DuMu^{x}: DUNE for Multi-{Phase, Component, Scale, Physics, ...} Flow and Transport in Porous Media

B. Flemisch[a], M. Darcis[a], K. Erbertseder[a], B. Faigle[a], A. Lauser[a], K. Mosthaf[a], S. Müthing[b], P. Nuske[a], A. Tatomir[a], M. Wolff[a], R. Helmig[a]

[a] *Universität Stuttgart, Department of Hydromechanics and Modeling of Hydrosystems, Pfaffenwaldring 61, 70569 Stuttgart, Germany*
[b] *Universität Stuttgart, Institute for Visualization and Interactive Systems, Universitätsstraße 38, 70569 Stuttgart, Germany*

Abstract

DuMu^{x} is a free and open-source simulator for flow and transport processes in porous media, based on the Distributed and Unified Numerics Environment DUNE. Its main intention is to provide a sustainable and consistent framework for the implementation and application of model concepts, constitutive relations, discretizations, and solvers. The paper provides an overview of DuMu^{x} with the focus on software-related aspects. Selected examples highlight the multi-scale and the parallel capabilities.

Keywords: porous media simulator, open source, multi-scale, multi-physics

1. Introduction

The quality of any type of computational modeling crucially depends on the quality of the employed software framework. Research codes very often fail to be developed and maintained in a continuous manner. On the contrary, software development at academic institutions usually is highly fragmented and driven by individual short-term needs. Furthermore, work is often done redundantly, diverting resources from the original focus of research projects by a need for reinventing the wheel. We are therefore convinced that the free and open source (FLOSS) idea provides a chance for sustainable high quality software development also in academia.

DuMu^{x} is a simulator for flow and transport processes in porous media. It is built on top of DUNE, the Distributed and Unified Numerics Environment, a modular toolbox for solving partial differential equations with grid-based

methods, [1, 2]. DuMu^x is licensed under the terms and conditions of the GNU General Public License (GPL) version 2 or later, [3]. Stable releases are available for download, [4], and anonymous read-access to the Subversion repository is granted.

DuMu^x includes several standard models of varying complexity, ranging from stationary isothermal single-phase single-component flow to transient non-isothermal multi-phase compositional flow. Active research is currently undertaken to include multi-scale and multi-physics concepts, as well as non-standard formulations like multiple continua approaches or models involving interfacial area as primary state variable. All models employ efficient nonlinear solvers in close combination with a sophisticated time step management. The capabilities of DUNE are heavily exploited to offer various spatial discretization schemes as well as the possibility of parallel computations. The applications currently targeted by DuMu^x include fuel cells, groundwater remediation, evaporation from partially saturated soils, CO_2 storage, and drug delivery into human tissue. Several scientists from diverse areas of expertise (computer science, engineering, mathematics) are involved in the code development.

Various porous media simulators are under ongoing development, we list some examples in the following. Most prominently, ECLIPSE is a simulation tool used extensively in the oil and gas industry, [5]. Another commercial tool is the Generalised Equation-of-state Model compositional reservoir simulator (GEM), [6]. Finite Element Heat and Mass Transfer Simulator (FEHM) is a porous media fluid flow simulator developed by the Los Alamos National Laboratory, [7]. Stanford's General Purpose Research Simulator (GPRS) serves as a research platform for reservoir simulation, [8, 9]. Integrated Parallel Accurate Reservoir Simulators (IPARS), developed at the University of Texas, is a framework for parallel models of subsurface flow and transport through porous media, [10]. OpenGeoSys is a project for the development of numerical methods for the simulation of thermo-hydro-mechanical-chemical processes in porous and fractured media, [11]. MUFTE-UG, the predecessor of DuMu^x, is capable of solving isothermal and non-isothermal multi-phase flow problems including compositional effects, [12]. The simulation code TOUGH2 was developed by researchers from Lawrence Berkeley National Laboratory, [13]. The MATLAB Reservoir Simulation Toolbox (MRST) is developed by SINTEF Applied Matemathics and is a result of their research on the development of new (multiscale) computational methodologies, [14]. The distribution and licensing policies for the listed simulators vary from pro-

prietary commercial to open source and free of charge. From the list, only MRST is released under a GPL license from the Free Software Foundation.

This paper is structured as follows: in the remaining part of this introduction, the vision, concept and design ideas behind DuMu^x are presented. In Section 2, the common base of all DuMu^x models is outlined. Section 3 describes the available models in DuMu^x. In Section 4 some examples highlighting the capabilites of DuMu^x are given. We summarize and give an outlook in Section 5. Concerning the notation it should be mentioned that quantities that have a direct representation within the code base of DuMu^x — be it a class name or the name of a folder containing a model — are set in `typewriter`.

1.1. Vision

Up to now, DuMu^x is an academic research code and thus primarily targeted towards researchers and particularly PhD students to code, test and apply new mathematical and numerical modeling approaches. Thanks to the abstraction principles employed in the DUNE framework, this can be achieved without any knowledge of the underlying detailed implementations. Still, a profound knowledge of advanced C++ programming techniques is required from the current users and developers. In the future, the capabilities of DuMu^x are expected to attract end-users, who are mainly interested in applying existing numerical models to their concrete problem at hand. For this group of users which includes prospective bachelor and master students as well as partners from industry, it will be important to offer a framework allowing general problem descriptions and a model selection without requiring in-depth programming knowledge.

It is evident that only one research group cannot cover all aspects of the computational modeling of porous media flow and transport processes. Therefore, it is important to combine the expertise from different groups, and free and open source software development is the most natural way to achieve a sufficiently flawless integration of individual solution components. For this purpose, DuMu^x is part of the recently funded Open Porous Media (OPM) initiative, [15], which is dedicated to develop a simulation suite that is capable of modeling industrially and scientifically relevant flow and transport processes in porous media and to bridge the gap between the different application areas of porous media modeling.

1.2. Concepts and Design Ideas

Modularity is the leitmotif to design the code. DuMu^x provides shelves of modularized objects, enabling the user to choose the appropriate parts according to the problem at hand. This can be compared to somebody, who can easily grab a different shirt out of a shelf without changing the trousers and without ending up with a combination of clothes that do not match. Following that design idea, DuMu^x is meant to provide a collection of shelves each holding interchangeable alternatives, that are still fully interactive to other shelves by using common interfaces. The user is able to select each part of the implementation at each shelve through an efficient compile-time `property system` (Section 2.4). Part of this modular setup (see Figure 1) are the shelves from which to choose

- model concepts (Section 3),
- numerical schemes,
- control strategies for the simulation (Section 2.3),
- multitude of substances, material laws (Section 2.2),
- small and large-scale examples and applications.

Furthermore, the following design principles are of high importance:

1. Like DUNE, DuMu^x is coded in C++ and employs high-level generic programming techniques. The design principles guiding the DUNE development, see Section 2.1, are also closely followed within DuMu^x.

2. The whole project is free and open source, and fully available to the public. By this, the project is open to all kinds of participation from different areas of expertise, which attracts new streams of ideas to improve DuMu^x.

3. A framework of this size (several 100,000 lines of code) has to be in good maintenance requiring a sound infrastructure: a common repository with a version control system through Subversion (SVN), [16], an online bugtracking and feature request system, [17], and the regular submission of build and test results to a dashboard, [18].

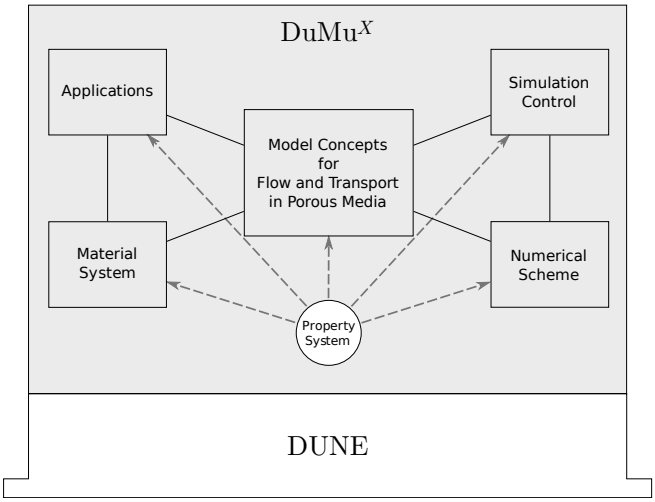


Figure 1: Modular design of DuMu^x.

2. The Backbone of DuMu^x

This section deals with the common structures employed by most DuMu^x models. Most prominently, these are the grid, solver and discretization interfaces provided by the DUNE framework described in Subsection 2.1. Of crucial importance for the ability to solve real life porous media flow problems is a flexible and extendable material system, introduced in Subsection 2.2, as well as a sophisticated simulation control, which is outlined in Subsection 2.3. Finally, we propose a convenient alternative to traits classes in Subsection 2.4.

2.1. DUNE

DUNE, the Distributed and Unified Numerics Environment, is a modular toolbox for solving partial differential equations with grid-based methods, [1, 2]. To quote from [19]:

> The underlying idea of DUNE is to create slim interfaces allowing an efficient use of legacy and/or new libraries. Modern C++ programming techniques enable very different implementations of the same concept (i.e. grids, solvers, ...) using a common interface at a very low overhead. Thus DUNE ensures efficiency in scientific computations and supports high-performance computing applications. DUNE is based on the following main principles:

- Separation of data structures and algorithms by abstract interfaces.
- Efficient implementation of these interfaces using generic programming techniques.
- Reuse of existing finite element packages with a large body of functionality.

DUNE is organized as a modular system. The current release 2.0 includes the core modules dune-common (basic classes), dune-grid (grid interface and implementations), dune-istl (iterative solver template library), and dune-localfunctions (interface for finite element shape functions). In addition to these, DuMu^x also uses the DUNE external module dune-pdelab which provides a large variety of finite element function spaces, global assembly of residuals and operators, linear and nonlinear solvers as well as explicit and implicit time discretizations based on the method of lines approach, [20]. Moreover, for the multi-scale and multi-physics approaches, the external module dune-multidomaingrid is employed which supplies a meta grid allowing the division of a given grid into separate sub-domains, [21].

The use of DUNE as basis on which DuMu^x is built on (Figure 1) offers several advantages. The most important one is the ability to use a wide range of different grid implementations and several linear solvers without having to care about the underlying data structures of the individual implementations. This particularly includes capabilities like parallelism and adaptivity, which comes at minimal additional programming cost for the user. Thus, the main part of the development of DuMu^x can concentrate on the implementation of physical and mathematical models. The key modules of DuMu^x (Figure 1) are introduced in the following subsections.

2.2. Material System

The biggest challenges in porous media simulation are the possibly highly heterogeneous distribution of parameters and the complex nonlinear material laws. The DuMu^x material system constitutes a framework that allows a convenient definition and usage of parameters and material laws. Due to the strong interconnection of these properties, it proves difficult to achieve modularity. Nevertheless, it is possible to achieve a modular structure by a separation into the following parts.

`Components`. The term component stands for constituents of the phases which can be associated with a unique chemical species, or, more generally, with a group of species exploiting similar physical behavior. Each component is implemented as a class consisting primarily of static member functions describing the physical properties of the component. This ranges from simple constants like the molar mass to possibly very complex functional relationships like the density depending on pressure and temperature.

`FluidSystems`. A `FluidSystem` describes the properties of the participating fluid phases. This includes phase densities and viscosities as well as fugacities and diffusion coefficients of components inside phases, where each phase may consist of one or more components. The properties of the fluid phases usually depend on their current composition which is described in a separate object of type `FluidState` containing the saturation and mole fraction values. A `FluidSystem` is implemented in the same way as a `Component`.

`FluidMatrixInteractions`. This part collects the material laws which are necessary for the description of the interaction of the fluid phases with the porous medium, i.e. capillarity and relative permeability. A collection of standard laws is provided, including e.g. VAN GENUCHTEN and BROOKS–COREY. For our extended model concepts, elaborate interfacial area - capillary pressure - saturation – relationships are included as well as standard hysteresis models. Through modular adapters, regularization schemes can be imposed for extreme values. Each material law uses a set of appropriately definable parameters of type `MaterialLawParams`, which may depend on the location inside the domain.

`SpatialParameters`. This part collects all parameters that may vary depending on the location within the porous medium. It admits a local assignment of purely intrinsic properties like porosity, permeability, or heat capacity as well as of the aforementioned parameters for the material law.

2.3. Simulation Control

Two standard approaches for the solution of porous media problems exist: a coupled fully-implicit approach and a decoupled semi-implicit approach. The fully-implicit approach discretizes the original coupled balance equations by an appropriate method in space and by an implicit method in time. The decoupled approach manipulates the balance equations towards one equation

for the pressure and one or more equations for transport (of phases, components, energy, etc.), where the pressure equation is solved implicitly while the transport equations are solved explicitly. In Figure 2, the algorithmic representations of both approaches down to the element level are illustrated.

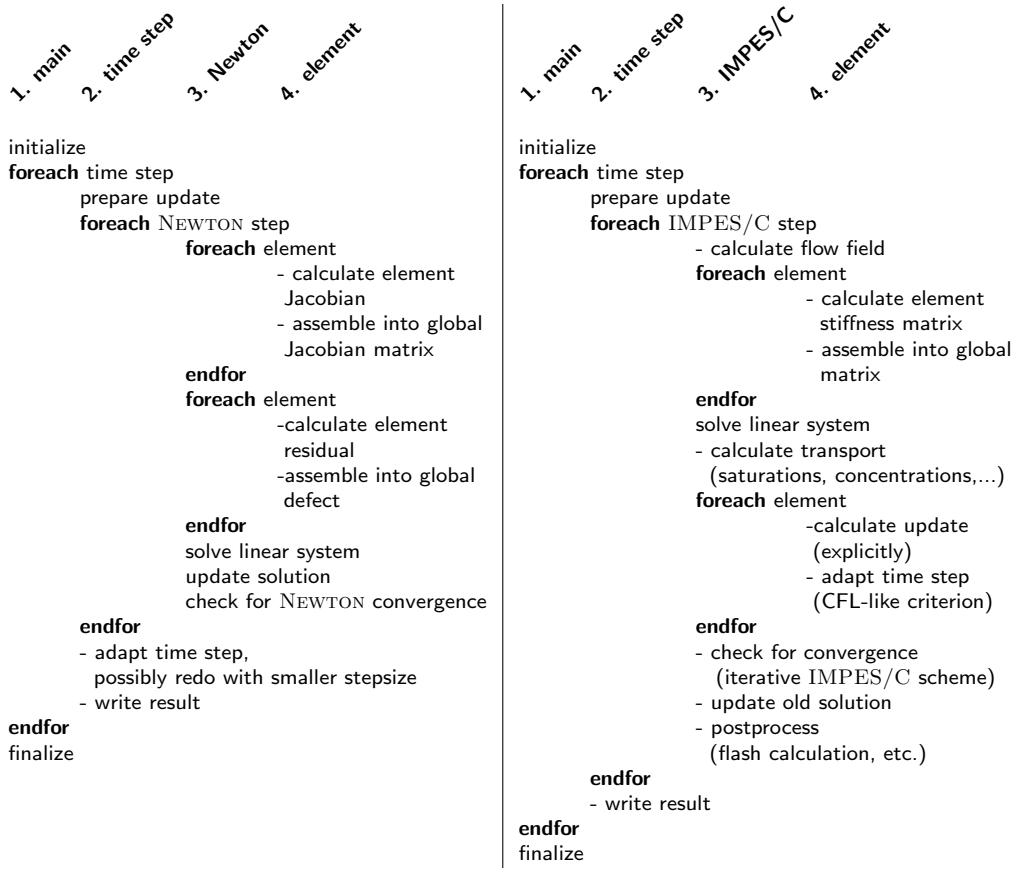


Figure 2: Structure of a coupled fully-implicit (**left**) and a decoupled semi-implicit (**right**) scheme in DuMu^{x}.

In DuMu^{x}, both the coupled fully-implicit and the decoupled semi-implicit models use the same code for the time-step control: The temporal domain is first divided into episodes, defined as time periods where boundary conditions, source terms and material parameters are differentiable with respect to time. Simulation time is then advanced by the minimum of the time-step size suggested by the underlying numerical model or the time span until the end of an episode. For the coupled fully-implicit models, the time-step size is

controlled based on the number of iterations required by the Newton method to achieve convergence for the last time integration: The time-step size is reduced, if the number of iterations exceeds a specified threshold, whereas it is increased, if the method converges within less iterations. The main influential parameters are the threshold value for the Newton convergence and how it is determined, and the factors for increasing and decreasing the time-step size. While a default implementation for each numerical model is available, these parameters can be changed problem-specific. For the decoupled models, the time-step size is calculated by CFL-like criteria.

2.4. Property System

On the one hand, DuMu^{x} modules can be freely combined, on the other hand, dynamic polymorphism is avoided for reasons of performance. Thus, a consistent set of parameters has to be provided throughout the module hierarchy at compile time. Examples for such parameters are the classes including the problem description, where initial and boundary conditions are typically defined on the highest level of the class hierarchy but are required by the code of the low-level spatial discretization.

Many such parameters are typically necessary and providing all of them as C++ template parameters would be very cumbersome and error-prone. One option is to use traits classes. In this approach a class hierarchy is created where each level stores the parameters required by the corresponding level of abstraction. A fundamental problem with this approach is the inability to change parameters of low levels at higher levels. This is because the parameters may be defined only using those of lower levels in the traits hierarchy.

To remedy this, the DuMu^{x} property system has been developed based on the C++ template specialization mechanism. In this system, a hierarchy of nodes – called type tags – is defined. Then all parameters are labeled and attached to the appropriate nodes in this acyclic graph. The labels are called property tags, whereas the parameters actually attached are called properties. The definition of properties may depend on arbitrary other properties which may be overwritten at any higher node of the acyclic graph. The only requirement for properties is that they may not exhibit cyclic dependencies. This is illustrated in the following short example:

Example 1. *Assume that the type used to represent floating point values should be parameterized as a property using the label* `Scalar`*. Next, assume*

that the property representing solution vectors is labeled `Vector`*, and defined within the spatial discretization module using the* `Scalar` *property. In the DuMu^x property system, a default value for the floating point representation can be specified at the level of the spatial discretization, but it can be changed at the level of the problem definition if higher accuracy is required. In the latter case, the definition of* `Vector` *changes automatically.*

3. Models

This section describes the models currently implemented in DuMu^x. In Table 1, an overview of the available models is given. They can be chosen – largely independent from the problem description – according to the task at hand. The individual models will be described in further detail here. As

Table 1: Currently available models within DuMu^x. With `p` standing for phase, `c` for component, `ni` for non-isothermal and `ia` for interfacial area.

	coupled fully-implicit	**decoupled semi-implicit**
Standard	`1p, 1p2c, 2p, 2pni, 2p2c, 2p2cni, Richards`	`1p, 2p, 2p2c, 2p2cni, 3p3c`
Extended	`2pia, 2p2cia, 2pNc, linear-elasticity 2pDFM, 2pMINC, 1DPipe3DPorousFlow, 1p2cDoubleContinuum`	
	`multiscale, multiphysics`	

outlined above, we distinguish between decoupled semi-implicit and coupled fully-implicit approaches. Furthermore, a brief sketch of the multi-scale and multi-physics capabilities is given. The following nomenclature will be used from this point on: `p` stands for phase, `c` for component, `ni` for non-isothermal and `ia` for interfacial area.

3.1. Decoupled Semi-implicit Numerical Models

As already described above, decoupled models solve a system of equations in which the single equations are only weakly coupled with each other. Thus, a sequential solution strategy can be applied where the standard scheme for multi-phase flow in porous media is an IMPES/IMPEC algorithm (IMplicit Pressure Explicit Saturation/Concentration). IMPES/IMPEC schemes first implicitly solve a pressure equation to get the flow field. Afterwards, the transport equations can be solved explicitly in the simplest case by using an explicit Euler scheme.

`1p`. This model solves an elliptic pressure equation in a fully-saturated porous medium. Available discretization methods are: Cell-centered finite volumes with TPFA (Two-Point-Flux-Approximation), or with MPFA (Multi-Point-Flux-Approximation, [22]), and mimetic finite differences, [23].

`2p`. Formulations for immiscible isothermal two-phase flow are implemented as classical fractional flow formulation (incompressible, e.g. [24]) or decoupled phase pressure formulation (slightly compressible/incompressible, see e.g. [25]). Transport equations for saturation are discretized by cell-centered finite volumes, for the solution of the pressure equations various discretization methods (see `1p` model) are available. An IMPES strategy is applied.

`2p2c/ni`. Miscible compressible two-phase flow can be modeled isothermally (`2p2c`) as well as non-isothermally (`2p2cni`). Capillary pressure is at the stage of this publication neglected. Instead of saturation equations, transport equations for concentrations of the components (`2c` - two components) are solved and flash calculations are performed afterwards to determine the phase composition. Details on the formulation can be found in [26]. Both pressure and transport equations for concentrations are discretized by cell-centered finite volumes, and an IMPEC scheme is used for the solution of the system.

`3p3c`. This model for three-phase three-component flow and transport is similar to the `2p2c` model.

3.2. Coupled Fully-implicit Numerical Models

We provide brief descriptions of standard and extended models treated in a coupled fully-implicit manner.

Standard models. The standard coupled models can be subdivided into several groups. It can be distinguished between the phase-based formulations (`1p`, `2p` and `2pni`) and the component-based models (`1p2c`, `2p2c`, `2p2cni`). Further, thermal effects can be taken into account (non-isothermal models: `2pni`, `2p2cni`). In general, a modular structure is used and the models with a higher complexity inherit the functionality from the simpler ones and extend it accordingly. As spatial discretization, a vertex-centered finite volume scheme (box method, [27]) is used. So far, the implicit Euler scheme is applied for the temporal discretization.

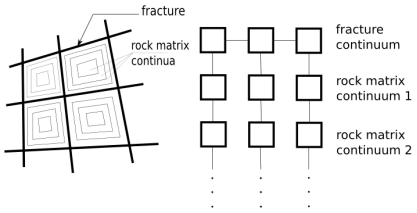
`interfacial area`. Usually, flow and transport in the bulk phases is modeled using the assumption of local thermodynamic equilibrium. The interfacial area models (abbreviated `2pia` and `2p2cia`) extend this approach by incorporating flow and evolution of the area separating the fluid phases [28]. This enables the description of kinetic mass and energy transfer between the fluid phases (thermal and chemical non-equilibrium). A natural description of hysteresis in the capillary pressure – saturation relationship is envisaged.

`linear-elasticity`. In the case of large pressure gradients the assumption of a rigid porous medium might be violated. The linear-elasticity models account for the deformation of the solid matrix and its interaction with a single- or two-phase flow system based on the theory of [29]. A first application of the DuMu^{x} single-phase linear-elasticity model is described in [30].

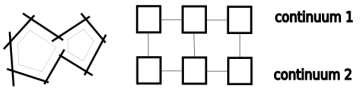
`2pNc`. In general it is desirable not to write a new model whenever an additional component is to be modeled. This, as well as the general prevention of switches in the primary variables – in the case of one phase disappearing – are the distinctions of the 2Nc model, with `Nc` standing for N-number of components .

`2pDFM`. The model simulates the two-phase flow in fractured porous systems using a discrete fracture model (DFM) approach [31], with a lower dimensional representation for the fractures [32]. The representative fracture network of the DFM is reconstructed with a geostatistic fracture generator [33].

`2pMINC`. In contrast to the `2pDFM`, this model does not discretize the fractures but simulates the two-phase flow in fractured porous media using the MINC method [34], [35].The fractures are treated as an equivalent homogeneous



(a) MINC discretization of a fractured porous system



	2pDFM	1p2cDouble Continuum	1DPipe3D PorousFlow
cont. 1	Discrete Fracture System	Capillary blood vessels	Pipe System
cont. 2	Rock Matrix Continuum	Biological Tissue	Rock Matrix Continuum

(b) Example of a double-continuum discretization for various applications

Figure 3: Sketch of the relation between model concepts and the continuum representation. Schematic diagrams of connectivity for the continuum models

porous medium which requires the determination of appropriate effective parameters and transfer functions between continua, but reduces considerably the geometrical complexity of the problem, see Figure 3a.

`1p2cDoubleContinuum`. It models the flow, transport and reaction processes through the terminal vascular bed (capillary vessels) and the surrounding biological tissues as an application to pulmonary cancer therapy. This is done by using the `1p2c` model concept, where both the capillary vessels and the surrounding tissue are described as two interacting porous media.

`1DPipe3DPorousFlow`. It simulates a coupled flow and transport system composed of quasi one-dimensional hollow structures embedded into a 3D porous medium [36]. The hollow structures are expressed with the cross-sectionally averaged one-dimensional pipe flow equation, like Hagen-Poiseuille, whereas for the flow within the porous medium the Darcy law is used.

The `2pDFM`, `1p2cDoubleContinuum`, and `1DPipe3DPorousFlow` models use the coupling strategy of a standard double-continuum approach, that is, the exchange terms between the two continua are implemented by additional source/sink terms, see Figure 3b.

3.3. Multi-Scale and Multi-Physics Models

The modular concept of DuMu^{x} allows the combination of the single models described before in multi-scale and/or multi-physics concepts. Multi-scale methods combine simulations on different length or time scales dependent on the processes that are to be modeled and on the information available. In

DuMu^x, this can be done in the context of classical numerical upscaling (or downscaling) methods (e.g. permeability upscaling) or in combination with multi-physics strategies. In space, multi-physics can be volume coupling (see e.g. multi-continuum models) or surface coupling of different model domains depending on the occurring processes that dominate in a sub-domain. In time, it could be a sequential solution strategy, where the model type is switched if the dominating processes change. The idea of both multi-scale and multi-physics methods is to be able to model complex processes occurring in large model domains. Examples on the capabilities of DuMu^x concerning multi-scale and multi-physics ideas can be found in Section 4.

4. Selected Examples

We present two examples that demonstrate the capabilities of DuMu^x for applications, relevant for the simulation of multi-phase flow in porous media. The multi-scale example presents an approach allowing the coarse-scale description of flow and transport in a fine-scale heterogeneous domain. This is accomplished by exploiting the capabilities of DUNE and DuMu^x for handling grids on multiple scales and solving small flow problems in order to incorporate the influence of sub-scale heterogeneities on the coarse scale.

Many real life applications require large modeling domains on the one hand and high model complexity on the other hand. Typically, this results in computationally demanding tasks. Therefore, the second example focuses mainly on the parallelization of code, which comes at little additional effort to the programmer. Furthermore, it is shown that the single models (described in Section 3) can be applied sequentially.

4.1. Multi-Scale

The multi-scale framework of DuMu^x that is illustrated in the following is a general implementation for local-global upscaling (downscaling) methods. It allows to carry out local fine-scale simulations – numerical experiments – to estimate effective parameters, which can be directly used in global coarse-scale simulations. In the following part the term fine-scale is used for the scale of the finest heterogeneities that are accounted for (*Darcy*-scale on which the previously described models (Section 3) are valid), whereas the term coarse-scale indicates the scale to which an upscaling is applied.

One basic idea is that any of the fine-scale models which are available in DuMu^x can be chosen for the local calculations without any changes in

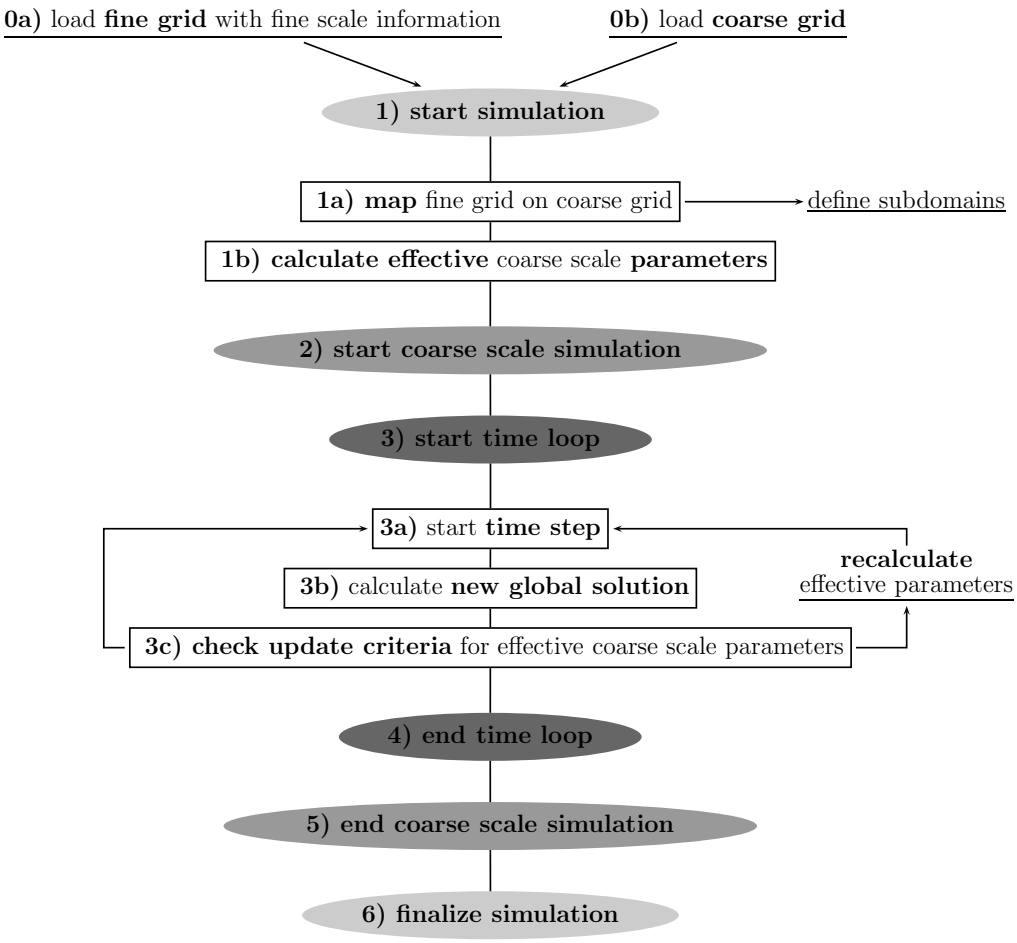


Figure 4: Flow of a multi-scale simulation in DuMu^{x}.

these model implementations. A second idea is that different kinds of post-processing routines which calculate the effective parameters are implemented once, and can be easily reused as well as combined in different kinds of local(-global) upscaling methods. Thirdly, depending on the upscaling approach one of the fine-scale model implementations could also be used on the coarse scale (with different parameters) or new coarse-scale models can be implemented, which again might be combined respectively with the different fine-scale and post-processing methods. Figure 4 shows the general flow of a local(-global) multi-scale simulation in DuMu^{x}. In the following paragraphs, we describe the conceptual and the mathematical model and show

some results.

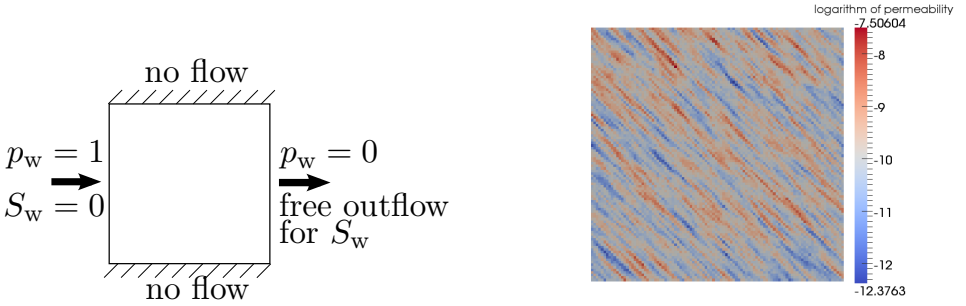
Conceptual Model. The multi-scale approach shown here is a local intrinsic permeability upscaling method (see e.g. [37, 38]), where a heterogeneous fine-scale permeability field is assumed to be known. It divides the global model domain into sub-domains assigned to coarse grid blocks (Figure 4, 1a)). The grids for the single sub-domains are managed using the DUNE module `dune-multidomaingrid` [21]. Effective coarse-scale intrinsic permeabilities are calculated for every coarse grid block using the results of local fine-scale flow simulations carried out on each of the sub-domains. As the fine-scale intrinsic permeability field is not dependent on saturation or pressure if the porous matrix is rigid, it is possible to calculate effective permeabilities that are also independent of the coarse-scale pressure and saturation field. This can be done in a preprocessing step (Figure 4, 1b)). After the preprocessing the global coarse-scale simulation can be directly started using the newly calculated effective intrinsic permeabilities (Figure 4, 2)). If an adaptive local-global approach is used in which effective permeabilities are updated depending on the flow field, a criterion which triggers the recalculation has to be checked at the end of every time step (Figure 4, 2c)). In this example a full tensor effective intrinsic permeability is calculated only once in a preprocessing step.

Mathematical Model. An isothermal incompressible two-phase flow model is used in a decoupled formulation (decoupled - `2p`) and in the simplest form, neglecting capillary pressure as well as gravity. The following equations have to be solved:

$$\nabla \cdot (-\lambda_{\mathrm{t}} \mathbf{K} \operatorname{grad} p_{\mathrm{w}}) = \sum_{\alpha} q_{\alpha}, \quad \alpha \in \{\mathrm{w}, \mathrm{n}\}, \tag{1}$$

$$\phi \frac{\partial (S_{\mathrm{w}})}{\partial t} + \operatorname{div} (-\lambda_{\mathrm{w}} \mathbf{K} \operatorname{grad} p_{\mathrm{w}}) = q_{\mathrm{w}}, \tag{2}$$

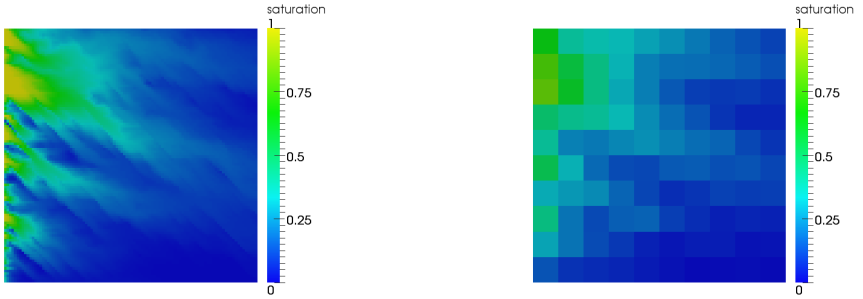
where $\mathbf{K}$ is the intrinsic permeability, ϕ is the porosity, λ_{α} is the mobility of phase α and $\lambda_{\mathrm{t}} = \sum_{\alpha} \lambda_{\alpha}$ is the total mobility, p_{w} is the pressure of the wetting phase, S_{w} is the saturation of the wetting phase, and q_{α} source or sink term of phase α. Equation 1 is called pressure equation and Equation 2 is called saturation equation. The same set of equations is used for both, fine scale and coarse scale, while effective (upscaled) coefficients are used on the coarse scale (e. g. $\mathbf{K} = \mathbf{K}_{\mathrm{eff}}$, see [38]).



(a) boundary conditions

(b) logarithmic intrinsic permeability distribution

Figure 5: Setup of the multi-scale simulation example.



(a) fine-scale reference

(b) coarse-scale solution (permeability upscaling method)

Figure 6: Saturation distribution at $t = t_{end}$.

Results. The models of this example are discretized by a cell-centered finite volume method. To account for the full tensor permeabilities a MPFA o-method is used (see [22], [39]) for discretization of the pressure equation. The setup of the simple test-example (2-D) is shown in Figure 5a. The heterogeneous permeability field (Figure 5b) is randomly generated by the open source tool GSTAT [40], which can be used for geostatistical modeling. The result of the coarse-scale simulation is shown in Figure 6b and shows good agreement with the fine-scale reference solution depicted in Figure 6a. Even for the quite simple and small test problem a significant speed-up of the multi-scale model (> factor 40) can be achieved.

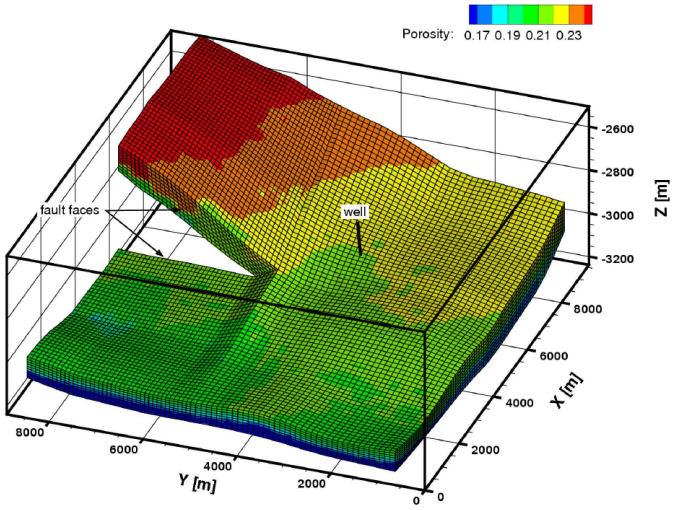


Figure 7: Geometry and porosity distribution of the Benchmark Problem 3.1 given in [41].

4.2. CO_2 Storage – A Large Scale Example

As a large scale application example, the injection of CO_2 into a heterogeneous saline aquifer is modeled with DuMu^x. The geometry and the hydraulic parameters are chosen according to the Benchmark Problem 3.1 given in [41]. The model domain and the position of the injection well are shown in Figure 7. CO_2 is injected for 25 years with a rate of 15 kg/s over the bottom 50 m of the vertical injection well. After the injection stop, the migration of the CO_2 plume is modeled for another 25 years. We present the conceptual and the mathematical model as well as some results in the following paragraphs.

Conceptual Model. In order to reduce the computational effort, the simulation of the injection and post-injection period is performed with a sequentially coupled model. Sequential model coupling can be applied if the dominating physical processes change over time. With respect to CO_2 storage it is possible to distinguish such different time periods. The injection period is governed by viscous and buoyancy driven multi-phase flow processes, which are hardly influenced by compositional processes such as CO_2 dissolution, diffusion and density-driven convection. After injection stop, the compositional effects become increasingly important and finally dominate the flow processes.

The sequential model applied here consists of two standard coupled fully-implicit models (Section 3.2). The injection period is simulated with a two-phase model (`2pni`) and the post-injection period is simulated with a more complex two-phase, two-component model (`2p2cni`). Both models include an energy balance equation to take into account non-isothermal effects e.g. the cooling caused by a CO_2 injection temperature, which is often below the reservoir temperature, or the reestablishment of the geothermal temperature gradient after injection stop by heat conduction.

Due to the modular structure of DuMu$^{\text{x}}$ the setup of a sequential model only requires the implementation of a suitable interface which allows a reasonable transfer of the primary variables between the coupled models. For more details on sequential model coupling in general and its application for the simulation of CO_2 storage, the reader is referred to [42], [43].

Mathematical Model. In the model of the first time period (`2pni`), the following mass balance equation is solved for both the wetting brine phase w and the non-wetting CO_2 phase n,

$$\phi \frac{\partial(\varrho_\alpha S_\alpha)}{\partial t} - \operatorname{div}\left\{\varrho_\alpha \lambda_\alpha \mathbf{K}\left(\operatorname{grad} p_\alpha - \varrho_\alpha \mathbf{g}\right)\right\} = q_\alpha, \quad \alpha \in \{\mathrm{w}, \mathrm{n}\}. \tag{3}$$

Here, ϱ_α represents the phase densities. Additionally, the energy balance equation of the fluid-solid mixture is solved to describe the non-isothermal processes. Assuming local thermal equilibrium, the energy balance equation can be written as

$$\begin{aligned} & \phi \frac{\partial\left(\sum_\alpha \varrho_\alpha u_\alpha S_\alpha\right)}{\partial t} + (1-\phi)\frac{\partial(\varrho_\mathrm{s} c_\mathrm{s} T)}{\partial t} \\ - & \sum_\alpha \operatorname{div}\left\{\varrho_\alpha h_\alpha \lambda_\alpha \mathbf{K}\left(\operatorname{grad} p_\alpha - \varrho_\alpha \mathbf{g}\right)\right\} - \operatorname{div}\left(\lambda_\mathrm{pm} \operatorname{grad} T\right) = q_h, \end{aligned} \tag{4}$$

including a summation over the phases $\alpha \in \{\mathrm{w}, \mathrm{n}\}$. T is the temperature, u_α and h_α stand for the internal energy and the enthalpy of the fluid phases. The density ϱ_s and the heat capacity c_s are properties of the rock matrix and λ_pm is the saturation-dependent thermal conductivity of the porous medium. To close this system of equations, the following auxiliary conditions are applied:

$$\sum_\alpha S_\alpha = 1 \quad \text{and} \quad p_\mathrm{w} = p_\mathrm{n} - p_\mathrm{c}(S_w), \tag{5}$$

with the primary drainage capillary pressure p_c.

The model of the second time period (`2p2cni` model) solves compositional mass balances for the components CO_2 and brine. Brine is a pseudo-component and represents water with a certain salinity. The component mass balances are described by

$$\phi \frac{\partial(\sum_\alpha \varrho_\alpha X_\alpha^\kappa S_\alpha)}{\partial t} - \sum_\alpha \operatorname{div}\left\{\varrho_\alpha X_\alpha^\kappa \lambda_\alpha \mathbf{K}(\operatorname{grad} p_\alpha - \varrho_\alpha \mathbf{g})\right\}$$
$$- \sum_\alpha \operatorname{div}\left\{D_{\alpha,\mathrm{pm}}^\kappa \varrho_\alpha \operatorname{grad} X_\alpha^\kappa\right\} = \sum_\alpha q_\alpha^\kappa \quad \kappa \in \{\text{brine, } CO_2\}. \tag{6}$$

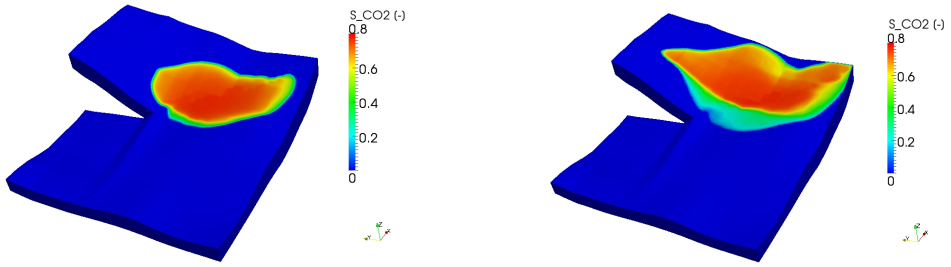
Here, X_α^κ is the mass fraction of the component κ, $D_{\alpha,\mathrm{pm}}^\kappa$ is the porous medium diffusion coefficient and q_α^κ is the source or sink term of the component κ in the phase α. The energy equation of the `2p2cni` model is similar to Equation (5), but the phase enthalpies and internal energies are functions of the dissolved components in addition to pressure and temperature. Besides the equations given in (5), a further auxiliary condition needs to be fulfilled for each phase in the `2p2cni` model:

$$\sum_\kappa X_\alpha^\kappa = 1. \tag{7}$$

The fluid properties of CO_2 are calculated as functions of pressure and temperature. The properties of brine additionally depend on the salinity, and in the compositional model, on the CO_2 mass fraction. The mutual solubilities of water and CO_2 are calculated according to [44]. For further information on the fluid property functions, the reader is referred to [45].

Results. The simulation is performed on a grid with 469,813 vertices leading to 1,409,439 degrees of freedom. For the parallelization of the DuMu^x models, almost no additional implementation work is required, since DUNE provides arbitrary data decomposition in a generic way and the employed assembly operator from dune-pdelab and linear solvers from dune-istl are designed correspondingly. To test the parallel scaling, several simulations with a differing number of cores are carried out.

Figure 8 shows the resulting CO_2 saturation distribution after 25 years (injection period) and after 50 years. The simulation results fit into the range of the results given in the benchmark study [41]. In Figure 9a, the computation time for the simulation of the benchmark problem is plotted against the number of cores applied in each simulation. The computation



(a) CO_2 plume after 25 years (b) CO_2 plume after 50 years

Figure 8: CO_2 saturation distribution after 25 years (end of the injection period) and after 50 years.

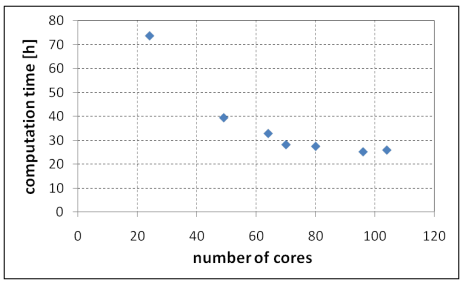
time is decreasing for an increasing number of cores up to a value of 96 cores. For a larger number the computation time slightly increases, which is due to a sub-optimal linear solver applied in the current implementation. In particular, a BiCGSTAB solver is used, preconditioned by an additive Schwarz method consisting of ILU applications locally on each process. No coarse grid correction is used and a linear parallel speed-up is not possible. This can also be seen by the increasing total CPU time which is plotted in Figure 9b.

In order to investigate this further, Figure 10a illustrates the average time required for assembling the global stiffness matrix with respect to the number of cores applied in each simulation. For an increasing number of cores the assembling time continuously decreases. The product of assembling time and number of cores is approximately constant for all simulations (Figure 10b), thus, with respect to assembling, linear scaling is observed.

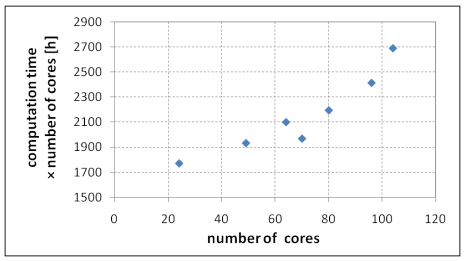
As already mentioned, the average time required for the solution of the global stiffness matrix by the iterative solver is not scaling linearly with the number of cores (Figure 11). An algebraic multigrid solver for non-overlapping grids is currently developed within DUNE and will be available for parallel simulations with DuMu^x in the future.

5. Summary and Outlook

This paper has been devoted to introduce DuMu^x, a free and open-source simulator for flow and transport processes in porous media. The vision, concept and design ideas have been presented. The common base of all DuMu^x

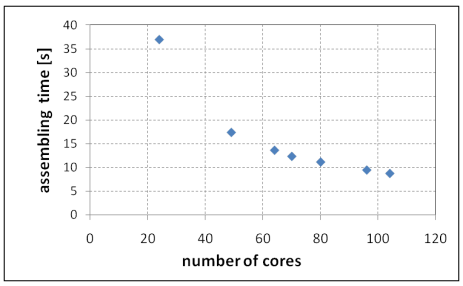


(a) Computation time

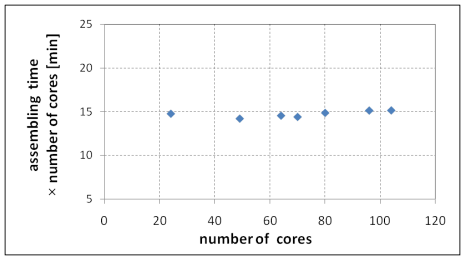


(b) Total CPU hours

Figure 9: Computation time for the simulation of the Benchmark Problem 3.1 versus number of cores (a) and total CPU hours (computation time × number of cores) versus number of cores (b).

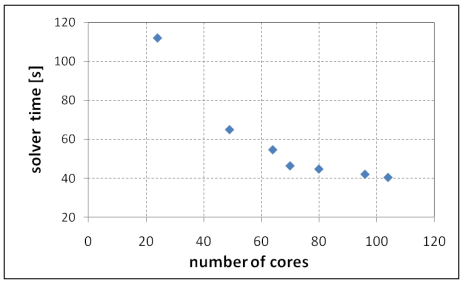


(a) Assembling time

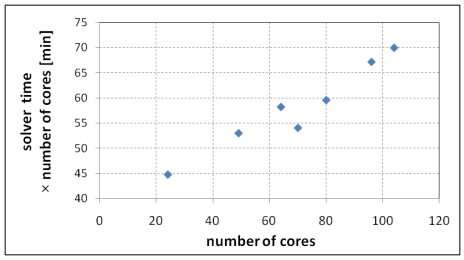


(b) Total CPU assembling time

Figure 10: Average assembling time per nonlinear iteration versus number of cores in seconds (a) and total CPU assembling time (assembling time × number of cores) versus number of cores (b).



(a) Linear solver time



(b) Total linear solver time

Figure 11: Average time required by the linear solver per nonlinear iteration versus number of cores (a) and total CPU time required by the linear solver (linear solver time × number of cores) versus number of cores (b).

models has been outlined, most importantly, the employed framework provided by DUNE, the Distributed and Unified Numerics Environment, as well as the flexible and extendable material system, the advanced simulation control, and the alternative to traits classes. The available models have been described, distinguishing between decoupled and fully coupled implicit approaches and briefly sketching the multi-scale and multi-physics capabilities. Two examples have been presented, one discussing the multi-scale framework in more detail and one focussing on a large scale application.

In the future, the modeling capabilities will be further extended. Increased emphasis will be devoted to the efficiency and robustness of the implemented numerical models. An enhanced user-friendliness will attract more users from outside of the developers group. Within the Open Porous Media (OPM) initiative, DuMu^{x} will become part of a simulation suite integrating the expertise of several academic and industrial partners.

Acknowledgments. This work was partially funded by the German research foundation (DFG) and the German Federal Ministry of Education and Research (BMBF). Some of the authors are members of the DFG International Research Unit MUSIS (FOR 1083), the Cluster of Excellence SimTech and of the International Research Training Group NUPUS, all funded by the German Research Foundation (DFG). Others are members of the CO2-MoPa joint project, BMBF's IPSWaT programme and of the FORSYS-Partner project. We thank the DFG, the BMBF and the partners from industry for their support.

References

[1] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, O. Sander, A generic grid interface for parallel and adaptive scientific computing. I. Abstract framework, Computing 82 (2-3) (2008) 103–119. doi:10.1007/s00607-008-0003-x.

[2] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, O. Sander, A generic grid interface for parallel and adaptive scientific computing. II. Implementation and tests in DUNE, Computing 82 (2-3) (2008) 121–138. doi:10.1007/s00607-008-0004-9.

[3] GNU General Public License version 2.
URL `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`

[4] The Dumux website.
URL http://dumux.org/

[5] Schlumberger, Eclipse technical description, 2007.

[6] Computer Modelling Group, GEM user guide.
URL http://www.cmgroup.com/software/brochures/GEM_FactSheet.pdf

[7] B. Robinson, H. Viswanathan, A. Valocchi, Efficient numerical techniques for modeling multi-component groundwater transport based upon simultaneous solution of strongly coupled subsets of chemical components, Adv. Water Res. 23 (2000) 307–324.

[8] H. Cao, Development of techniques for general purpose simulators, Ph.D. thesis, Stanford University (2002).

[9] Y. Jiang, Techniques for modeling complex reservoirs and advanced wells, Ph.D. thesis, Stanford University (2007).

[10] J. Wheeler, M. Wheeler, Integrated parallel and accurate reservoir simulator, Tech. rep., TICAM01-25, CSM, University of Texas at Austin (2001).

[11] The OpenGeoSys project.
URL http://www.ufz.de/data/OGS-5_concept_V110981.pdf

[12] A. Assteerawatt, P. Bastian, A. Bielinski, T. Breiting, H. Class, A. Ebigbo, H. Eichel, S. Freiboth, R. Helmig, A. Kopp, J. Niessner, S. Ochs, A. Papafotiou, M. Paul, H. Sheta, D. Werner, U. Ölmann, MUFTE-UG: structure, applications and numerical methods, Newsletter, International Groundwater Modeling Centre, Colorado School of Mines 23(2), 2005.

[13] K. Pruess, The TOUGH codes – a family of simulation tools for multiphase flow and transport processes in permeable media, Vadose Zone J. 3 (2004) 738–746.

[14] K.-A. Lie, S. Krogstad, I. S. Ligaarden, J. R. Natvig, H. M. Nilsen, B. Skaflestad, Discretisation on complex grids – open source matlab implementation, in: Proceedings of ECMOR XII, Oxford, UK, 6–9 September 2010, 2010.

[15] Open Porous Media Initiative.
URL http://www.sintef.no/Projectweb/GeoScale/Simulators/OPM/

[16] Apache Subversion.
URL http://subversion.apache.org/

[17] Flyspray bugtracking system.
URL http://flyspray.org/

[18] Cdash software testing server.
URL http://www.cdash.org/

[19] The DUNE website.
URL http://www.dune-project.org/

[20] P. Bastian, F. Heimann, S. Marnach, Generic implementation of finite element methods in the distributed and unified numerics environment DUNE, Kybernetika 46 (2) (2010) 294–315.

[21] S. Müthing, dune-multidomaingrid.
URL http://gitorious.org/dune-multidomaingrid

[22] I. Aavatsmark, An introduction to multipoint flux approximations for quadrilateral grids, Computational Geosciences 6 (2002) 405 – 432. doi:10.1023/A:1021291114475.

[23] B. Flemisch, R. Helmig, Numerical investigation of a mimetic finite difference method, in: R. Eymard, J. Hérard (Eds.), Finite volumes for complex applications V - problems and perspectives, Wiley - VCH, 2008, pp. 815–824.

[24] P. Binning, M. A. Celia, Practical implementation of the fractional flow approach to multi-phase flow simulation, Advances in Water Resources 22 (5) (1999) 461–478. doi:10.1016/S0309-1708(98)00022-0.

[25] Z. Chen, G. Huan, Y. Ma, Computational Methods for Multiphase Flows in Porous Media, SIAM, Computational Science & Engineering, 2006.

[26] J. Fritz, B. Flemisch, R. Helmig, Multiphysics modeling of advection-dominated two-phase compositional flow in porous media, International Journal of Numerical Analysis & Modeling. To appear 2010.

[27] R. Huber, R. Helmig, Node-centered finite volume discretizations for the numerical simulation of multiphase flow in heterogeneous porous media, Comput. Geosci. 4 (2000) 141–164.

[28] J. Niessner, S. M. Hassanizadeh, Modeling kinetic interphase mass transfer for two-phase flow in porous media including fluid-fluid interfacial area, Transport in Porous Mediadoi:10.1007/s11242-009-9.

[29] M. A. Biot, Theory of elasticity and consolidation for a porous anisotropic solid, Journal of Applied Physics 25 (1955) 182–185.

[30] K. H. Støverud, S. M. Hassanizadeh, R. Helmig, M. Darcis, Modeling convection-enhanced drug delivery into brain tissue including information from magnetic resonance imaging, submitted to: Biomechanics and Modeling in Mechanobiology.

[31] B. Berkowitz, Characterizing flow and transport in fractured geological media: A review, Advances in Water Resources 25 (8-12) (2002) 861–884. doi:10.1016/S0309-1708(02)00042-8.

[32] V. Reichenberger, H. Jakobs, P. Bastian, R. Helmig, A mixed-dimensional finite volume method for two-phase flow in fractured porous media, Advances in Water Resources 29 (7) (2006) 1020–1036. doi:10.1016/j.advwatres.2005.09.001.

[33] A. Assteerawatt, Flow and transport modelling of fractured aquifers based on geostatistical approach, Ph.D. thesis (Jul. 2008).

[34] K. Pruess, T. N. Narasimhan, A practical method for modeling fluid and heat flow in fractured porous media, Society of Petroleum Engineers.

[35] K. Pruess, Brief guide to the MINC-Method for modeling flow and transport in fractured media, Tech. Rep. LBNL-32195 Lawrence Berkeley National Laboratory (1992).

[36] O. Dogan, H. Class, R. Helmig, Different concepts for the coupling of porous-media flow with lower-dimensional pipe flow, Computer Modeling in Engineering & Sciences 53 (3) (2009) 207–233.

[37] L. J. Durlofsky, Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media, Water Resources Research 27 (5) (1991) 699–708.

[38] X. H. Wen, L. J. Durlofsky, M. G. Edwards, Use of border regions for improved permeability upscaling, Mathematical Geology 35 (5) (2003) 521–547. doi:10.1023/A:1026230617943.

[39] G. T. Eigestad, R. A. Klausen, On the convergence of the multi-point flux approximation o-method: Numerical experiments for discontinuous permeability., Numerical methods for partial differential equations 21 (6) (2005) 1079–1098. doi:10.1002/num.20079.

[40] J. E. Pebesma, C. G. Wesseling, Gstat: a program for geostatistical modelling, prediction and simulation., Computers & Geosciences 24 (1998) 17–31.

[41] H. Class, A. Ebigbo, R. Helmig, H. K. Dahle, J. M. Nordbotten, M. A. Celia, P. Audigane, M. Darcis, J. Ennis-King, Y. Fan, B. Flemisch, S. E. Gasda, M. Jin, S. Krug, D. Labregere, B. A. Naderi, R. J. Pawar, A. Sbai, S. G. Thomas, L. Trenty, L. Wei, A benchmark study on problems related to CO_2 storage in geologic formations, Computational geosciences 13 (4) (2009) 409–434.

[42] H. Class, R. Helmig, I. Neuweiler, Sequential coupling of models for contaminant spreading in the vadose zone, Vadose Zone Journal 7 (2).

[43] M. Darcis, H. Class, B. Flemisch, R. Helmig, Sequential model coupling for feasibility studies of CO_2 storage in deep saline aquifers, submitted to: Oil & Gas Science and Technology – Rev. IFP.

[44] N. Spycher, K. Pruess, CO_2-H_2O mixtures in the geological sequestration of CO_2. ii. partitioning in chloride brines at 12–100°c and up to 600 bar, Geochimica et Cosmochimica Acta 69 (13) (2005) 3309–3320.

[45] A. Bielinski, Numerical simulation of CO_2 sequestration in geological formations, Ph.D. thesis (2006).