values $c_1, \dots, c_N$. For simplicity, only one concentration c (i.e. $N = 1$) is considered. Depending on the mixing model, the following law can be stated for the evolution of f:

$$\partial_t f + \nabla_{\boldsymbol{x}} \cdot (\boldsymbol{u} f) - \nabla_{\boldsymbol{x}} \cdot (D \nabla_{\boldsymbol{x}} f) = \partial_c [\Omega_m \cdot (c - \langle c \rangle) \cdot f], \tag{3.31}$$

where $\boldsymbol{u}$ is the convective velocity, D is the diffusion coefficient (without the dispersion tensor), $\langle c \rangle$ is the mean concentration at given time t and point $\boldsymbol{x}$,

$$\langle c \rangle (t, \boldsymbol{x}) = \int_0^{+\infty} f(t, \boldsymbol{x}, c) \cdot c \, \mathrm{dc}, \tag{3.32}$$

and the constant parameter Ω_m of the mixing model is given by

$$\Omega_m = \frac{C_\Omega D_{\text{ens}}}{\lambda^2}, \tag{3.33}$$

D_{ens} being the ensemble dispersion coefficient, λ the correlation length of the permeability field and C_Ω a scaling factor.

Equation (3.31) is a convection-diffusion PDE. Nevertheless it has some special features. First of all, it is formulated in a high-dimensional domain $\boldsymbol{\Omega} \times \mathbb{R}_+$. Taking into account, that $\boldsymbol{\Omega}$ is 3-dimensional, the dimensionality of the whole problem is at least 4. (It can be greater if several transported concentrations are considered). This requires a special discretization approach since the usual triangulations of the whole domain become inefficient due to a large number of degrees of freedom. Therefore, a sparse grid method described in Section 3.5.2 is used.

Furthermore, one should take into account the non-local coupling due to the presence of the mean concentration $\langle c \rangle$ in (3.31), cf. (3.32). This term is evaluated using the data from the old time step, cf. Section 3.5.2.

Problem (3.31) must be closed by a specification of boundary and initial conditions. As (3.31) has convection as well as diffusion terms in the geometric space, boundary conditions should be imposed on f for all $\boldsymbol{x} \in \partial\boldsymbol{\Omega}$. A particular example for this is the Dirichlet boundary condition

$$f(t, \boldsymbol{x}, c) = f_D(t, \boldsymbol{x}, c), \qquad \boldsymbol{x} \in \partial\boldsymbol{\Omega}. \tag{3.34}$$

In contrast to $\boldsymbol{x}$, (3.31) is completely hyperbolic in c. The direction of the corresponding component of the velocity depends on the sign of the factor $(c - \langle c \rangle)$. Due to this factor, the characteristics run out of the domain, so that no boundary conditions at $c = 0$ and $c = +\infty$ are needed. Therefore, $\partial\boldsymbol{\Omega} \times \mathbb{R}_+$ is the only part of the high-dimensional domain where the boundary conditions must be set.

The initial conditions must be set on the whole $\boldsymbol{\Omega} \times \mathbb{R}_+$. Note that typically, only the dependence of the concentration on $\boldsymbol{x}$ is known for the initial and the Dirichlet boundary conditions. To extend this dependence into the c-direction, it is multiplied by some standard distribution, for example, by the Gaussian pulse.

In the numerical methods, the interval $\mathbb{R}_+$ for c in the definition of $\boldsymbol{\Omega} \times \mathbb{R}_+$ and in (3.32) is replaced by some finite interval $[0, c_{\max}]$. This restriction influences the choice of possible initial and boundary distributions for c, or introduces errors in $\langle c \rangle$, but is in accordance with the presence of physical bounds for the concentration.

3.5.2 Sparse grids for discretization of high-dimensional PDEs

A discretization replaces the domain $\boldsymbol{\Omega} \times \mathbb{R}_+^N$ (the space of the geometrical and probability dimensions) by a finite grid. At elements of the grid (typically at vertices), degrees of freedom describing possible approximations of f are placed. The PDEs (3.31) are replaced by a system of algebraic equations for these degrees of freedom, and this system is solved. The obtained numerical solution has an error tending to zero, as the grid is refined. Therefore, possibly finer grids should be used. However the grids, grid functions and grid operators have to be represented in the computer memory and this restricts their maximum size. Furthermore the numerical complexity of the algebraic solvers grows, as the grid is refined, in particular due to reduction of the efficiency of the algorithms. This leads to essential problems in the application of the numerical methods to the high-dimensional problems because the number of degrees of freedom is increasing rapidly with grid refinement. For example, as the total dimensionality of $\widetilde{\mathbf{\Omega}} \coloneqq \mathbf{\Omega} \times \mathbb{R}_+^N$ is $\tilde{d} = d + N$, the number of grid points for a quasi-regular triangulation is proportional to $h^{-\tilde{d}}$, where h is the grid length. This results in extremely large discretized systems for an appropriate accuracy of the numerical solution.

A principal reduction of the necessary number of degrees of freedom can be achieved by application of the so-called sparse grids, cf. /ZEN 91/, /GSZ 92/, /REI 04/, /REI 07/. This approach exists in two forms:

- The sparse grid can be represented by a set of vertices located at special positions in the domain. A suitable finite-difference discretization is used for the PDE.

- A set of specially refined Cartesian grids is considered, and the PDEs are discretized separately on every of them. (The sparse grid in the sense of the previous item is the union of vertices of all these grids.) The numerical solutions obtained on these grids are then summarized to an approximation of the analytical solution. This method is said to be the combination technique for the sparse grid.

Whereas the first approach needs quite specific and relatively inefficient data structures, the combination technique is very flexible and can be used for a wide class of problems. This approach has been used for the discretization of (3.31).

To introduce the method, the following sets of grids are defined recursively:

1. $\boldsymbol{G}_0$ consists of only one initially specified grid covering $\widetilde{\mathbf{\Omega}}$. This is the coarsest grid that can contain for example only boundary grid nodes.
2. $\boldsymbol{G}_{l+1}$ is obtained from $\boldsymbol{G}_l$ by refining every $\widetilde{\mathbf{\Omega}}_h \in \boldsymbol{G}_l$ separately in every dimension.

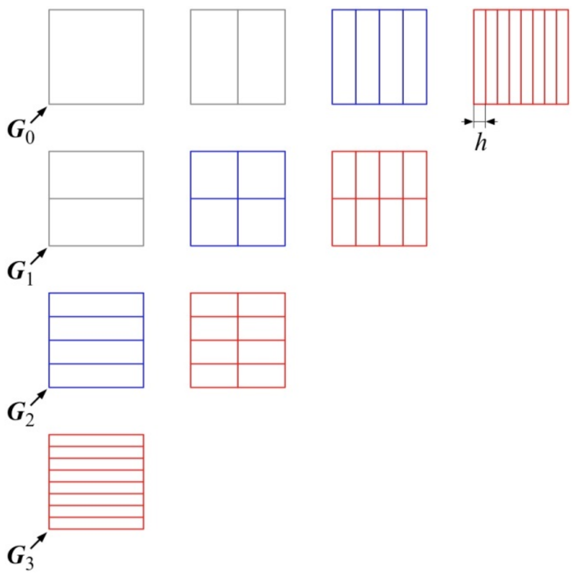


Fig. 3.8 Grids in the combination technique approach for the sparse grids. Only the blue and the red sets ($\boldsymbol{G_2}$ and $\boldsymbol{G_3}$) are used in the computation on grid level 2, cf. (3.35)

This construction is illustrated in Fig. 3.8 for a 2-dimensional square domain and $l \leq 3$. For a computation on *grid level* l, only grids from $\boldsymbol{G}_l, \dots, \boldsymbol{G}_{l+d-1}$ are used. The total number of grid nodes on all these grids is $O\left(h^{-1}|\log_2 h^{-1}|^{\tilde{d}-1}\right)$, i.e. essentially less then that for the regularly refined grid.

On every grid $\widetilde{\boldsymbol{\Omega}}_h \in \boldsymbol{G}_l \cup \dots \cup \boldsymbol{G}_{l+d-1}$, Equation (3.31) and the boundary conditions are discretized by a finite-difference scheme. In particular, the full-upwind scheme is used for the convective terms. The discretization is implicit in time. This results in a large sparse linear system, which is solved by the linear iteration preconditioned with the V-cycle of the geometric multigrid method with the Gauss-Seidel smoothers. Although most of the grids in $\boldsymbol{G}_l \cup \dots \cup \boldsymbol{G}_{l+d-1}$ are strongly anisotropic, this solver converged only in several iterations.

Denote the solution computed on the grid $\widetilde{\boldsymbol{\Omega}}_h$ in time step k by $f^k_{\widetilde{\boldsymbol{\Omega}}_h}$. In terms of the multilinear basis functions, this function is defined on the whole domain $\widetilde{\boldsymbol{\Omega}}$. Then the sparse-grid approximation of the analytical solution is

$$f_h^k = \sum_{i=0}^{d-1} \sum_{\widetilde{\mathbf{\Omega}}_h \in \boldsymbol{G}_{l+i}} a_{\widetilde{\mathbf{\Omega}}_h}^{l+i} f_{\widetilde{\mathbf{\Omega}}_h}^k, \tag{3.35}$$

where $a_{\widetilde{\mathbf{\Omega}}_h}^l$ are scalar coefficients depending on the grid sizes of $\widetilde{\mathbf{\Omega}}_h$, cf. /REI 04/. Function f_h^k converges to the analytical solution as $h \to 0$. For (3.31) the discretization error is $O(h)$.

The time discretization is based on the implicit Euler method. Note that the time steps are computed for every $\widetilde{\mathbf{\Omega}}_h \in \boldsymbol{G}_l \cup \ldots \cup \boldsymbol{G}_{l+d-1}$ separately (with the same time step length), so that $f_{\widetilde{\mathbf{\Omega}}_h}^k$ depends only on $f_{\widetilde{\mathbf{\Omega}}_h}^{k-1}$, but not on the solutions on the other grids. This concerns in particular the term with $\langle c \rangle$ in (3.31): For the spatial discretization in time step k on grid $\widetilde{\mathbf{\Omega}}_h$, this term is computed from $f_{\widetilde{\mathbf{\Omega}}_h}^{k-1}$:

$$\langle c \rangle_{\widetilde{\mathbf{\Omega}}_h}^k(\boldsymbol{x}) = \int_0^{+\infty} f_{\widetilde{\mathbf{\Omega}}_h}^{k-1}(\boldsymbol{x}, c) \cdot c \, dc, \tag{3.36}$$

(cf. (3.32)) and this $\langle c \rangle_{\widetilde{\mathbf{\Omega}}_h}^k$ is used for the computation of $f_{\widetilde{\mathbf{\Omega}}_h}^k$. The formula (3.35) is therefore used only for the output of the numerical solution.

Parallelization of the computations is based on the distribution of the grids from $\boldsymbol{G}_l \cup \ldots \cup \boldsymbol{G}_{l+d-1}$ between processors. Communication is only necessary for the computation of f_h^k by (3.35) for example after every time step.

3.5.3 Numerical tests

As an example, results of simple numerical tests are presented. A rectangular geometric domain $\mathbf{\Omega} = [0,4] \times [0,2] \times [0,2]$ $[m^3]$ and segment $[0,1]$ $[-]$ for the normalized concentration are taken, so that (3.31) is considered on the 4-dimensional domain $\widetilde{\mathbf{\Omega}} = [0,4] \times [0,2] \times [0,2] \times [0,1]$.

Furthermore, the Dirichlet-0 boundary conditions are imposed on $\partial\mathbf{\Omega}$. Specification of the initial conditions is based on the distribution of concentration

$$c_{\text{ave},0}(\boldsymbol{x}) = \begin{cases} c_0 \cdot \left(\frac{R^2 - \|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2}{R^2}\right)^3, & \|\boldsymbol{x} - \boldsymbol{x}_0\| \geq R, \\ 0, & \|\boldsymbol{x} - \boldsymbol{x}_0\| \geq R \end{cases} \tag{3.37}$$

around the *injection point* $\boldsymbol{x}_0$ in $\boldsymbol{\Omega}$: At every geometric point, the initial concentration is defined as

$$f(0, \boldsymbol{x}, c) = \frac{c_{\text{ave},0}(\boldsymbol{x})}{\sigma\sqrt{2\pi}} \cdot \exp\left(\frac{\left(c - c_{\text{ave},0}(\boldsymbol{x})\right)^2}{\sigma\hat{}2}\right). \tag{3.38}$$

Values of the coefficients for (3.31), (3.33), (3.37) and (3.38) are listed in Tab. 3.6.

Tab. 3.6 Coefficients for the numerical test

Parameter	**Value**	**Unit**
$\boldsymbol{u}$	$(1.1574 \cdot 10^{-5}, 0, 0)$	$[m \cdot s^{-1}]$
D	$1.1574 \cdot 10^{-7}$	$[m^2 \cdot s^{-1}]$
C_Ω	3	$[-]$
D_{ens}	$1.1574 \cdot 10^{-7}$	$[m^2 \cdot s^{-1}]$
λ	1	$[m]$
$\boldsymbol{x}_0$	$(1, 1, 1)$	$[m]$
R	0.5	$[m]$
c_0	0.5	$[-]$
σ	0.1	$[-]$

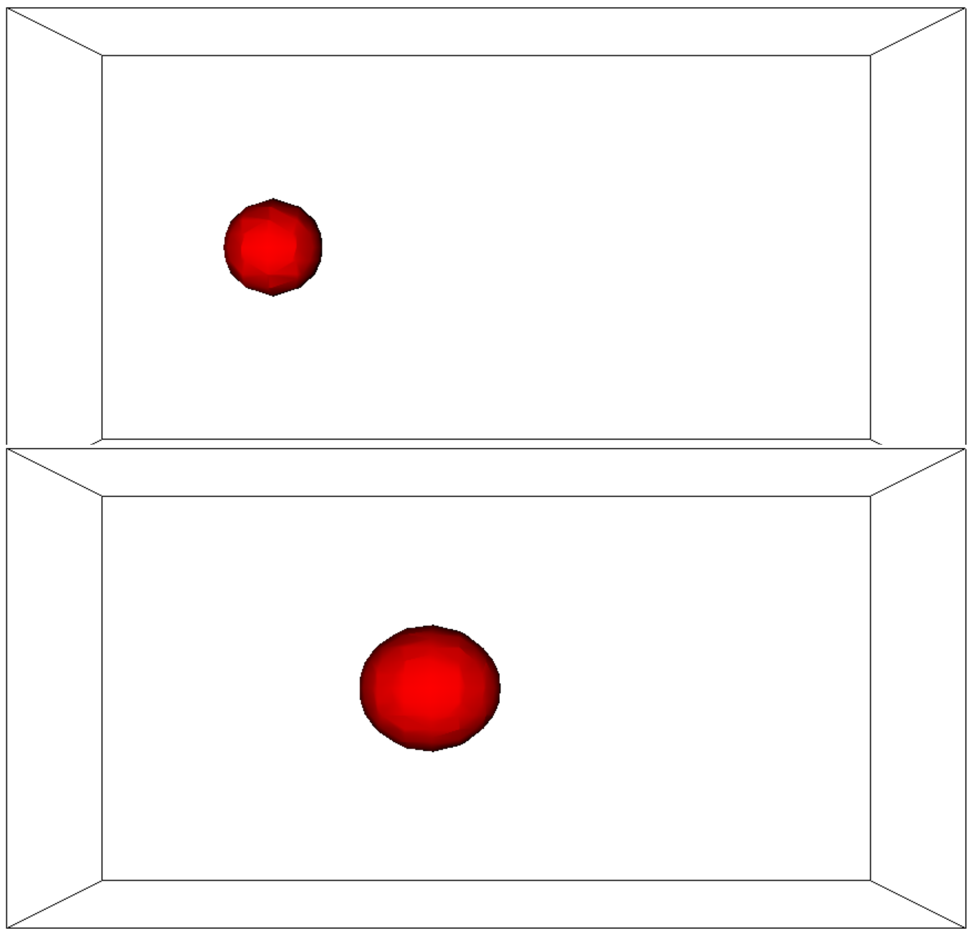


Fig. 3.9 Isosurface of the averaged concentration (0.25 of the averaged maximum concentration) in the test computation: Initial condition (above) and solution at time step 64 (below)

The computation has been performed on the sparse grid level 5 with the time step 1000 s. The set $\boldsymbol{G}_0$ consisted of the Cartesian $5 \times 5 \times 5 \times 5$-grid. Fig. 3.9 presents the evolution of the averaged concentration $\langle c \rangle$ in this computation.

In the second test, a stochastic velocity field is considered, and the numerical solution of the problem is compared with the analytical one. For the comparison, the averaged concentration $\langle c \rangle$ (for the numerical solution: computed by combination technique on the regular $33 \times 33 \times 33$-grid) was integrated over the yz-planes for every x. These distributions in a neighbourhood of the injection point x_0 are presented in Fig. 3.8. The computations have been performed for same parameters of the transport problem (3.31), as above (cf. Tab. 3.6), except for the velocity $\boldsymbol{u}$. The mean value of the stochastic velocity field was $1.1574 \cdot 10^{-5}\, m \cdot s^{-1}$ (i.e. 1 m a day), the correlation length $1\, m$, and the variance 0.005.

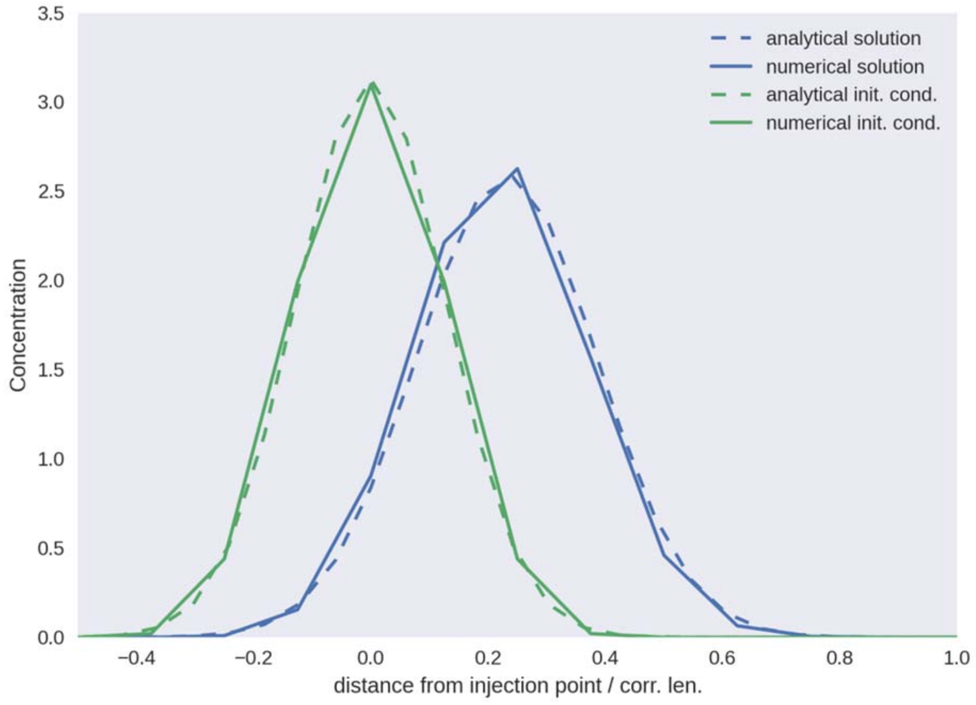


Fig. 3.10 Comparison of numerical and analytical solution in the experiment with the stochastic velocity field. Integrals of the mean concentration over the $\boldsymbol{yz}$-planes in $\boldsymbol{\Omega}$ are plotted along the $\boldsymbol{y}$-axis of the graph. The distance from the injection point (the $\boldsymbol{x}$-axis of the graph) is measured in the units of the correlation length of the stochastic velocity

4 Modelling uncertainty in salt transport

4.1 Motivation

Transport of dissolved substances is determined by groundwater flow, which in turn is strongly influenced by the properties of the formation. Formation properties like the hydraulic conductivity are generally highly heterogeneous on many different scales. These heterogeneities range from the order of magnitude of individual grains to large geologic structures like facies, fractures and sediment layers.

Conversely, the heterogeneities of the formation make the transport of the contaminants also highly heterogeneous. Contaminated water volume elements which travel very closely together can be separated and follow distinct and separated flow paths. An enhanced spreading of the plume is the result of it. Thus, in order to predict the transport of contaminants deterministically, it is necessary to know all aquifer properties influencing the transport everywhere. Monitoring the complete formation down to the smallest scales on which heterogeneities appear is neither feasible nor possible. If only partial knowledge of the formation can be retrieved, the formation properties and flow and transport parameters remain partially unknown. This uncertainty of formation properties and model parameters can be taken into account by using a stochastic representation of the formation. By applying a stochastic framework, hydraulic properties become stochastic and in turn the contaminant concentration as well.

The mean transport behavior can be calculated by taking the ensemble average of the heterogeneous transport equation. The resulting equation is a transport equation for the ensemble averaged concentration with the following characteristics: The highly heterogeneous and spatially fluctuating groundwater velocity is replaced by an ensemble averaged velocity field and the effect of the fluctuating velocity on the transport is modelled by an enhanced dispersion called macro dispersion or ensemble dispersion. The limitation of this approach is that the ensemble averaged concentration first of all describes the mean plume behavior, sometimes also the most probable behavior, but not necessarily the behavior of a specific plume in a single formation (see Fig. 4.1). Only if the plume has sampled a representative part of the formation it becomes ergodic, and the individual transport behavior can be modelled by the ensemble average behavior.

Deviations from the mean behavior can be quantified by calculating the concentration variance. It is not only transported by advection and dispersion, like the mean concentration, but is also generated by mean concentration gradients and destroyed by dissipation processes. The latter need a closure model in order to limit the computational effort. There exist different suggestions to close the equation of the concentration variance. Unfortunately, these closure models were developed in turbulence theory and have not been fully adapted yet to flows in porous media. Here it is supposed that adapting the closure model is necessary since the mixing behavior in turbulent flows and in porous media flow is very different. The most important difference is the speed at which heterogeneity induced mixing takes place. In turbulent flows it is very fast due to chaotic flow behavior and the large scale dispersion coefficients can be approximated by their asymptotic limits and thus by a constant. In porous media flows, flow is not chaotic and heterogeneity impacted mixing takes much longer times to develop. /DEN 00/. This effective mixing or dispersion is small at early times and increase only slowly with time. Therefore, early time concentration gradients are steep and remain steep for a prolonged time, which in turn prevents smoothing variable concentrations and preserves concentration uncertainty.

In addition, the closure problem becomes even more difficult for reactive transport, since the concentration gradients and the reactions both dominantly influence the behavior of the reactive contaminants. Often, the reaction terms are non-linear and are especially difficult to model at steep concentration gradients.

If the predictions made by a contaminant transport model are to be used for risk analysis, even more information than the mean concentration and the variance is needed. In risk analysis, the quantity of interest is the exceedance probability, which can only be calculated if the complete one-point probability density function of the concentration is known. The exceedance probability is defined as

$$\mathrm{Prob}(\mathrm{R} > \mathrm{r_{crit}}) = 1 - \mathrm{F}(\mathrm{r_{crit}}). \tag{4.1}$$

with F being the cumulative distribution function and $\mathrm{F}(\mathrm{r_{crit}})$ being a threshold, regulated for example by an environmental agency /AND 96/. The concentration variance can only be used as an upper limit to the exceedance probability. Therefore, a second approach – the PDF approach- is very promising. It yields an equation for the whole pdf of the concentration.

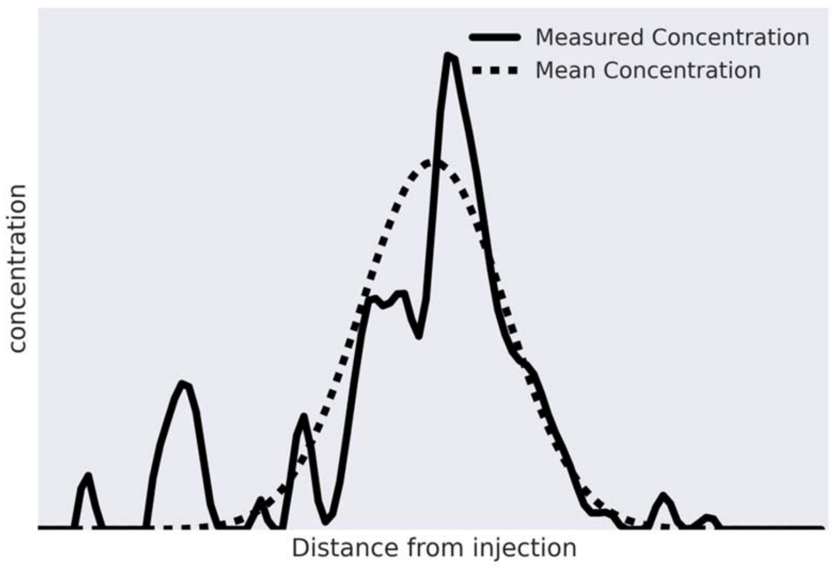


Fig. 4.1 A measure is needed to quantify how good the mean concentration approximates the measured concentration

In this project, the transport equations for the mean concentration and the concentration variance have been derived and parametrized. The parameters of the equation for the mean concentration are given by the mean velocity and the ensemble dispersion coefficients which have been explicitly evaluated already in previous projects, see e. g. /SCH 12/ and /SCH 13/. The equation for the concentration variance needs a new closure for the mixing model adapted to porous media flow. Making use of these two equations and the new closure model, an approximation for transport equation for the whole concentration PDF could be derived, parametrized and finally verified by numerical simulations. Our new closure model enables to establish transport equations for the whole pdf of non-reactive and reactive transport. The new pdf transport equations are defined in a higher dimensional space but with the new numerical methods for higher dimensional problems developed as presented in Section 3.5 it is now possible to numerically solve the pdf equation very efficiently.

4.2 Methods

The transport of a solute in groundwater can be described by

$$\frac{\partial \mathrm{C}}{\partial \mathrm{t}} + \mathbf{V} \cdot \nabla \mathrm{C} = \mathrm{D}\Delta\mathrm{C}, \tag{4.2}$$

where V is a random velocity field and D the dispersion coefficient. Important statistical quantities of a probability distribution PDF are its moments, like e.g. mean and variance. The mean concentration and its variance are defined by

$$\langle C \rangle(\mathbf{x},\mathrm{t}) := \int \mathrm{cP}(\mathrm{c};\ \mathbf{x},\mathrm{t})\,\mathrm{dc} \tag{4.3}$$

$$\sigma_{\mathrm{c}}^2(\mathbf{x},\mathrm{t}) := \int \mathrm{c}^2\mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})\,\mathrm{dc} - \langle C \rangle^2(\mathbf{x},\mathrm{t}). \tag{4.4}$$

Thus, if the parameters of the transport equation are known the transport of the PDF can be derived.

4.2.1 PDF transport equations

The derivation is shown in detail for example in /SUC 15/. Here only the most important steps are sketched. The PDF is defined as the ensemble average of the so-called fine-grained PDF, which is a delta function

$$\mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}) := \langle \delta(\mathrm{C}(\mathbf{x},\mathrm{t}) - \mathrm{c}) \rangle. \tag{4.5}$$

The PDF evolution equation can be derived by taking the time derivative of (4.5) which can be evaluated using the following relation

$$\nabla\delta(\mathrm{C}(\mathbf{x},\mathrm{t}) - \mathrm{c}) = -\nabla\mathrm{C}(\mathbf{x},\mathrm{t})\frac{\partial}{\partial \mathrm{c}}\delta(\mathrm{C}(\mathbf{x},\mathrm{t}) - \mathrm{c}). \tag{4.6}$$

Applying this relation to the PDF (4.5) and inserting the definition of the conditional average $\langle \mathrm{Q}(\mathbf{x},\mathrm{t})|\mathrm{c}\rangle = \langle \mathrm{Q}(\mathbf{x},\mathrm{t})\delta(\mathrm{C}(\mathbf{x},\mathrm{t}) - \mathrm{c})\rangle / \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})$ results in

$$\frac{\partial}{\partial \mathrm{t}}\mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}) = -\frac{\partial}{\partial \mathrm{c}}\left[\left\langle \frac{\partial \mathrm{C}(\mathbf{x},\mathrm{t})}{\partial \mathrm{t}} \middle| \mathrm{c} \right\rangle \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})\right]. \tag{4.7}$$

Evolution equation (4.2) can now be used to obtain the PDF transition

$$\frac{\partial}{\partial \mathrm{t}}\mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}) = \frac{\partial}{\partial \mathrm{c}}[\{\langle \mathbf{V} \cdot \nabla\mathrm{C}(\mathbf{x},\mathrm{t})|\mathrm{c}\rangle + \langle \mathrm{D}\Delta\mathrm{C}(\mathbf{x},\mathrm{t})|\mathrm{c}\rangle\}\mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})]. \tag{4.8}$$

The two terms on the right hand side are unclosed and a closure model is needed in order to be able to calculate the time evolution of the PDF. The advective term can be closed by making use of the knowledge that the impact of velocity fluctuations on transport can be modelled by an enhanced dispersion

$$\frac{\partial}{\partial \mathrm{c}}[\langle \mathbf{V} \cdot \nabla \mathrm{C}(\mathbf{x},\mathrm{t}) | \mathrm{c} \rangle \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})] = -\nabla[\langle \mathbf{V} \rangle \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})] + \nabla \cdot \mathrm{D}^{\mathrm{ens}} \nabla \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}), \tag{4.9}$$

where $\mathrm{D}^{\mathrm{ens}}$ is an upscaled dispersion coefficient. The unclosed difference can be transformed to

$$\frac{\partial}{\partial \mathrm{c}}[\langle \mathrm{D}\Delta \mathrm{C}(\mathbf{x},\mathrm{t}) | \mathrm{c} \rangle \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})] = \nabla \cdot \mathrm{D} \nabla \mathrm{P} - \frac{\partial^2}{\partial \mathrm{c}^2}[\langle \mathrm{D}(\nabla \mathrm{C})^2 | \mathrm{c} \rangle], \tag{4.10}$$

For the last term on the right hand side, a mixing model $M = \langle \mathrm{D}(\nabla \mathrm{C})^2 | \mathrm{c} \rangle$ is needed. /DOP 75/ formulated a closure model called ***I**nteraction **E**xchange with the **M**ean* (IEM) which is widely used for modelling reactive and turbulent flows (see /AND 98/; /POP 14/). It closes the mixing term by approximating it with

$$\frac{\partial^2}{\partial \mathrm{c}^2}\langle \mathrm{D} \nabla \mathbf{C} \cdot \nabla \mathbf{C} | \mathrm{c} \rangle = \frac{\partial}{\partial \mathrm{c}}\big(\mathrm{M}[\mathrm{c} - \langle \mathrm{C}(\mathbf{x},\mathrm{t}) \rangle] \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})\big), \tag{4.11}$$

where M is a dissipation rate, which will be discussed in Section 4.3. The dissipation model causes concentration fluctuations to relax exponentially towards the mean concentration.

With the IEM model inserted into equation (4.9), a closed transport equation for the PDF can be stated

$$\begin{aligned} \frac{\partial \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})}{\partial \mathrm{t}} &+ \langle \mathbf{V} \rangle \cdot \nabla \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}) - \nabla \cdot \mathrm{D}^{\mathrm{ens}} \nabla \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t}) \\ &= \frac{\partial}{\partial \mathrm{c}}(\mathrm{M}[\mathrm{c} - \langle \mathrm{C}(\mathbf{x},\mathrm{t}) \rangle] \mathrm{P}(\mathrm{c};\ \mathbf{x},\mathrm{t})). \end{aligned} \tag{4.12}$$

4.2.2 Mean and Variance Transport Equations

The transport equation for the first statistical moment, the mean concentration, can be derived from the PDF transport equation (4.12) by multiplying it with c and integrating over the complete concentration space,

$$\int \mathrm{c}\frac{\partial \mathrm{P}}{\partial \mathrm{t}} \mathrm{dc} + \int \mathrm{c}\langle \mathbf{V} \rangle \cdot \nabla \mathrm{P}\, \mathrm{dc} - \int \mathrm{c}\mathrm{D}^{\mathrm{ens}} \Delta \mathrm{P}\, \mathrm{dc} = \int \mathrm{c}\frac{\partial}{\partial \mathrm{c}}(\mathrm{M}[\mathrm{c} - \langle \mathrm{C} \rangle]\mathrm{P})\, \mathrm{dc}. \tag{4.13}$$

The order of integration and derivation can be interchanged and on the right hand side the product rule can be applied.

$$\frac{\partial}{\partial \mathrm{t}}\int \mathrm{cP}\,\mathrm{dc} + \langle \mathbf{V}\rangle \cdot \nabla \int \mathrm{cP}\,\mathrm{dc} - \mathrm{D}^{\mathrm{ens}}\Delta \int \mathrm{c\,P}\,\mathrm{dc} = \int \left\{\frac{\partial}{\partial \mathrm{c}}(\mathrm{cM}[\mathrm{c}-\langle \mathrm{C}\rangle]\mathrm{P}) - \mathrm{M}[\mathrm{c}-\langle \mathrm{C}\rangle]\mathrm{P}\frac{\partial \mathrm{c}}{\partial \mathrm{c}}\right\}\mathrm{dc}. \tag{4.14}$$

On the left hand side, the definition of the mean concentration can be inserted and on the right hand side, the integral is evaluated:

$$\frac{\partial \langle \mathrm{C}\rangle}{\partial \mathrm{t}} + \langle \mathbf{V}\rangle \cdot \langle \mathrm{C}\rangle - \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle = \mathrm{cM}[\mathrm{c}-\langle \mathrm{C}\rangle]\mathrm{P}|_{\mathrm{c}=0}^{1} - \mathrm{M}[\langle \mathrm{C}\rangle - \langle \mathrm{C}\rangle]. \tag{4.15}$$

Both terms on the right hand side vanish. The second one obviously cancels out, but the first one needs further explanations. The lower boundary case c = 0 does not contribute, but the upper boundary could potentially result in a non-zero value, if all concentration is located at one singular point as a Dirac function. But this situation does not exist in real transport situations. Thus, the transport equation for the mean concentration is given by

$$\frac{\partial \langle \mathrm{C}\rangle}{\partial \mathrm{t}} + \langle \mathbf{V}\rangle \cdot \langle \mathrm{C}\rangle - \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle = 0. \tag{4.16}$$

The M-term does not influence the mean motion since it cancels out.

The second statistical moment, the variance, is defined by

$$\sigma_{\mathrm{c}}^2 := \int \mathrm{c}^2\mathrm{P}\,\mathrm{dc} - \langle \mathrm{C}\rangle^2. \tag{4.17}$$

Now, the PDF transport equation is multiplied and integrated over the whole concentration space. The order of integration and derivation is also being interchanged and the product rule is applied on the right hand side:

$$\frac{\partial}{\partial \mathrm{t}}\int \mathrm{c}^2\mathrm{P}\,\mathrm{dc} + \langle \mathbf{V}\rangle \cdot \nabla \int \mathrm{c}^2\mathrm{P}\,\mathrm{dc} - \mathrm{D}^{\mathrm{ens}}\Delta \int \mathrm{c}^2\,\mathrm{P}\,\mathrm{dc} = \int \left\{\frac{\partial}{\partial \mathrm{c}}(\mathrm{cM}[\mathrm{c}-\langle \mathrm{C}\rangle]\mathrm{P}) - 2\mathrm{cM}[\mathrm{c}^2-\langle \mathrm{C}\rangle]\mathrm{P}\right\}\mathrm{dc}. \tag{4.18}$$

The first term on the right hand side vanishes for the same reason as in the derivation of the mean concentration.

$$\frac{\partial}{\partial \mathrm{t}}\int \mathrm{c}^2 \mathrm{P}\,\mathrm{dc} + \langle \mathbf{V}\rangle \cdot \nabla \int \mathrm{c}^2 \mathrm{P}\,\mathrm{dc} - \mathrm{D}^{\mathrm{ens}}\Delta \int \mathrm{c}^2\,\mathrm{P}\,\mathrm{dc} = \\ -2\mathrm{M}\left\{\int \mathrm{c}^2\,\mathrm{P}\,\mathrm{dc} - \langle \mathrm{C}\rangle^2\right\}. \tag{4.19}$$

The brackets on the right hand side could already be replaced by the concentration variance, but in order to do, the transport equation for $\langle \mathrm{C}\rangle^2$ needs to be subtracted from equation (4.19) and the equation for the squared mean concentration needs to be derived first. Equation (4.16) is multiplied by $\langle \mathrm{C}\rangle$:

$$\langle \mathrm{C}\rangle \frac{\partial \langle \mathrm{C}\rangle}{\partial \mathrm{t}} + \langle \mathrm{C}\rangle\langle \mathbf{V}\rangle \cdot \nabla\langle \mathrm{C}\rangle - \langle \mathrm{C}\rangle \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle = 0. \tag{4.20}$$

By making extensive use of the product rule one arrives at

$$\frac{\partial \langle \mathrm{C}\rangle^2}{\partial \mathrm{t}} - \langle \mathrm{C}\rangle \frac{\partial \langle \mathrm{C}\rangle}{\partial \mathrm{t}} + \langle \mathbf{V}\rangle \cdot \nabla\langle \mathrm{C}\rangle^2 - \langle \mathrm{C}\rangle\langle \mathbf{V}\rangle \cdot \nabla\langle \mathrm{C}\rangle \\ - \mathrm{D}^{\mathrm{ens}}[\nabla \cdot (\langle \mathrm{C}\rangle\nabla\langle \mathrm{C}\rangle) - (\nabla\langle \mathrm{C}\rangle)^2] = 0. \tag{4.21}$$

The dispersion term can be further modified by using the product

$$\mathrm{D}^{\mathrm{ens}}[\nabla \cdot (\langle \mathrm{C}\rangle\nabla\langle \mathrm{C}\rangle) - (\nabla\langle \mathrm{C}\rangle)^2] = \\ \mathrm{D}^{\mathrm{ens}}[\nabla \cdot (\nabla\langle \mathrm{C}\rangle^2 - \langle \mathrm{C}\rangle\nabla\langle \mathrm{C}\rangle) - (\nabla\langle \mathrm{C}\rangle)^2] = \\ \mathrm{D}^{\mathrm{ens}}[\Delta\langle \mathrm{C}\rangle^2 - (\nabla\langle \mathrm{C}\rangle)^2 + \langle \mathrm{C}\rangle\Delta\langle \mathrm{C}\rangle - (\nabla\langle \mathrm{C}\rangle)^2]. \tag{4.22}$$

Thus, equation (4.21) can be transformed to

$$\frac{\partial \langle \mathrm{C}\rangle^2}{\partial \mathrm{t}} + \langle \mathbf{V}\rangle \cdot \nabla\langle \mathrm{C}\rangle^2 - \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle^2 + 2\mathrm{D}^{\mathrm{ens}}(\nabla\langle \mathrm{C}\rangle^2) - \langle \mathrm{C}\rangle \frac{\partial \langle \mathrm{C}\rangle}{\partial \mathrm{t}} - \langle \mathrm{C}\rangle\langle \mathbf{U}\rangle \cdot \nabla\langle \mathrm{C}\rangle \\ - \langle \mathrm{C}\rangle \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle = 0. \tag{4.23}$$

Comparing the second line of equation (4.23) with equation (4.21), the transport equation for the squared mean concentration $\langle \mathrm{C}\rangle^2$ follows as

$$\frac{\partial \langle \mathrm{C}\rangle^2}{\partial \mathrm{t}} + \langle \mathbf{V}\rangle \cdot \nabla\langle \mathrm{C}\rangle^2 - \mathrm{D}^{\mathrm{ens}}\Delta\langle \mathrm{C}\rangle^2 + 2\mathrm{D}^{\mathrm{ens}}(\nabla\langle \mathrm{C}\rangle^2) = 0. \tag{4.24}$$

Now, equation (4.24) can be subtracted from equation (4.19):

$$\frac{\partial}{\partial t}\left[\int c^2 P\,dc - \langle C\rangle^2\right] + \langle \mathbf{V}\rangle\cdot\nabla\left[\int c^2 P\,dc - \langle C\rangle^2\right] - D^{ens}\Delta\left[\int c^2 P\,dc - \langle C\rangle^2\right] = 2D^{ens}(\nabla\langle C\rangle^2) - 2M\left[\int c^2\,P\,dc - \langle C\rangle^2\right]. \tag{4.25}$$

Finally, the definition of the concentration variance (4.18) can be inserted, which yields the final transport equation of the concentration variance

$$\frac{\partial}{\partial t}\sigma_c^2 + \langle \mathbf{V}\rangle\cdot\nabla\sigma_c^2 - D^{ens}\Delta\sigma_c^2 = 2D^{ens}(\nabla\langle C\rangle^2) - 2M\sigma_c^2. \tag{4.26}$$

The most interesting term of equation (4.26) is the last one on the right hand side. The same dissipation rate M as in the transport equation of the full PDF (4.12) appears here. Therefore, different propositions of the dissipation rate can be tested as a closure assumption for the transport equation of the concentration variance. The big advantage of testing different closures for the variance is that this equation is much easier to handle than the equation for the full concentration PDF.

4.2.3 Analytical Solutions of the First Moment

An analytical solution of the transport equation of the mean concentration (4.24) with a Gaussian initial condition centered at the position $\mathbf{x}_0$ evolving from time t_0 can easily be found, for example by transforming equation (4.24) into the frequency domain, which makes it an ordinary differential equation. This ODE can be solved and the solution can then be transformed back into the time domain, which gives:

$$\langle C\rangle = \left(4\pi D^{ens}(t+t_0)\right)^{-d/2}\exp\left\{-\frac{(\mathbf{x}-\mathbf{x_0}-\langle\mathbf{V}\rangle t)^2}{2D^{ens}(t+t_0)}\right\}. \tag{4.27}$$

4.2.4 Analytical Solution of the Second Moment

An analytical solution for the variance transport equation (4.26) needs a more tedious derivation. A semi-analytical solution can be found by using Green's function. A derivation similar to the one presented by /KAP 94/ will be presented here. The differential operator $L(\mathbf{x},t)$ is defined by

$$L\sigma_c^2 = \frac{\partial}{\partial t}\sigma_c^2 + \langle\mathbf{V}\rangle\cdot\nabla\sigma_c^2 - D^{ens}\Delta\sigma_c^2 - 2M\sigma_c^2 = 2D^{ens}(\nabla\langle C\rangle^2), \tag{4.28}$$

with the inhomogeneity

$$g(\mathbf{x},t) = 2D^{ens}(\nabla\langle C\rangle^2). \tag{4.29}$$

Then, equation (4.26) can be rewritten as

$$L(\mathbf{x},t)\sigma_c^2(\mathbf{x},t) = g(\mathbf{x},t). \tag{4.30}$$

The Green's function $G(\mathbf{x}-\mathbf{x}',t,t')$ is defined as the solution to the differential operator $L(\mathbf{x},t)$ with delta functions on the right hand side

$$L(\mathbf{x},t)G(\mathbf{x}-\mathbf{x}',t,t') = \delta(\mathbf{x}-\mathbf{x}')\delta(t-t'). \tag{4.31}$$

It can be shown that the general solution of equation (4.26) is given by

$$\sigma_c^2(\mathbf{x},t) = \sigma_{c_h}^2(\mathbf{x},t) + \int_{\mathbb{R}}\int_{\mathbb{R}^d} G(\mathbf{x}-\mathbf{x}',t,t')\, g(\mathbf{x}',t')d\mathbf{x}'dt'. \tag{4.32}$$

where $\sigma_{c_h}^2(\mathbf{x},t)$ is the solution of equation (4.26) without an inhomogeneity. Assuming that the initial condition is known without any uncertainty, the variance at time t = 0 is

$$\sigma_c^2(\mathbf{x},t=0) = 0.$$

But without the inhomogeneity, which acts as the only source term, the solution of the homogeneous PDE is $\sigma_{c_h}^2(\mathbf{x},t) = 0$ for all times. Therefore, the solution to the homogeneous problem can be dropped. If the Green's function is known, the solution to equation (4.26) can be calculated from equation (4.32), which is a convolution in space of Green's function and the inhomogeneity:

$$\begin{aligned}\sigma_c^2(\mathbf{x},t) &= \int_{\mathbb{R}}\int_{\mathbb{R}^d} dt'\, G(\mathbf{x}-\mathbf{x}',t,t')\, g(\mathbf{x}',t')d\mathbf{x}' \\ &= \int_{\mathbb{R}} dt'\, G(\mathbf{x},t,t') * g(\mathbf{x}',t') \\ &= \int_{\mathbb{R}} dt'\, \mathcal{F}^{-1}\left\{\tilde{G}(\mathbf{x},t,t')\tilde{g}(\mathbf{x}',t')\right\},\end{aligned} \tag{4.33}$$

where a tilde denotes the Fourier transform of a variable. Hence, $\tilde{G}$ and $\tilde{g}$ need to be calculated in order to obtain the solution σ_c^2. Fourier transforming both sides of equation (4.30) gives an ODE in the frequency domain, which can be solved by separation of variables, resulting in

$$\begin{aligned}\tilde{G}(\mathbf{k},t,t') &= \Theta(t-t')\exp\{-(i\langle\mathbf{V}\rangle\cdot\mathbf{k} + D^{ens}\mathbf{k}^2)(t-t')\} \\ &\cdot \exp\left\{-2\int_{t'}^{t} dt''\, M(t'')\right\}.\end{aligned} \tag{4.34}$$

In order to transform the inhomogeneity (4.29), the transformed mean concentration $\langle \mathrm{C} \rangle$ from equation (4.27) needs to be inserted:

$$\tilde{\mathrm{g}}(\mathbf{k},\mathrm{t}) = \mathcal{F}[2\mathrm{D}^{\mathrm{ens}}(\nabla\langle \mathrm{C}\rangle^2)] = \frac{-1}{(2\pi)^{\mathrm{d}/2}}\,\mathbf{k}\langle\tilde{\mathrm{C}}\rangle * \mathbf{k}\langle\tilde{\mathrm{C}}\rangle. \tag{4.35}$$

Simply using equation (4.27) leads to a singularity for t = 0, as the Gaussian distribution tends to a delta function for small times. This unnatural behavior can be avoided by modifying solution (4.27) in a way that for t = 0 it stays a finite Gaussian distribution. If a time t_0 is introduced by which the solution is shifted forward in time exactly the wished behavior is achieved. By inserting the shifted concentration

$$\langle\tilde{\mathrm{C}}\rangle = \frac{\langle \mathrm{C}\rangle_0}{(2\pi)^{\mathrm{d}/2}} \exp\left\{-\frac{\mathrm{D}^{\mathrm{ens}}}{\mathrm{n}}\mathbf{k}(\mathrm{t}+\mathrm{t}_0) - \mathrm{i}\mathbf{k}\cdot\left(\mathbf{x_0} + \langle\mathbf{V}\rangle\frac{\mathrm{t}}{\mathrm{n}}\right)\right\} \tag{4.36}$$

into the transport equation, it can be shown that this is still a correct solution to the original PDE (4.27). With this new solution, the Fourier transformed inhomogeneity can be calculated:

$$\begin{aligned}\tilde{\mathrm{g}}(\mathbf{k},\mathrm{t}) = \frac{\langle \mathrm{C}\rangle_0^2\,\mathrm{D}^{\mathrm{ens}}}{2(2\pi)^{\mathrm{d}}} \frac{1}{\left(2\,\mathrm{D}^{\mathrm{ens}}(\mathrm{t}+\mathrm{t}_0)\right)^{\mathrm{d}/2}} \Bigg[\frac{1}{\mathrm{D}^{\mathrm{ens}}(\mathrm{t}+\mathrm{t}_0)} \\ - \mathbf{k}^2\Bigg] \exp\left(\frac{1}{2}\,\mathrm{D}^{\mathrm{ens}}(\mathrm{t}+\mathrm{t}_0)\mathbf{k}^2 - \mathrm{i}(\mathbf{x_0} + \langle \mathrm{U}\rangle \mathrm{t})\cdot\mathbf{k}\right).\end{aligned} \tag{4.37}$$

Finally, everything can be combined into equation (4.32) and a final time integral remains to be calculated:

$$\begin{aligned}\sigma_{\mathrm{c}}^2 = 2\mathrm{D}^{\mathrm{ens}} \int_0^{\mathrm{t}} \mathrm{dt}' \quad \left(4\pi^2(\mathrm{D}^{\mathrm{ens}})^2(\mathrm{t}'+\mathrm{t}_0)(2\mathrm{t}+\mathrm{t}_0-\mathrm{t}')\right)^{-\frac{\mathrm{d}}{2}} \\ \cdot\left[\frac{d(\mathrm{t}-\mathrm{t}')}{2\mathrm{D}^{\mathrm{ens}}(\mathrm{t}'+\mathrm{t}_0)(2\mathrm{t}+\mathrm{t}_0-\mathrm{t}')} + \frac{(\mathbf{x}-\langle\mathbf{V}\rangle\mathrm{t})^2}{\left(2\mathrm{D}^{\mathrm{ens}}(2\mathrm{t}+\mathrm{t}_0-\mathrm{t}')\right)^2}\right] \\ \cdot\exp\left(-\frac{(\mathbf{x}-\langle\mathbf{V}\rangle\mathrm{t})^2}{\left(2\mathrm{D}^{\mathrm{ens}}(2\mathrm{t}+\mathrm{t}_0-\mathrm{t}')\right)^2}\right)\exp\left(-2\int_{\mathrm{t}'}^{\mathrm{t}}\mathrm{dt}''\,\mathrm{M}(\mathrm{t}'')\right).\end{aligned} \tag{4.38}$$

This integral can either be evaluated analytically by using a long time approximation or by applying numerical methods. In this case, the short time behavior is also of interest, thus the integral is solved numerically using an adaptive quadrature.

4.3 A Time Dependent Extension of the IEM Model time dependent mixing model

The original IEM model approximates the conditional diffusion term by

$$\frac{\partial^2}{\partial c^2}\langle D\nabla\mathbf{C}\cdot\nabla\mathbf{C}|c\rangle = \frac{\partial}{\partial c}\big(M[c-\langle C(\mathbf{x},t)\rangle]P(c;\ \mathbf{x},t)\big), \tag{4.39}$$

where the dissipation rate M is assumed to be constant. M is determined by the ensemble dispersion divided by a characteristic mixing length to the square. As discussed before, in heterogeneity induced mixing in porous media flow needs more time to become ergodic. A more realistic dissipation rate should be determined by Deff(t) instead of Dens. since the ensemble dispersion coefficient accounts for the fluctuations of the center of mass of the concentration plume from realization to realization. These fluctuations do not exist in a single realization and play no role for the mixing. The long time behavior of both coefficients is identical and because the mixing in turbulent flows is so much faster than it is in groundwater flows, the difference does not matter for turbulent flows. Therefore, the mathematically simpler to handle ensemble dispersion coefficient is used in studies concerning turbulent flows. In addition, the characteristic squared mixing length scale is given by Dt where D is the local dispersion.

Therefore, a time-dependent dissipation rate is proposed

$$M(t) = k_M\frac{D^{\mathrm{eff}}(t)}{Dt}, \tag{4.40}$$

M(t) has larger values at early times than the constant one, which causes a stronger dissipation. But then it drops below the constant mixing frequency and approaches the limit M(t) = 0.

With this adaptation, the simplicity and low computational costs of the IEM model are preserved, while at the same time, the dissipation rate accounts for the physical mixing and dissipation effects in porous media flows.

As demonstrated in Fig. 4.2, the standard IEM model predicts a constant dissipation rate of the solute plume. The new time dependent IEM model, models a stronger dissipation at early times and a weaker dissipation at later times.

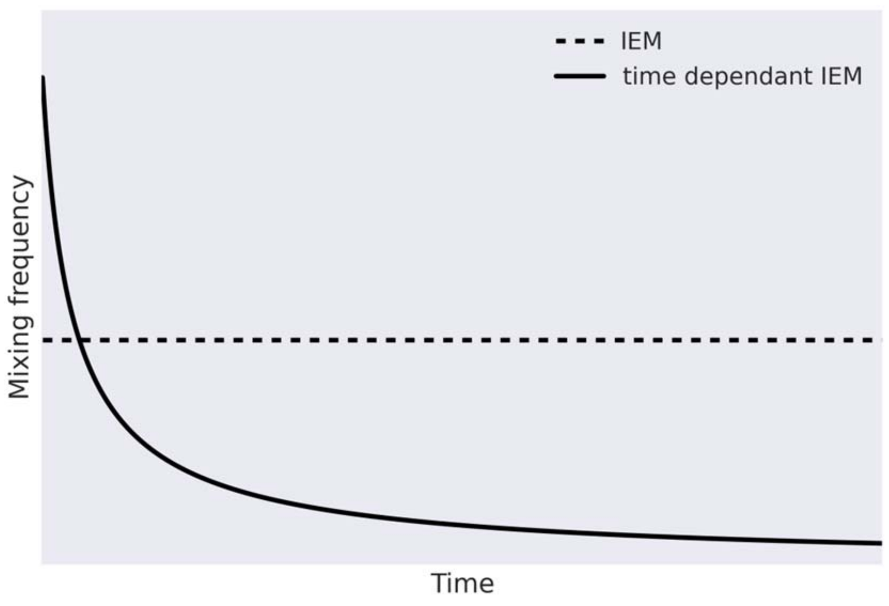


Fig. 4.2 Dissipation M of the standard IEM model
and the newly proposed time dependent IEM model plotted against time t

4.4 Simulations

4.4.1 Simulation Setup

In order to verify the theoretical results and the newly time dependent IEM model, numerical simulations were performed. The simulations follow the set-up described in /DEN 02/. The heterogeneous flow field was modelled as a solution of the linearised Darcy equation by the Kraichnan algorithm /KRA 70/. The mean flow velocity is set to $\langle \mathrm{V} \rangle = 1\mathrm{m}\ \mathrm{d}^{-1}$, and a Gaussian covariance structure with an integral scale of $\lambda = 1\mathrm{m}$ and a variance of $\sigma^2 = 0.1$ is chosen. The flow fields are generated by using 6400 Fourier modes. The particles moving in the velocity field and performing random jumps were modelled according to

$$\mathrm{dX_i(t)} = \mathrm{V_i}(\mathbf{X})\mathrm{dt} + \sqrt{2\mathrm{D}}\, d\mathrm{W_i(t)}, \tag{4.41}$$

where $\mathrm{W_i(t)}$ are independent standard Wiener processes /SUC 15/. An extended Runge-Kutta scheme /DRU 84/ with an accuracy of order $(\Delta t)^{3/2}$ is used to discretize the Langevin equations (4.41).

The particles undergo diffusive jumps with isotropic local dispersion coefficients of $\mathrm{D} = 0.01\mathrm{m}^2\ \mathrm{d}^{-1}$. The particles are distributed uniformly in a rectangle with side lengths according to an initial diffusion time of t_0 = 10 d. A time step of $\Delta t = 0.5d$ is used. 1000 realisations are calculated to create a statistical ensemble.

In order to reduce the computational effort, the so called GRW-algorithm is used here as a second option according to the same set up as described by /SUC 15/. The GRW uses a superposition of many weak solutions to Langevin equations projected on a regular grid. The particles solving the Langevin equations are spread on the grid globally according to the advection and dispersion coefficients of the transport equation. By construction, this algorithm is free of numerical diffusion and can be used for practically arbitrary numbers of particles without an impact on the computational costs. The transport parameters are set to the same as for the standard particle tracking. A normalized 2-dimensional histogram on grid cells with a size of $1m \times 1m$ was performed to calculate concentrations from the particle positions.

4.4.2 Concentration Variance

Monte Carlo Simulations are used to determine the reference concentration variance. This solution is compared to the analytical solution (4.38) of equation (4.26) making use of the two different proposed dissipation rates. At early times, the concentration variance shows a unimodal distribution with a transition to a bimodal distribution at later times as described already in /AND 98/.

The concentration variance computed from both, the Monte Carlo particle tracking and the GRW simulations are compared to the analytical solution (4.26) with the two different mixing models, the standard IEM model and the time dependent IEM model (4.40) in Fig. 4.3. The different solutions are plotted 50 d, 100 d, 200 d, 300 d, 400 d and 500 d after injection. First of all, it should be noted that the two independent numerical simulations match well. But the most obvious feature of the figure is the large discrepancy at early times between the analytical solution with the standard IEM closure and the numerical reference runs.

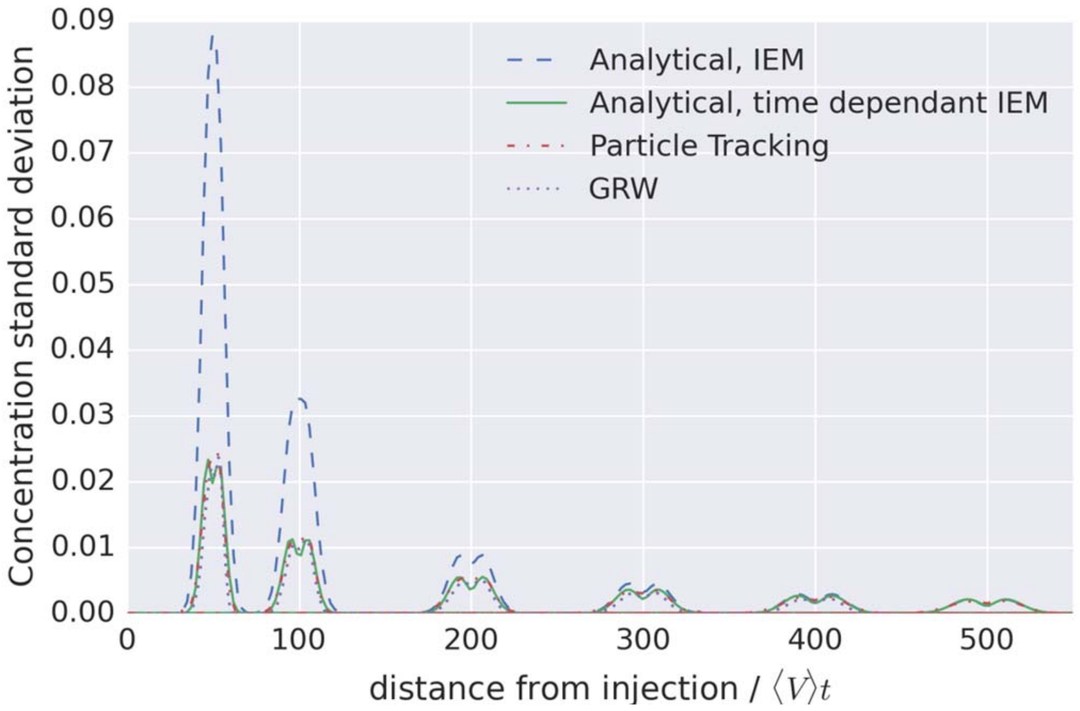


Fig. 4.3 Analytical solution with the classical and time dependent IEM mixing model compared to particle tracking and GRW simulations at different times

The variance dissipation at short times is underestimated by the standard IEM model whereas the time-dependent IEM-model causes a much larger dissipation at short times. Consequently, the temporal evolvement of the concentration variance models by the new IEM model matches the reference simulation runs much better.

4.4.3 PDF Modelling

The solutions of the PDF evolution equations depend on $3 + \mathrm{N}_\alpha$ independent variables plus the time variable, with $\mathrm{N}_\alpha = 1$ being the number of species forming a higher dimensional system of equations. In turbulence theory, mostly particles methods are used to solve for the PDF. These particle methods avoid errors by numerical diffusion /RAD 11/, unavoidable in classical schemes for PDF equations, which usually result in strongly advection-dominated problems. Here solutions of particle methods are presented which are also used a test case for higher dimensional numerical solvers presented in Section 3.4. In the case of Pope's /POP 85/ particles approach and also other particle tracking methods, the computational cost increases linearly with the number of particles. The GRW algorithm /VAM 03/ is insensitive to the increase of the number of particles. The GRW algorithm is equivalent to a finite difference scheme for advection–diffusion problems with constant coefficients. But in case of variable coefficients it is faster and free of numerical diffusion.

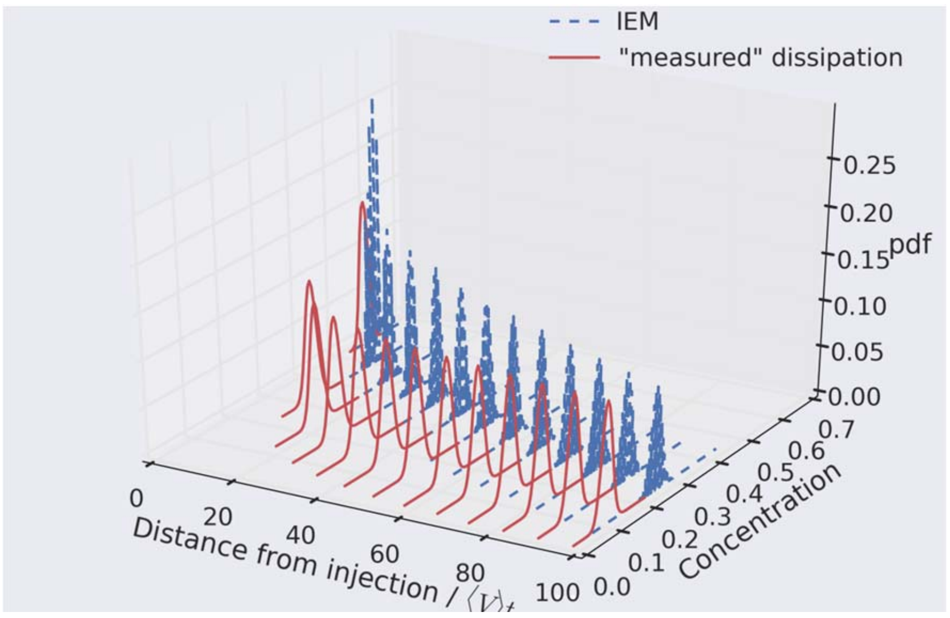


Fig. 4.4 The concentration PDF at the center of the plume $x = \langle V \rangle t$ with the standard IEM model compared to the PDF with empirically determined dissipation rates every 10 days. The standard IEM model underestimates the dissipation, which causes the gap between the peaks

For instance, when solving flow and transport problems in highly heterogeneous media by finite element schemes and the transport solvers are replaced by a GRW scheme, then the numerical diffusion is completely removed and the total computation time is about 20 times smaller /SUC 13/. Note that for a grid-free particle tracking scheme the computing time is of the order of the number of particles and for a GRW solution to the same problem and the same number of time steps it is of the order of grid points occupied by particles. For the solution of the PDF problem solved here, using 10^{24} particles which move on a lattice with $\sim 10^5$ points, the GRW computing time is about 0.5s while a sequential particle tracking would require a computing time $\sim 10^{19}$ times larger. To illustrate the GRW-PDF approach, a two dimensional PDF problem is considered for joint concentration–position PDF $P(c; x, t)$, solution of the particular form of (4.12)

$$\frac{\partial P}{\partial t} + \mathcal{V}\frac{\partial P}{\partial x} + \mathcal{V}_c\frac{\partial P}{\partial c} = \mathcal{D}\frac{\partial^2 P}{\partial x^2} + \mathcal{D}_c\frac{\partial^2 P}{\partial c^2}, \tag{4.42}$$

with $\mathcal{D}$ and $\mathcal{D}_c$ being upscaled diffusion coefficients in physical and concentration space, respectively. $\mathcal{V}$ and $\mathcal{V}_c$ are upscaled drift coefficients. The solution of the Fokker–Planck equation (4.42) is approximated by the point-density at lattice sites of a large number of computational particles evolving according to Itô equations

$$d\mathbf{X}(t) = \mathcal{V}(\mathbf{X}(t), t)dt + d\mathbf{W}(\mathbf{X}(t), t) \tag{4.43}$$

$$d\mathbf{C}(t) = \mathbf{M}(\mathbf{C}(t), \mathbf{X}(t), t)dt. \tag{4.44}$$

At a given time step, the computational particles on a lattice site are globally scattered in groups of particles which remain at the position determined by the drift coefficients and particles undergoing diffusive jumps. The numbers of particles in each group are binomial random variables with parameters determined by the coefficients of the Itô equations, the time step, and the lattice constants. The GRW algorithm is thus a superposition of many weak Euler-schemes for systems of Itô equations projected on a regular lattice. The GRW algorithm is free of numerical diffusion, because the diffusive jumps and the nominal diffusion coefficients are related to the lattice. It is practically insensitive to the increase of the number of particles (see /SUC 13/ for implementation details and convergence tests). Since the particles move between lattice sites on which mean values are also defined through particle densities, the GRW-PDF solution presented here avoids the artificial diffusion generated in classical PDF methods by interpolation to particle positions of the means computed by averaging over computational cells.

Monte Carlo simulations of 2d passive transport of a single chemical species in saturated aquifers were used to estimate the PDF $P(c;x,t)$ of the cross-section space-average concentration $C(x,t) = \int_0^{L_y} \quad c(x,y,t)dy$,where L_y is the transverse dimension of the computational domain, estimated at the x-coordinate of the plume centre of mass, $x = \langle V \rangle t$. The upscaled drift coefficient $\mathcal{V}$ in eq. (4.43) is the ensemble mean velocity, equal to the velocity of the center of mass $\langle V \rangle = 1$m/day. The upscaled diffusion coefficient $\mathcal{D}$ is the longitudinal component of the time dependent ensemble Dispersion coefficient. The latter is a self-averaging quantity for transport in random velocity fields with finite correlation range, as considered here. Hence, $\mathcal{D}(t)$ was efficiently determined by using a single particle trajectory of diffusion in a single realization of the random velocity field. Two different mixing models M were considered in the concentration Itô equation (4.44). The first one is the IEM model, a drift term given by the attenuation of the mean concentration due to the local diffusion, and an additional noise term modeled as a Wiener process, needed to control the shape of the PDF /FOX 03/.

$$dC(t) = -a(t)[C(t) - \langle C(t) \rangle]dt + \Delta\langle C(t) \rangle + bdW(t). \tag{4.43}$$

The coefficient $a(t)$ was estimated, similarly to turbulence problems by the inverse of the diffusion time scale $\mathcal{D}(t)/\lambda^2$, with $\lambda = 1$ m, the characteristic correlation scale of the transport problem. $\Delta\langle C(t) \rangle$ was estimated from a fractional step, performed at each time step of the GRW-PDF simulation, consisting of a GRW simulation of the 1d diffusion with the constant local diffusion coefficient $D = 0.01$m²/d considered in the Monte Carlo simulations. The amplitude $b \approx 10^{-6}$ 1/d of the Wiener process W(t) was adjusted by comparisons with the Monte-Carlo inferred Eulerian PDF $P(c;x,t)$. The results obtained by this IEM model are represented by the dotted curves in Fig. 4.3

The second model consists of a drift term equal to the rate of decay of the mean concentration at mass center $\langle C(x,t) \rangle$, determined from the ensemble of Monte Carlo simulations /SUC 06/ and a small diffusion in the concentration space. The corresponding diffusion coefficient starts with an initial value adjusted in the same way as the noise term in equation (4.45) and decays exponentially in time, as suggested by a preliminary analysis of concentration time series generated during the Monte Carlo simulations, carried out with an automatic algorithm /VAM 12/. The Eulerian concentration PDF $P(c;x,t)$ was simulated by the GRW algorithm and compared with the Monte Carlo results. The cross-section concentration $C(x,t)$ is ergodic with a good approximation /SUC 06/, thus $C(x,t) \simeq \langle C(x,t) \rangle$. The initial distribution of particles in the (x,c) plane was approximated by multiplying the Monte Carlo PDF at $t = 1$ d by 10^{24} particles.

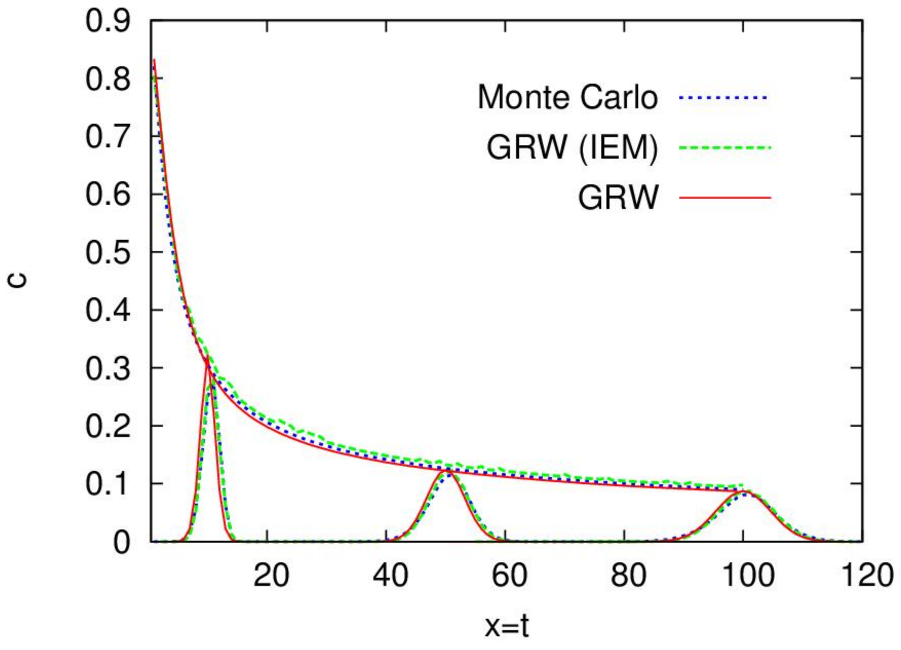


Fig. 4.5 The mean concentration $\langle C\rangle(x)$ at fixed times t = 10, 50, 100 days (peaks) and $\langle C\rangle(t)$ at the centre of mass $x = \mathcal{V}t$ (monotone curves) calculated by Monte Carlo methods and by the GRW with the two mixing methods

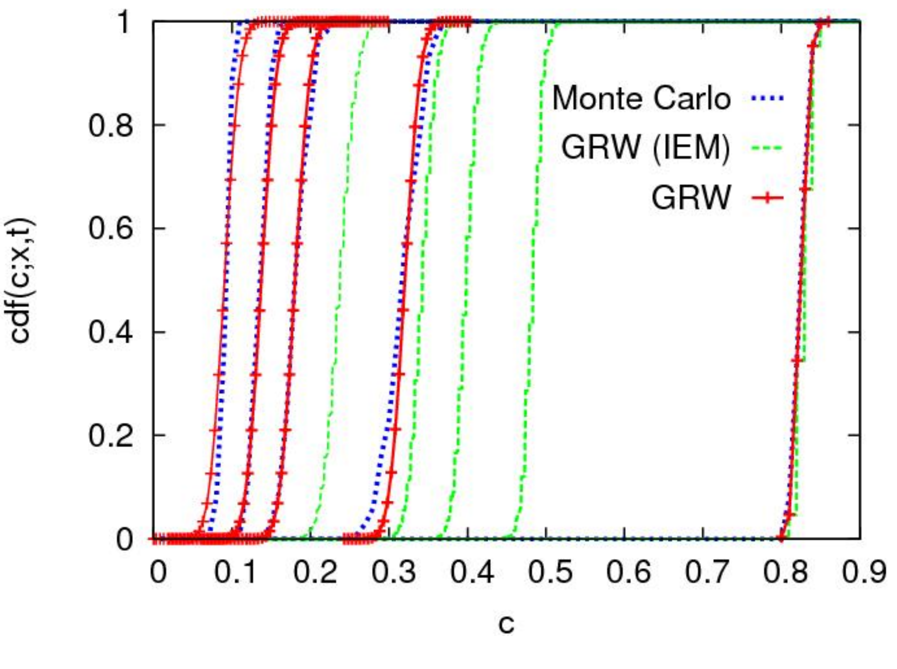


Fig. 4.6 The cumulative distribution functions cdf$(c; x, t)$, $x = \mathcal{V}t$, at t = 0, 10, 30, 50, 100 d (from right to left) calculated by Monte Carlo methods and the GRW with two mixing models

Fig. 4.5 shows a good agreement of the mean concentration, for both mixing models, between Monte Carlo results and GRW-PDF simulations. This result is already expected because $\langle C(x,t)\rangle$ is the probability density of the computational particles governed by the Itô equation (4.43), which is independent of the concentration Itô equation (4.44) and of the dissipation model. The comparisons presented in Fig. 4.4 and Fig. 4.6 show that the dissipation model M based on the rate of decay of "measured" (i.e. simulated) concentrations resembles the Monte Carlo results quite well. The IEM model (4.45) instead fails to capture the transport of the PDF in physical and concentration space. The discrepancy may be attributed to structural differences between groundwater and turbulent flows. Such models are better suited for homogeneous systems.

As demonstrated in Fig. 4.4, the standard IEM model predicts dissipation rates of the solute plume, which are too small. The underestimation possibly stems from the interplay of steep concentration gradients at these early times together with the heterogeneous velocity fluctuations, because these two effects cause enhanced mixing. Heterogeneity induced mixing is very efficient in the elimination of small scale concentration fluctuations but it needs time to develop in porous media flows. The IEM model does not account for this transient behavior, because it is constant in time. To overcome this problem, a time dependent extension of the IEM model is proposed. As already described in Section 4.4.2, the time dependent IEM model greatly improves the results for the concentration variance. Similar comparisons were done for the concentration PDF. These results are shown in Fig. 4.7. The concentration PDFs with the standard IEM model, the time dependent IEM model, and with empirically determined dissipation rates at the mass center of the plume are compared. It can be seen that the dissipation rate is larger for the time dependent model than for the standard IEM model. With this enhanced dissipation rate, the empirically determined rates are matched to a good extent. Although the shape of the PDF is narrowed too much, it gives much better results compared to the standard IEM model.

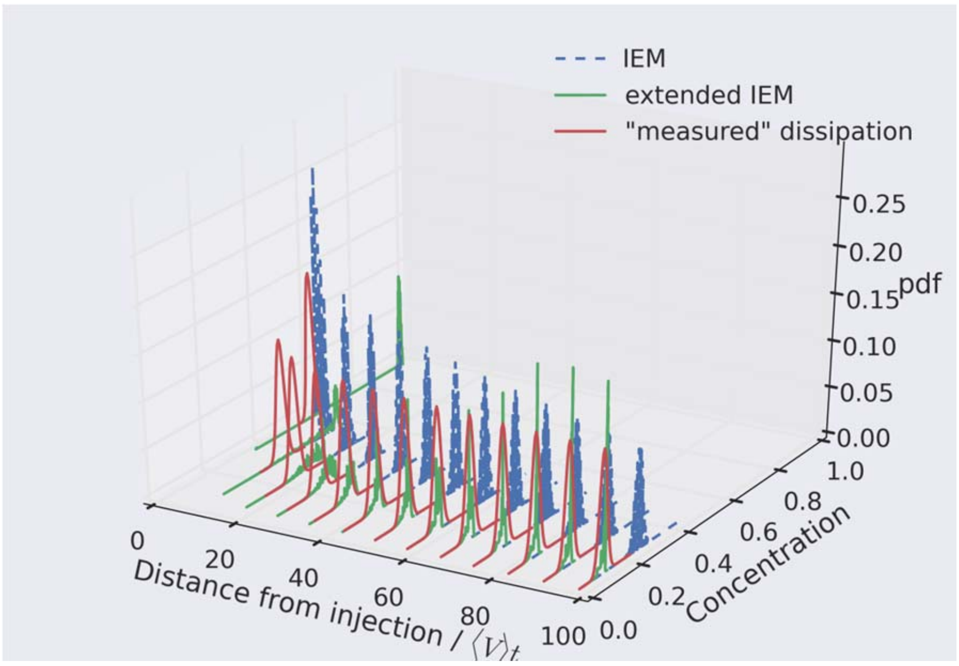


Fig. 4.7 The concentration PDF at the center of the plume $x = \langle V \rangle t$ with the standard IEM model (blue) and the time dependent IEM model (green) compared to the PDF with empirically determined dissipation rates (red) every 10 days

4.5 Conclusions

Equations for the mean concentration, the concentration variance and the complete concentration PDF have been established. To achieve this goal, explicit results for the ensemble dispersion and the concentration dissipation rate have been derived and inserted into the different equation. Solving these three equations and comparing them with an ensemble of Monte-Carlo simulations showed an excellent agreement. The results build the basis for a computationally efficient replacement of computer resource demanding huge Monte-Carlo simulations by simpler equations for the mean concentration, concentration variance or the concentration PDF. The concentration PDF follows as a solution of a higher dimensional problem that is computationally more demanding than solving only for mean and variance of the concentration. However, the reduction of higher computational costs can be reduced by making use of the efficient numerical solvers for high dimensional problems as described in Section 3.5.

5 Code verification and applications

5.1 Viscosity-dependent flow

5.1.1 Model description

A vertical two-dimensional flow and heat transport model established in the E-DuR project /SCH 12/ was modified to investigate the effect of variable viscosity on the temperature field.

The model domain is depicted in Fig. 5.1. It represents an area of about 814 m width and 1057 m depth and includes four hydrogeological units (I to IV) as well as five discrete fractures (F1 to F5). The permeability of the hydrogeological layers decreases strongly with depth. The fractures have a rather high permeability compared to the surrounding rock matrix. They can be traced through the lower two hydrogeological units and are dipping at angles between 50 and 90 degrees.

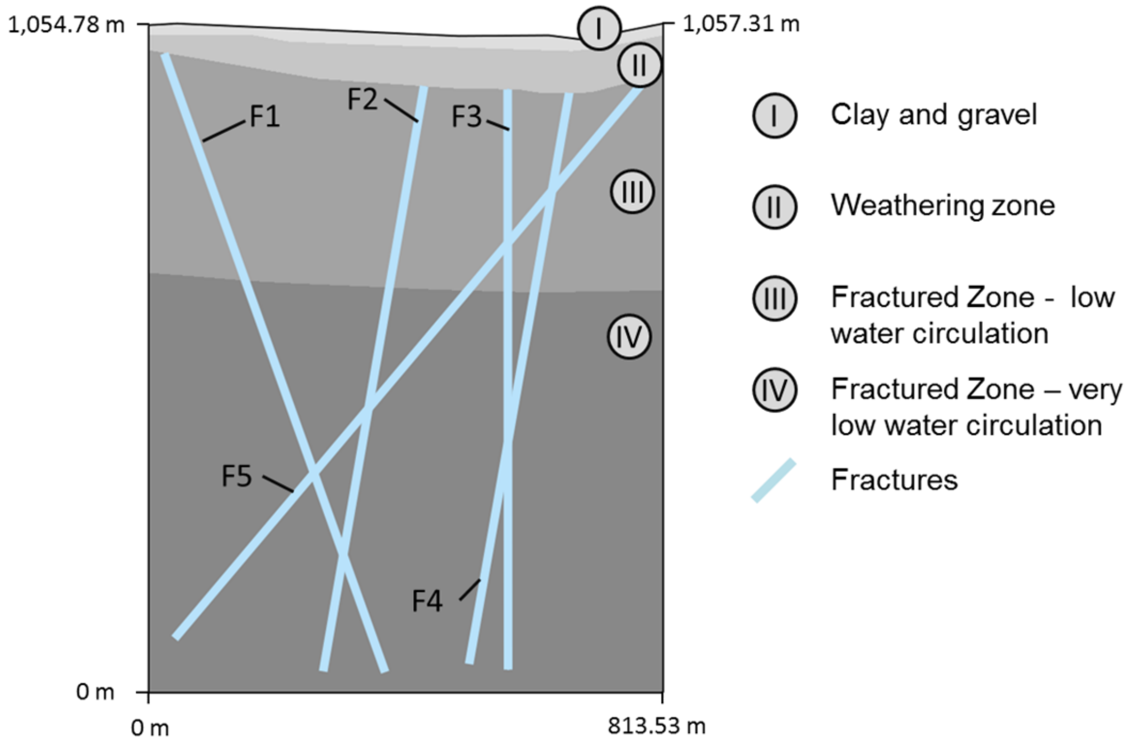


Fig. 5.1 Model geometry based on the schematic cross-section

Pressure boundary conditions at the left and right boundary represent constant hydraulic heads of 1048.8 m and 1048.4 m, respectively (cp. Fig. 5.2). A groundwater recharge of 41.3 mm a^{-1} is assigned to the top boundary. The bottom of the model is impermeable to flow.

A temperature of 2 °C is set at the top and of 32 °C at the bottom of the model. An in-/outflow boundary condition is assigned to the left boundary which means that the inflow temperature reflects the general temperature gradient of about 2.84 °C per 100 m and the outflow temperature is allowed to vary. On the right boundary, only outflow occurs allowing for variable temperatures. An initial temperature distribution according to the general temperature gradient is assumed.

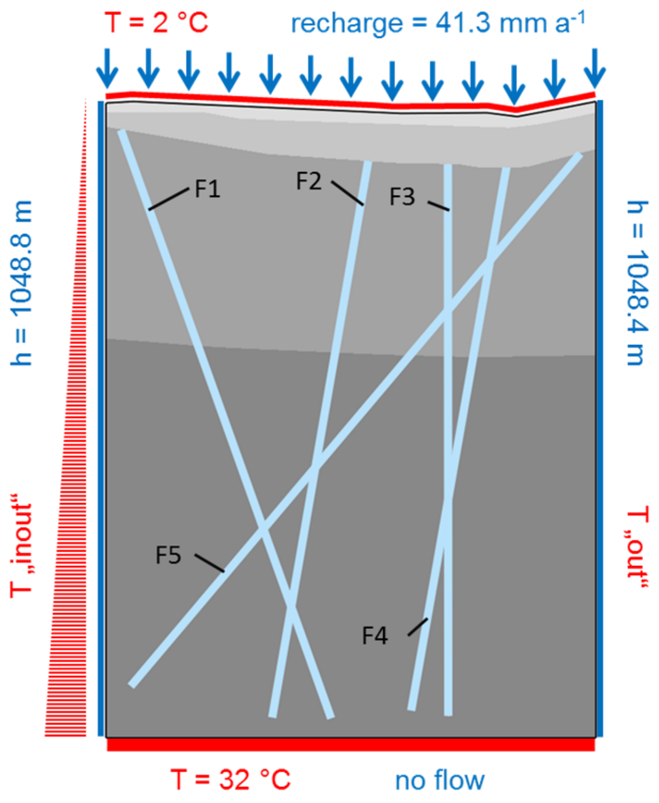


Fig. 5.2 Boundary conditions for the groundwater flow and heat transport

Two model variations were set up using the same hydrogeological and heat transport parameters (cp. Tab. 5.1 and Tab. 5.2). In the constant viscosity model, viscosity is fixed to $1.002 \cdot 10^{-3}$ Pa s^{-1} which corresponds to a temperature of 20 °C. In the variable viscosity model, however, the influence of temperature $T_{°C}$ on viscosity η is considered according to the following formulation /KRO 10/:

$$\eta(T_{°C}) = \frac{3.5}{(17 \cdot T_{°C} + 575)^{1.18}} + 6 \cdot 10^{-5} \left(\frac{T_{°C}}{200}\right)^2 - 1.2 \cdot 10^{-4} \; [Pa\, s^{-1}] \qquad (5.1)$$

Viscosity values of 1.7 Pa s at the top to 0.7 Pa s at the bottom of the model follow the temperature distribution (cp. Fig. 5.3). The influence of temperature on density is neglected because it is very low (changes by 0.5 % between 2 °C and 32 °C) compared to the effect of temperature on viscosity (changes by 55 % between 2 °C and 32 °C).

Tab. 5.1 Permeability of the four hydrogeological units and the fractures

Hydrogeological unit	**Permeability [m²]**
I: Clay and gravel	$2.08 \cdot 10^{-12}$
II: Weathering zone	$5.79 \cdot 10^{-13}$
III: Fractured zone (low water circulation)	$1.16 \cdot 10^{-17}$
IV: Fractured zone (very low water circulation)	$1.16 \cdot 10^{-21}$
Fractures	$7.52 \cdot 10^{-9}$

Tab. 5.2 Flow and heat transport parameters for the rock matrix and the fractures

	Matrix	**Fracture**
Porosity [-]	$4.65 \cdot 10^{-3}$	$5.0 \cdot 10^{-3}$
Tortuosity [-]	1.0	1.0
Molecular diffusivity [m^2 s^{-1}]	$1.0 \cdot 10^{-9}$	$1.0 \cdot 10^{-9}$
Longitudinal dispersivity [m]	5.0	5.0
Transversal dispersivity [m]	$5.0 \cdot 10^{-1}$	$5.0 \cdot 10^{-1}$
Heat capacity of the fluid [J kg^{-1} K^{-1}]	$4.17 \cdot 10^{3}$	$4.17 \cdot 10^{3}$
Heat capacity of the solid [J kg^{-1} K^{-1}]	733	733
Thermal conductivity of the fluid [J s^{-1} m^{-1} K^{-1}]	2.65	2.65
Mass density of the solid phase [kg m^{-3}]	$3.0 \cdot 10^{3}$	$3.0 \cdot 10^{3}$

Mass density of the fluid phase [kg m^{-3}]	998.2	998.2
Aperture [m]	-	$1 \cdot 10^{-2}$

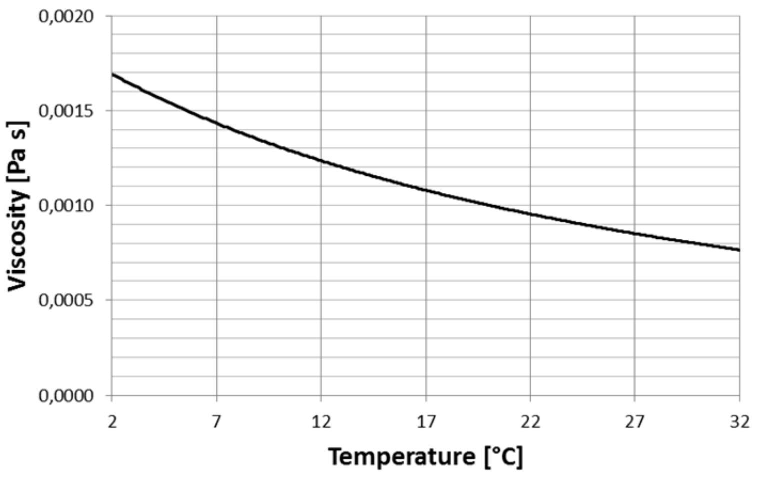


Fig. 5.3 Viscosity of pure water according to /KRO 10/

5.1.2 Results

Flow and heat transport simulations were run for about 28 years model time ($8.7 \cdot 10^8$ s) for the constant and the variable viscosity model. The resulting flow fields agree in their main characteristics (Fig. 5.4): the flow velocity corresponds by and large to the permeability of the matrix. The general flow direction is from left to right but vertical components are induced on one hand by the groundwater recharge and on the other hand by the draining effect of the fractures. Flow through the fracture system follows basically fractures F1 and F5 (cp. Fig. 5.5a,b). They constitute a hydraulic bypass between the left and the right boundary. Very low flow velocities occur in the lower thirds of the fractures which form dead ends within the very lowly permeable matrix.

The main difference between the flow fields from the constant and the variable viscosity model concern the position of a flow divide formed in the upper left corner of the model (Fig. 5.4). This divide arises where a part of the recharge flows to the left boundary, while the other part flows to the right. In the constant viscosity model, the divide appears at about 130 m from the left boundary. Considering variable viscosity causes it to move by about 100 m to the right. Further explanations for the appearance of a divide in this model are given in Appendix B.

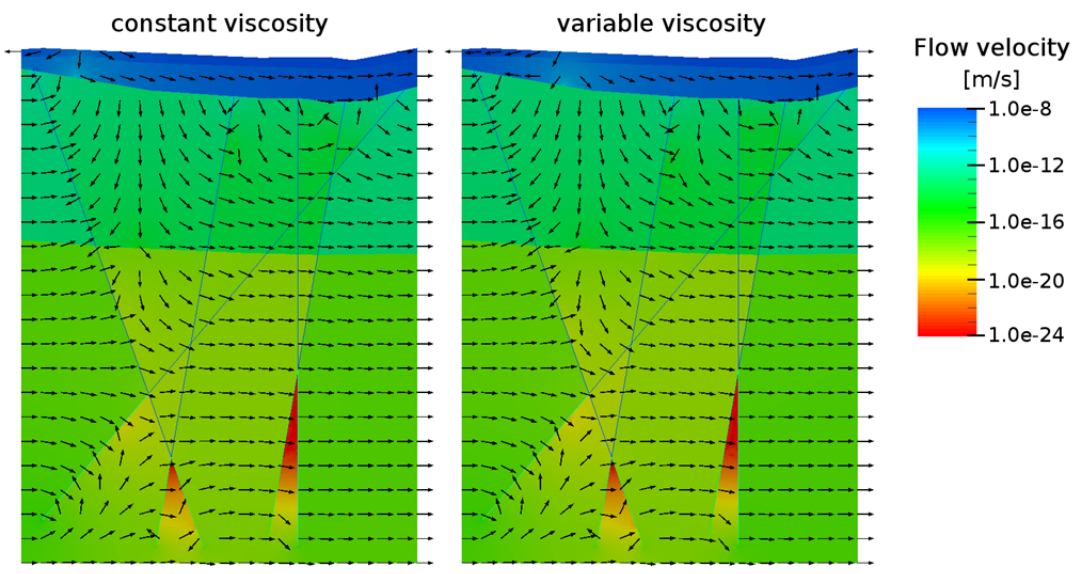


Fig. 5.4 Flow field for simulations with constant and variable viscosity

At the same time, the impact of the fracture system on the overall flow field decreases by applying variable viscosity: the flow velocity in the main flow path is diminished by about 50 % in F1 and about 20 to 50 % in fracture F5. This is mirrored in the outflow rates of the lateral boundaries (Fig. 5.5 c) which are dominated in the variable viscosity model by the outflow from unit I and II. On the left side, the outflow rate of the variable viscosity model is two times higher than the outflow rate of the constant viscosity model while it amounts only to 20 % of the outflow rate at the right boundary. The integral of the outflow rates is the same in both models and corresponds to the recharge.

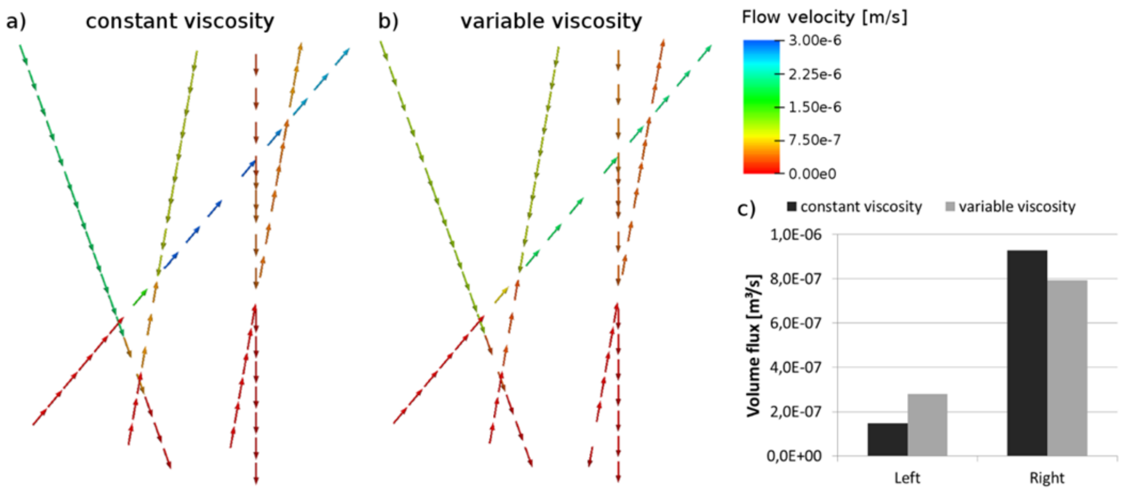


Fig. 5.5 Flow velocities in the fracture system (a, b) and outflow rates for the lateral boundaries (c) for the constant and variable viscosity model

It was shown in a former project that the fracture system influences the temperature field in its vicinity /SCH 12/. The initially imposed geothermal temperature field is distorted depending on the flow direction and velocity within the fractures. Thus, the differences between the constant and the variable viscosity model concerning the flow velocity within the fracture system should lead to differences in the temperature field.

The temperature profiles along the two cross-sections depicted in Fig. 5.6 show the differences that arise from the different treatment of the viscosity. If the fractures had no effect on the thermal field, the temperature would be constant in the two sections. For both models, however, deviations from the mean value are visible in the vicinity of the fractures. These deviations are very small at the end of the simulation but are still growing as steady-state has not yet been reached.

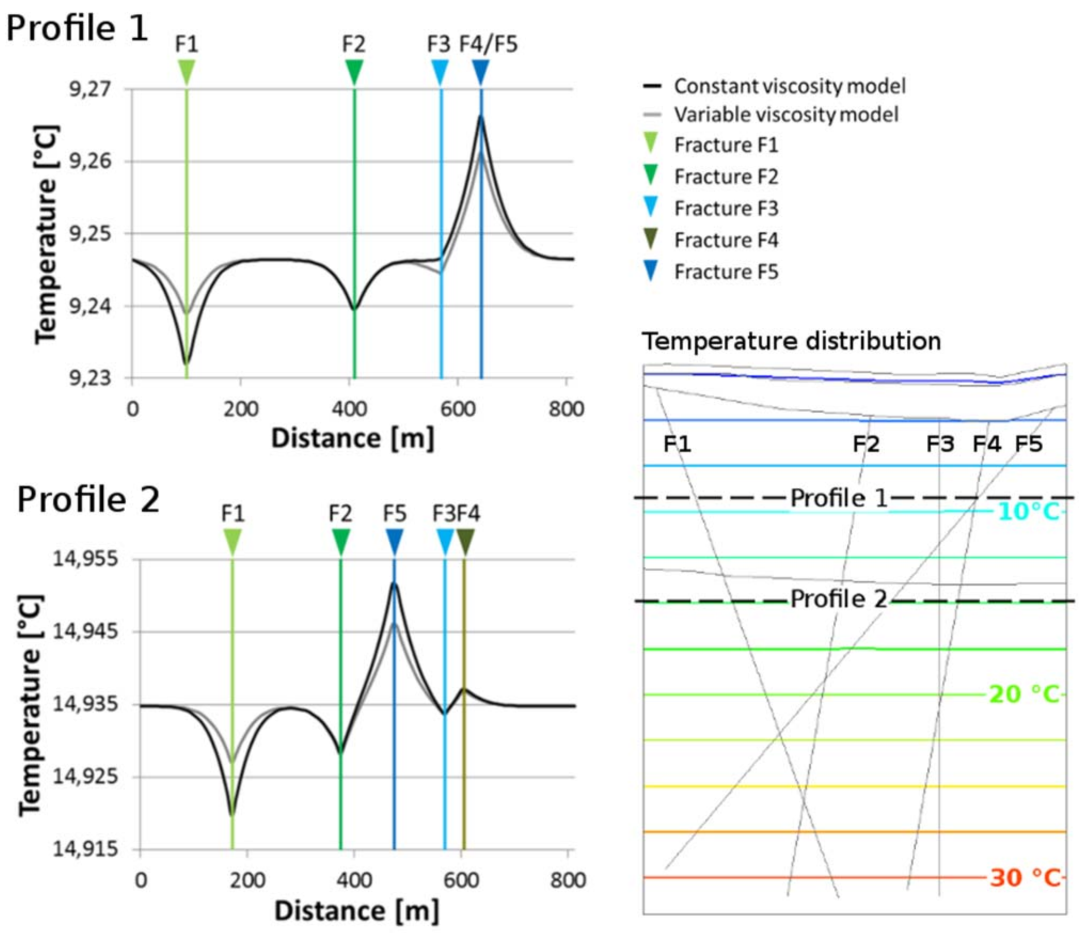


Fig. 5.6 Temperature profiles for the constant and the variable viscosity model

The temperature in the matrix around the fractures is either reduced (F1, F2, F3) or increased (F4, F5) to a varying degree. The underlying mechanism is heat exchange between water in the fracture and the surrounding water-saturated matrix. Along fracture F1 relatively cold water is drawn down from Unit II and getting in contact with increasingly warmer parts of the matrix. The temperature in the matrix is thus controlled by the complex interplay of several processes:

- velocity of water
    - in the fracture
    - exchange with the matrix
- heat flow
    - according to the temperature gradient between fracture and matrix,
    - due to the geothermal temperature gradient.

The same applies in principle also to fracture F5, except that relatively warm water flows through increasingly colder parts of the matrix so that the temperature in the matrix is increased.

Differences between the temperature profiles of the constant and the variable viscosity model thus correlate to the differences in the flow velocities in the fracture system described above. In both sections, the temperature deviation at fracture F1 is two times lower in the variable viscosity model than in the constant viscosity model. At fracture F5 the values for the variable viscosity model are 25 to 20 % less than the values in the constant velocity model. These differences match the deviances found for the flow velocities in fractures F1 and F5.

The results of this study indicate that the temperature dependence of water viscosity has a strong influence on groundwater flow. In addition, the resulting effect on the flow field is hardly to predict, especially when fractures are involved. Exchange of water and heat between fractures and matrix is strongly affected in the investigated model by the accuracy with which the water viscosity is described. Even more complex becomes heat transport in a non-isothermal temperature field. This can obviously become highly relevant for the prediction of solute transport in the near-filed of a repository. It is therefore highly advisable to consider the temperature-dependency of water viscosity in cases of non-isothermal groundwater flow.

5.2 A 3d fracture flow and transport model

5.2.1 Background

The following modelling exercise is based on the work of the TRUE[4] Block Scale Project at the Hard Rock Laboratory Äspö /WIN 02/. As one result of this project the so-called “TRUE Block Scale hydrostructural model” had been defined in the framework of Task 6c of the Äspö Task Force on Modelling of Groundwater Flow and Transport of Solutes.This hydrostructural model had a size of 200 m x 200 m x 200 m and formed the basis for deriving a block scale fracture flow and transport model. The TRUE Block Scale model consisted of 22 deterministic structures[5] with an extension larger than 50 m which were all actually detected in-situ. Smaller ones were called background fractures and can only be described in terms of geostatistics.

Eleven of the deterministic structures were eliminated in the framework of Task 6c for different reasons /DER 03/. The remaining structures were treated as planar. Where actual field data were not available a geostatistical model was set up to fill in the gaps. 19 synthetic 100 m scale structures were generated in just one realisation and then added to the model. These were therefore also called deterministic adding up to a total of 30 multiple intersecting discrete fractures. Additionally, 5660 synthetic background features were also generated and added.

In the framework of the TRUE Block Scale project also the tracer test “C2” was performed /AND 02/. It included injection of an ideal tracer in a certain structure and extraction from a different but hydraulically connected structure. This test formed the basis for Tasks 6D as well as 6E.

The work described here comprises modelling flow in this complex domain as well as the tracer transport across several fractures. Some new conclusions will also be drawn with respect to the flow field in general as well as to the role of the matrix for the tracer migration.

---

[4] Tracer Retention Understanding Experiments

[5] The terms ‘structure’ or ‘feature’ were used to describe the sum of a fracture including fracture fillings as well as the complex changes in the host rock around the fracture (c.f. section 0). For the sake of simplicity the term “fracture” will be used synonymously further on.

5.2.2 Conceptual model

Review of models used in Task 6

All models used in the framework of Task 6 incorporated a discrete fracture network (DFN). This was generally achieved by either one of two basic approaches. One way was to emulate fractures in a 3d-grid as highly conductive zones based on the "smeared fracture approach" (e.g. /FOU 03/). Alternatively, special 1d- or 2d-elements in space were used. Obviously, the first approach works on a reasonably fine grid only with a limited number of fractures.

In case of more complex fracture systems as in Task 6E matrix flow was simply ignored by several models e.g. /CRA 06/, (q.v. /MOR 08/), /CHE 06/. Where the matrix was not modelled explicitly, analytical solutions for diffusion in the matrix were applied e.g. /POT 06/ (q.v. /POT 09/), /DER 06/ (q.v. /UCH 09/). One model from a Japanese team and one from a French team /GRE 06/ was based on an equivalent continuum replacing matrix and background features by a "smeared fracture model". Unfortunately, there exists no citable source for the Japanese model. The only hint was found in /TAN 03/. The procedure for finding parameters for the equivalent continuum is conceptually rather simple. For a 3D volume element an equivalent conductivity *K*$_e$ is calculated as the volume weighed mean of fracture and matrix permeabilities:

$$K_e = \frac{K_m V_m + \sum_i^n K_i V_{f\,i}}{V_e} \qquad (5.2)$$

Using the data from the semisynthetic hydrogeological model presented in /DER 03/ this method yielded an equivalent conductivity for the continuum of $K_e = 10^{-9}$ m/s.

In some cases also the background fractures were at least partly ignored /GRE 06/ (q.v. /GRE 08/). In /SVE 06/ fractures below a predefined cell size were only considered as storage volumes but did not take part in the flow.

Adopted approach

Fractured rock is often conceptualised as a so-called "intact rock" that is interspersed with stochastically distributed fractures. Other than in the Canadian Shield, where the granite can be sparsely fractured (e.g. /KUZ 08/), the granite at Äspö proves to be comparatively highly fractured. It is even commonly assumed here that fractures exist on all length-scales implying that the hydraulically relevant rock porosity is basically made up of micro fractures (e.g. /DER 03/).

Obviously, there is a lower limit to the size of the fractures that can be incorporated in any of these models. However, the minimum size of the considered fractures can be considerably less in a stochastic fracture network model without a matrix than in a case where the matrix is included.

It is common practice to distinguish between large fractures and the remaining set of smaller fractures, also called "background fractures". There is no general definition of background fractures, though. Basically, they are defined by their size (e.g. /DER 03/, /BLE 11/) or by their transmissivity (e.g. /CVE 06/, /BLE 13/) or a combination. /CVE 10/ add that background fractures typically come in numbers so that a discrete description of them is only possible by means of geostatistics. Since the background fractures are less transmissive than the large fractures[6] and more evenly distributed they can arguably be replaced by a second continuum that effectively increases the matrix permeability (e.g. /BLE 11/). It has to be noted, though, that outflow from the rock into the probing boreholes at the BRIE-site varied roughly over two orders of magnitude already over a distance of just 1.5 m /BOC 13/. This observation suggests that relevant water-bearing fractures at Äspö may become scarce on a scale of a few metres or less /KRO 16a/.

In the light of these considerations a model concept has been developed in the framework of Task 8 where the undisturbed matrix and the background fractures are combined to one continuum with effective flow parameters /KRO 16a/ as proposed earlier by /GRE 06/. The large fractures, however, are represented discretely by 2d-elements in space. The following calculations of flow and transport according to Task 6 are based on this concept. As an approach to the conceptual model outlined in /DER 03/ only the 100-m scale structures are considered to be large.

[6] Note that a direct proportionality between fracture size and transmissivity has been established for the granite at Äspö (e.g. /BOC 13/).

However, there is little information coming from the Task 6 descriptions that can be used to derive the effective flow parameters for the continuum. Main reason for this is that the focus of Task 6 had lain on the modelling of tracer tests as well as the appropriate description of the influence of microstructural effects in the fractures on the transport behaviour. Characterising the flow field played a secondary role. Data from /KRO 16a/ are thus adopted for the Task 6 model where applicable.

5.2.3 Fractures

Structure

On a micro scale level of observation a fracture is a rather complex structure as shown in Fig. 5.7 so that in principle no fracture is identical to another one. Basically, the structures investigated in the TRUE Block consist of open fractures with mineral coating. They are embedded in a comparatively large altered zone and often include subparallel fractures. Some fractures contain additionally breccia[7] and fine-grained fault gouge[8]. Also adjacent cataclasit[9] and mylonite[10] can be found. The thickness data for the different structure components are compiled in Tab. 5.3. Despite these complexities, however, the fractures are conceptualised as planar parallel plates for the purpose of modelling fracture flow and transport.

[7] Fault breccia: a medium- to coarse-grained cataclasite containing >30% visible fragments /WIK 10/.Millimetre to centimeter (> 2 mm) sized pieces of altered wall rock/cataclasite and/or mylonite. The chemical and mineralogical composition is usually similar to that of the (altered) wall rock /POT 02/.

[8] Fault gouge: an incohesive, clay-rich fine- to ultrafine-grained cataclasite, which may possess a planar fabric and containing <30% visible fragments. Rock clasts may be present /WIK 10/. Fragments and mineral grains (≤ 2 mm) of altered wall rock and secondary minerals (clay minerals and calcite) /POT02/.

[9] Cataclasite: a fault rock which is cohesive with a poorly developed or absent planar fabric, or which is incohesive, characterised by generally angular clasts and rock fragments in a finer-grained matrix of similar composition /WIK 10/.

[10] Mylonite: a fault rock which is cohesive and characterized by a well developed planar fabric resulting from tectonic reduction of grain size, and commonly containing rounded porphyroclasts and rock fragments of similar composition to minerals in the matrix /WIK 10/.

Tab. 5.3 Extent of structure components

Rock type	Extent [cm]
Intact wall rock	–
Altered zone	10 - 20
Cataclasited$_{cat}$	2
Fault gouge d_g	0.5
Fracture coating d_c	0.05

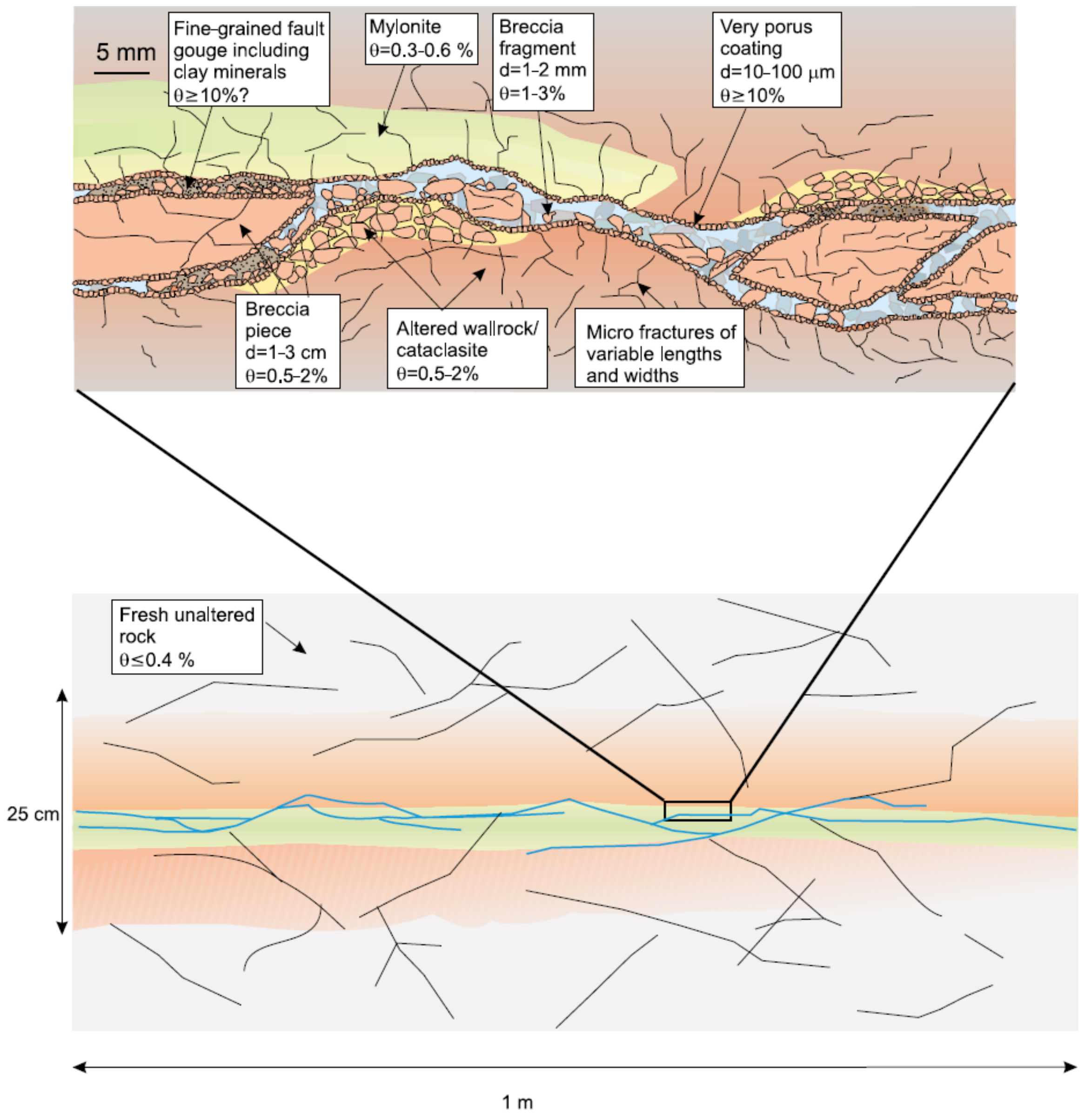

5 mm
Fine-grained fault gouge including clay minerals θ≥10%?
Mylonite θ=0.3-0.6 %
Breccia fragment d=1-2 mm θ=1-3%
Very porus coating d=10-100 µm θ≥10%
Breccia piece d=1-3 cm θ=0.5-2%
Altered wallrock/ cataclasite θ=0.5-2%
Micro fractures of variable lengths and widths
Fresh unaltered rock θ≤0.4 %
25 cm
1 m

Fig. 5.7 Generalised conceptual model of a typical conductive structure; from /WIN 02/

Geometry of the fracture system

Coordinates are generally given in relation to an Äspö-specific reference point. The centre of the cube-shaped model domain is defined in these local coordinates by

- easting: 1900 m
- northing: 7170 m
- elevation: -450 amsl

Real as well as synthetic 100-m scale structures are defined by three, four or five corner points given also in terms of the Äspö reference system.The names of the structures are taken over from /DER 03/, definitions are compiled in Tab. C.1 and Tab. C.2 in Appendix C. Also included are the corner coordinates corrected for fitting into the cubic model domain where applicable. Note that clipping produced further pentagonal and triangular structures in some cases. Note further that the assumption of planar fractures introduced an aberration of the fracture location of up to 10 m compared to the real fracture system.

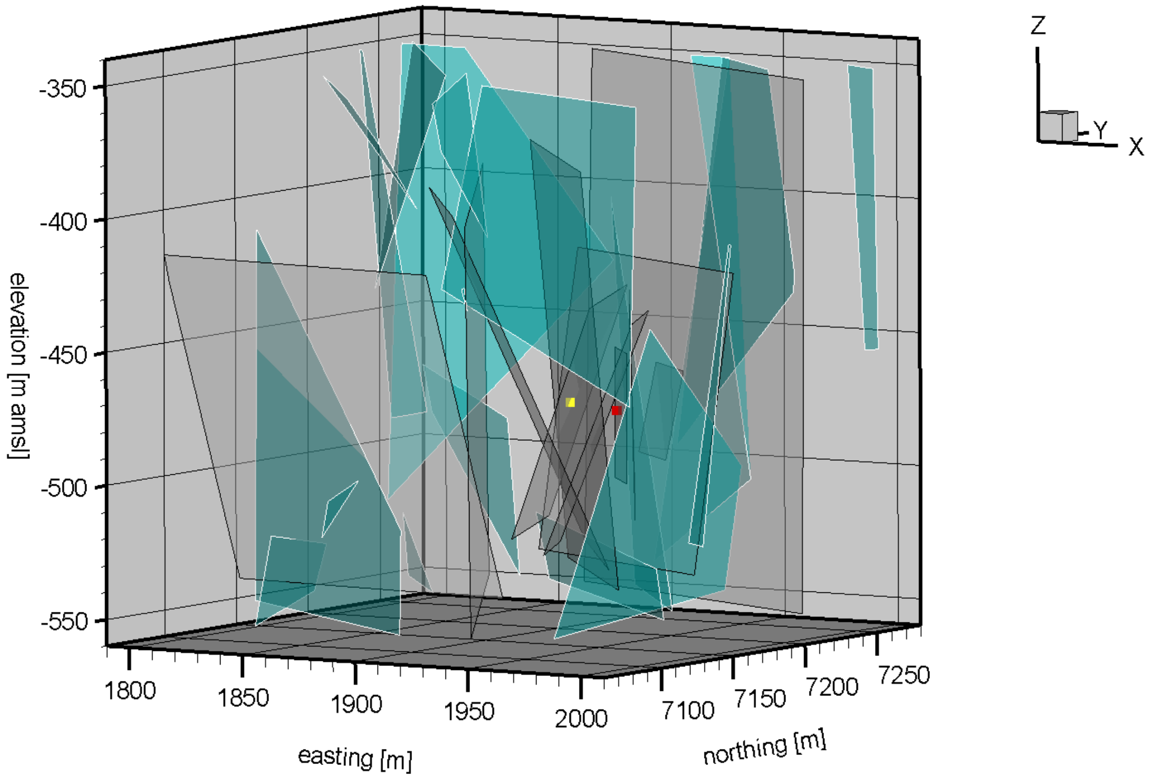


Fig. 5.8 100-m structures in the 200-m model block; real in grey, synthetic in green

5.2.4 Hydraulic properties

It was possible to assign characteristic hydraulic parameters to each fracture in the TRUE block by means of in-situ tests and modelling exercises /DER 03/. Resistance to flow is given in terms of transmissivities. Also listed are the transport relevant apertures so that permeabilities can be calculated. These three properties are compiled in Tab. C.3. Variability of permeability and aperture are illustrated in Fig. 5.9 and Fig. 5.10.

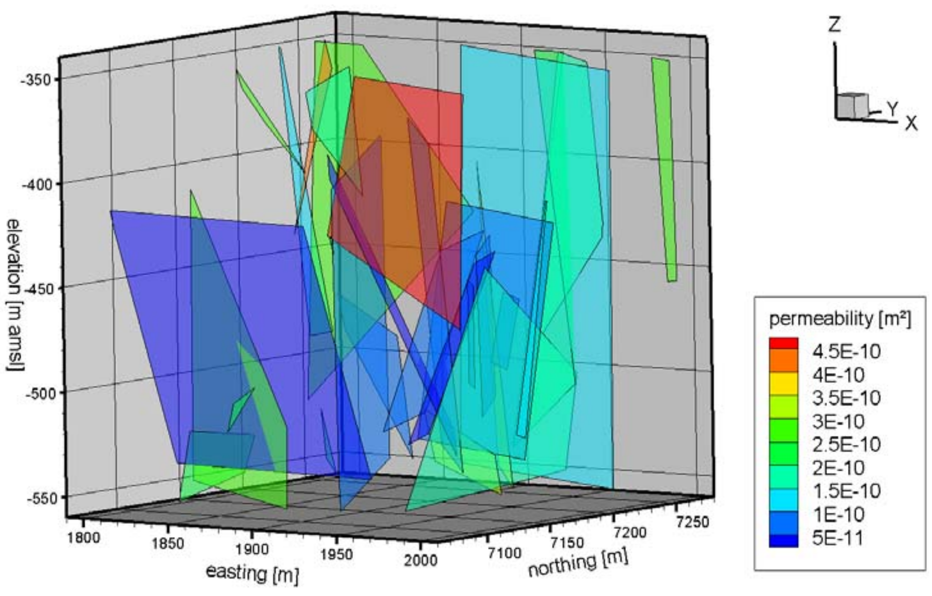


Fig. 5.9 Permeability of the fractures

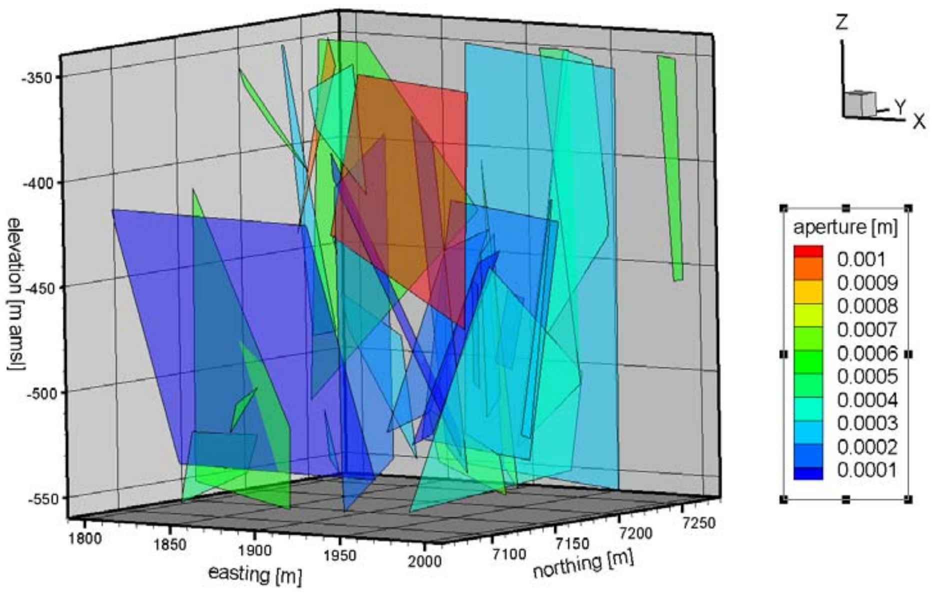


Fig. 5.10 Aperture of the fractures

An impression of the system of background fractures as well as the related transmissivities gives Fig. 5.11. It strongly supports the notion of representing this fracture system by an equivalent continuum. While the data from /GRE 06/ indicated a permeability of 10^{-16} m² for the equivalent continuum representing matrix and background fractures, modelling Task 8 provided a related value of 10^{-17} m² /KRO 16a/, /KRO 16b/. Adopted was the latter value for the modelling as it was confirmed by two other participants of Task 8. Also adopted was the suggested porosity value of Φ = 0.005.

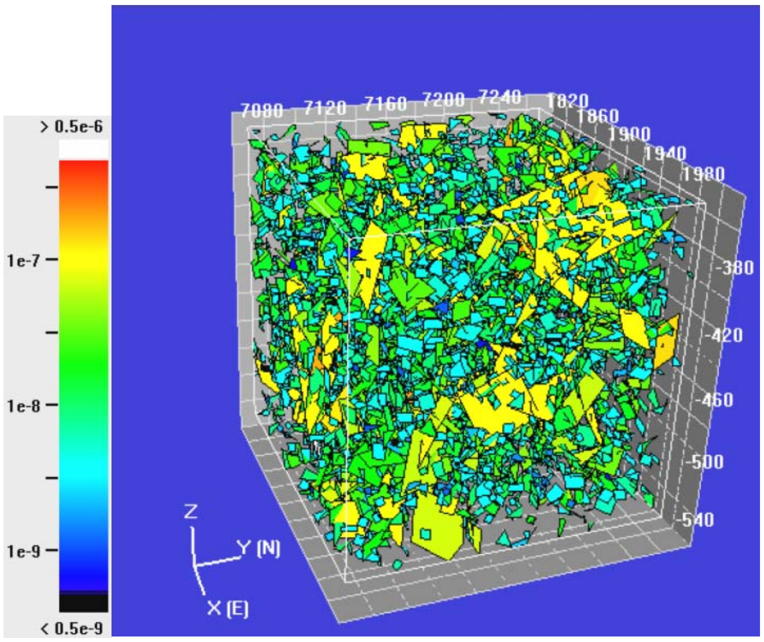


Fig. 5.11 Transmissivities of the background fractures; from /DER 03/

5.2.5 Hydraulic boundary conditions

Boundary conditions were given in terms of piezometric heads across the surface of the 200 m cube. These heads were calibrated in such a way that modelling the underground openings like tunnels and boreholes is not necessary /DER 03/. A characteristic head value is given for each 22 m x 22 m square on the surface. In the two plots in Fig. 5.12 these values are already interpolated. The original roughness can be appreciated, though, by the regular 22 m grid that is also shown.

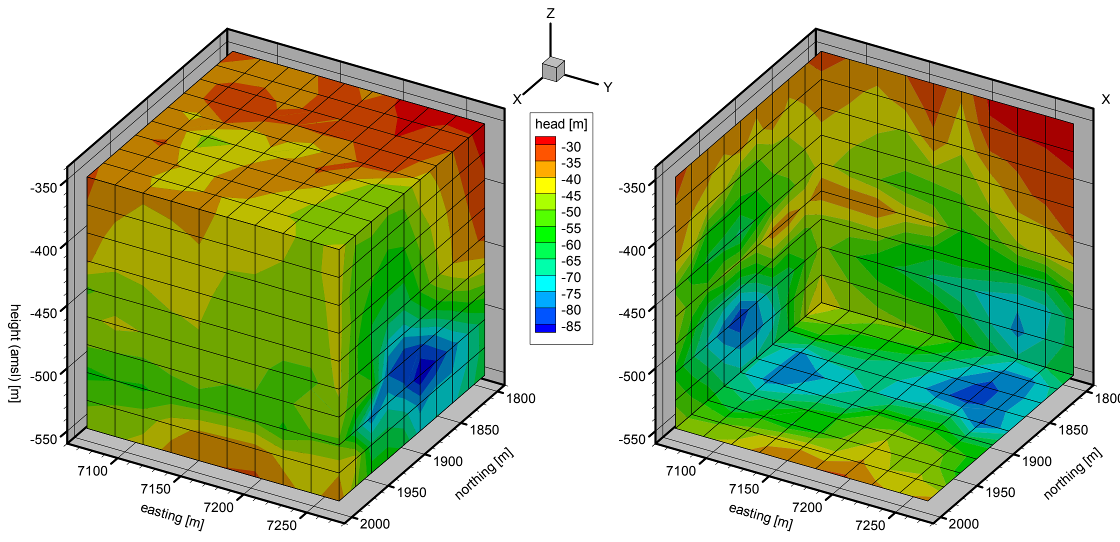


Fig. 5.12 Hydraulic heads on the surface of the 200-m model block

5.2.6 Tracer test

The tracer test "C2" included a source in structure 23 at (1929.741, 7194.840, -476.100)[11] and a sink in structure 21 at (1914.628, 7186.294, -473.065). Direct distance between source and sink amounted to 17 m but the actual flow path covering structures 23, 22, 20, and 21 (see Fig. 5.13) had a length of about 97 m /AND 02/.

According to /DER 03/ the injection rate was 1.5 10^{-7} m³/s and the pumping rate was 3.27 10^{-5} m³/s. These data were later modified to 1.67$10^{-7}$ m³/s and 3.25 10^{-5} m³/s, respectively /DER 06/.

The concentration of the injected tracer was given in terms of activity per mass (see Appendix Tab. C.4). The total amount of the injected activity amounted to $1.71 \cdot 10^{8}$ Bq with an uncertainty of $1.89 \cdot 10^{6}$ Bq. As a result of the tracer test the measured extraction concentration over time is given (see Appendix Tab. C.5). The data is plotted in Fig. 5.14.

[11] Äspö coordinate system

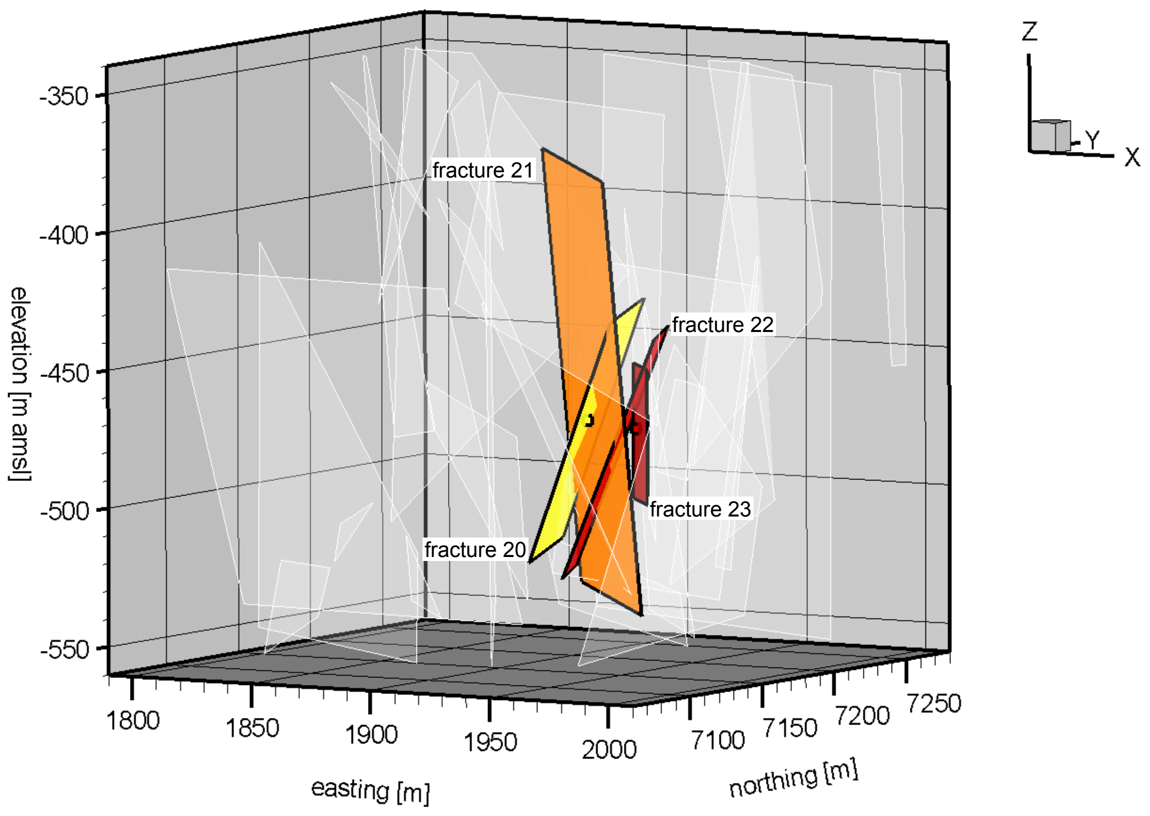


Fig. 5.13 Structures involved in the C2-tracer test

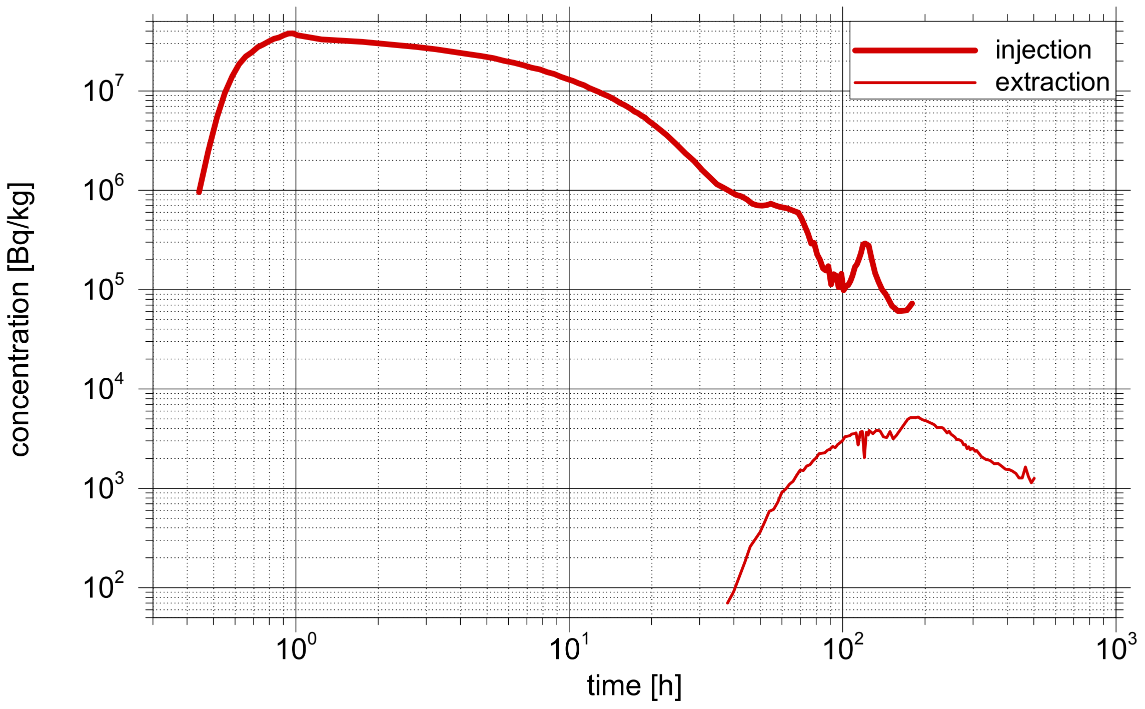


Fig. 5.14 Activity concentrations for the tracer in the C2-test

The apparent or effective diffusion is taken here to be a product of the coefficient for free diffusion in water and a so-called “formation factor” which includes effects from porosity, tortuosity, constrictivity etc. The referring data with respect to each material found in the fractures is compiled in Tab. 5.4. However, since the parallel plate model is applied here a formation factor of $2\cdot10^{-4}$ is adopted for the fractures and $7.5\cdot10^{-5}$ for the matrix in the calculations.

Tab. 5.4 Transport relevant parameters in different materials;from /DER 03/

	Fracture coating	**Fault gouge**	**Cataclasite**	**Altered zone**	**Unaltered wall rock**
Porosity (%)	5	20	1	0.6	0.3
Formation factor, F	6.2E-03	5.6E-02	4.9E-04	2.2E-04	7.3E-05

5.2.7 Results

Pressure

The pressure distribution prescribed on the model boundary was rather coarse in comparison to the size of the elements representing the matrix. In order to provide a somewhat smoother distribution for modeling purposes the pressure data for the boundary was processed by an inverse distance weighing scheme. It appears, though, that the real data distribution was too complex to be reconstructed by this procedure. Either the orthogonal pattern from the provided data (c.v. Fig. 5.12) remained to be visible or certain features of this distribution got lost. Fig. 5.15 shows the adopted compromise. For reference the used grid is depicted as well.

The resulting pressure distribution in the fracture system is depicted in Fig. 5.16. Clearly visible is a considerable pressure drop in the middle of the system apparently relating to the extraction point in the C2-test. This observation is underpinned by the pressure distribution in the matrix in the vicinity of this location as shown in Fig. 5.17

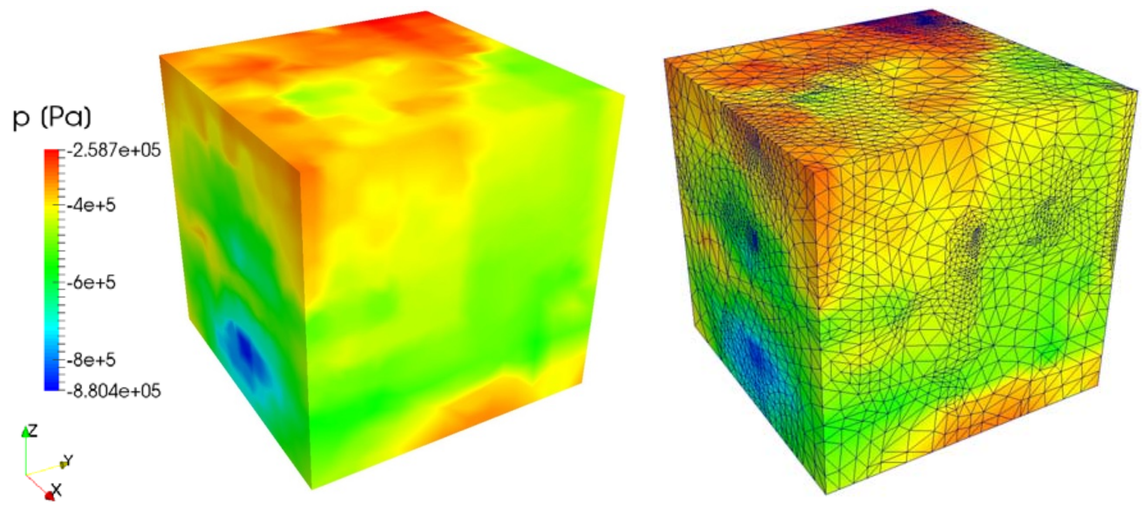


Fig. 5.15 Pressure distribution and numerical grid on the model boundary

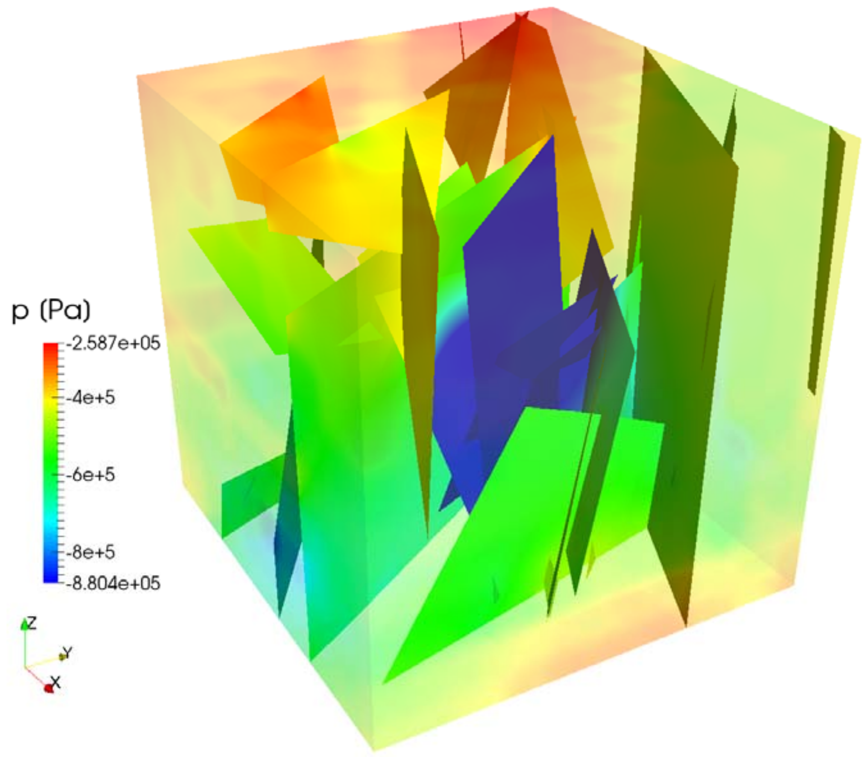


Fig. 5.16 Pressure distribution in the fractures

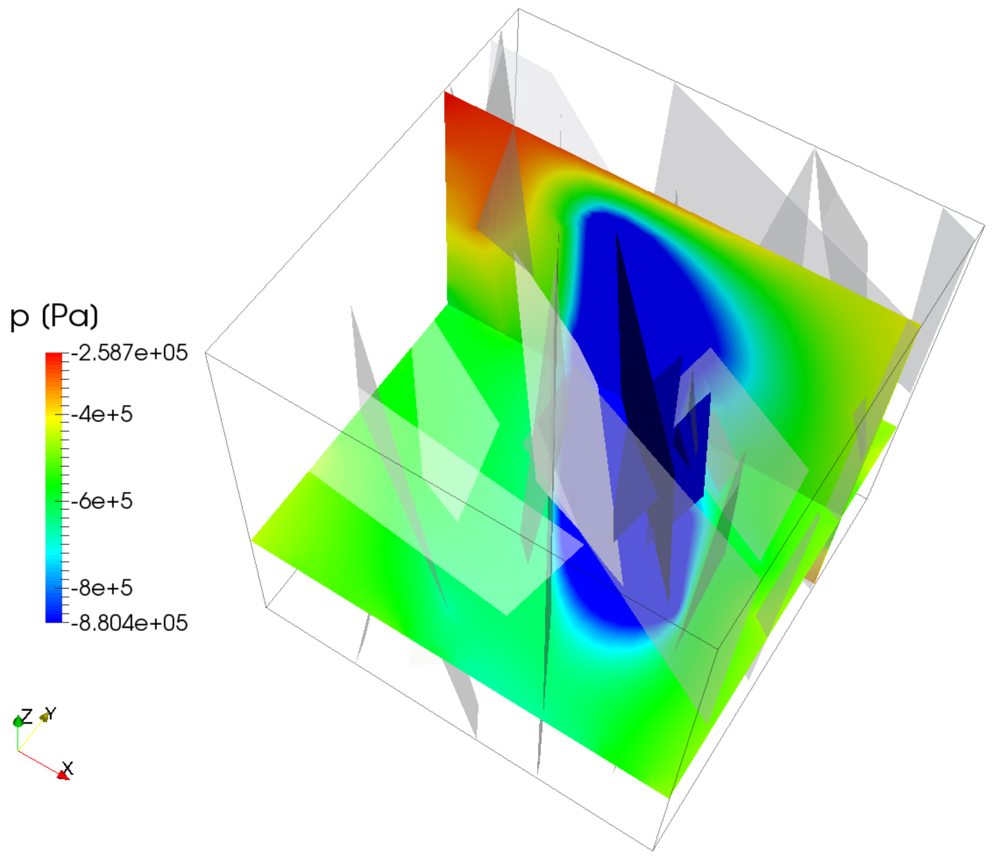


Fig. 5.17 Pressure distribution in the matrix at the C2-test

Finally confirmed is this conclusion by a closer look at fractures 23, 22, 20, and 21 which are immediately affected by the C2-test. Injection is located in fracture 23 and extraction in fracture 21. Plotting the related pressure field at a different scale as in the left graphic of Fig. 5.18 reveals that the pressure drop is indeed caused by the extraction of water in the C2-test.

Almost not visible by comparison is the slight pressure increase due to the injection of the tracer in the right plot in Fig. 5.18 showing the pressure at a yet different scale. This is of course a consequence of the fact that the extraction rate is about 200 times higer than the injection rate.

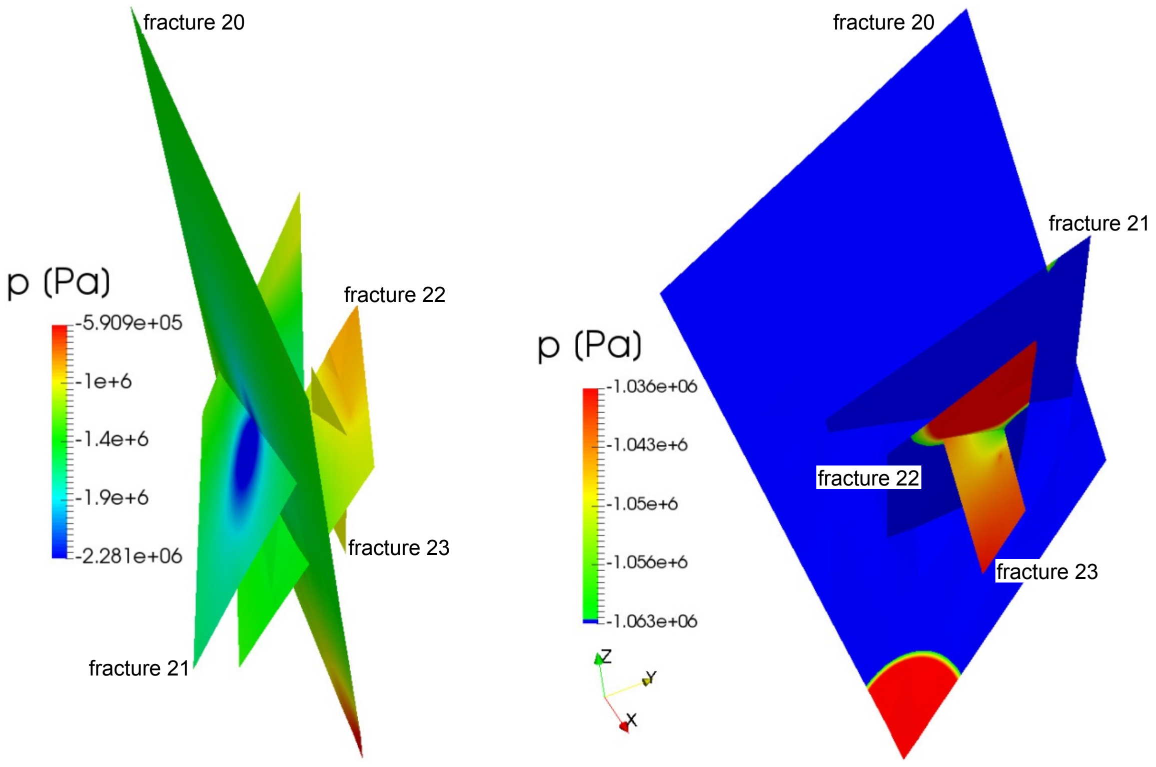


Fig. 5.18 Two views of the pressure distribution in the fractures related to the C2-test at different scales

Velocity

While the pressure field looks rather straight forward the resulting velocity field is quite complex because of the hydraulic interaction of the fractures. Exemplarily for the intricacy of the whole fracture flow system a closer look is taken at the flow fields in the four fractures immediately involved in the C2-test. However, a much better understanding can be gained if the fractures intersecting this "core group" of four fractures are also taken into account. The list of fractures connecting to the fractures of the core group shows already a certain complexity:

- fracture 23 (including injection) connects to fracture 22,
- fracture 22 connects to fractures 23, 20, 06, 07, and 13,
- fracture 20 connects to fractures 22, 21, 07 and 13, and
- fracture 21 (including extraction) to fractures 20, 07, and 13.

Note that fracture 22 does not penetrate fracture 07 but just has a touching line in common with fracture 07.

All fractures from the list are depicted in Fig. 5.19. For the sake of clarity fractures 23, 22, 20, and 21 are plotted in grey while fractures 06, 07, and 13 are showing the hydraulic pressure.

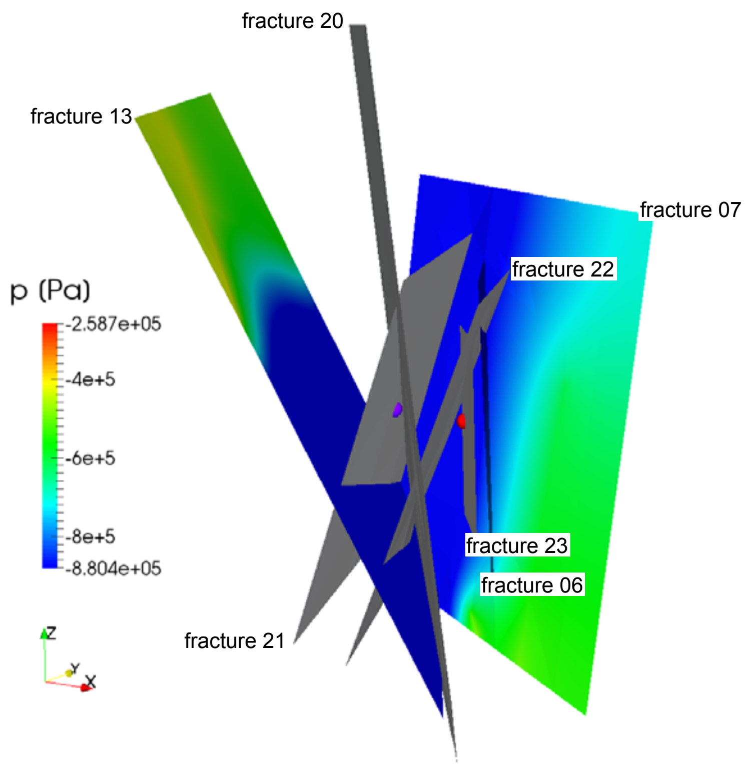


Fig. 5.19 Pressure distribution in the fractures cutting through the fractures related to the C2-test

In the following a series of plots depicting the velocity field in a fracture from the core group as well as a second fracture intersecting the first one is shown to elucidate the velocity field in the 3d-fracture system. Injection and extraction points are represented by a red and a blue sphere, respectively, where applicable. Note that a velocity vector is plotted for each fracture element which leads in some cases to a local concentration of vectors that is simply caused by a local grid refinement.

The tracer carrying water is injected into fracture 23 at the location marked by the red sphere in Fig. 5.20. Fracture 23 is not intersected by any other fracture than fracture 22 which subsequently takes up much of the injected tracer. The flow field in fracture 23 is therefore basically determined by the injection, by fracture 22 and of course by the surrounding matrix. Clearly visible is the local increase of velocity at the source. The resulting increase of velocity at the intersection with fracture 22 is small, though, compared to the flow coming from fracture 22. A considerable decrease of tracer concentration due to mixing with fresh water can thus be expected at this intersection.

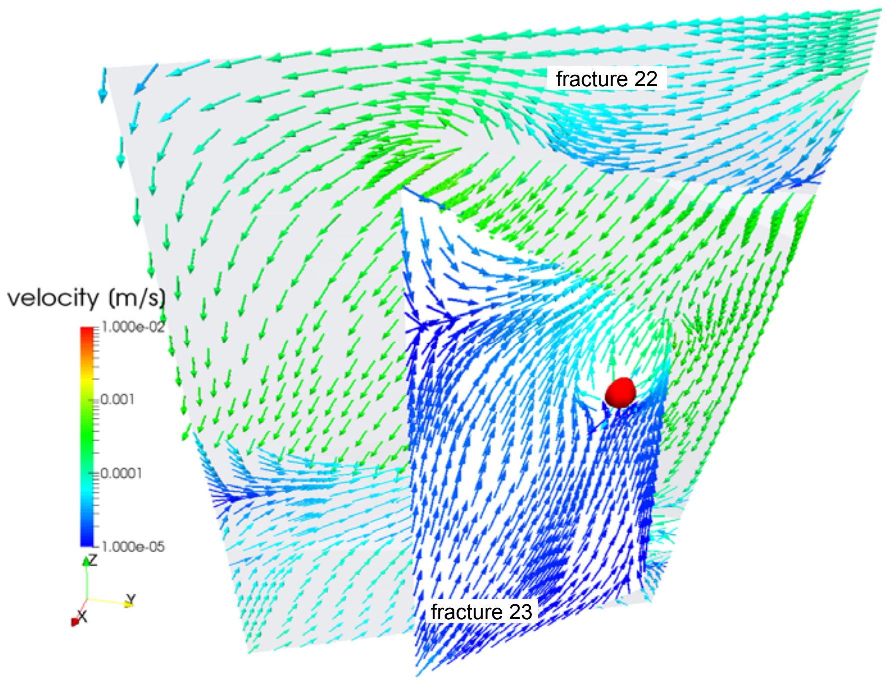


Fig. 5.20 Velocity in fractures 23 and 22

Contrary to fracture 23 there are several intersections of other fractures with fracture 22. Fig. 5.21 to Fig. 5.22 show the velocity field of fracture 22 together with one of these other intersecting fractures. In Fig. 5.21 fracture 22 is depicted together with fracture 20 which is the next fracture along the main flowpath of the tracer. Some mixing is again to be expected at the intersection of fractures 22 and 20. Also easily recognisable in Fig. 5.21 is the influence of fractures 06 and 07 on the flow field in fracture 22. Both fractures are significantly feeding water into fracture 22 which can be also seen in Fig. 5.22 and Fig. 5.23. Only fracture 13 seems not to influence fracture 22 as depicted in Fig. 5.24.

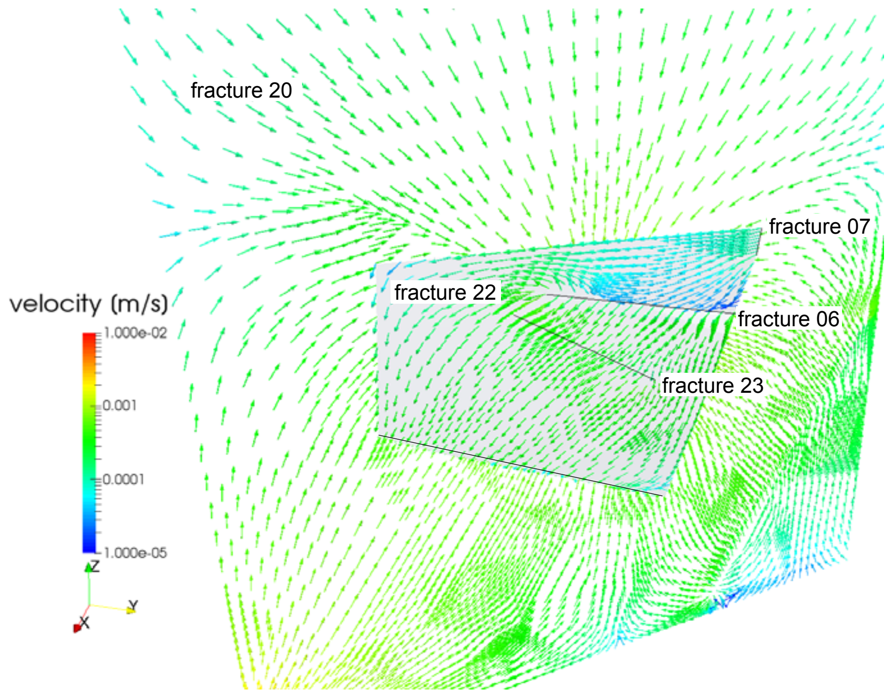


Fig. 5.21 Velocity in fractures 22 and 20

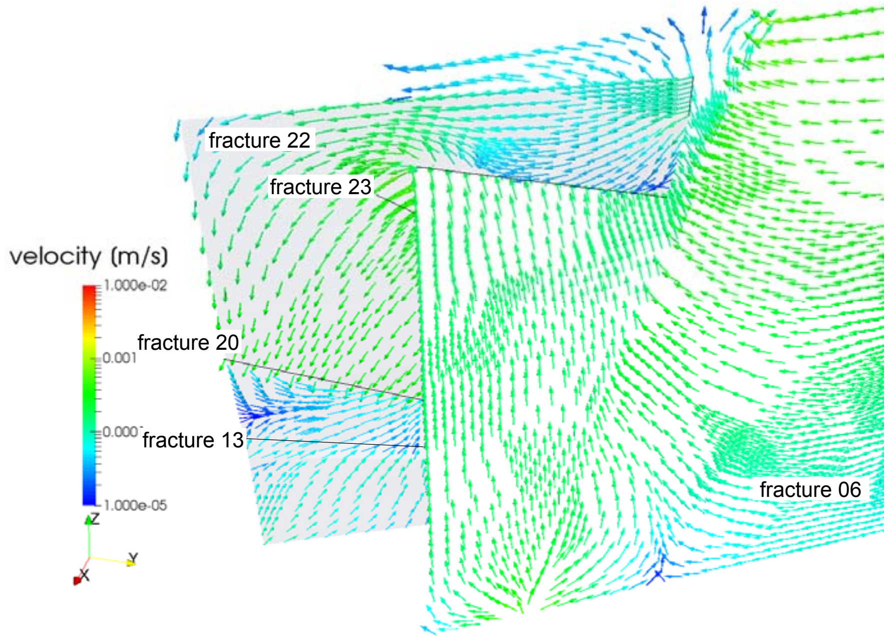


Fig. 5.22 Velocity in fractures 22 and 06

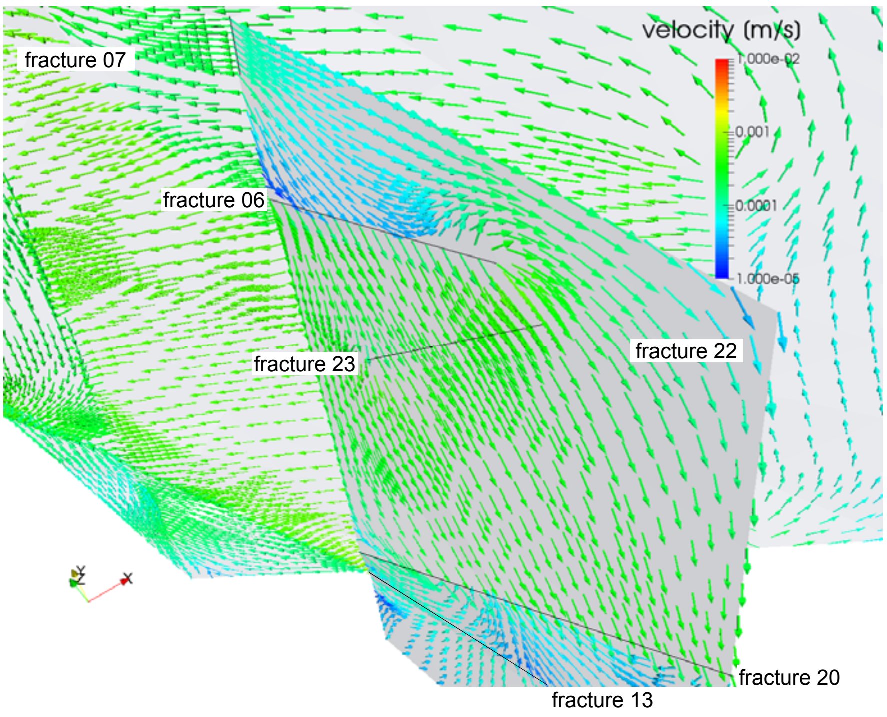


Fig. 5.23 Velocity in fractures 22 and 07

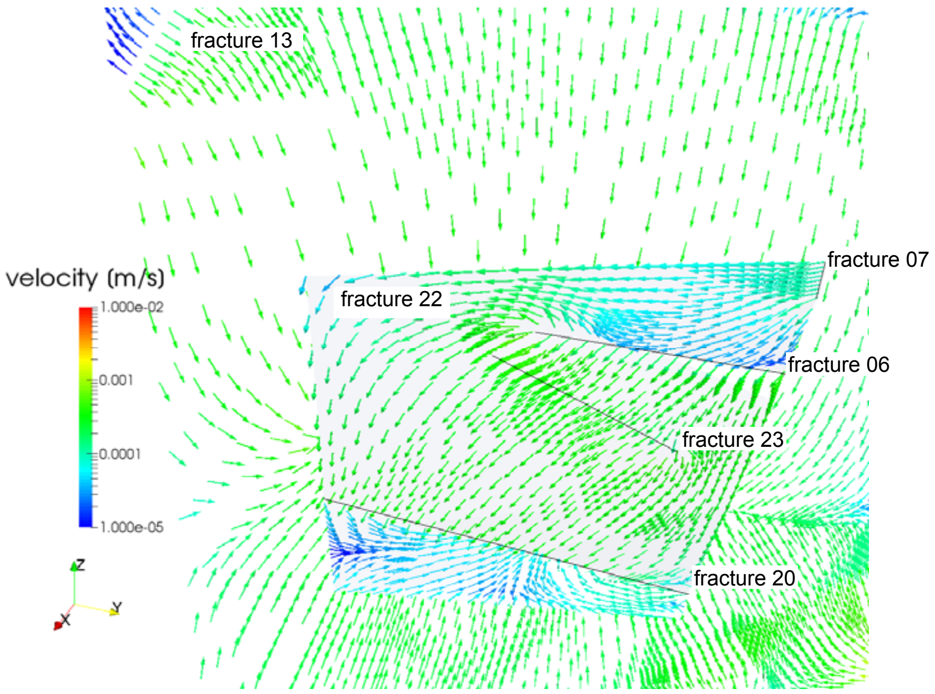


Fig. 5.24 Velocity in fractures 22 and 13

Fracture 22 is the second to last fracture before the tracer reaches the extraction point in fracture 21. Fig. 5.25 shows both fractures but fracture 21 is seen from the side opposite to the sink. The influence from the sink on the velocity field in fracture 21 is clearly visible by reference to the flow pattern close to the middle of the intersection line. Similarly, the water coming from the direction of fracture 20 appears to converge already towards the same location at the intersection with fracture 21.

Water flow from fracture 22, though, seems to be little as it has no strong influence on the flow in fracture 20. Yet another considerable dilution of the tracer plume is therefore to be expected at this intersection.

Fig. 5.25 seems to indicate also only a minor influence of fracture 13 on the flow of fracture 20 while Fig. 5.26 shows a contribution to fracture 20 that is in the same order as flow in fracture 20. An increase in flow by a factor of 2 or 3 gets apparently lost in the logarithmic colour scale for the velocity.

Flow is only significantly changed across the intersection with fracture 07 which is confirmed by Fig. 5.27.

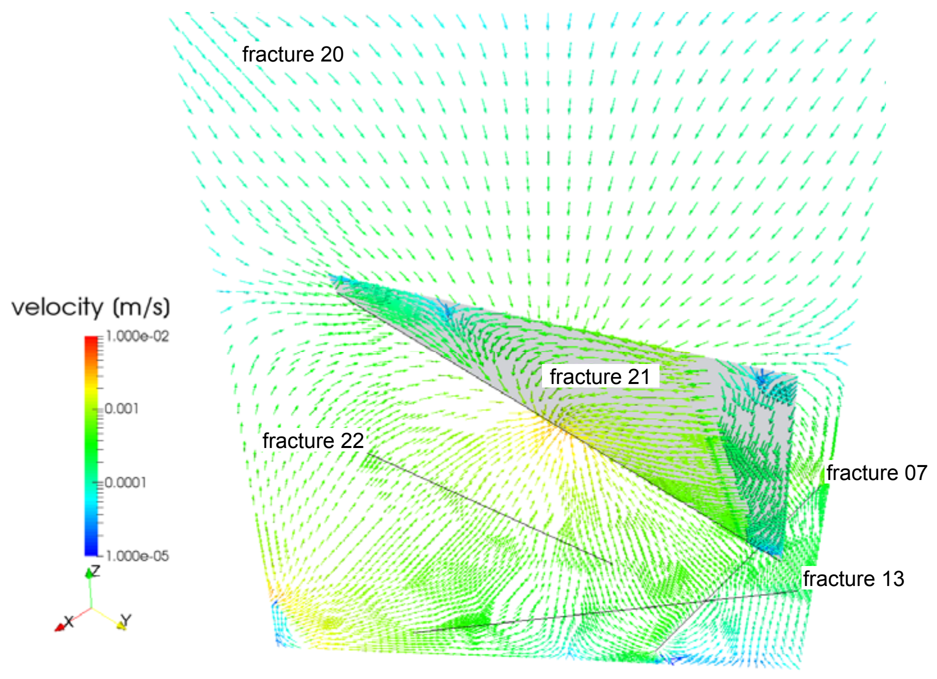


Fig. 5.25 Velocity in fractures 20 and 21

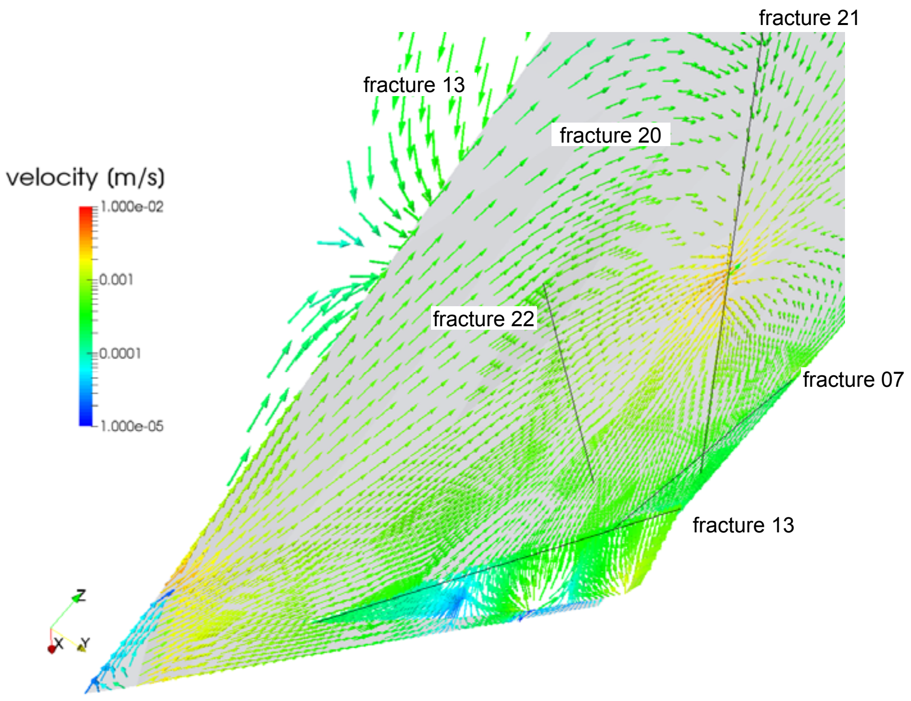


Fig. 5.26 Velocity in fractures 20 and 13

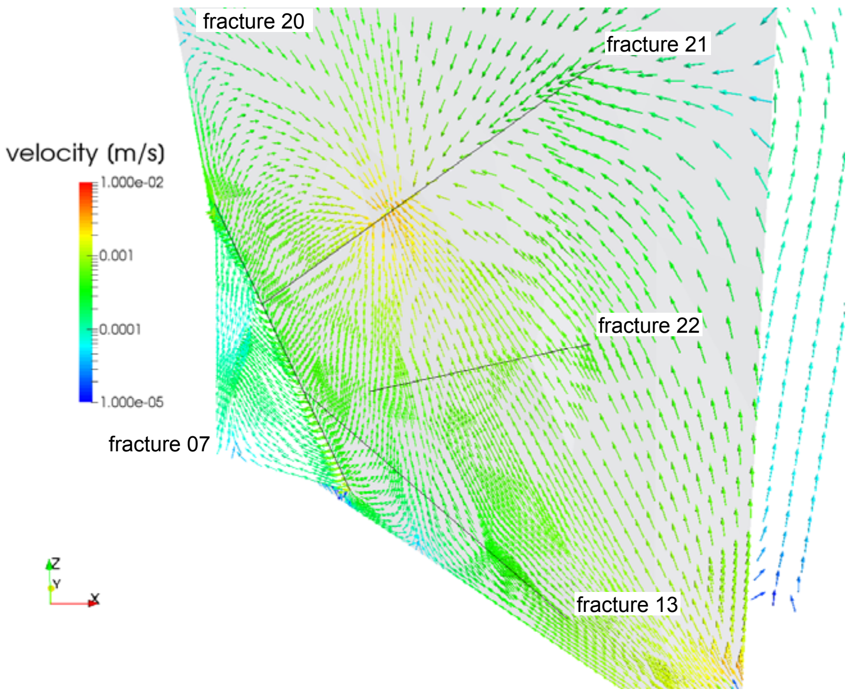


Fig. 5.27 Velocity in fractures 20 and 07

Finally, flow in fracture 21 is not surprisingly controlled by the extraction at the point marked with the blue sphere as shown in Fig. 5.28. Again only fracture 07 delivers enough water to change the flow pattern in fracture 21 at the intersection significantly (see also Fig. 5.29).

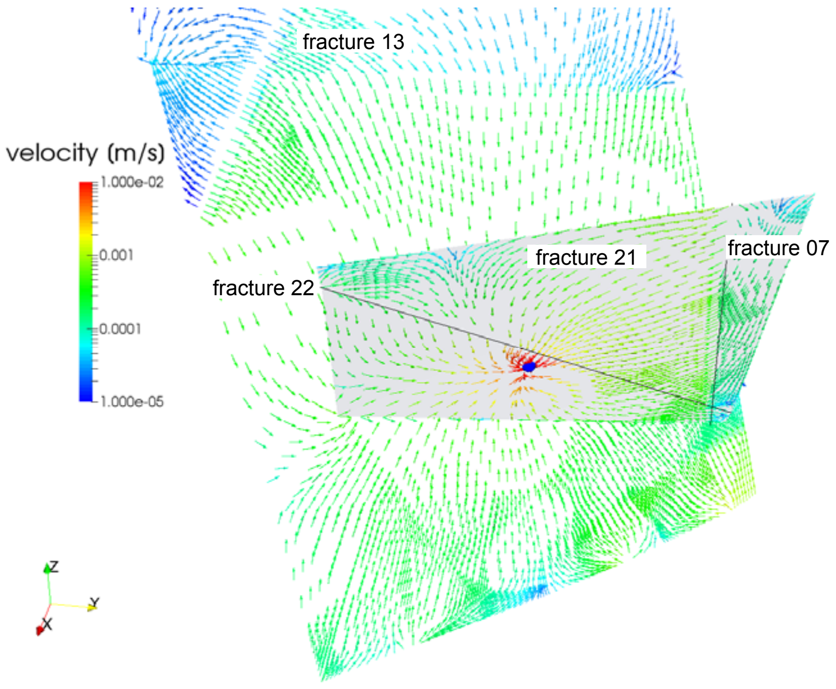


Fig. 5.28 Velocity in fractures 21 and 13

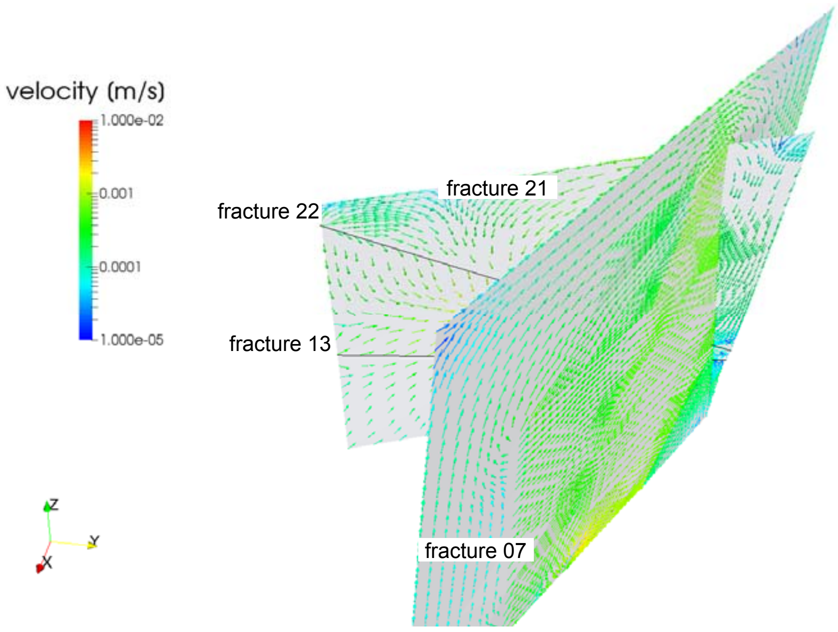


Fig. 5.29 Velocity in fractures 21 and 07

Tracer concentration

The transport simulation was checked against the measured breakthrough curve of the tracer. However, breakthrough was not satisfyingly reproduced by the model – neither in terms of breakthrough time nor in terms of peak value – when using the hydraulic input parameters described in the previous subsections. The tracer plume took too long to reach the extraction point and did not reach the measured concentration.

Since too much spreading in the matrix and too little velocity in the fractures was suspected the porosity values for matrix and fractures were reduced. A satisfying fit was achieved reducing the formation factor for the matrix by a factor of 30 and the formation factor for the fracture by a factor of 10. Note that a related increase of the permeability would have had the same effect except that higher fluid and solute flow rates would have been involved. The resulting breakthrough curve with reference to the measured data is shown in Fig. 5.30. The results presented in the following refer to the model with modified porosities.

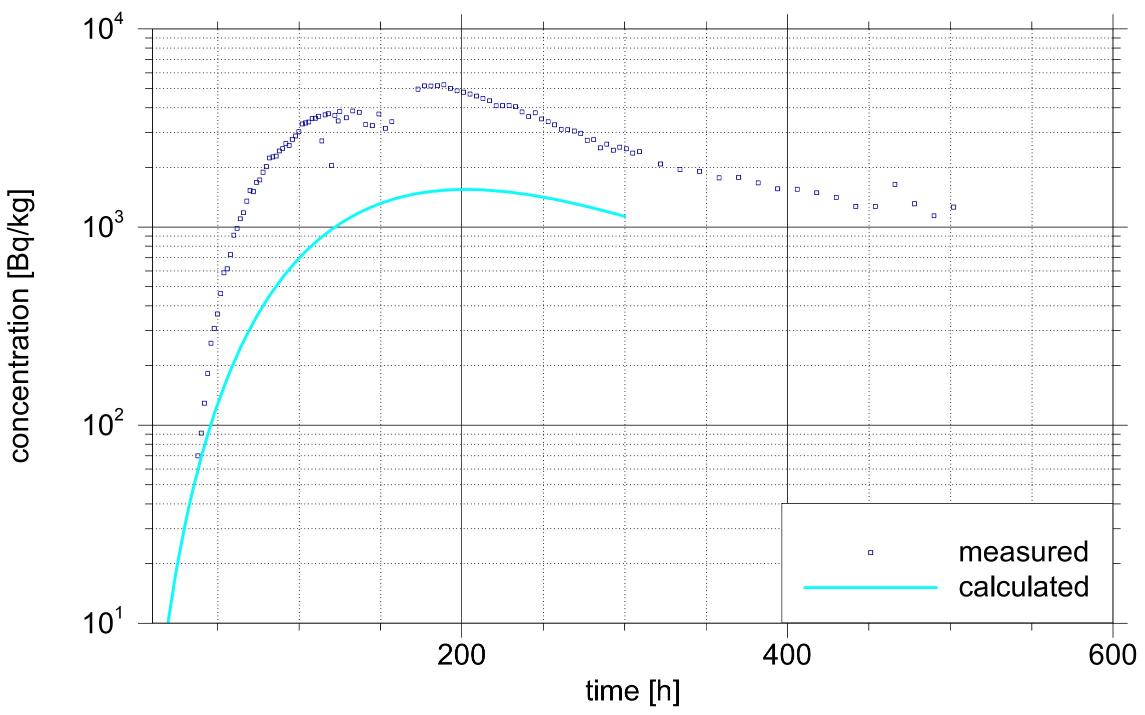


Fig. 5.30 Measured and calculated breakthrough curve

A comparison of the breakthrough curve with the injection curve shows essentially a decrease in the peak concentration and a phase shift (c.p. Fig. 5.14). These two properties reflect the effects of the travel time between injection and extraction point and the effect of mixing with fresh water. Mixing occurs always on a microscopic scale due to the hydrodynamic dispersion leading to a certain spreading of the tracer plume. A second mixing effect can be observed in the C2-test when the concentration plume is crossing an intersection of two fractures thus mixing the solution with fresh water.

While the hydrodynamic dispersion smoothens the concentration distribution and is therefore numerically favourable, mixing at fracture intersections is quite difficult to reproduce in a numerical model. The problem that arises in this situation is exemplarily illustrated in Fig. 5.31. Assuming the same flow rate in two identical intersecting fractures and assuming further a tracer plume with the concentration c_0 arriving at the intersection from one fracture while the second fracture carries no tracer results in a mixing concentration of ½c_0. The model must therefore represent three different stepwise changing concentration values at the intersection, c_0 upstream of the intersection in one fracture, 0 in the other fracture upstream of the intersection and ½c_0 in both fractures downstream of the intersection. These step functions can lead to severe oscillations in the solution of numerical schemes that are based on continuous basis functions (e.g. /KRO 91/) if they persist as part of the exact solution and if hydrodynamic dispersion is insufficient to dampen the oscillations. Since the chosen dispersion lengths amount only to 10 cm thus leading to high Peclet-numbers it is of specific interest how d^{3}f++ handles this situation.

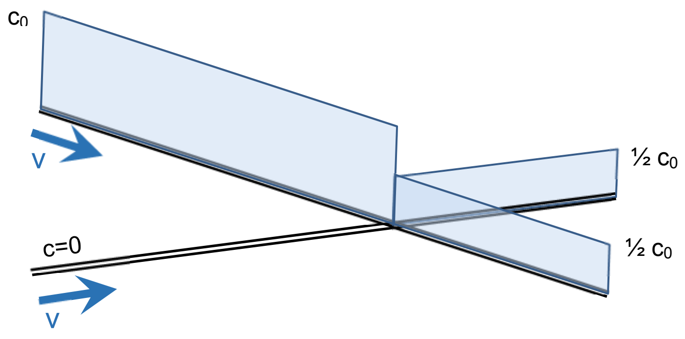


Fig. 5.31 Illustration of concentration changes at a fracture intersection

Additionally, the numerical model faces a similar problem where the streamlines converge towards the sink. Without hydrodynamic dispersion the boundary of the tracer plume would be characterized by a step function orthogonal to the flow direction. Even more problematic for the numerical scheme is calculating the concentration distribution at the very sink because a step function is also required in the direction of flow. This situation is illustrated by Fig. 5.32. Ususally, a sufficient amount of hydrodynamic dispersion is required to minimize numerical problems from step functions in the solution which can be interpreted as introducing longitudinal and transverse mixing.

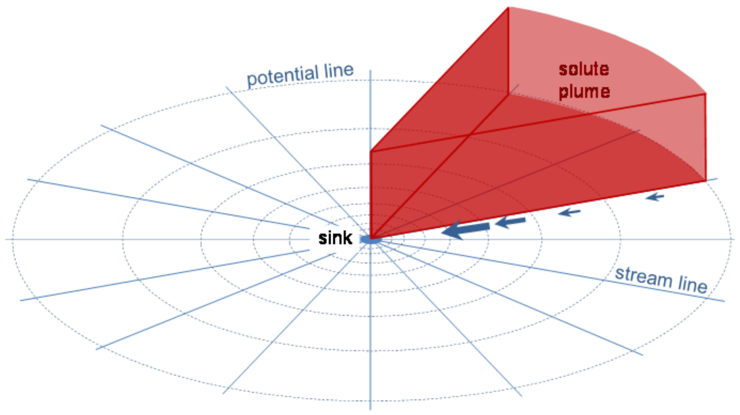


Fig. 5.32 Illustration of a solute plume migrating towards a sink without hydrodynamic dispersion

An example for the consequences of this sort of problem is given in Fig. 5.33 showing the model results for a steady-state gas saturation distribution. In a dipole test gas had been injected and extracted in a fracture at the HRL Äspö /KUL 02/. While the gas in the model forms a plume that correlates nicely with the depicted flow field, a curious saturation peak is visible close to the sink. It shows higher values than the maximum saturation at the source which can in reality not be exceeded and is thus an artefact that is caused by the numerical difficulties at the sink.

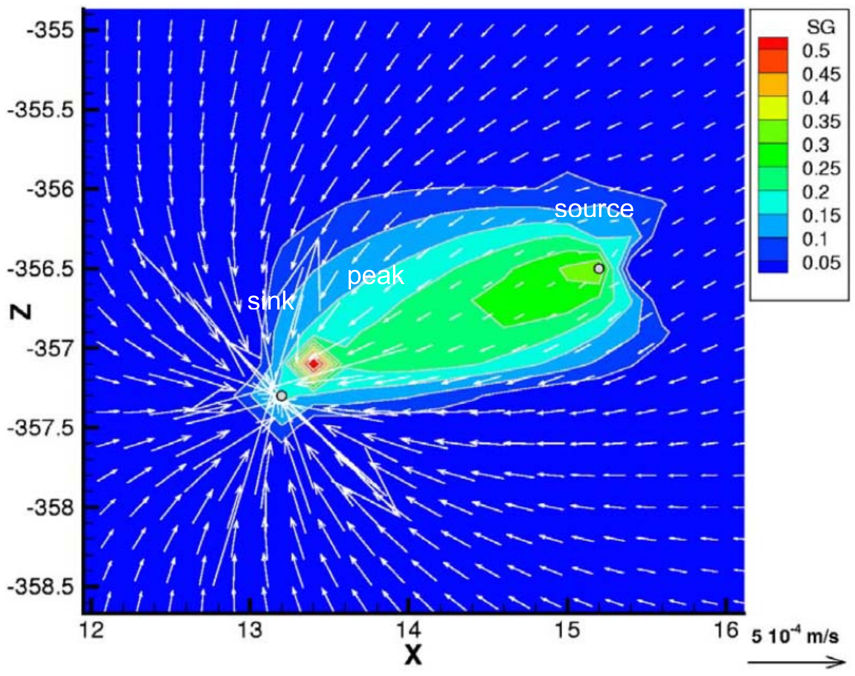


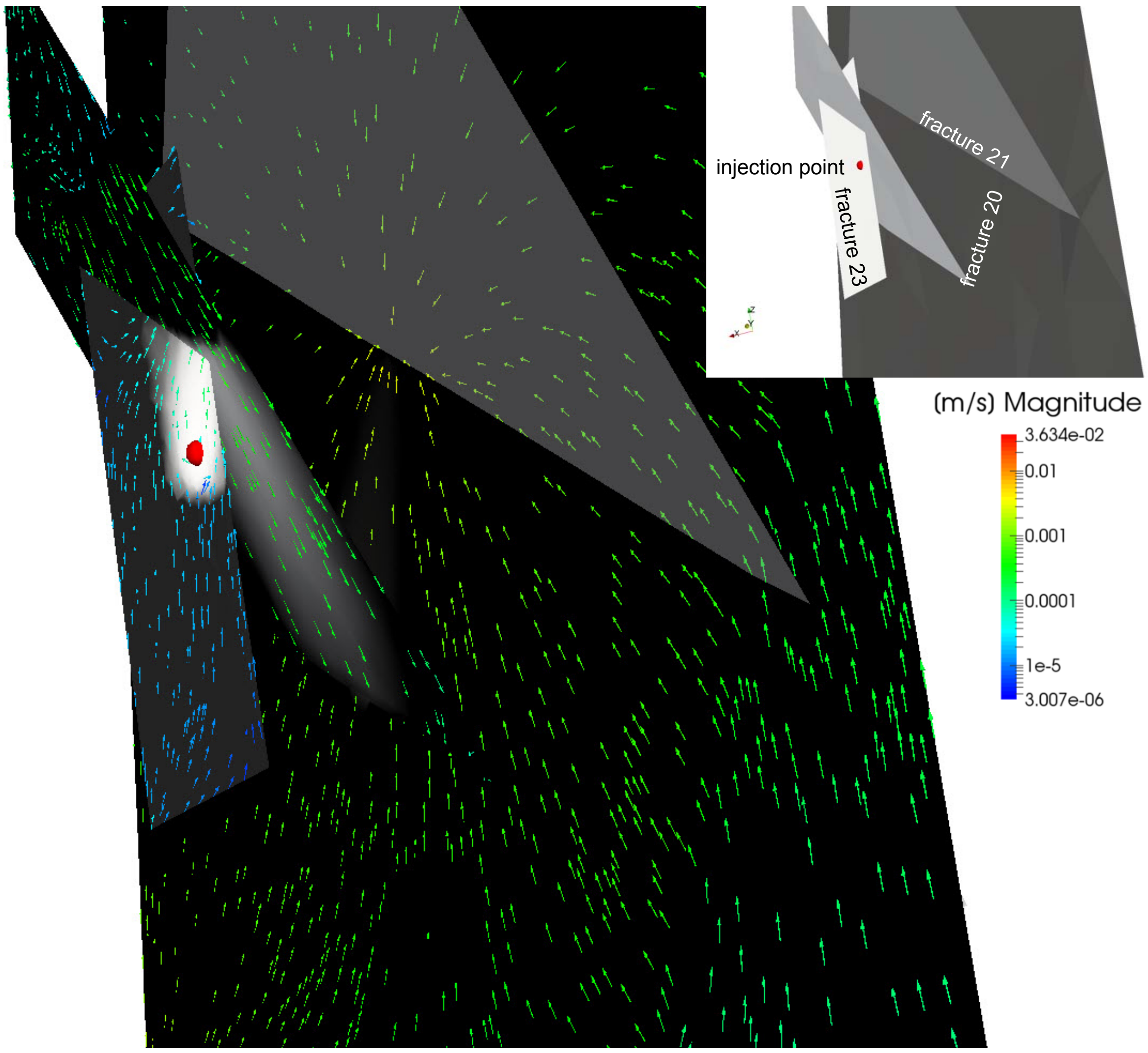
Fig. 5.33 Model results for the steady-state gas saturation (SG) in a dipole test; from /KUL 02/

The results for tracer migration in Task 6 are illustrated in Fig. 5.34 to Fig. 5.36 by plots showing the concentration distribution as well as the velocity field in fractures 23, 22, 20, and 21 at 10h, 50h, and 200h. The adopted three points in time represent the end of the high tracer injection rate and the beginning of a strong reduction of this rate as shown in Fig. 5.14 (the fluid injection was kept constant for 10 hours), an interesting phase of the transport dynamics where the tracer plume has already affected three of the four fractures (at 50 hours), and reaching the maximum concentration at the sink at 200 hours.

Since there is no meaningful view from which the injection and the extraction point is visible, the group of the four fractures is depicted from two sides. Apparently, the plume follows the streamlines at the source that are indicated in Fig. 5.20 at all times. This includes flow from the source against the general flow direction over a short distance in fracture 23. After the tracer injection rate had almost dropped to zero, a ringlike concentration distribution developes around the injection point that can be seen in the plot for 200 h in Fig. 5.36.

The concentration distribution along the fractures 23, 22, 20, and towards the extraction point in fracture 21 shows a large decrease of concentration with increasing distance to the source. The peak concentration is reduced by 4 orders of magnitude while reliable concentration values span a range of more than 6 orders of magnitude. From this fact a strong dilution of the tracer at the fracture intersections can be inferred which could easily provoke large oscillations due to the little influence of hydrodynamic dispersion. However, looking at the concentration distribution in fractures 20 and 21 at 200 h model time (when the tracer reached the maximum at the sink) reveals no such errors in the numerical solution as depicted in Fig. 5.37. On the contrary, the concentration contrasts at the intersections are quite sharp.

To corroborate this observation the four fractures were scanned closely for negative concentration values as indicators for oscillations. But only three pointlike locations could be identified, two in fracture 23 and one in fracture 20 where a negative concentration developed. These negative concentrations did not exceed 16 Bq/kg, though, in comparison to the maximum of $2.7 \cdot 10^7$ Bq/kg at the injection point.



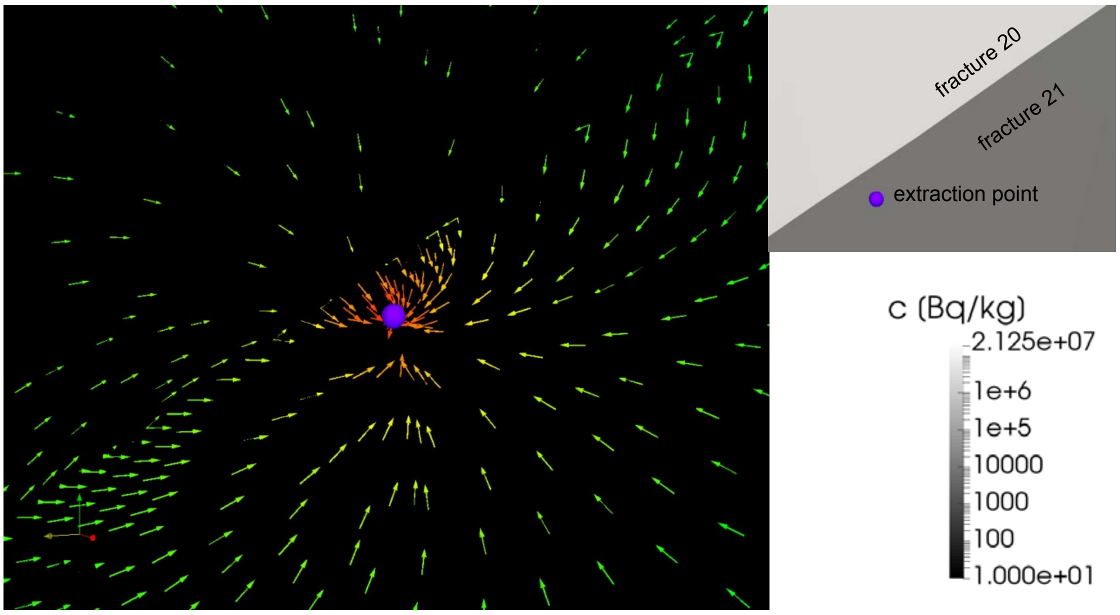
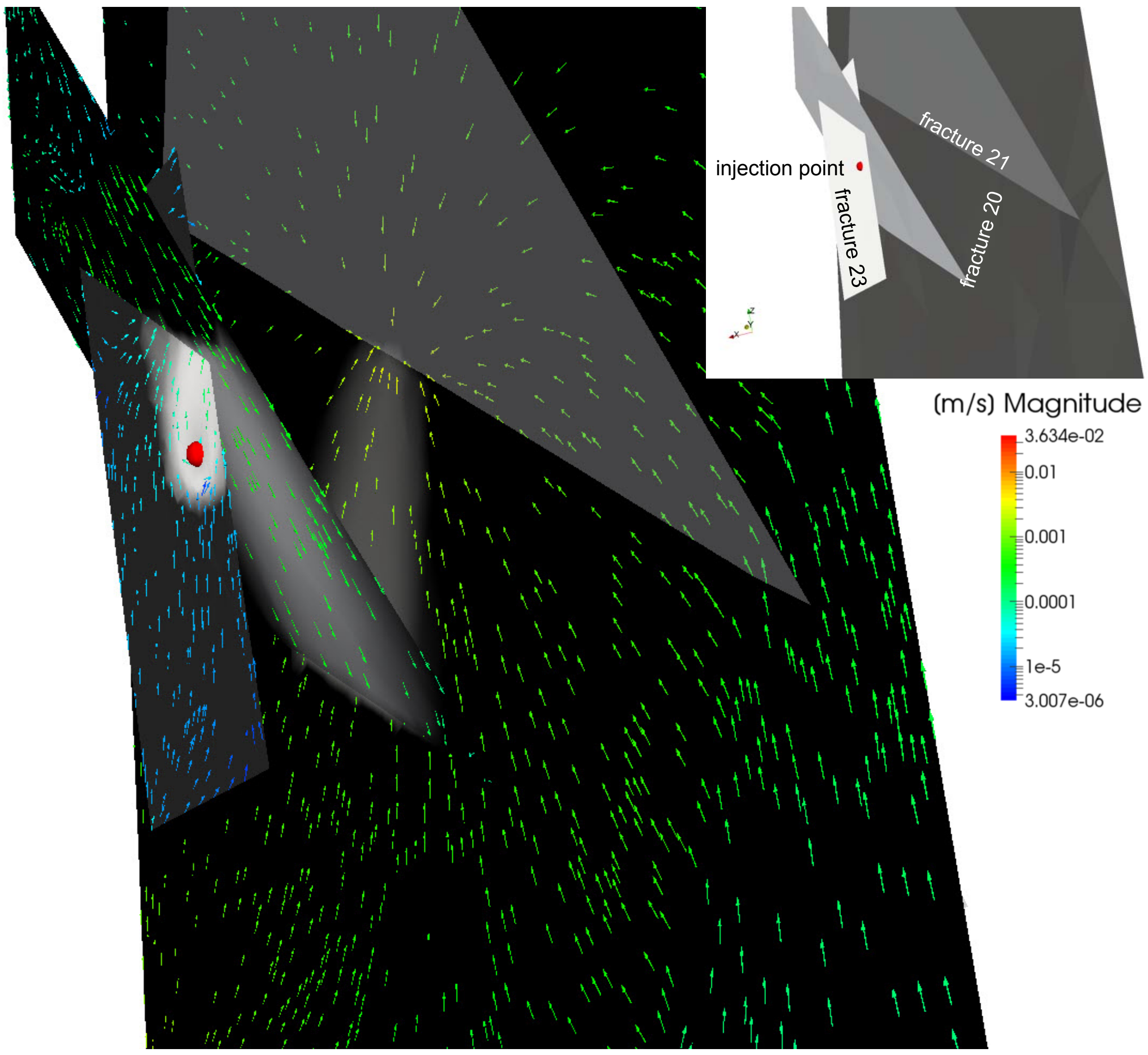


Fig. 5.34 Concentrations in fractures 23, 22, 20, and 21 at 10h



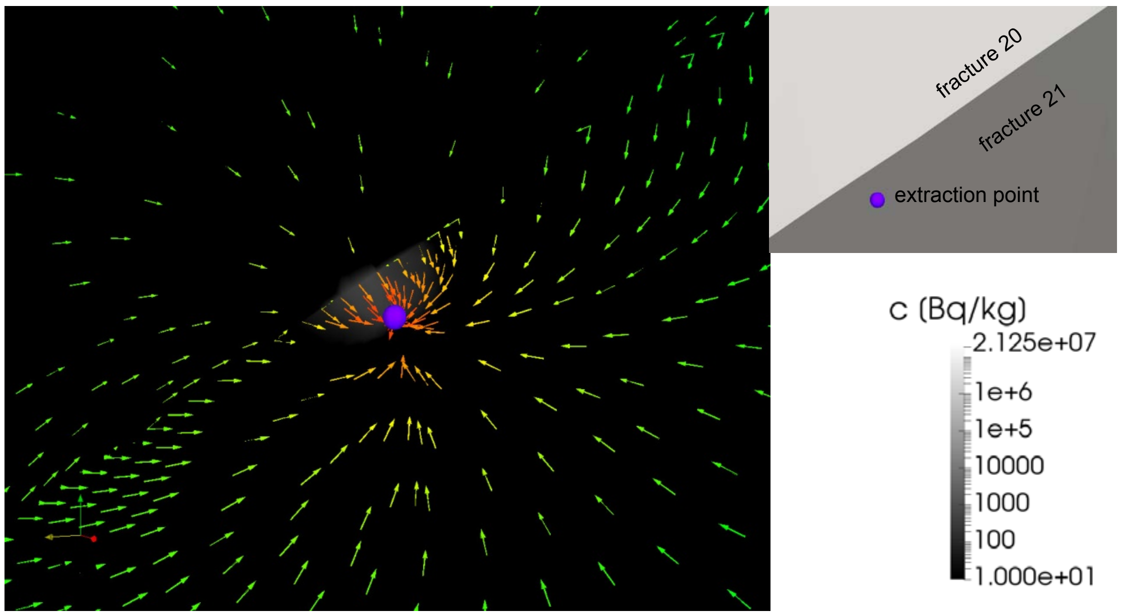
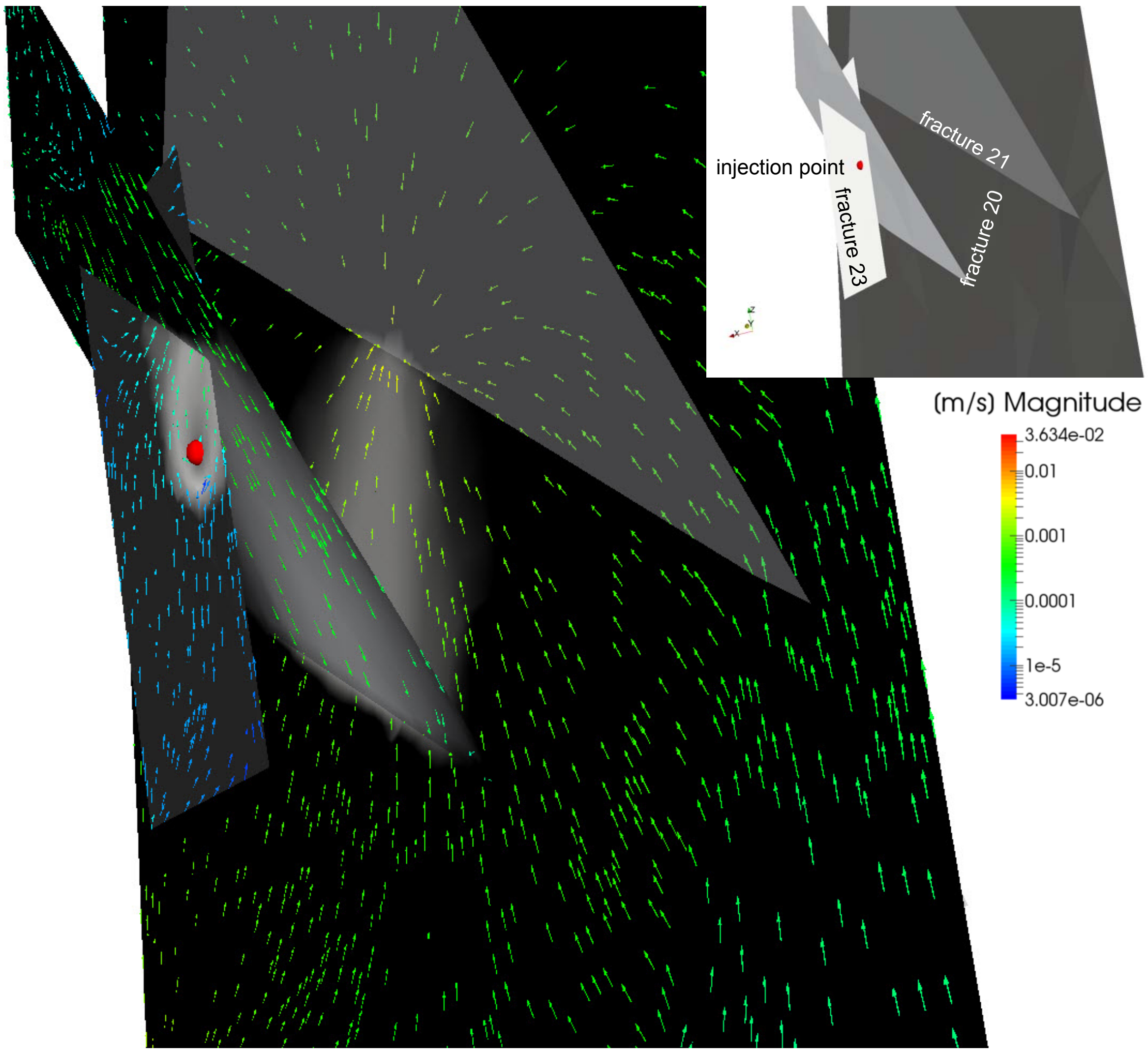


Fig. 5.35 Concentrations in fractures 23, 22, 20, and 21 at 50h



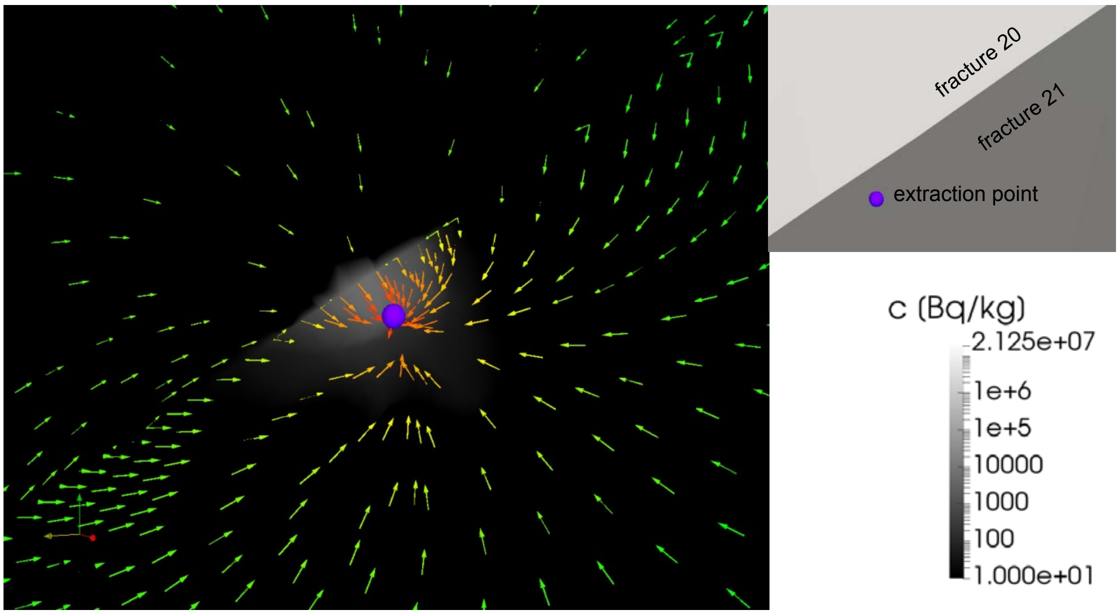


Fig. 5.36 Concentrations in fractures 23, 22, 20, and 21 at 200h

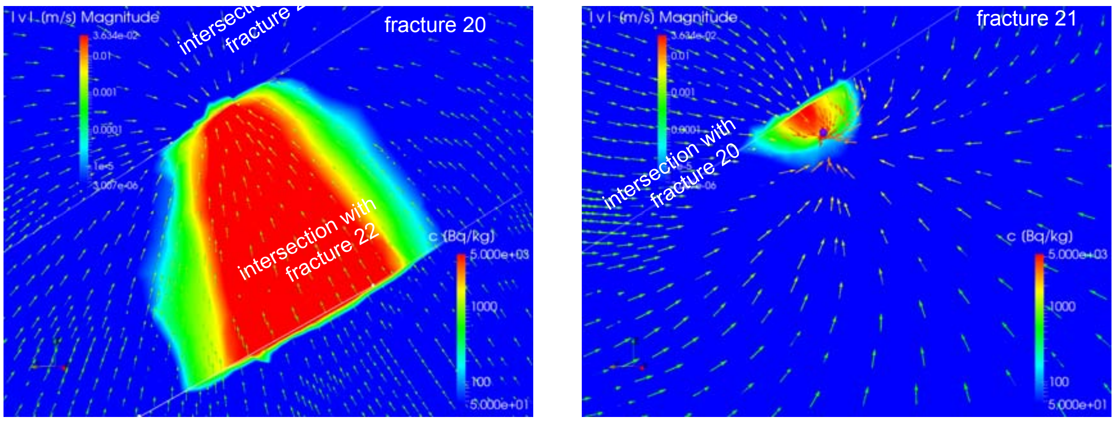


Fig. 5.37 Concentrations in fractures 20 and 21 at 200h

5.2.8 Conclusions

Modelling flow and tracer transport according to the description of Task 6 of the Task Force on Groundwater Flow and Transport of Solutes demonstrated effectively the ability of d^{3}f++ to cope with flow and tracer transport in highly fractured porous media. Even if reproducing the C2-test required some parameter modifications the measured breakthrough curve was met rather well. This is even more remarkable as the peak concentration was reduced by four orders of magnitude and reliable concentration values were calculated even at 1 % of the reduced peak concentration at the sink. The sharp concentration changes at the fracture intersections due to mixing were very well captured without provoking oscillations as often seen in other models. The same applies also to the model results for the sink area where sharp concentration contrasts are naturally part of the solution.

While the model performed technically very well, explanations for the necessary considerable reduction of the formation factor remain somewhat in the dark. Referring to the fractures an error in the transformation from transmissivities to permeabilities for the fractures was suspected. To confirm this, the exact measurement procedures as well as a clear understanding of the physical implications of the transformation with a view to the complex fracture structure would have been required. However, this could not be achieved based on the available reports only.With respect to the matrix the reduction of the formation factor seems to indicate deficiencies in the conceptual model. While replacing the matrix as well as the background fractures by one effective continuum appears to be justifiable when simulating flow this might not be the case if simulating solute transport. In case of transport modelling additional hydraulic properties are required. While the permeability is sufficient to characterize steady-state flow, transport is also based on the pore velocity thus requiring the porosity. Moreover, solute migration is additionally sensitive to the actual transport path. This includes spreading of a plume not only because of the microscopic dispersion but also because of macroscopic spreading in the network of background fractures. The model concept for tracer transport in fractured porous media should thus be reconsidered in the future.

Further conclusions can be drawn with respect to the C2-test. Analyzing the pressure distribution in the fractures as well as in the matrix shows that the flow field is dominated by the extraction of water. Water is basically drawn from the boundary to the sink. The sophisticated pressure distribution at the boundary that has been applied for the model is therefore only of secondary relevance for this task. Flow, however, does not follow a straight line through the matrix but takes mostly detours along the hydraulically better conducting fractures. This phenomenon ensures that water drawn from the injection point to the extraction location is essentially flowing along fractures.

In general, this leads to highly complex flow patterns as the fracture orientations and intersections are seemingly at random. The complexity of the flow field became obvious by analyzing the velocity field along the transport path of the injected solution. While only four fractures covered this path the additionally three fractures that intersect these four fractures influenced the underlying velocity field significantly. Further analyses in this detail were not feasible in a reasonable time frame as the model comprised a total of 30 discrete fractures.

Finally, the difficulties should be acknowledged that are encountered when visualizing physical quantities in a fracture network. One is the inherent problem with the results of 3d-models that can only be presented in 2d thereby loosing easily the orientation of objects and the relation of two or more objects in space to each other. Another problem arises clarifying graphically interactions between two fractures that are intersecting at narrow angles. Scalar or vector plots of the two fracture faces are not discernable if they are meeting under narrow angles. And of course one half of the graphics are always on the far side of the observer which becomes even more serious if several fractures are shown. These challenges culminated at illustrating the time-dependent tracer transport by combined plots depicting concentration distributions and the underlying velocity field at the same time.

5.3 3d regional free surface flow with well pumping, variable recharge conditions and river discharge

The d³f++ code handles free surface groundwater flow using level set methods, see /FRO 12/ and /SCH 12/. Previously, this implementation has only been tested on the basis of small 2d examples. Within the BMBF-funded NAWAK project ("Development of sustainable adaption strategies for the water supply and distribution infrastructure on condition of climatic and demographic change", identification number 033W007) d³f++ is applied to a regional 3d model of a coastal aquifer near the German North Sea, taking into account variable recharge, river discharge and the pumping wells of three waterworks. The objective is forecasting the impact of several demographic and climate change scenarios on the position of the seawater-freshwater interface. Here, this model is used as a huge 3d test case for the correct implementation of the level set methods in d³f++, the increased performance of the new code and the implementation of new features such as as variable recharge rates in space and time as well as river discharge.

5.3.1 Free groundwater table in d³f++

In d³f++ the time dependent position of the groundwater table $\Gamma(t)$ in a fixed domain D is described implicitly as the zero level set of a level set function $\phi(x,t)$, i. e. $x \epsilon \Gamma(t)$ $\Leftrightarrow$ $\phi(x,t) = 0$. $\Gamma(t)$ divides D into the fully saturated zone $\Omega(t)$, where Darcy's law is valid, and the partially saturated zone $\mathrm{D}\backslash\Omega(t)$ that is regarded to be outside of the model domain, see Fig. 5.38.

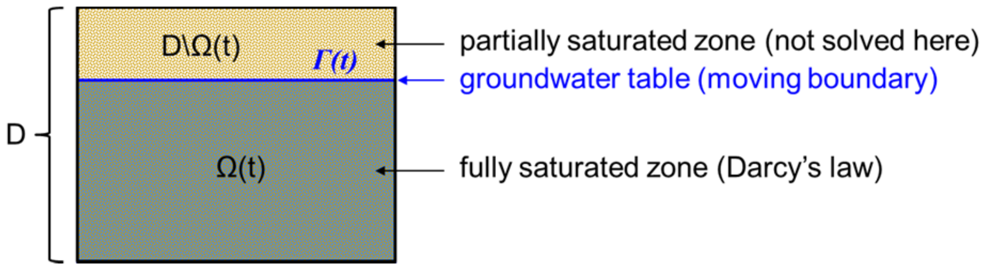

D\Ω(t)
Γ(t)
D
Ω(t)
partially saturated zone (not solved here)
groundwater table (moving boundary)
fully saturated zone (Darcy's law)

Fig. 5.38 Model domain D divided by $\Gamma(t)$ D into a fully saturated zone $\Omega(t)$ and a partially saturated zone $\mathrm{D}\backslash\Omega(t)$

As a proper choice of a level set function the signed distance function Φ is used with $\| \nabla \Phi \| = 1$ and $\Phi(\gamma) = 0, \gamma \epsilon \Gamma(t)$. $\Gamma(t)$ forms the time dependent part of the boundary of Ω, moving with a normal velocity $S(\gamma) := \vec{N}(\gamma) \cdot \vec{u}(\gamma, t),\ \gamma \epsilon \Gamma(t)$. These normal velocity can easily be interpolated to $D \backslash \Omega(t)$, which is necessary to compute the movement of $\Gamma(t)$ /FRO 12/.

Using level set functions for the description of $\Gamma(t)$ implicates some restrictions to these part of the model boundary: Regarding the pressure, the boundary condition $p = 0$ on $\Gamma(t)$ at any fixed time t is set directly by numerical discretization and cannot be changed /SCH 12/. That means groundwater recharge as well as discharge may not be treated as boundary conditions, both effects have to be modelled as factors directly influencing the normal velocity S, see /SCH 12/:

$$S = \left(\frac{1}{\phi}\, \boldsymbol{q} + \frac{r}{\phi} \boldsymbol{e}_z + \frac{c}{\phi} \boldsymbol{e}_z\right) \cdot \vec{N} \tag{5.3}$$

where ϕ stands for the effective porosity of the medium in the corresponding subdomain, $r = r\,(t, \gamma)$ is the groundwater recharge caused by precipitation and $c = c\,(t, \gamma, \Gamma)$ is a special source due to the rivers on the top of the domain. ($\boldsymbol{e}_z$ is the unit vector of the z-direction.) The values of $r = r\,(t, \gamma)$ are specified as a piecewise constant function over the projection of the domain to the xy-plane, and are periodic in time.

The rivers are considered to be a one-dimensional projection to the xy-plane, but to hve an arbitrary depth in the z-direction. They are represented as a separate grid of segments in the xy-plane. The source $C = C\,(t, \gamma, \Gamma)$ due to the rivers is given by the following formula:

$$C = -\frac{k_\mathrm{c} \cdot \rho_\mathrm{r} \cdot g}{\varepsilon \cdot \mu_\mathrm{r}} \cdot \big(h_\mathrm{s}(\gamma) - h_\mathrm{fs}(t, \gamma, \Gamma)\big) \cdot w. \tag{5.4}$$

Here, g is the gravity, w is the width of the river, k_c and ε are the effective permeability and the thickness of the colmation layer, ρ_r and μ_r are the effective density and viscosity of the water in the river, $h_\mathrm{s}(\gamma)$ is the specified water level in the river (absolutely or optionally relatively to ground level, depending on the specification of $h_\mathrm{s}(\gamma)$), whereas $h_\mathrm{fs}(t, \gamma, \Gamma)$ is the depths of the groundwater table. The function c is smoothed C:

$$c(t,\gamma,\Gamma) = \int_{B_\delta^{xy}(\gamma)} C(t,\gamma',\Gamma) \cdot M(\gamma,\gamma') \cdot d\gamma', \tag{5.5}$$

where δ is the specified smoothing length, $B_\delta^{xy}(\gamma)$ is the circle of radius δ in the xy-plane around the projection of γ to that plane, and $M(\gamma,\gamma')$ is the mollifier

$$M(\gamma,\gamma') = \begin{cases} \dfrac{\delta - d(\gamma,\gamma')}{\pi\delta^3/3}, & d(\gamma,\gamma') < \delta \\ 0, & d(\gamma,\gamma') \geq \delta \end{cases} \tag{5.6}$$

with $d(\gamma,\gamma')$ being the distance between the projections of γ and γ' to the xy-plane. There is a further option to restrict the value $h_s(\gamma) - h_{fs}(t,\gamma,\Gamma)$ if it exceeds some specified maximum (to avoid unphysically strong sources).

5.3.2 The Sandelermöns model

The Sandelermöns model covers an area of about 1,000 km² situated at the German North sea coast as shown in Fig. 5.39. Almost half of the model region is low-lying and formed part of the North Sea some hundred years ago so that the aquifers lead saline groundwaters. The south-western part consists of moraines with notable recharge rates and freshwater aquifers below. The three waterworks Sandelermöns, Feldhausen and Klein Horsten are situated in these areas, competing for pumping concessions and worrying about the close-by saline waters.

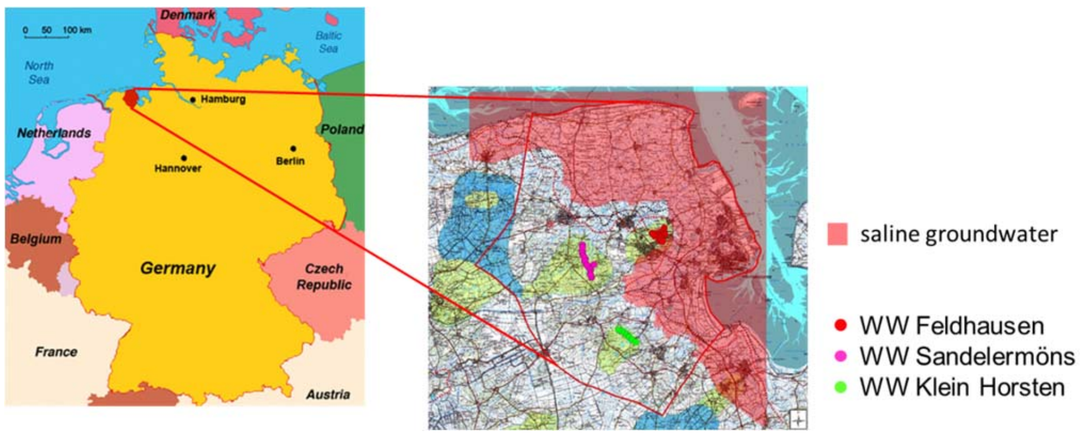
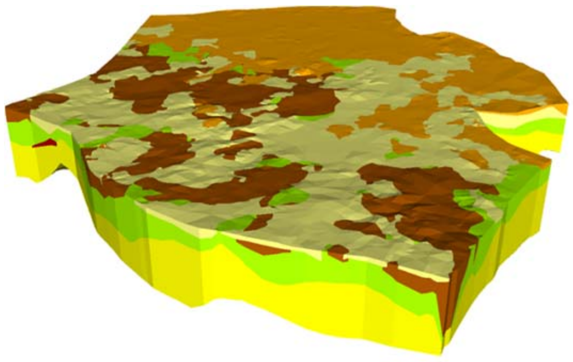


Fig. 5.39 Situation of the Sandelermöns model area including the wells of the three waterworks and area of saline groundwater

The hydrogeological model was set up by the Oldenburg-Ostfriesian Water Company (OOWV). Six formations are distinguished, starting with a thick layer of fluvial sands at the bottom, followed by small fields of Tergast clay and a continuous layer of melt-water sands. The top of the model consists of thin layers of Lauenburg clay, dune sands and silty materials. The base surface of each hydrogeological layer was read-in into the ProMesh tool and together with ground surface data composed to a 3d model. Fig. 5.40 shows the hydrogeological model created and the start values for the permeabilities and porosities.



	unit	permeability [m²]	porosity [-]
🟧	silt	$1 \cdot 10^{-12}$	0.1
🟨	sands	$1 \cdot 10^{-10}$	0.2
🟫	Lauenburg clay	$1 \cdot 10^{-13}$	0.05
🟩	melt-water sands	$4 \cdot 10^{-11}$	0.2
🟥	Tergast clay	$1 \cdot 10^{-13}$	0.05
🟨	fluvial sands	$2.8 \cdot 10^{-11}$	0.2

Fig. 5.40 3d hydrogeological model of the Sandelermöns region, exaggerated in vertical direction by a factor of 30

One challenge was the generation of a coarse grid with advantageous numerical properties. Because tetrahedral grids turned out to be not suitable, d³f++ had to be enabled to handle prism grids, including the option of anisotrope grid refinements and adaptions of the implementation of the levelset method. The result was a coarse grid consisting of only 18,000 prisms.

The south-western boundary of the model is located on a watershed and therefore assumed to be impermeable. The salt concentration is set to zero. The north-western and south-eastern boundaries are choosen to be perpendicular to the water table isohypses and therefore also regarded as impermeable for the flow. The coastal boundary is equipped with a hydrostatic pressure for seawater and a salt concentration of 34 kg kg^{-1}. The bottom of the model is also assumed to be impermeable because geological knowledge suggests almost impermeable clayey formations here.

On the upper boundary, a Dirichlet boundary condition $p = 0$ has to be choosen for the pressure as already mentioned in Chapter 5.3.1. The concentration is also set to zero.

Time dependent groundwater recharge data were provided by the Braunschweig Technical University. 147 polygonal recharge zones were distinguished as shown in Fig. 5.42.

The north-eastern region of the model domain is characterized by a dense net of small draining ditches and rivers conducting water to the coastal pumping stations in the dikes to keep the groundwater level below land surface. This drainage plays a crucial role in the hydraulic regime and had to be incorporated in the model. Because an explicite mapping of all ditches in a regional model is impossible, only the rivers of first and second orders were integrated, and their influence was smeared over a user defined range of surface elements.

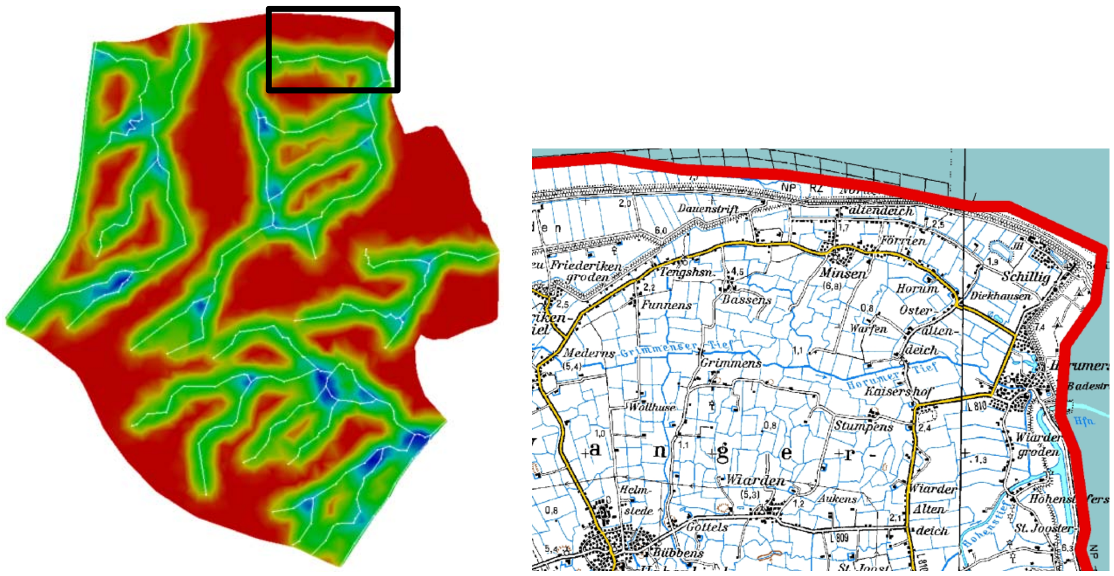


Fig. 5.41 Influence of river drainage, exemplarily (red: 0.0, blue: 1e-8 m³ s^{-1}) right: real network of draining ditches in the black marked detail

Regrettably there exists only little information about the water levels of the receiving streams in the Sandelermöns region as well as the pumping rates of the coastal stations. Missing information was replaced by reasonable assumptions to meet the natural hydraulic regime by a careful calibration process.

Furthermore, the 51 pumping wells of the waterworks were included into the model. Additionally, 36 private wells are regarded.

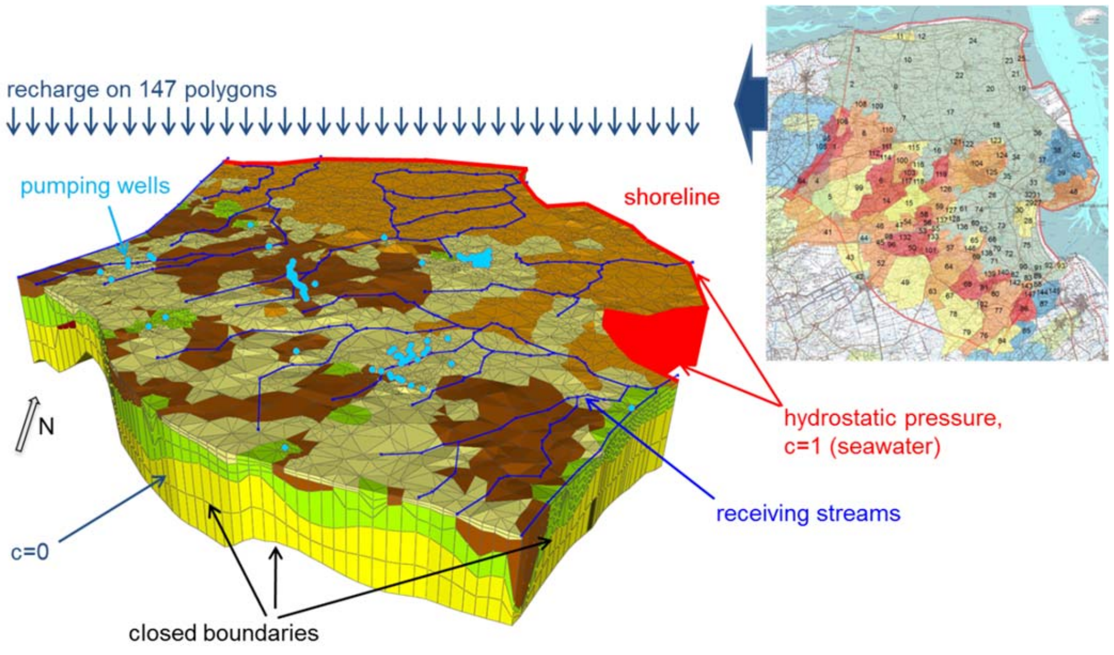


Fig. 5.42 Sandelermöns model with prism grid, boundary conditions, receiving streams (blue) and pumping wells (light blue); right: groundwater recharche distinguished on 147 polygons

To find an appropriate initial condition for the free groundwater table the data of 284 gauge wells were averaged over the year 2011, interpolated and converted into a ProMesh data set. Unfortunately, only few gauges exist near the coastal line, and the model reacts very sensitive on changes in these settings.

The initial condition for the salt concentration is based on geoelectromagnetic data provided by the Leibniz Institute for Applied Geology, see Fig. 5.43. Therefore, specific resistance measurements had to be converted into salt concentrations and the resulting 3d data field had to be read in by the ProMesh tool.

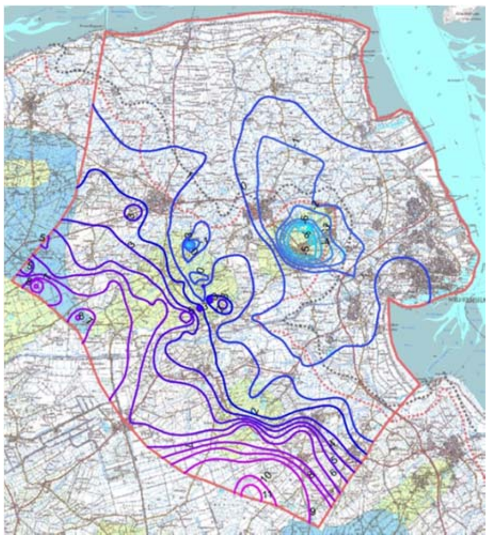
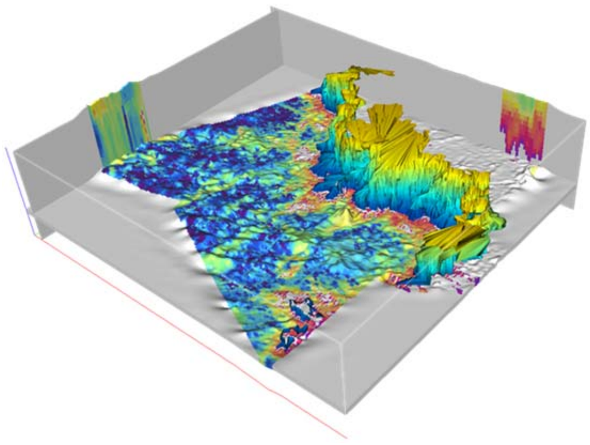



Fig. 5.43 Initial conditions,
left: groundwater surface
right: 3-Ωm-isosurface resulting from geoelectromagnetic measurements

5.3.3 Sandelermöns simulations

Simulations started with the initial and boundary conditions as described in Chapter 5.3.2. Fig. 5.44 shows the salt concentration at the first step of simulation, where $c = 1$ means seawater concentration (34 kg kg^{-1}), as well as well fields and rivers.

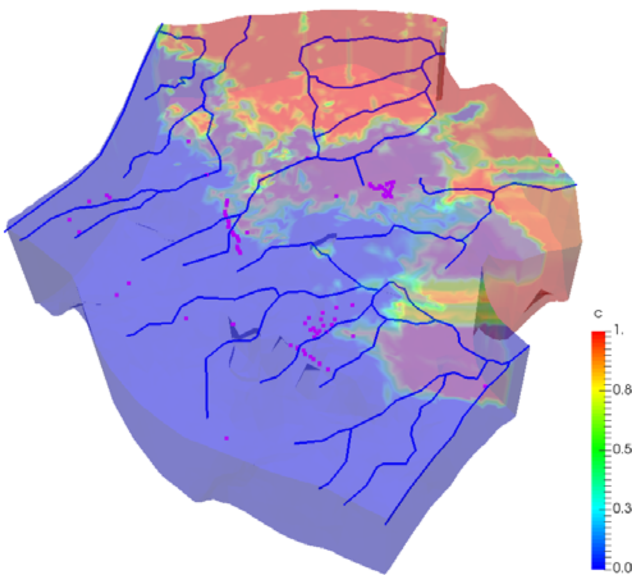


Fig. 5.44 Initial state of the Sandelermöns model with river system (blue), pumping wells (red) and salt concentration

The first simulations were performed on grid level 1 consisting of only 72 000 nodes with the objective to calibrate the model. In a first step, the fluid volume in the model had to be stabilized. Factors influencing this volume are the initial state of the groundwater level, sea level, recharge rates, river drainage and pumping rates.

The model reacts mostly sensitive against variations in sea level, represented by the hydrostatic pressure boundary condition at the north-eastern boundary, relatively to the initial groundwater level at the coastal line. River drainage relatively to the recharche rates is much more influencing the results than well pumping, which plays a less important role. Theses sensitivities form a huge problem because of the poor knowledge about groundwater levels in the coastal zone as well as river drainage. Another problem is that the results for the fluid volume may completely fail without an adequate grid refinement.

Fig. 5.45 shows the simulated groundwater surface after a model time of 6 years as a result of the calibration process as a contour map. In the scatter plot on the right simulated groundwater levels are compared with measured data. Therefore, the simulated values had to be approximated from the grid nodes next to the gauge points which leads to errors especially in the depression cones of the wells.

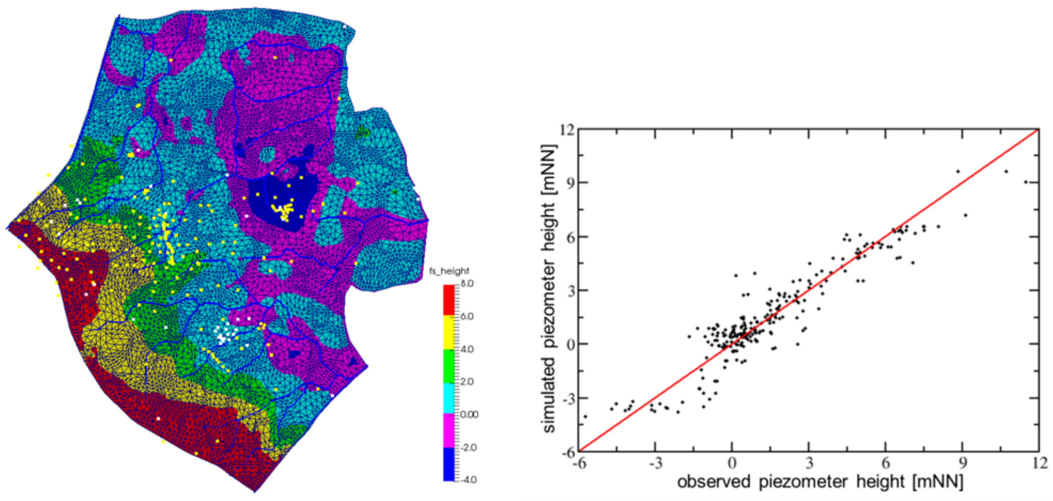


Fig. 5.45 Simulated groundwater level with gauge wells;
right: comparison of measured and simulated data during calibration after a model time of 6 years

In consideration of the lack of data and the coarse grid the calibration results are rather good and plausible. Nevertheless, further grid refinements are necessary for a better calibration. To improve simulation results, it is desireable to achieve grid convergence, that means at least grid levels 3 or 4 (1.1 or 4.6 millions of nodes) have to be used.

As shown in Fig. 5.46, the influence of the hydrostatic pressure boundary condition is restricted to a zone near the coastal boundary. Saline waters detected by measures outside this zone must result from former inundations hundreds of years ago, before the areas were protected by dikes. That means the simulated salt distribution is mainly determined of the initial condition.

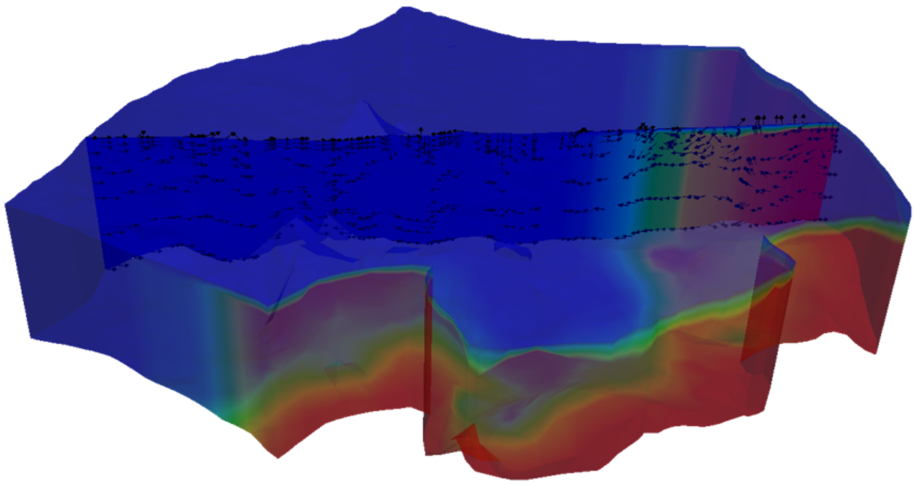


Fig. 5.46 Simulated flow field on a cutting plane

The main objective of the NAWAK project is predicting the situation of the freshwater-/saltwater interface up to the year 2050. This work is in progress. Fig. 5.47 shows exemplarily the concentration isosurface of 500 mg l^{-1} (drinking water standard) relatively to the well fields.

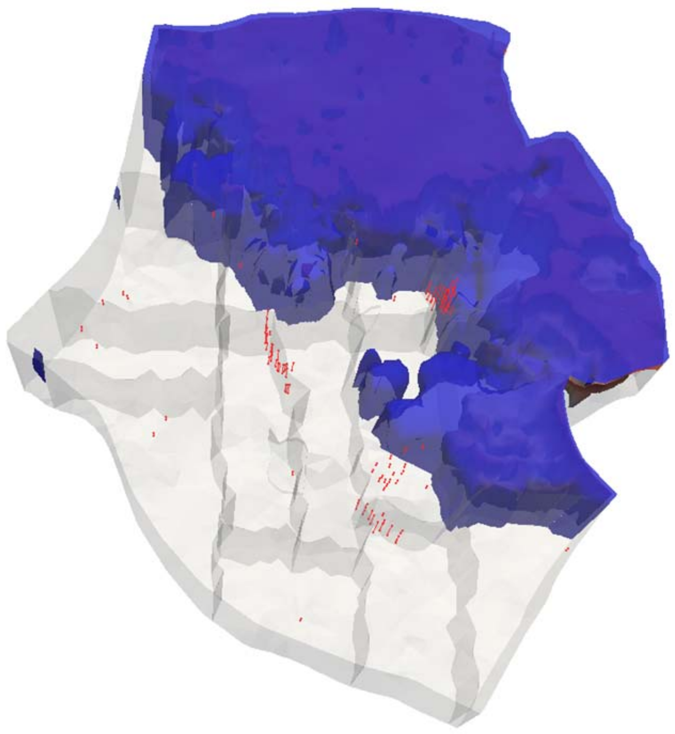


Fig. 5.47 Simulated drinking water/seawater interface in relation to the pumping wells

5.3.4 Conclusions

The Sandelermöns model represents an highly challenging test case for d³f++ with respect to free groundwater surface modelling and provoked several crucial improvements of the code. At the same time, it allows a broad examination of the correct transfer of the level set methods from UG3 to UG4. Additionally, Sandelermöns posed challenges to the d³f++ preprocessing and initiated a series of very useful enhancements of ProMesh. In this way it made also a contribution to an improved applicability of d³f++ to large, regional problems with thin layers.

Furthermore it was detected that a minimum grid refinement is crucial for getting correct results for the fluid volume or the position of the groundwater surface, respectively. Thereby it turned out again that grid convergence is an indispensable precondition for confidable simulation results, especially regarding forcasts.

6 Summary

The codes d^3f and r^3t are well established for modelling density-driven flow and nuclide transport in the far field of repositories for hazardous material in deep geological formations. They may be used in porous media as well as fractured rock or mudstone, for modelling salt- and heat transport as well as in models with free groundwater surface.

Both codes were applications of the software platform UG /BAS 94/ that was developed in the early nineties. In order to adapt the flow and transport simulations to the growing requirements of modern efficient numerical software, the renewed code basis UG4 for the simulation of coupled partial differential equations has been developed /VOG 13/. The new implementation is grounded on an object-oriented software design and written in C++.

To be able to participate in the current and future enhancements and numerical advances the UG-applications d^3f and r^3t had also to be transformed to this new software platform. Benfitting the fact that coupling between different sets of equations is natively supportet by UG4, both codes were coupled in this process to the new conjoint code d^3f++. This allows the simultaneous simulation of density-driven groundwater flow and pollutant transport.

State-of-the art computer codes have not only to run on massively parallel computers, they also have to use modern multicore and hybrid computer structures. Each processor consists of multiple cores that are accessing at the same, hierarchically structured main memory, and, moreover, the cache memory may be organized in a much more complex way. In many cases processors of this type are supplemented by very specialized processors like GPUs and Cell processors. The efficient use of these modern computer architectures relies on appropriate data structures that had to be implemented in UG4 and d^3f++, respectively. Additionally, the multigrid solvers had to be adapted to these new structures and advanced. In tests with up to131,000 processors a very good scaling behavior could be shown.

To improve efficiency, different types of non-linear solvers were compared. As an alternative to Newton-type fixed point iterations linear-implicit schemes were introduced. These schemes allow to linearize the problem only once, and thus, when compared to the Full Newton algoritm, the number of linear iterations is reduced considerably. Moreover, this method can elegantly be combined with a timestep control and also with spatial adaptivity.

In every site investigated some of the the rock properties as well as flow and transport parameters remain partially unknown. These uncertainties are usually taken into account by stochastic modeling. However, Monte Carlo simulations with respect to regional groundwater flow are not applicable because of their high computational costs. A new stochastic approach was therefore adapted and applied. It is based on the idea that a scalar variable like the salt concentration is replaced by a so called "filtered probability density function"in the differential equation system. This leads, though, to higher dimensional equation systems requiring new numerical solvers. These were also developed.

UG4 applications are controlled by scripting or a graphical user interface. Therefore, lua-scripting and the Visual Reflection Language VRL had to be adapted to the needs of d^{3}f++. Additionally, d^{3}f++ profits from the UG4 pre-processor ProMesh that enables the user to set-up model geometries based on different types of data input and to generate the computational grid. ProMesh was also enhanced by several features that are helpful in the buildup of hydrogeological models. Data exchange between d^{3}f++ and pre- and postprocessors has been standardised using exclusively the new grid format ugx. For the output of the simulation results an interface to the well established VTK framework has been created, offering the possibility to use e. g. PARAVIEW or VISIT for visualisation.

The new, modern code d^{3}f++ underwent a series of tests, and it was already successfully applied to large complex models such as Task 6 of the Task Force on Groundwater Flow and Transport of Solutes and a regional groundwater flow model near the German North Sea coast. This demonstrated the ability of d^{3}f++ to handle models in crystalline as well as in sedimentary rock at acceptable computational effort. The models were thereby of a higher complexity than previous models calculated with the former versions d^{3}f and r^{3}t.

Furthermore, the handling of d^{3}f++ has become much more user-friendly, and, finally, the data interface by standardised files opens up the use of a wide spectrum of post-processing software.

7 References

/ACK 04/ Ackerer, P., Younes, A., Mancip, M.: A new coupling algorithm for density-driven flow in porous media, GEOPHYSICAL RESEARCH LETTERS 31 (2004) L12506, 1–4. doi:10.1029/2004GL019496.

/AND 96/ Andricevic, R., V. Cvetkovic (1996), Evaluation of Risk from Contaminants Migrating by Groundwater, Water Resour. Res., 32 (3), 611-621.

/AND 98/ Andricevic, R. (1998), Effects of local dispersion and sampling volume on the evolution of concentration uctuations in aquifers, Water Resour. Res., 34 (5), 1115-1129.

/AND 02/ Andersson, P., Byegård, J., Winberg, A.: Final report of the TRUE Block Scale project - 2. Tracer tests in the block scale. Technical Report TR-02-14, Swedish Nuclear Fuel and Waste Management Company (SKB), 2002.

/BAS 94/ Bastian, P., and Wittum, G.: Robustness and adaptivity: The UG concept. In: Multigrid Methods IV, proceedings of the fourth european multigrid conference, ed. by Hemker, P., Wesseling, P., 1994.

/BAS 97/ Bastian, P., Birken, K., Johannsen, K., Lang, S., Neuß, N., Rentz-Reichert, H., Wieners, C.: UG – A flexible software tool for solving partial differential equations. Computing and Visualization in Science 1, 27-40, 1997.

/BEA 91/ Bear, J., Bachmat, Y.: Introduction to Modeling of Transport Phenomena in Porous Media, Theory and applications of transport in porous media, Kluwer Academic, Dordrecht, 1991.

/BLE 11/ Blessent, D., Therrien, R. and Lemieux J.-M.: Inverse modeling of hydraulic tests in fractured crystalline rock based on a transition probability geostatistical approach. Water Resources Research, Vol. 47, W12530, doi: 10.1029/2011WR011037, 2011.

/BLE 13/ Blessant, D.: Stochastic fractured rock facies for groundwater flow modelling. Dyna rev.fac.nac.minas vol.80 no.182, Medellín, 2013.

/BOC 13/ Bockgård, N., Vidstrand, P., Åkesson, M.: Modelling the interaction between engineered and natural barriers – An assessment of a fractured bedrock description in the wetting process of bentonite at deposition tunnel scale. Technical Committee of Task 8, SKB, Rev. 2013-10-27, 2013.

/CHE 06/ Cheng, H., Cvetcovic, V.: Äspö Task Force on modelling of groundwater flow and transport of solutes; Modelling of Task 6D, 6E, and 6F, flow and transport simulations in fracture networks. International Progress Report IPR-06-20, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/CRA 06/ Crawford, J., Moreno, L.: Äspö Task Force on modelling of groundwater flow and transport of solutes; Modelling of Task 6D, 6E and 6F, using CHAN3D. International Progress Report IPR-06-19, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/CVE 10/ Cvetkovic, V. and Frampton, A.: Transport and retention from single to multiple fracturesin crystalline rock at Äspö (Sweden): 2. Fracture network simulations and generic retention model. Water Resources Research, Vol. 46, W05506, doi: 10.1029/2009WR008030, 2010.

/DEN 00/ Dentz, M., H. Kinzelbach, S. Attinger, and W. Kinzelbach (2000), Temporal behavior of a solute cloud in a heterogeneous porous medium 1. Point-like injection, Water Resour. Res., 36 (12), 3591-3604.

/DEN 02/ Dentz, M., H. Kinzelbach, S. Attinger, and W. Kinzelbach (2002), Temporal behavior of a solute cloud in a heterogeneous porous medium 3. Numerical simulations, Water Resour. Res., 38 (7), 23-1-23-13.

/DER 03/ Dershowitz, W., Winberg, A., Hermanson, J., Byegård, J., Tullborg, E.-L., Andersson, P., Mazurek, M.: Äspö Task Force on modelling of groundwater flow and transport solutes; Task 6c - A semi-sythetic model of block scale conductive structures at the Äspö HRL. International Progress Report IPR-03-13, Swedish Nuclear Fuel and Waste Management Company (SKB), 2003.

/DER 06/ Dershowitz, W.,Fox, A., Lee, G., van Fossen, M., Uchida, M.: Äspö Task Force on modelling of groundwater flow and transport solutes; Discrete fracture network flow and transport modelling at the rock block scale: Task 6D, 6E, 6F and 6F2. International Progress Report IPR-06-22, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/DEU 02/ Deuflhard, P., Bornemann, F.: Scientific computing with ordinary differential equations. Springer Science & Business Media, 2002, 42.

/DEU 90/ Deuflhard, P., Nowak, U., Wulkow, M.: Recent Developments in Chemical Computing, Computers & Chemical Engineering 14(11) (1990) 1249–1258.

/DIE 98/ Diersch, H.-J. G., Kolditz, O.: Coupled groundwater flow and transport: 2. Thermohaline and 3D convection systems, Advances in Water Resources 21 (5) (1998) 401 – 425. doi:10.1016/S0309-1708(97)00003-1.

/DIE 09/ Diersch, H.-J. G., Kolditz, O.: Variable-density flow and transport in porous media: approaches and challenges, in: FEFLOW © White Paper Volume II, DHI-WASY GmbH, Berlin, 2009.

/DOP 75/ Dopazo, C.: Probability density function approach for a turbulent axisymmetric heated jet centerline evolution, Phys. Fluids, 18 (4), 397-404, 1975.

/DRU 84/ Drummond, I. T., S. Duane, and R. R. Horgan (1984), Scalar diffusion in simulated helical turbulence with molecular diffusivity, J. Fluid Mech.,138, 75-91.

/FEI 99/ Fein, E.; Schneider, A. (eds.): d^3f – ein Programmpaket zur Modellierung von Dichteströmungen. Final report. FKZ-02 C 0465 0. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, GRS-139, Braunschweig 1999.

/FEI 04/ Fein, E. (eds.): Software Package r^3t. Model for Transport and Retention in Porous Media. Final report. FKZ-02 E 9148/2. Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, GRS-192, Braunschweig 2004.

/FOU 03/ Fourno, A., Grenier, C., Mouche, E., Benabderrahmane, H.: Qualification and Validity of a Smeared Fracture Modelling Approach for Transfers in Fractured Media. Proceedings of: Groundwater in Fractured Rocks, 15–19 September 2003, Prag (Czech Republic). IHP-VI, Series on Groundwater, No. 7, 2003. (cited in /HOD 07/

/FOX 03/ Fox, R. O.: Computational Models for Turbulent Reacting Flows, Cambridge University Press, New York, 2003.

/FRO 12/ Frolkovič, P.: Application of level set method for groundwater flow with moving boundary, Adv. Wat. Res. 2012

/GRE 06/ Grenier, C., Bernard-Michel, G.: Äspö Task Force on modelling of groundwater flow and transport of solutes; Modelling of Task 6D, 6E, 6F and 6F2, using Cast3M code. International Progress Report IPR-06-18, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/GRE 08/ Grenier, C., Bernard-Michel, G., Benabderrahmane, H.: Evaluation of retention properties of a semi-synthetic fractured block from modelling at performance assessment time scales (Äspö Hard Rock Laboratory, Sweden). Hydrogeology Journal 17: 1051–1066, 2009.

/GRI 10/ Grillo, A., Lampe, M., Wittum, G.: Three-dimensional simulation of the thermohaline-driven buoyancy of a brine parcel, Comput Visual Sci 13 287–297, 2010.

/GSZ 92/ Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens (editors), Iterative Methods in Linear Algebra. IMACS, Elsevier, North Holland, 1992.

/HEI 04/ Heil, M.: An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems, Computer Methods in Applied Mechanics and Engineering 193,1-2, 1 – 23. doi:http://dx.doi.org/10.1016/j.cma.2003.09.006 2004.

/HEP 13/ Heppner, I., Lampe, M., Nägel, A., Reiter, S., Rupp, M., Vogel, A., Wittum, G.: Software framework UG4: Parallel multigrid on the hermit supercomputer. In: High Performance Computing in Science and Engineering 12, p. 435-449, Springer, 2013.

/HOD 07/ Hodgekinson, D.: Äspö Task Force on modelling of groundwater flow and transport of solutes. Technical Report TR-07-03, Swedish Nuclear Fuel and Waste Management Company (SKB), 2007.

/HOF 13/ Hoffer, M., Poliwoda, C., and Wittum, G.: Visual reflection library: a framework for declarative GUI programming on the Java platform. Computing and Visualization in Science 16 (4), 181-192, 2013.

/HOL 98/ Holzbecher, E.: Modeling density-driven flow in porous media., Springer, Berlin, Heidelberg, 1998.

/JOH 02/ Johannsen, K., Kinzelbach, W., Oswald, S., Wittum, G.: The saltpool benchmark problem - numerical simulation of saltwater upconing in a porous medium, ADVANCES IN WATER RESOURCES 25 (2002) 309–1708.

/JOH 04/ Johannsen, K.: Numerische Aspekte dichtegetriebener StrÂšmung in porösen Medien, Habilitationsschrift, 2004.

/JOH 06/ Johannsen, K.: Numerical aspects of density driven flow in porous media, in: XVI International Conference on Computational Methods in Water Resources, 2006. doi:10.4122/1.1000000246 .

/KAP 94/ Kapoor, V., and L. W. Gelhar: Transport in three-dimensionally heterogeneous aquifers: 1. Dynamics of concentration fluctuations, Water Resour. Res., 30 (6), 1775-1788, 1994.

/KIM 11a/ Kim, J., Tchelepi, H., Juanes, R.: Stability, accuracy, and efficiency of sequential methods for coupled flow and geomechanics, SPE Journal 16 (2). doi:10.2118/119084-PA, 2011.

/KIM 11b/ Kim, J., Tchelepi, H., Juanes, R.: Stability and convergence of sequential methods for coupled flow and geomechanics: Fixed-stress and fixed-strain splits, Computer Methods in Applied Mechanics and Engineering 200,13-16 (2011) 1591 – 1606. doi:http://dx.doi.org/10.1016/j.cma.2010.12.022 2011.

/KRA 70/ Kraichnan, R. H. (1970), Diffusion by a Random Velocity Field, Phys. Fluids, 13 (1), 22-31.

/KRO 91/ Kröhn, K.-P.: Simulation von Transportvorgängen im klüftigen Gestein mit der Methode der Finiten Elemente. Bericht 29/1991, Institut für Strömungsmechanik und Elektronisches Rechnen im Bauwesen, Universität Hannover, 1991.

/KRO 10/ Kröhn, K.-P.: State Variables for Modelling Thermohaline Flow in Rocks. GRS-268 BMWi-FKZ 02 E 10336, 92 p., Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) GmbH, Braunschweig (2010)

/KRO 16a/ Kröhn, K.-P.: Hydraulic Interaction of Engineered and Natural Barriers – Task 8b-8d,8f of SKB. Summary report, FKZ 02 E 11213 and 02 E 11102 (BMWi), Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Köln, 2016. (in preparation)

/KRO 16b/ Kröhn, K.-P.: Hydraulic Interaction of Engineered and Natural Barriers – Task 8e of SKB. Summary report, FKZ 02 E 11213 and 02 E 11102 (BMWi), Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) mbH, Köln, 2016. (in preparation)

/KUL 02/ Kull, H. (ed.), Helmig, R., Jacobs, H., Jockwer, N., Kröhn, K.-P., Zimmer, U.: Two-Phase-Flow Experiment in the Fractured Crystelline Rock of the Äspö Hard Rock laboratory. Final report FKZ 02 E 9027 8 (BMFT), Report GRS-183, GRS Braunschweig, 2002.

/KUZ 08/ Kuzyk, G.W. and Martino, J.B.: URL Excavation Design, Construction and Performance. Report NWMO TR-2008-17, Nuclear Waste Management Organization (NMWO), Toronto 2008.

/LAC 01/ Lacroix, S., Vassilevski, Y. V., Wheeler, M. F.: Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS), Numerical Linear Algebra with Applications 8 (8) (2001) 537–549. doi:10.1002/nla.264.

/LAN 06/ Langevin, C. D., Guo, W.: MODFLOW/MT3DMS-Based Simulation of Variable-Density Ground Water Flow and Transport, Ground Water 44 (3) (2006) 339–351. doi:10.1111/j.1745-6584.2005.00156.x .

/LAN 08/ Langevin, C. D., Daniel, J., Thorne, T., Dausman, A. M., Sukop, M. C., Guo, W.: U.S. Geological Survey Techniques and Methods Book 6, U.S. Geological Survey, Reston, Virginia, 2008, Ch. A22: SEAWAT Version 4: A Computer Program for Simulation of Multi-Species Solute and Heat Transport.

/LAN 05/ Lang, S., Wittum, G.: Large scale density driven flow simulations using parallel unstructured grid adaptation and local multigrid methods., Concurrency Computat. 17 (11) (2005) 1415–1440.

/LEI 92/ Leijnse, A.: Three-dimensional modeling of coupled flow and transport in porous media., Ph.D. thesis, University of Notre Dame, Indiana (1992).

/LU 09/ Lu, B., Wheeler, M.: Iterative coupling reservoir simulation on high performance computers, Petroleum Science 6 (2009) 43–50 doi:10.1007/s12182-009-0008-x.

/MAT 06/ Matthies, H., Niekamp, R., Steindorf, J.: Algorithms for strong coupling procedures, Computer Methods in Applied Mechanics and Engineering 195 (17-18) (2006) 2028–2049. doi:10.1016/j.cma.2004.11.032 .

/MIK 13/ Mikelicʹ, A., Wheeler, M. F.: Convergence of iterative coupling for coupled flow and geomechanics, Computational Geosciences 17 (3) (2013) 455–461.

/MIK 14/ Mikelicʹ, A., Wang, B., Wheeler, M. F.: Numerical convergence study of iterative coupling for coupled flow and geomechanics, Computational Geosciences (2014) 1–17 doi:10.1007/s10596-013-9393-8.

/MOR 08/ Moreno, L. , Crawford, J.: Can we use tracer tests to obtain data for performance assessment of repositories for nuclear waste? Hydrogeology Journal 17: 1067–1080, 2009.

/NAE 15/ Nägel, A., Vogel, A., Wittum, G.: Evaluating linear and nonlinear solvers for density driven flow, Computer Methods in Applied Mechanics and Engineering, Elsevier BV, 2015, 292, 3-15.

/NOS 12/ Noseck, U., Brendler, V., Flügge, J., Stockmann, M., Britz, S., Lampe, M., Schikora, J., Schneider, A..: Realistic integration of sorption processes in transport codes for long-term safety assessments, Final Report GRS-297, BMWi-FKZ 02E10548. Gesellschaft für Anlagen- und Reaktorsicherheit (mbH), Braunschweig, (2012), 293 p.

/POP 85/ Pope, S. B.: PDF methods for turbulent reactive flows, Prog. Energy Combust. Sci. 11(2), 119-192, 1985.

/POP 14/ Popov, P. P., and S. B. Pope (2014), Implicit and explicit schemes for massconsistency preservation in hybrid particle/finite-volume algorithms for turbulent reactive flows, J. Comput. Phys., 257, 352-373.

/POT 06/ Poteri, A.: Äspö Task Force on modelling of groundwater flow and transport of solutes; Modelling of Task 6D, 6E, 6F and 6F2 using the Posiva streamtube approach. International Progress Report IPR-06-17, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/POT 09/ Poteri, A.: Retention properties of flow paths in fractured rock. Hydrogeology Journal 17: 1081–1092, 2009.

/PUT 95/ Putti, M., Paniconi, C.: Picard and newton linearization for the coupled model for saltwater intrusion in aquifers, Advances in Water Resources 18 (3) (1995) 159–170. doi:10.1016/0309-1708(95)00006-5.

/RAD 11/ Radu, F. A., N. Suciu, J. Hoffmann, A. Vogel, O. Kolditz, C.-H. Park, S. Attinger, Accuracy of numerical simulations of contaminant transport in heterogeneous aquifers: a comparative study, Adv. Water Res. 34 (1), 47-61, 2011.

/REI 04/ Reisinger, C., Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben. PhD Thesis, University of Heidelberg, 2004.

/REI 07/ Reisinger, C., Wittum, G.: Efficient Hierarchical Approximation of High-Dimensional Option Pricing Problems, SIAM Journal on Scientific Computing, 29(1), 2007.

/REI 13/ Reiter, S., Vogel, A., Heppner, I., Rupp, M., Wittum, G.: A massively parallel geometric multigrid solver on hierarchically distributed grids. Computing and Visualization in Science 16 (4), 151-164, 2013.

/REI 14/ Reiter, S.: Effiziente Algorithmen und Datenstrukturen für die Realisierung von adaptiven, hierarchischen Gittern auf massiv parallelen Systemen. Dissertation, Universität Frankfurt, 2014.

/RHE 98/ W. C. Rheinboldt, Methods for Solving Systems of Nonlinear Equations, 2nd Edition, SIAM, 1998.

/SCH 12/ Schneider, A. (ed.): Enhancement of the codes d3f and r3t. GRS-292 BMWi-FKZ 02 E 10336 , 365 p.; Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) GmbH, Braunschweig, 2012.

/SCH 13/ Schneider, A. (ed.), Representation of inhomogeneities in the flow and transport codes d^3f and r^3t, GRS 311, Braunschweig, 2013.

/SUC 06/ Suciu, N., C. Vamos, J. Vanderborght, H. Hardelauf, H. Vereecken: Numerical investigations on ergodicity of solute transport in heterogeneous aquifers, Water Resour. Res., 42 (4), 2006.

/SUC 13/ Suciu, N., F. A. Radu, A. Prechtel, F. Brunner, P. Knabner: A coupled finite element-global random walk approach to advection-dominated transport in porous media with random hydraulic conductivity, J. Comput. Appl. Math., 246, 27-37, 2013.

/SUC 15/ Suciu, N., F. A. Radu, S. Attinger, L. Schüler, and P. Knabner: A Fokker-Planck approach for probability distributions of species concentrations transported in heterogeneous media, J. Comput. Appl. Math., 289, 241-252, 2015.

/SVE 06/ Svensson, U.: Äspö Task Force on modelling of groundwater flow and transport of solutes; Modelling of Task 6D, 6E, 6F and 6F2. Flow, transport and retention in a sparsely fractured granite. International Progress Report IPR-06-21, Swedish Nuclear Fuel and Waste Management Company (SKB), 2006.

/TAN 03/ Tanaka, Y.: Preliminary results of application of FEGM to Task 6D. Presentation at the 17th Workshop of SKB's Task Force on Groundwater Flow and Transport of Solutes, Thun, Switzerland, 2003. (Cited in /HOD 07/)

/UCH 09/ Uchida, M., Dershowitz, W., Lee, G., Shuttle, D.: An empirical probabilistic approach for constraining the uncertainty of long-term solute transport predictions in fractured rock using in situ tracer experiments. Hydrogeology Journal 17: 1093–1110, 2009.

/VAM 03/ Vamos, C., N. Suciu, H. Vereecken: Generalized random walk algorithm for the numerical modeling of complex diffusion processes, J. Comp. Phys. 186 (2), 527-544, 2003

/VAM 12/ Vamos, C., M. Craciun: Automatic Trend Estimation, Springer, Dortrecht, 2012.

/VOG 13/ Vogel, A., Reiter, S., Rupp, M., Nägel, A., Wittum, G.: UG 4: A novel flexible software system for simulating PDE based models on high performance computers. Computing and Visualization in Science 16 (4), 165-179, 2013.

/VOG 14/ Vogel, A.: Flexible und kombinierbare Implementierung von Finite-Volumen-Verfahren höherer Ordnung mit Anwendungen für die Konvektions-Diffusions-, Navier-Stokes- und Nernst-Planck- Gleichungen sowie dichtegetriebene Grundwasserströmung in porösen Medien. Doktorarbeit, Universität Frankfurt am Main, 2014.

/VTK 06/ Schröder, W., Martin, K., Lorensen, B.: Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition, Kitware; 2006.

/WIK 10/ http://en.wikipedia.org

/WIN 02/ Winberg, A., Andersson, P., Byegård, J., Poteri, A.,Cvetkovic, V., Dershowitz, B., Doe, T., Hermanson, J., Gómez-Hernández, J-J., Hautojärvi, A., Billaux, D.,Tullborg, E-L., Meier, P. and A. Medina:TRUE Block Scale Project; FinalReport – 4. Synthesis of flow, transport and retention in the block scale.Technical Report TR-02-16, Swedish Nuclear Fuel and Waste Management Company (SKB),2002.

/ZEN 91/ Zenger, C.: Sparse Grids. Parallel Algorithms for Partial Differential Equations. In: Hackbusch, W. (ed.): Notes on Numerical Fluid Dynamics 31. Proceedings of the Sixth GAMM-Seminar, 1990.

Table of figures

List of tables

A Notation

The most general notations used in this report are given below.

c	-	volumetric solute concentration [mol m^{-3}]
C	-	specific heat capacity [J kg^{-1} K^{-1}]
$\mathbf{D}$	-	hydrodynamic-dispersion tensor $\{ \mathbf{D} = \phi\, D_m\, \mathbf{T} + \mathbf{D}_{\text{disp}}(\mathbf{q}) \}$ $[m^2 s^{-1}]$
$\mathbf{D}_{\text{disp}}$	-	mechanical dispersion tensor [m^2 s^{-1}]
D_m	-	molecular diffusion coefficient [m^2 s^{-1}]
$\mathbf{g}$	-	gravity [m s^{-2}]
$\mathbf{I}$	-	identity matrix [-]
$\mathbf{J}_{\text{d}}$	-	diffusive mass flux [kg m^{-2} s^{-1})]
$\mathbf{J}_{\text{T}}$	-	heat flux vector [J m^{-2} s^{-1})]
$\mathbf{k}$	-	permeability tensor [m^2]
K_d	-	distribution coefficient [m^3 kg^{-1}]
p	-	hydraulic pressure [Pa]
$\mathbf{q}$	-	Darcy velocity vector [m s^{-1}]
Q	-	mass source/sink terms [kg m^{-3} s^{-1}]
$\mathbf{T}$	-	tortuosity tensor [-]
$T_{1/2}$	-	half-life [s]
$\mathbf{u}$	-	flow velocity vector [m s^{-1}]
α_L	-	longitudinal dispersion length [m]
α_T	-	transverse dispersion length [m]
Θ	-	temperature [K]
λ	-	decay constant $[s^{-1}]$
Λ	-	thermal conductivity [W m^{-1} K^{-1}]
μ	-	viscosity [Pa s]

ϱ	-	density [kg m^{-3}]
ϕ	-	porosity [-]
ω	-	solute mass fraction [-]

B Viscosity-dependent flow

One characteristic of the flow field obtained from flow simulations for the test case Mayak is a divide in the upper three hydrogeological units where a part of the water leaves the model area to the left, whereas another part leaves it to the right. This characteristic shall be explained in the following section. To simplify matters, the effect is studied in a system without fractures.

Neglecting the fractures the flow field is controlled by the ration of groundwater recharge and horizontal flow imposed by the Dirichlet boundary conditions on the left and right boundary. Since the recharge is assumed to fixed this ratio can be changed only by varying the

- horizontal hydraulic gradient,
- permeability, or
- viscosity of water (changes of the density are not considered here)

In this context, the permeability and the viscosity determine whether the vertical component introduced by the groundwater recharge or the horizontal component caused by the hydraulic gradient dominates the flow field. Also the relation between the groundwater recharge and the hydraulic gradient plays a role.

In the following, it shall be looked at the permeability and the viscosity. The permeability is varied in the upper two hydrogeological units to both higher and lower values compared to the basic scenario described in Chapter 5.1.1. Three different values for the viscosity are applied for the whole model area mirroring the conditions at 2 °C, 20 °C and 32 °C according to (5.1). The resulting flow fields are shown in Fig. B.1 and Fig. B.2, respectively.

Flow fields for different distributions of the permeability are shown in Fig. B.1. If the permeability is rather low or the groundwater recharge is high compared to the applied horizontal hydraulic gradient, water enters the model area exclusively through the top boundary and leaves it in equal parts to the right and left forming a vertical divide in the middle of the model area (Fig. B.1a). This divide shifts to the left for higher permeability values (Fig. B.1b). If the permeability goes beyond a certain value or if the hydraulic gradient is high compared to the recharge, the horizontal component dominates the flow field and a flow from the left to the right occurs (Fig. B.1c).

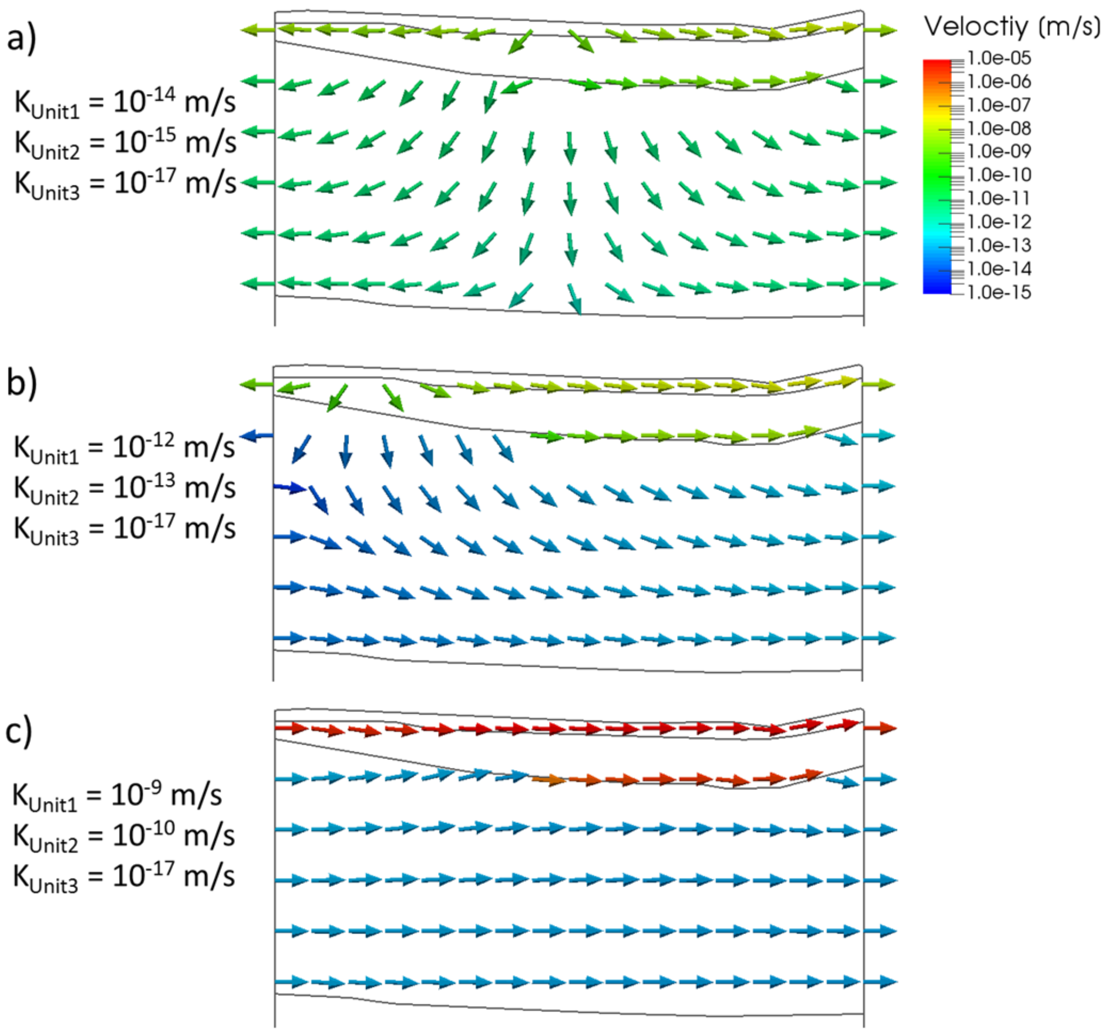


Fig. B.1 Flow fields obtained with different distributions of the permeability

A similar effect is generated by varying the viscosity, only that the relation between hydraulic conductivity and viscosity is reciprocal. The values used to obtain the flow fields in Fig. B.2 correspond to temperatures of 2 °C (Fig. B.2a), 20 °C (Fig. B.2b) and 32 °C (Fig. B.2c). However, the values vary only slightly compared to the permeability values used above so that the effect is considerably smaller.

Nevertheless, also in these flow fields a divide occurs at the top boundary (Fig. B.2a) and is shifted to the left using decreasing values for the viscosity (Fig. B.2b and c). Simultaneously, also the amount of fluid that leaves the model area to the left decreases and the amount that leaves to the right increases.

Further simulations showed that viscosity related by (5.1) to a temperature gradient of 2 °C at the top and 32 °C at the bottom of the model results in a flow field similar to Fig. B.**2**a which corresponds to a temperature of 2 °C. Thus, it seems that the viscosity in the upper hydrogeological units is crucial to the flow pattern that evolves.

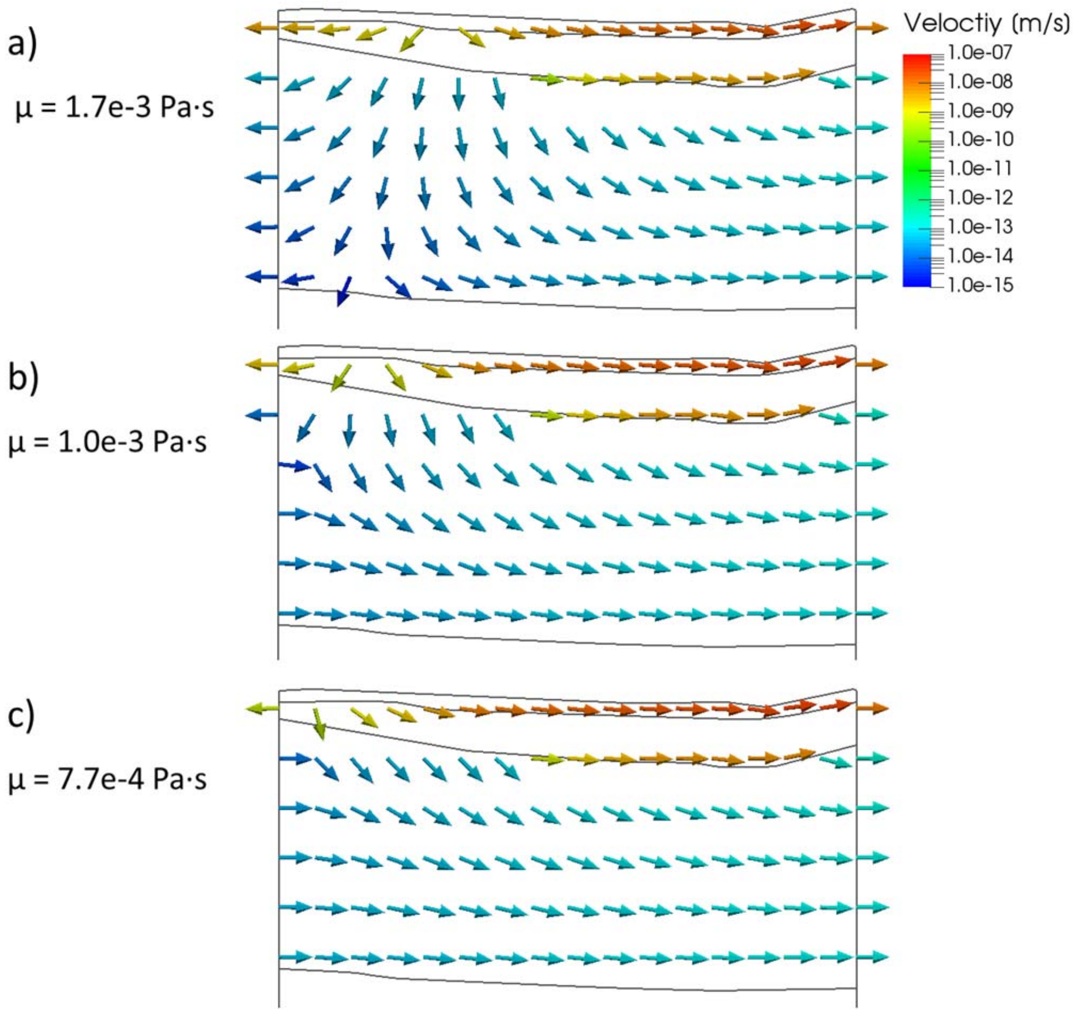


Fig. B.2 Flow fields obtained with different values for the viscosity

C Data for the Task 6 model

Tab. C.1 Coordinates of measured deterministic fractures

structure	easting [m]		northing [m]		elevation [m amsl]	
	original	clipped	original	clipped	original	clipped
5	1736.03	1871.32	7329.37	7270.00	-200.00	-350.00
	2150.00	2000.00	7151.68	7214.77	-200.00	-350.00
	2150.00	2000.00	7147.35	7213.04	-700.00	-550.00
	1849.52	1867.29	7276.33	7270.00	-700.00	-550.00
6	1894.27	as original	7259.38	as original	-438.46	as original
	1898.04		7257.10		-515.54	
	1943.68		7185.50		-515.54	
	1942.01		7184.49		-438.46	
7	1885.40	as original	7237.62	as original	-420.87	as original
	1877.60		7222.43		-533.13	
	1978.11		7172.43		-533.13	
	1985.91		7187.62		-420.87	
10	1799.34	1800.00	7084.83	7084.77	-414.76	-414.76
	1807.46	1800.00	7125.05	7088.10	-539.24	-424.88
	1931.36	1807.46	7113.05	7125.05	-539.24	-539.24
	1923.24	1931.36	7072.83	7113.05	-414.76	-539.24
		1923.24		7072.83		-414.76
13	1844.39	as original	7198.82	as original	-397.01	as original
	1890.79		7234.04		-530.33	
	1955.21		7149.04		-530.33	
	1908.82		7113.82		-397.01	
19	1794.96	1822.89	7316.79	7270.00	-395.48	-395.48
	1813.76	1825.15	7289.29	7270.00	-558.52	-550.00
	1941.28	1942.19	7075.63	7073.89	-558.52	-550.00
	1958.78	1944.24	7042.31	7070.00	-395.48	-530.97
		1942.25		7070.00		-395.48
20	1873.00	as original	7224.29	as original	-380.00	as original
	1883.44		7233.52		-537.06	
	1962.98		7143.52		-537.06	
	1952.54		7134.29		-380.00	
21	1908.28	as original	7235.88	as original	-433.46	as original
	1881.06		7224.18		-520.54	
	1915.45		7144.18		-520.54	
	1942.67		7155.88		-433.46	
22	1933.48	as original	7211.17	as original	-439.65	as original
	1903.29		7196.85		-526.80	
	1924.62		7151.85		-526.80	
	1954.81		7166.17		-439.65	
23	1926.76	as original	7198.00	as original	-452.47	as original
	1926.76		7198.00		-501.53	
	1943.43		7180.00		-501.53	
	1943.43		7180.00		-452.47	

24	1931.11	as original	7220.00	as original	-459.97	as original
	1923.34		7220.00		-494.03	
	1949.34		7198.00		-494.03	
	1957.10		7198.00		-459.97	

Tab. C.2 Coordinates of synthetic deterministic fractures

structure	easting	northing	elevation	structure	easting	northing	elevation
1S	2000,00	7144,09	-518,82	13S	2000,00	7098,52	-350,00
	1988,50	7152,26	-518,73		2000,00	7093,43	-462,08
	1998,41	7164,92	-407,44		1886,75	7140,90	-427,98
	2000,00	7163,79	-407,46		1908,09	7135,79	-350,00
3S	1967,54	7167,72	-550,00	14S	1943,25	7070,00	-426,75
	1851,89	7270,00	-550,00		1940,37	7070,63	-418,98
	1846,00	7270,00	-525,30		1941,97	7070,00	-418,34
	1966,04	7165,00	-530,80				
4S	1800,00	7270,00	-490,05	15S	1984,20	7270,00	-350,00
	1800,00	7264,18	-471,84		2000,00	7262,59	-350,00
	1849,24	7245,23	-487,83		2000,00	7265,99	-455,46
	1838,78	7270,00	-549,39		1991,53	7270,00	-456,66
6S	1928,78	7270,00	-350,00	17S	1800,00	7249,89	-350,00
	1968,34	7239,78	-350,00		1829,13	7248,04	-350,00
	2000,00	7207,97	-421,90		1900,37	7238,94	-425,00
	2000,00	7207,23	-428,90		1800,00	7239,55	-520,41
	1908,37	7270,00	-497,02				
7S	1969,48	7088,66	-550,00	18S	1913,80	7270,00	-350,00
	1929,53	7269,74	-550,00		1915,76	7268,70	-350,00
	1929,55	7270,00	-549,53		1969,83	7224,69	-502,29
	1936,92	7270,00	-503,11		1898,19	7270,00	-540,82
	1973,86	7148,75	-438,96				
8S	1913,12	7070,00	-550,00	20S	1890,40	7070,00	-350,00
	1800,00	7147,47	-550,00		1904,07	7070,00	-366,00
	1800,00	7148,98	-410,96		1884,90	7126,63	-396,50
	1913,75	7070,00	-510,78		1837,86	7135,75	-350,00
9S	1800,00	7220,75	-350,00	21S	1800,00	7253,47	-550,00
	1802,23	7219,19	-350,00		1800,00	7249,86	-526,09
	1829,53	7220,51	-484,26		1840,89	7206,24	-550,00
	1800,00	7242,03	-490,06				
10S	1878,97	7170,65	-350,00	24S	1800,00	7196,89	-533,05
	1928,15	7070,00	-350,00		1856,41	7070,00	-515,95
	1931,88	7070,00	-367,60		1849,30	7070,00	-550,00
	1901,94	7149,02	-408,54		1800,00	7188,38	-550,00
11S	1879,46	7270,00	-405,10	25S	1800,00	7257,32	-350,00
	1954,11	7194,85	-550,00		1876,56	7160,15	-350,00
	1889,59	7270,00	-550,00		1800,00	7231,02	-440,42
12S	1878,81	7070,00	-515,33	First node of 18S lies not in the plane of the following nodes and is thus skipped			
	1876,14	7078,44	-502,67				
	1895,09	7070,00	-493,05				

Tab. C.3 Hydraulic properties of deterministic fractures

structure	transmissivity [m²/s]	aperture[12] [m]	permeability [m²]
5	4,020E-07	2,917E-04	1,405E-10
6	1,910E-07	2,010E-04	9,687E-11
7	9,760E-08	1,437E-04	6,923E-11
10	2,980E-08	7,941E-05	3,825E-11
13	1,380E-08	5,404E-05	2,603E-11
19	1,020E-07	1,469E-04	7,078E-11
20	1,430E-07	1,740E-04	8,378E-11
21	6,020E-08	1,129E-04	5,435E-11
22	2,190E-08	6,807E-05	3,280E-11
23	1,660E-07	1,874E-04	9,030E-11
24	8,510E-08	1,342E-04	6,464E-11
1S	3,140E-07	2,576E-04	1,219E-10
3S	2,290E-06	6,965E-04	3,288E-10
4S	1,900E-07	2,007E-04	9,466E-11
6S	6,930E-07	3,830E-04	1,809E-10
7S	7,060E-07	3,865E-04	1,827E-10
8S	1,490E-06	5,611E-04	2,656E-10
9S	3,080E-07	2,552E-04	1,207E-10
10S	5,710E-07	3,475E-04	1,643E-10
11S	2,010E-06	6,528E-04	3,079E-10
12S	1,240E-06	5,131E-04	2,417E-10
13S	5,080E-06	1,037E-03	4,899E-10
14S	6,720E-07	3,772E-04	1,782E-10
15S	1,600E-06	5,810E-04	2,754E-10
17S	1,540E-06	5,701E-04	2,701E-10
18S	1,160E-06	4,946E-04	2,345E-10
20S	1,350E-06	5,336E-04	2,530E-10
21S	1,040E-06	4,688E-04	2,218E-10
24S	9,920E-07	4,581E-04	2,165E-10
25S	3,960E-06	9,158E-04	4,324E-10

[12] transport relevant aperture, being 1/8 of hydraulic aperture

Tab. C.4 Injection history

Elapsed time [h]	C [Bq/kg]	Elapsed time [h]	C [Bq/kg]	Elapsed time [h]	C [Bq/kg]
0,44	9,54E+05	13,83	8,89E+06	80,72	2,26E+05
0,48	2,45E+06	14,34	8,48E+06	82,72	1,99E+05
0,51	5,42E+06	14,84	8,00E+06	84,72	1,64E+05
0,55	9,50E+06	15,34	7,56E+06	86,73	1,56E+05
0,59	1,41E+07	15,84	7,23E+06	88,73	1,72E+05
0,62	1,85E+07	16,35	6,91E+06	90,73	1,13E+05
0,66	2,22E+07	16,85	6,56E+06	92,74	1,43E+05
0,69	2,45E+07	17,35	6,18E+06	94,74	1,38E+05
0,73	2,76E+07	17,85	5,94E+06	96,75	1,06E+05
0,76	2,94E+07	18,36	5,64E+06	98,75	1,44E+05
0,80	3,15E+07	18,86	5,41E+06	100,75	9,83E+04
0,83	3,35E+07	19,36	5,08E+06	102,75	1,08E+05
0,87	3,44E+07	20,61	4,47E+06	104,76	1,11E+05
0,90	3,64E+07	22,61	3,63E+06	106,76	1,22E+05
0,94	3,79E+07	24,62	2,91E+06	108,76	1,39E+05
0,98	3,80E+07	26,63	2,35E+06	110,76	1,69E+05
1,01	3,63E+07	28,63	1,97E+06	112,76	1,81E+05
1,24	3,30E+07	30,64	1,59E+06	114,77	2,05E+05
1,75	3,12E+07	32,64	1,35E+06	116,77	2,34E+05
2,26	2,92E+07	34,65	1,15E+06	118,77	2,85E+05
2,76	2,77E+07	36,65	1,06E+06	120,78	2,92E+05
3,27	2,62E+07	38,65	9,78E+05	122,78	2,83E+05
3,77	2,47E+07	40,66	8,99E+05	124,60	2,78E+05
4,28	2,34E+07	42,66	8,69E+05	127,40	2,08E+05
4,78	2,24E+07	44,66	8,07E+05	131,41	1,47E+05
5,28	2,14E+07	46,67	7,32E+05	135,41	1,19E+05
5,79	2,01E+07	48,67	7,06E+05	139,42	1,00E+05
6,29	1,92E+07	50,67	6,98E+05	143,42	9,06E+04
6,79	1,81E+07	52,68	7,09E+05	147,42	7,89E+04
7,30	1,71E+07	54,68	7,31E+05	151,42	6,86E+04
7,80	1,65E+07	56,68	7,03E+05	159,43	6,07E+04
8,30	1,55E+07	58,68	6,82E+05	171,44	6,18E+04
8,81	1,48E+07	60,69	6,71E+05	179,45	7,25E+04
9,31	1,39E+07	62,69	6,57E+05		
9,81	1,32E+07	64,69	6,36E+05		
10,31	1,26E+07	66,69	6,15E+05		
10,82	1,19E+07	68,70	5,98E+05		
11,32	1,14E+07	70,70	5,22E+05		
11,82	1,07E+07	72,70	4,36E+05		
12,33	1,02E+07	74,71	3,64E+05		
12,83	9,75E+06	76,71	2,90E+05		
13,33	9,24E+06	78,71	2,95E+05		

Tab. C.5 Extraction concentrations

Elapsed time [h]	C [Bq/kg]	Elapsed time [h]	C [Bq/kg]	Elapsed time [h]	C [Bq/kg]
37,88	70,02	121,88	3655,75	297,17	2526,12
39,88	91,15	123,88	3430,74	301,17	2483,65
41,88	129,41	125,02	3833,57	305,17	2360,97
43,88	181,85	129,03	3562,69	309,17	2401,65
45,88	258,72	133,02	3847,86	321,55	2082,17
47,88	307,04	137,03	3792,36	333,55	1945,74
49,88	363,91	141,03	3294,10	345,55	1908,74
51,88	461,23	145,03	3246,17	357,54	1773,43
53,88	587,23	149,03	3717,57	369,54	1779,78
55,88	614,55	153,02	3150,95	381,54	1674,82
57,88	725,80	157,00	3404,15	393,53	1564,90
59,88	909,40	173,19	4960,98	405,53	1546,99
61,88	984,42	177,19	5157,41	417,53	1488,56
63,88	1095,03	181,19	5148,77	429,53	1405,23
65,88	1182,88	185,19	5169,04	441,52	1270,19
67,88	1348,96	189,19	5219,89	453,52	1270,16
69,88	1527,51	193,19	5014,16	465,52	1644,08
71,88	1508,80	197,19	4873,89	477,51	1308,08
73,88	1677,21	201,19	4787,48	489,51	1137,64
75,88	1728,36	205,18	4690,09	501,51	1258,13
77,88	1891,79	209,18	4580,08		
79,88	2017,03	213,18	4462,09		
81,88	2229,05	217,18	4339,77		
83,88	2259,00	221,18	4089,17		
85,88	2279,17	225,18	4100,47		
87,88	2415,71	229,18	4108,44		
89,88	2493,42	233,18	4049,61		
91,88	2644,78	237,18	3814,63		
93,88	2579,00	241,18	3609,55		
95,88	2766,19	245,18	3765,10		
97,88	2875,98	249,18	3514,16		
99,88	3029,70	253,18	3395,17		
101,88	3305,83	257,18	3280,61		
103,88	3351,97	261,18	3110,13		
105,88	3388,53	265,17	3097,54		
107,88	3529,45	269,17	3046,72		
109,88	3544,41	273,17	2967,31		
111,88	3621,85	277,17	2737,81		
113,88	2715,41	281,17	2760,15		
115,88	3678,35	285,17	2514,42		
117,88	3734,53	289,17	2623,87		
119,88	2044,85	293,17	2441,04		

Gesellschaft für Anlagen- und Reaktorsicherheit (GRS) gGmbH

Schwertnergasse 1
50667 Köln
Telefon +49 221 2068-0
Telefax +49 221 2068-888

Forschungszentrum
Boltzmannstraße 14
85748 Garching b. München
Telefon +49 89 32004-0
Telefax +49 89 32004-300

Kurfürstendamm 200
10719 Berlin
Telefon +49 30 88589-0
Telefax +49 30 88589-111

Theodor-Heuss-Straße 4
38122 Braunschweig
Telefon +49 531 8012-0
Telefax +49 531 8012-200

www.grs.de

ISBN 978-3-944161-73-0