Numerical Computation of Multiphase Flows in Porous Media

Peter Bastian

Habilitationsschrift

vorgelegt an der Technischen Fakultät der
Christian–Albrechts–Universität Kiel

zur Erlangung der Venia legendi im Fachgebiet

Informatik (Wissenschaftliches Rechnen)

Preface

Groundwater is a precious resource that is important for all forms of life on earth. The quality of groundwater is impaired by leaking disposal dumps and tanks or accidental release. Cleanup of contaminated sites is very difficult, if at all possible, and estimated costs amount to hundreds of billions of DM in Germany. Underground waste repositories currently being planned in many countries have to be designed in such a way that groundwater quality is not harmed.

In all these problems numerical simulation can help to gain a better process understanding, to make predictive studies and to ultimately optimize remediation techniques with respect to cost and efficiency. Clearly, this is a long term goal and considerable progress is necessary in all aspects of the modeling process. The present work is a contribution to the fast numerical solution of the partial differential equations (PDE) governing multiphase flow in porous media. A fully–coupled Newton–multigrid procedure has been implemented on the basis of the general purpose PDE software UG which allows the treatment of large–scale problems with millions of unknowns in three space dimensions on contemporary parallel computer architectures.

I am very grateful to G. Wittum for continuously encouraging this (and previous) work. His unlimited support of UG and the productive atmosphere at ICA III provided the basis of this work. R. Helmig introduced me to the field of multiphase flow in porous media. His enthusiasm for the subject was always a source of inspiration for me and I thank him for the years of excellent collaboration.

I am deeply indebted to my colleagues K. Birken, K. Johannsen, S. Lang, N. Neuß, H. Rentz–Reichert and C. Wieners who were involved in the development of the software package UG. Without the unselfish and cooperative style of work in our group this work would not have been possible. Special thanks also to V. Reichenberger who carefully read some versions of the manuscript.

Finally, my personal acknowledgments go to my family for their patience and support.

Heidelberg, June 1999 P. Bastian

Contents

Notation

Scalar values, functions and sets are denoted by normal letters, like $p_c, S_n, \rho, \ldots$ etc. Vectors are typeset in boldface symbols like e. g. in $\mathbf{x}, \mathbf{u}$ whereas tensors are written in boldface italic letters as in K.

Latin Symbols

A	edge set of a graph, p. 126
A	Jacobian matrix, system matrix, p. 100
$A_{\alpha h}$	dual form for flux term, p. 78
$\mathbf{A}_\alpha$	vector function for flux term, p. 79
B_h	box mesh, secondary mesh, p. 72
b_i, b_j	boxes, control volumes, p. 72
b_i^k	sub–control volume, p. 75
C	volume fraction, p. 14
C_α^κ	volume fraction of component κ in phase α, p. 29
$C_i(n)$	cluster of vertex n on level i, p. 134
$c_i(n)$	cluster map on level i, p. 134
D_m	molecular diffusion constant, p. 14, $[m^2/s]$
D	hydrodynamic dispersion, p. 14, $[m^2/s]$
$E(i)$	indices of elements touching vertex v_i, p. 71
E_h	mesh, set of elements, p. 71
E_l	elements of level l, p. 99
$E_l^{(p)}$	elements of level l stored by processor p, p. 117
e_i, e_j	elements, p. 71
$\mathbf{F}$	nonlinear defect, p. 100
f	flux term in 1D hyperbolic model problem, p. 47
$f(e)$	father element of element e, p. 116
f_α	fractional flow function, p. 35
G	graph for partitioning problem, p. 126
$\mathbf{G}$	modified gravity vector, p. 35, $[m/s^2]$
$\mathbf{g}$	gravity vector, p. 13, $[m/s^2]$
$H_0^1(\Omega)$	Hilbert space of functions with first order derivatives in $L^2(\Omega)$ and vanishing on the boundary, p. 36
$H_l^{(p)}$	maps to degrees of freedom handled by processor p on level l, p. 118
$H_l^{(p)}$	subspace corresponding to $H_l^{(p)}$, p. 118

I	index set, p. 72
I_d	index set of non–Dirichlet vertices, p. 72
$I_{\alpha d}$	index set of non–Dirichlet vertices for phase α, p. 77
$I_l^{(p)}$	maps to all degrees of freedom stored by processor p on level l, p. 118
$I_l^{(p)}$	subspace corresponding to $I_l^{(p)}$, p. 118
J	J–Leverett function, p. 42
J	finest level in multigrid structure, p. 99
$\mathbf{j}$	flux vector, p. 71
K	number of elements in a mesh, p. 71
K	absolute permeability tensor, p. 13, $[m^2]$
K_α	phase permeability tensor, p. 20, $[m^2]$
k	number of partitions in graph partitioning, p. 126
$k_{r\alpha}$	relative permeability, p. 20
$k_{r\alpha h}$	finite element approximation of relative permeability field, p. 79
$L^2(\Omega)$	Hilbert space of measurable, square integrable functions on Ω, p. 72
M	number of time steps, p. 88
$M_{\alpha h}$	dual form for accumulation term, p. 78
$\mathbf{M}_\alpha$	vector function for accumulation term, p. 79
m	Van Genuchten parameter, p. 23
m_l	maps elements to processors on level l, p. 116
$\mathrm{meas}(\Omega)$	length, area or volume of argument depending on dimension, p. 10, $[m^d]$
N	number of vertices, p. 71
N	vertex set of a graph, p. 126
N'	set of constrained vertices, p. 127
N''	set of free vertices, p. 127
$N(i)$	partition, p. 126
$NB_l(e)$	neighboring elements of element e on level l, p. 116
n	Van Genuchten parameter, p. 23
$\mathbf{n}$	normal vector, p. 14
$\mathbf{n}_{ij}^k$	sub–control volume face normal, p. 75
$\mathbf{n}_i^{kf}$	boundary sub–control volume face normal, p. 75
P_l	prolongation operator, p. 106
P	maps coefficient vector to finite element function, p. 73
P	set of processors, p. 116
p	single phase pressure, p. 13, $[Pa]$

p	global pressure, p. 38, $[Pa]$
p_c	capillary pressure, p. 17, $[Pa]$
p_{ch}	finite element approximation of $p_c(\mathbf{x})$ at time t, p. 79, $[Pa]$
$\mathbf{p}_{cmin}$	vector of minimum nodal capillary pressure, p. 82, $[Pa]$
p_d	entry pressure, p. 22, $[Pa]$
p_n	non–wetting phase pressure, p. 17, $[Pa]$
p_w	wetting phase pressure, p. 17, $[Pa]$
$\mathbf{p}_w$	coefficient vector for wetting phase pressure, p. 78, $[Pa]$
$p_{\alpha h}$	finite element approximation of phase pressure, p. 78, $[Pa]$
$Q_{\alpha h}$	dual form for source/sink term, p. 78
$\mathbf{Q}_\alpha$	vector function for accumulation term, p. 79
Q	permutation matrix, p. 109
$Q_l^{(p)}$	maps to degrees of freedom owned only by processor p on level l, p. 118
$Q_l^{(p)}$	subspace corresponding to $Q_l^{(p)}$, p. 118
q, q_α	source/sink term, p. 13, $[s^{-1}]$
R	individual gas constant, p. 12, $[kJ/(kg\,K)]$
R_l	restriction operator, p. 106
$\mathbb{R}$	real numbers, p. 12
r_α^κ	interphase mass transfer, p. 30, $[kg/(m^3\,s)]$
$\mathbf{r}$	flow field in advection–diffusion equation, p. 71
$\mathbf{S}_n$	coefficient vector of non–wetting phase saturation, p. 78
S_α	saturation of phase α, p. 20
$S_{\alpha h}$	finite element approximation of saturation of phase α, p. 78
$\overline{S}_\alpha$	effective saturation, p. 23
$S_{\alpha r}$	residual saturation, p. 23
S_w^L, S_w^R	left and right states in Riemann problem, 48
s	shock speed, p. 49
s	vertex migration cost, p. 126
T	temperature, p. 12, $[K]$
T	end of time interval, p. 88, $[s]$
t	time, p. 12, $[s]$
t^n	time step n, p. 88, $[s]$
U	boundary condition for total velocity, p. 41, 46
$\mathbf{u}$	single phase Darcy velocity, p. 13, $[m/s]$
$\mathbf{u}$	total velocity, p. 35, $[m/s]$
$\mathbf{u}_\alpha$	multiphase Darcy velocity, p. 21, $[m/s]$
V	vertex set, p. 71
$V(k)$	indices of vertices of element e_k, p. 71

V_h	standard conforming finite element space, p. 72
V_{hd}	finite element space with Dirichlet conditions incorporated, p. 72
$V_{\alpha hd}$	finite element space with Dirichlet conditions of phase α incorporated, p. 77
$V_l^{(p)}$	vertices on level l stored by processor p, p. 117
$V_l^{(p)}$	maps to vertical ghost nodes, p. 118
$V_l^{(p)}$	subspace corresponding to $V_l^{(p)}$, p. 118
v_i, v_j	vertices, p. 71
W	total weight of a graph, p. 126
W'	total weight of constrained vertices, p. 127
W''	total weight of free vertices, p. 127
$\bar{W}_i$	average cluster weight on level i, p. 134
W_h	test space piecewise constant on boxes, p. 72
W_{hd}	test space with Dirichlet conditions incorporated, p. 72
$W_{\alpha hd}$	test space with Dirichlet conditions of phase α incorporated, p. 78
w	weights for vertices and edges of a graph, p. 126
X	edge separator, p. 126
X_α^κ	mass fraction of component κ in phase α, p. 29
$\mathbf{x}, \mathbf{x}_0, \ldots$	points in $\mathbb{R}^d$, p. 10, $[m^d]$
$\mathbf{x}^k$	barycenter of element e_k, p. 80
$\mathbf{x}_{ij}^k$	barycenter of sub–control volume face, p. 75
$\mathbf{x}_i^{kf}$	barycenter of boundary sub–control volume face, p. 75

Greek Symbols

α	Van Genuchten parameter, p. 23, $[Pa^{-1}]$
α_T, α_L	dispersivity constants, p. 14
Γ_d, Γ_n	Dirichlet boundary, Neumann boundary, p. 14
$\Gamma_{\alpha d}, \Gamma_{\alpha n}$	Dirichlet and Neumann boundary for phase α, p. 77
γ	void space indicator function, p. 10
γ	multigrid parameter, p. 106
γ_α	phase indicator function, p. 19
γ_{ij}^k	sub–control volume face, p. 75
γ_i^{kf}	boundary sub–control volume face, p. 75
Δt^n	length of n-th time step, p. 88
δ	load imbalance factor, p. 126
ε_{lin}^κ	residual reduction of linear solver in Newton step κ, p. 101

ε_{nl}	residual reduction in nonlinear solver, p. 101
ε_0	minimum reduction required in linear solver, p. 102
θ	contact angle, p. 16, $[rad]$
θ	parameter in one step θ–scheme, p. 88
λ	Brooks–Corey parameter, p. 24
λ	total mobility, p. 35, $[(Pa\ s)^{-1}]$
λ_α	phase mobility, p. 21, $[(Pa\ s)^{-1}]$
$\lambda_{\alpha h}$	finite element approximation of phase mobility field, p. 79, $[(Pa\ s)^{-1}]$
μ	dynamic viscosity, p. 13, $[Pa\ s]$
μ_h	finite element approximation of dynamic viscosity field, p. 79, $[Pa\ s]$
ν_1, ν_2	number of pre– and postsmoothing steps, p. 106
π, π^0	partition maps, p. 126, 126
π_w	$p - p_n$, p. 38
ρ	convergence factor of iterative method, p. 104
ρ, ρ_α	density, p. 12, $[kg/m^3]$
$\rho_{\alpha h}$	finite element approximation of density, p. 79, $[kg/m^3]$
ρ_α^κ	intrinsic mass density of component κ in phase α, p. 29, $[kg/m^3]$
σ	surface tension, p. 17, $[J/m^2]$
τ	tortuosity, p. 14
Φ	porosity, p. 10
Φ_h	finite element approximation of porosity field, p. 79
ϕ	normal flux, p. 13, $[kg/(s\ m^2)]$
φ	nodal basis function of V_h, p. 73
ψ	basis function of W_h, p. 73
Ω, Ω_i	domain in $\mathbb{R}^2$ or $\mathbb{R}^3$, p. 10

Norms, Operators, ...

∇	divergence operator, p. 12
∇	gradient operator, p. 13
$\|.\|_2$	Euclidean vector norm, p. 101

Indices

$\alpha, \beta, \ldots$	phase index, p. 19
h	mesh size, p. 71
l	multigrid level, p. 106

w, n, g	wetting phase, non–wetting phase, gas phase

Exponents

κ	component, p. 29
κ	iteration index in nonlinear solver, p. 101
μ	iteration index in linear solver, p. 104
(p)	processor number, p. 117

Introduction

Flow and transport of hazardous substances in the subsurface is of enormous importance to society. Estimated cleanup costs of contaminated sites in Germany are in the range of 100 to 300 billion DM (Kobus 1996). The present work is a contribution to the efficient numerical solution of the mathematical equations governing multiphase flow in the subsurface. A fully–coupled Newton–multigrid method is applied to various formulations of the two–phase flow problem with special emphasis on heterogeneous porous media. The applicability and effectiveness of the methods is shown in numerical experiments in two and three space dimensions. Moreover, the developed computer code is able to exploit the capabilities of large–scale parallel computer systems.

Groundwater Contamination

In Germany and many other countries more than half of the population depend on groundwater as their supply in drinking water (Jahresbericht der Wasserwirtschaft 1993). Problems with groundwater quality arise from disposal dumps, leaking storage tanks and accidental spills of substances used in industry.

Removing such substances from the subsurface is extremely complex and costly, if at all possible, see Kobus (1996). In order to design effective remediation strategies it is necessary to fully understand the governing physical processes of flow and transport in porous media. Mathematical modeling is one important tool that helps to achieve this goal. Incorporation of more detailed physics and geometric detail into the mathematical models requires the use of efficient numerical algorithms and large–scale parallel computers, both are of major concern in this work.

Among the most toxic and prevalent substances threatening groundwater quality are so–called nonaqueous phase liquids (NAPLs) such as petroleum products or chlorinated hydrocarbons. These volatile chemicals have low solubility in water and are to be considered as separate phases in the subsurface.

Fig. 1 illustrates the qualitative flow behavior of different NAPLs in the subsurface. In case A a light NAPL (LNAPL) with density smaller than water is released. It migrates downward through the unsaturated zone until it reaches the water table where it continues to spread horizontally. Typically, these substances contain volatile components which are then transported in the air phase. If the supply of LNAPL stops, a certain amount of it remains immobile in the soil at residual saturation as shown in case B.

The flow of a dense NAPL (DNAPL) being heavier than water is shown in case C. Its flow behavior in the unsaturated zone is similar but due to its greater density it migrates downward also through the saturated zone. Due to capillary

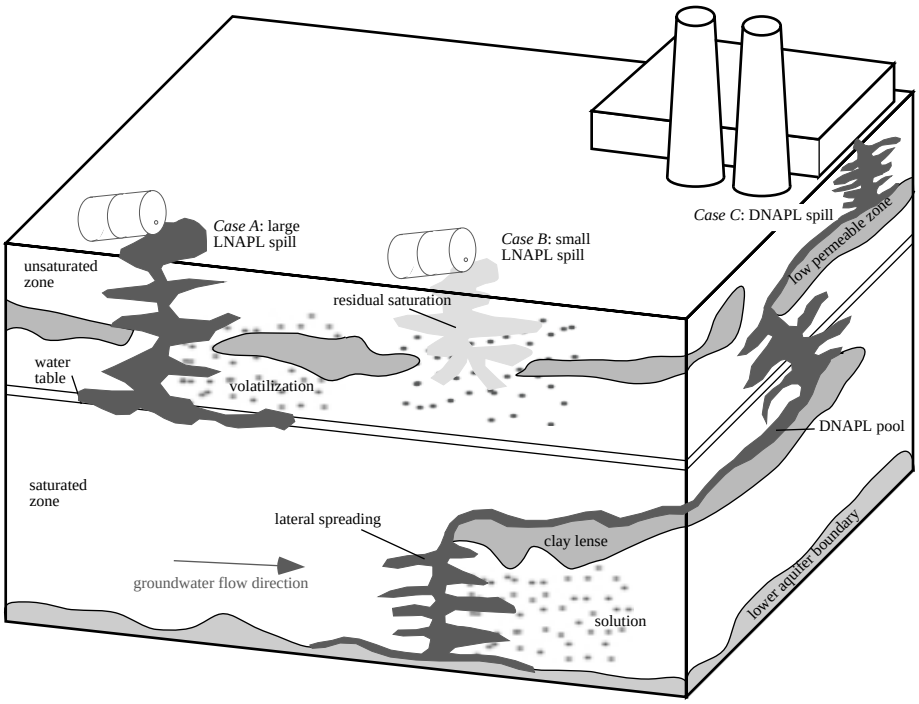


Figure 1: Qualitative behavior of NAPLs in the subsurface, after Helmig (1997).

effects heterogeneities in the soil (differences in grain size and therefore pore width) play an extremely important rôle in multiphase flows. Regions of lower permeability (smaller pores) are not penetrated by the fluid until a critical fluid saturation has accumulated. The size of these regions may vary from centimeters leading to an irregular lateral spreading of the NAPL to (many) meters with the formation of DNAPL pools.

NAPLs pose a long term threat to groundwater quality. The initial infiltration may happen in hours or days while the solution process may go on for years. Very small concentrations of NAPL on the order of $10[\mu g/l]$ make the water unusable for drinking water supply.

Depending on the situation different in–situ remediation strategies are possible, cf. Kobus (1996):

- Hydraulic schemes: extraction of contaminant in phase and/or solution by means of flushing and pumping. This so–called “pump and treat” strategy may be inadequate for hydrophobic substances due to capillary effects. It is very effective (and standard) for soluble contaminants.

- Degradation of contaminant by chemical reaction and/or microbiological decay.

- Soil air venting for volatile substances, can be enhanced thermally by use of steam.

- Air sparging for volatile substances in the saturated zone.
- Remobilization of (immobile) contaminant by lowering surface tension and/or viscosity ratio through supply of heat or chemicals (surfactants). Must be used with care since contaminant may move further downward.

From the large number of physical processes mentioned in this list it is evident that mathematical modeling of remediation processes can be very complicated. In the simplest example of two phase immiscible flow the mathematical model consists of two coupled nonlinear time–dependent partial differential equations. Since the detailed geometry of a natural porous medium is impossible to determine its complicated structure is effectively characterized by several parameters in the mathematical equations. It is the fundamental problem of all porous medium flow models to determine these parameters. Moreover, due to the heterogeneity of the porous medium on different length scales these effective parameters are also scale–dependent. Several techniques have been proposed to address this problem, we mention stochastic modeling (Kinzelbach and Schäfer 1992), upscaling (Ewing 1997) and parameter identification (Watson et al. 1994).

So far we concentrated on groundwater remediation problems as our motivation for the consideration of multiphase fluid flow in porous media. In addition there are other important applications for these models such as oil reservoir exploitation (historically the dominant application) and security assessment of underground waste repositories. The latter application is often complicated by the existence of fractures in hard rock, cf. (Helmig 1997).

Scientific Computing

The construction of a computer code that is able to simulate the processes described above involves different tasks from a variety of disciplines. The tasks are now subsumed under the evolving field of “Scientific Computing” in order to emphasize that multidisciplinary cooperation is the key to a successful simulation of these complex physical phenomena.

The first step in the modeling process is the derivation of the conceptual model. The conceptual model consists of a verbal description of the relevant physical processes, e. g. the number of phases and components, which components are present in which phase, existence of fractures and the like. Since all subsequent steps depend on the conceptual model it has to be considered very carefully.

In the next step a mathematical model describing the physical processes in a quantitative way is derived. It usually involves coupled systems of nonlinear time–dependent partial differential equations. In Chapter 1 we will review the mathematical models for single– and multiphase flow in porous media. Subsequently, mathematical analysis addresses questions of existence, uniqueness and regularity of solutions of the mathematical model.

Since a solution of the mathematical model in closed form is seldom possible a discrete numerical model suitable for computer solution is now sought (see Chapter 3). The numerical model consists of a large set of (non–) linear algebraic equations to be solved per time–step. The convergence of its solution to the solution of the (continuous) mathematical model is the fundamental question in numerical analysis. The actual determination of the discrete solution (see Chapter 4) may require enormous computational resources which are only available from large–scale parallel computers. The complete specification of the numerical model includes the geometric description of the domain and a computational mesh. From a practical point of view this may be the most time–consuming process especially since it requires human interaction.

A variety of techniques have been developed to speed up the solution of the numerical model. Multigrid methods (Hackbusch 1985), adaptive local mesh refinement (Eriksson et al. 1995) and parallelization (Smith et al. 1996) are important developments in this respect. However, the introduction of these techniques lead to an enormous increase in the complexity of numerical software and software design for scientific computing applications has recently received much attention in the scientific community.

The increasing complexity of PDE software lead to the development of "tools" that allow the incorporation of different problems and solution schemes into a standardized environment. To mention but a few we refer to Diffpack (Bruaset and Langtangen 1997), PETSc (Balay et al. 1997) and UG (Bastian et al. 1997), which is the basis of this work. Finally, the interpretation of the results obtained by large–scale simulations requires a powerful visualization tool. The sheer amount of data often exceeds the capabilities of conventional visualization programs and new techniques are also required in this area, cf. Rumpf et al. (1997).

The total modeling process is now complete and numerical results can be compared with experimental measurements. Often it is then necessary to do more iterations of the modeling cycle and to improve upon conceptual, mathematical and numerical model in order to match experimental results with sufficient accuracy.

In order to handle the complexity of the total modeling process a "divide and conquer" approach has been often applied in the past. The extraction of simplified "model problems" and their detailed investigation certainly was a very successful approach. However, as the solution of the individual subproblems is more understood their interaction becomes more important. It can very well happen that problems encountered in later stages of the modeling process can be circumvented by a different choice in an earlier stage. In order to illustrate this rather general remark we mention an example. In Chapter 2, a number of different formulations of the two–phase flow equations will be discussed in detail. It is very important to recognize the limitations and advantages of each formulation, e. g. the phase pressure formulations lead to difficulties in the nonlinear solver if both fluids are present at residual saturation in the domain. It is

of no use to try to improve the nonlinear solver, instead one should use a global pressure formulation in this case. In the case of a porous medium with a discontinuity the formulation with interface conditions leads to more accurate results and produces algebraic systems that are easier to solve (see Subs. 7.3.2).

Objective and Structure of this Work

This book starts with a discussion of various mathematical models of subsurface flow and the underlying concepts in Chapter 1.

Then the basic properties of the two–phase flow model for homogeneous and heterogeneous porous media are addressed in Chapter 2. Their extension to three–phase flow models is discussed briefly.

Chapter 3 concentrates on the discretization of the two–phase flow equations. A vertex centered finite volume scheme with upwind mobility weighting has been selected due to its monotone behavior and applicability to unstructured multi–element type meshes in two and three space dimensions. Time discretization is fully implicit.

Chapter 4 then treats the solution of the resulting (non–) linear algebraic equations. Special emphasis is put on the construction of a multigrid method for the linear systems arising from a fully–coupled Newton procedure. Step length control and nested iteration are used to ensure global convergence of the Newton method.

A data–parallel implementation and load balancing is discussed in Chapter 5 while the concepts of the PDE software toolbox UG are contained in Chapter 6.

Extensive numerical results for realistic problems are then presented in Chapter 7 in order to assess the quality of the numerical solutions obtained and to illustrate the excellent convergence behavior of the (non–) linear iterative schemes.

1

Modeling Immiscible Fluid Flow in Porous Media

This chapter provides an introduction to the models used in porous medium simulations. We begin with a definition of porous media, their basic properties and a motivation of macroscopic flow models. The subsequent sections are devoted to the development of models for single–phase flow and transport, multiphase flow and multiphase/multicomponent flows.

1.1 Porous Media

This subsection introduces the basic characteristics of porous media. Of special importance is the consideration of different length scales.

1.1.1 Definitions

A porous medium is a body composed of a persistent solid part, called *solid matrix*, and the remaining *void space* (or *pore space*) that can be filled with one or more fluids (e. g. water, oil and gas). Typical examples of a porous medium are soil, sand, cemented sandstone, karstic limestone, foam rubber, bread, lungs or kidneys.

A *phase* is defined in (Bear and Bachmat 1991) as a chemically homogeneous portion of a system under consideration that is separated from other such portions by a definite physical boundary. In the case of a *single–phase system* the void space of the porous medium is filled by a single fluid (e. g. water) or by several fluids completely *miscible* with each other (e. g. fresh water and salt water). In a *multiphase system* the void space is filled by two or more fluids that are *immiscible* with each other, i. e. they maintain a distinct boundary between them (e. g. water and oil). There may only be one gaseous phase since gases are always completely miscible. Formally the solid matrix of the porous medium can also be considered as a phase called the *solid phase*. Fig. 1.1 shows a two–dimensional cross section of a porous medium filled with water (single–phase system, left) or filled with water and oil (two–phase system, right).

Bear and Bachmat (1991) define a *component* to be part of a phase that is composed of an identifiable homogeneous chemical species or of an assembly of species (ions, molecules). The number of components needed to describe a phase is given by the *conceptual model*, i. e. it depends on the physical processes to be modeled. The example of fresh and salt water given above is described by a single–phase two component system.

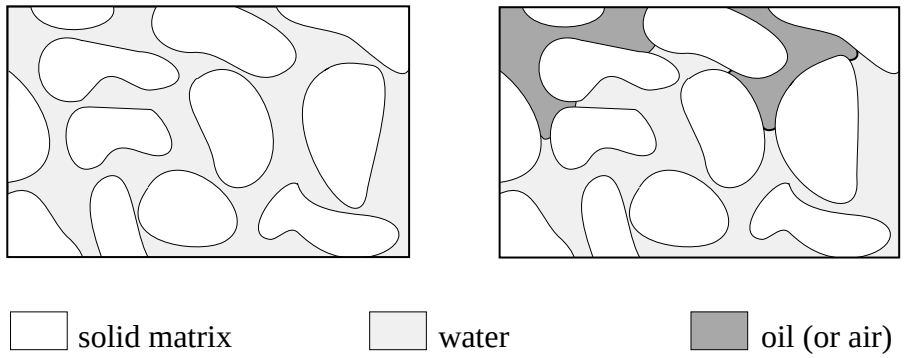


Figure 1.1: Schematic drawing of a porous medium filled with one or two fluids.

In order to derive mathematical models for fluid flow in porous media some restrictions are placed upon the geometry of the porous medium (Corey 1994, p. 1):

(P1) The void space of the porous medium is interconnected.

(P2) The dimensions of the void space must be large compared to the mean free path length[1] of the fluid molecules.

(P3) The dimensions of the void space must be small enough so that the fluid flow is controlled by adhesive forces at fluid–solid interfaces and cohesive forces at fluid–fluid interfaces (multiphase systems).

The first assumption (P1) is obvious since no flow can take place in a disconnected void space. The second property (P2) will enable us to replace the fluid molecules in the void space by a hypothetical continuum (see next chapter). Finally, property (P3) excludes cases like a network of pipes from the definition of a porous medium.

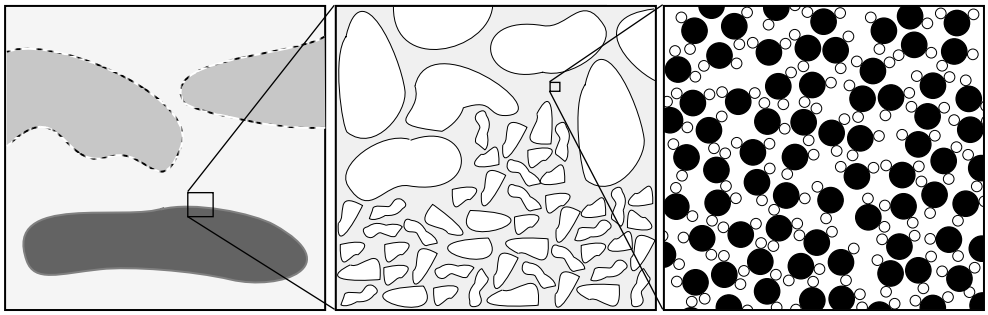
1.1.2 Continuum Approach

The important feature in modeling porous media flow is the consideration of different length scales. Fig. 1.2 shows a cross section through a porous medium consisting of different types of sands on three length scales.

In Fig. 1.2a the cross section is on the order of 10 meters wide. This scale is called the *macroscopic scale*. There we can identify several types of sand with different average grain sizes. A larger scale than the macroscopic scale is often called regional scale but is not considered here, see Helmig (1997).

If we zoom in to a scale of about 10^{-3}m as shown in Fig. 1.2b we arrive at the *microscopic scale* where individual sand grains and pore channels are visible.

[1] The average distance a molecule travels between successive collisions with other molecules. Mean free path of air at standard temperature is about $6 \cdot 10^{-8}$m.



(a) macroscopic scale (b) microscopic scale (c) molecular scale

Figure 1.2: Different scales in a porous medium.

In the figure we see the transition zone from a fine sand to a coarser sand. The void space is supposed to be filled with water.

Magnifying further into the water–filled void space one would finally see individual water molecules as shown in Fig. 1.2c. The larger black circles are oxygen atoms, the smaller white circles are the hydrogen atoms. This scale of about 10^{-9}m will be referred to as the *molecular scale*.

It is important to note that the behavior of the flow is influenced by effects on all these different length scales. Fluid properties like viscosity, density, binary diffusion coefficient and miscibility are determined on the molecular scale by the individual properties of the molecules. On the microscopic scale the configuration of the void space influences the flow behavior through properties like the tortuosity of the flow channels or the pore size distribution, whereas on the macroscopic scale the large scale inhomogeneities play a rôle.

In classical continuum mechanics, see e. g. (Chung 1996), the individual molecules on the molecular scale are replaced by a hypothetical continuum on the microscopic scale. Quantities like mass (density) or velocity are now considered to be (piecewise) continuous functions in space and time. The continuum approach is a valid approximation if the mean free path length of the fluid molecules is much smaller than the physical domain of interest. This is ensured by property (P2) from the last subsection.

Accordingly, the flow of a single newtonian fluid in the void space of a porous medium is described on the microscopic level by the Navier–Stokes system of equations (cf. (Chung 1996)) with appropriate boundary conditions. However, the void space configuration is usually not known in such detail to make this description feasible. Moreover, a numerical simulation on that level is beyond the capabilities of todays computers and methods.

In order to derive a mathematical model on the macroscopic level another continuum is considered. Each point in the continuum on the macroscopic level is assigned average values over *elementary volumes* of quantities on the micro-

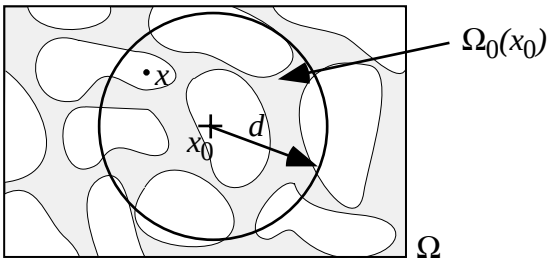


Figure 1.3: Illustration of the averaging volume.

scopic level. This process leads to *macroscopic equations* that do not need an exact description of the microscopic configuration. Only measurable statistical properties of the porous medium and the fluids are required.

1.1.3 Representative Elementary Volume

The averaging process used for passing from the microscopic to the macroscopic level is illustrated for the *porosity*, a simple geometric property of the porous medium.

The porous medium is supposed to fill the domain Ω with volume meas(Ω). Let $\Omega_0(\mathbf{x}_0) \subset \Omega$ be a subdomain of Ω centered at the point $\mathbf{x}_0$ on the macroscopic level as shown in Fig. 1.3.
Further we define the void space indicator function on the microscopic level:

$$\gamma(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \text{void space} \\ 0 & \mathbf{x} \in \text{solid matrix} \end{cases} \qquad \forall \mathbf{x} \in \Omega. \tag{1.1}$$

Then the porosity $\Phi(\mathbf{x}_0)$ at position $\mathbf{x}_0$ with respect to the averaging volume $\Omega_0(\mathbf{x}_0)$ is defined as

$$\Phi(\mathbf{x}_0) = \frac{1}{\text{meas}(\Omega_0(\mathbf{x}_0))} \int\limits_{\Omega_0(\mathbf{x}_0)} \gamma(\mathbf{x}) d\mathbf{x} \quad . \tag{1.2}$$

The macroscopic quantity porosity is obtained by averaging over the microscopic void space indicator function. If we plot the value of $\Phi(\mathbf{x}_0)$ at a fixed position $\mathbf{x}_0$ for different diameters d of the averaging volume $\Omega_0(\mathbf{x}_0)$ we observe a behavior as shown in Fig. 1.4. For very small averaging volumes the discontinuity of γ produces large variations in Φ, then at diameter l the average stabilizes and for averaging volumes with diameter larger than L the large scale inhomogeneities of the porous medium destabilize the average again, cf. (Bear and Bachmat 1991; Helmig 1997).

The averaging volume $\Omega_0(\mathbf{x}_0)$ is called a *representative elementary volume (REV)* if such length scales l and L as in Fig. 1.4 can be identified where the

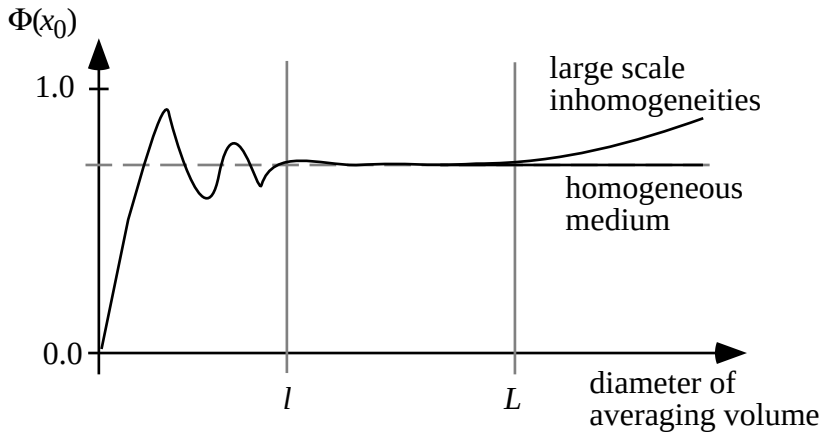


Figure 1.4: Porosity Φ for different sizes of averaging volumes.

value of the averaged quantity does not depend on the size of the averaging volume. In that case we can choose the averaging volume anywhere in the range

$$l \ll \mathrm{diam}\,(\Omega_0(\mathbf{x}_0)) \ll L \quad . \tag{1.3}$$

If a REV cannot be identified for the porous medium at hand the macroscopic theories of fluid flow in porous media cannot be applied (Hassanizadeh and Gray 1979a).

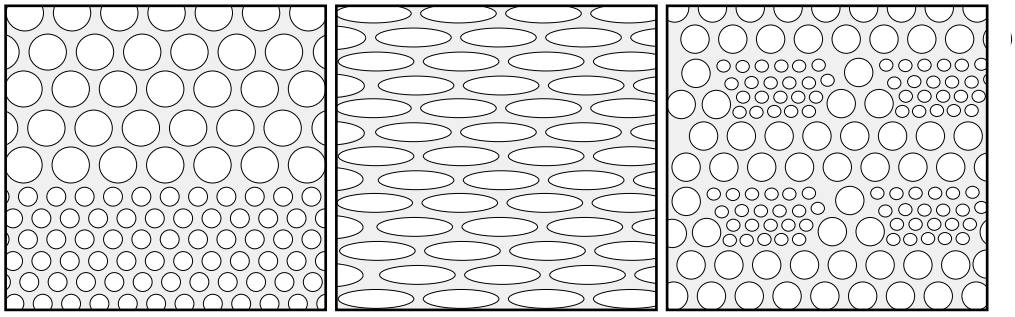
The following table with typical values of porosity is taken from (Corey 1994):

Consolidated sandstones	0.1–0.3
Uniform spheres with minimal porosity packing	0.26
Uniform spheres with normal packing	0.35
Unconsolidated sands with normal packing	0.39–0.41
Soils with structure	0.45–0.55

1.1.4 Heterogeneity and Anisotropy

A porous medium is said to be *homogeneous* with respect to a macroscopic (averaged) quantity if that parameter has the same value throughout the domain. Otherwise it is called *heterogeneous*. For example the porous medium shown in Fig. 1.5a has a different porosity in the parts with large and small sand grains and is therefore heterogeneous with respect to porosity.

Macroscopic tensorial quantities can also vary with direction, in that case the porous medium is called *anisotropic* with respect to that quantity. Otherwise it is called *isotropic*. As an example consider Fig. 1.5b. It is obvious that the porous medium is more resistive to fluid flow in the y-direction than in the x-direction. The corresponding macroscopic quantity called *permeability* will be anisotropic. Note that a similar effect as in Fig. 1.5b can also be achieved with the grain distribution shown in Fig. 1.5c.



(a) (b) (c)

Figure 1.5: Porous media illustrating the concepts of heterogeneity and anisotropy.

1.2 Single–Phase Fluid Flow and Transport

In this subsection we consider macroscopic equations for flow and transport in porous media when the void space is filled with a single fluid, e. g. water, or several completely miscible fluids.

1.2.1 Fluid Mass Conservation

Suppose that the porous medium fills the domain $\Omega \subset \mathbb{R}^3$, then the macroscopic fluid mass conservation is expressed by the partial differential equation

$$\frac{\partial(\Phi\rho)}{\partial t} + \nabla \cdot \{\rho \mathbf{u}\} = \rho q \quad \text{in } \Omega. \tag{1.4}$$

In its integral form this equation states that the rate of change of fluid mass in an arbitrary control volume $V \subseteq \Omega$ is equal to the net flow over the surface ∂V and the contribution of sources or sinks within V. The quantities in Eq. (1.4) have the following meaning.

$\Phi(\mathbf{x})$ Porosity of the porous medium as defined in Eq. (1.2). It is a function of position in the case of heterogeneous media. In general it could depend on the fluid pressure (introduced below) or on time (e. g. swelling of clay) but these effects are not considered here.

$\rho(\mathbf{x},t)$ *Density* of the fluid given in $[kg/m^3]$. In this work density is either constant when the fluid is *incompressible* or we assume an equation of state for ideal gases where density is connected to fluid pressure p (see below):

$$p = \rho RT. \tag{1.5}$$

Here R is the individual gas constant and T the temperature in $[K]$, cf. (Helmig 1997). Note that the time derivative in Eq. 1.4 vanishes when the density is constant.

$\mathbf{u}(\mathbf{x},t)$ *Macroscopic apparent velocity* in $[m/s]$. This velocity is obtained by a macroscopic observer. On the microscopic level the flow takes only place through the pore channels of the porous medium where an average velocity of $\mathbf{u}/\Phi$ is observed.

$q(x,t)$ Specific source/sink term with dimensions $[s^{-1}]$.

1.2.2 DARCY'S LAW

By using local averaging techniques, see e. g. (Whitaker 1986a), or homogenization, see (Hornung 1997), it can be shown that under appropriate assumptions (see below) the momentum conservation of the Navier–Stokes equation reduces to the Darcy–Law on the macroscopic level which is given by

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu}\left(\nabla p - \rho\mathbf{g}\right). \tag{1.6}$$

This relation was discovered experimentally for the one–dimensional case by H. Darcy in 1856. It is basically a consequence of property (P3) of the porous medium. The new quantities in Eq. (1.6) have the following meaning.

$p(x,t)$ Fluid *pressure* in $[Pa] = [N/m^2]$. This will be the unknown function to be determined by the flow model.

$\mathbf{g}$ Gravity vector pointing in the direction of gravity with size g (gravitational acceleration). Dimension is $[m/s^2]$. When the z–coordinate points upward we have $\mathbf{g} = (0,0,-9.81)^T$.

$\mathbf{K}(\mathbf{x})$ Symmetric tensor of *absolute permeability* with dimensions $[m^2]$. It is a parameter of the solid matrix only and may depend on position in the case of a heterogeneous porous medium. Furthermore $\mathbf{K}$ may be anisotropic if the porous medium has a preferred flow direction as explained in subsection 1.1.4.

$\mu(\mathbf{x},t)$ *Dynamic viscosity* of the fluid given in $[Pa\ s]$. In the applications considered here μ is either constant or a function of pressure.

Darcy's Law is valid for the slow flow (inertial effects can be neglected) of a Newtonian fluid through a porous medium with rigid solid matrix. No slip boundary conditions are assumed at the fluid–solid boundary on the microscopic level. For details we refer to (Bear 1972; Whitaker 1986a; Whitaker 1986b; Hassanizadeh and Gray 1979a; Hassanizadeh and Gray 1979b; Hassanizadeh and Gray 1980).

Inserting Eq. (1.6) into Eq. (1.4) yields a single equation for the fluid pressure p,

$$\frac{\partial(\Phi\rho)}{\partial t} - \nabla\cdot\left\{\rho\frac{\mathbf{K}}{\mu}\left(\nabla p - \rho\mathbf{g}\right)\right\} = \rho q \qquad \text{in } \Omega \tag{1.7}$$

with initial and boundary conditions

$$p(\mathbf{x},0) = p_0(\mathbf{x}) \quad \text{in } \Omega, \tag{1.8a}$$

$$p(\mathbf{x},t) = p_d(\mathbf{x},t) \quad \text{on } \Gamma_d, \qquad \rho\mathbf{u}\cdot\mathbf{n} = \phi(\mathbf{x},t) \quad \text{on } \Gamma_n. \tag{1.8b}$$

In the case of a compressible fluid Eq. (1.7) is of parabolic type, in the incompressible case it is of elliptic type (then the initial condition (1.8a) is not necessary).

1.2.3 TRACER TRANSPORT

We now consider the flow of two fluids F and T which are completely miscible. We assume that the amount of fluid T contained in the mixture has no influence on the flow of the mixture, hence the name tracer.

The volume fraction $C(\mathbf{x},t)$ of fluid T is defined as

$$C(\mathbf{x},t) = \frac{\text{volume of } T \text{ in REV}}{\text{volume of mixture in REV}} \tag{1.9}$$

Further we assume that T and F have the same density ρ. The conservation of mass for fluid T is then modeled by the equation

$$\frac{\partial(\Phi\rho C)}{\partial t} + \nabla\cdot\{\rho\mathbf{u}C - D\nabla C\} = \rho q_T \qquad \text{in } \Omega \tag{1.10}$$

together with appropriate initial and boundary conditions.

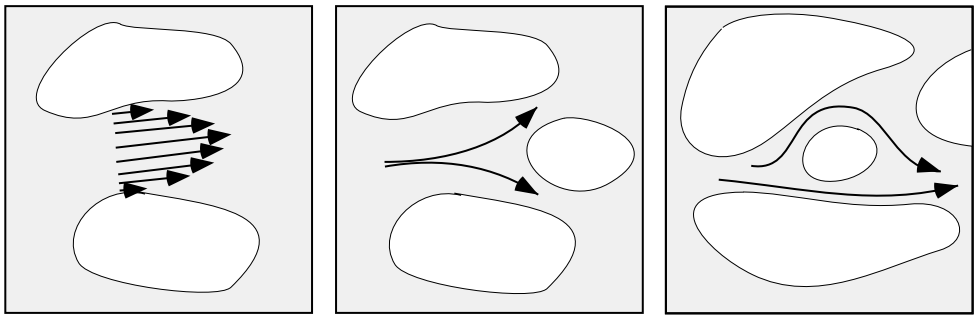
The velocity $\mathbf{u}$ is given by Eq.(1.7) and D is the tensor of *hydrodynamic dispersion*. It is composed of two terms describing *molecular diffusion* and *mechanical dispersion* (see (Scheidegger 1961; Bear 1979)):

$$D = \underbrace{(\Phi/\tau)D_m\mathbf{I}}_{\text{mol. diff.}} + \underbrace{\alpha_T\|\mathbf{u}\|\mathbf{I} + \frac{\alpha_L-\alpha_T}{\|\mathbf{u}\|}\mathbf{u}\mathbf{u}^T}_{\text{mech. dispersion}}. \tag{1.11}$$

Here D_m denotes the molecular diffusion constant and τ the tortuosity of the porous medium which is the average ratio of distance traveled in the microscopic pores of the medium to the net macroscopic distance traveled. The factors α_L and α_T are the parameters of longitudinal and transversal dispersivity. Mechanical dispersion models the spreading of the tracer on the macroscopic level due to the random structure of the porous medium and depends on the size and direction of the flow velocity. After (Allen et al. 1992) we mention three effects illustrated schematically in Fig. 1.6. The non–uniform velocity profile due to the no–slip boundary condition (a) leads to a longitudinal spreading of the tracer. The stream splitting shown in (b) leads to a transversal spreading. Similarly the tortuosity effect illustrated in (c) leads to a longitudinal spreading.

1.2.4 MISCIBLE DISPLACEMENT

We consider again the flow of two completely miscible fluids F and T in a porous medium filling the domain Ω. In contrast to the last subsection, however,



(a) (b) (c)

Figure 1.6: Illustration of mechanical dispersion: (a) Taylor diffusion, (b) stream splitting and (c) tortuosity effect.

the flow of the mixture depends on its composition. The dependence is through density ρ and viscosity μ depending on concentration and possibly on pressure:

$$\rho = \rho(p,C) \qquad \text{density of mixture,} \tag{1.12a}$$

$$\mu = \mu(p,C) \qquad \text{viscosity of mixture.} \tag{1.12b}$$

Furthermore we denote the density of the fluid T by $\rho_T(p)$. The pressure p of the mixture and concentration C of fluid T are now described by two coupled, in general nonlinear, equations

$$\frac{\partial(\Phi\rho(p,C))}{\partial t} - \nabla\cdot\left\{\frac{\rho(p,C)}{\mu(p,C)}\mathbf{K}(\nabla p - \rho(p,C)\mathbf{g})\right\} = \rho(p,C)q, \tag{1.13a}$$

$$\frac{\partial(\Phi\rho_T(p)C)}{\partial t} + \nabla\cdot\{\rho_T(p)\mathbf{u}C - D\nabla C\} = \rho_T(p)q_T \tag{1.13b}$$

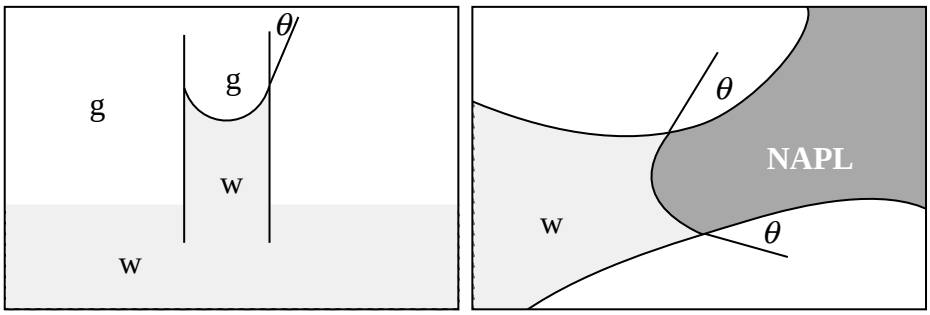
and appropriate boundary and initial conditions.

The first equation, the pressure equation, is coupled to the second via ρ and μ. The second equation, called the concentration equation, is coupled to the first via pressure p and velocity $\mathbf{u}$ (containing pressure). Note that a nonlinear coupling of the equations also exists through the dispersion tensor D depending on $\mathbf{u}$.

Eqs. (1.13) describe for example the miscible flow of fresh and salt water. There the coupling is via the density and the viscosity can be taken constant. Other applications are the miscible displacement of water with certain hydrocarbons. There the dependence of density on pressure and concentration can usually be neglected since the coupling through viscosity is dominant. In that case the equations reduce to

$$-\nabla\cdot\left\{\frac{\mathbf{K}}{\mu(p,C)}(\nabla p - \rho\mathbf{g})\right\} = \rho q, \tag{1.14a}$$

$$\frac{\partial(\Phi\rho C)}{\partial t} + \nabla\cdot\{\rho\mathbf{u}C - D\nabla C\} = \rho q_T. \tag{1.14b}$$



(a) (b)

Figure 1.7: Curved fluid–fluid interface due to capillarity in a capillary tube (a) and in a porous medium (b).

The numerical solution of these equations has been studied extensively, see e. g. (Ewing and Wheeler 1980; Ewing 1983).

1.3 Microscopic Considerations of Multiphase Systems

Single–phase flow is governed by pressure forces arising from pressure differences within the reservoir and the exterior gravitational force. In multiphase flows the sharp interfaces between fluid phases on the microscopic level give rise to a *capillary force* that plays an important rôle in these flows.

1.3.1 CAPILLARITY

Fig. 1.7 shows the interface between two phases in more detail. Part (a) shows a capillary tube in water, i. e. a water–air interface. Part (b) shows a water–NAPL interface in a pore channel between two sand grains. On the molecular level *adhesive forces* are attracting fluid molecules to the solid and *cohesive* forces are attracting molecules of one fluid to each other. At the fluid–fluid interface these forces are not balanced leading to the curved form of the interface (see below).

Wettability. The magnitude of the adhesive forces is decreasing rapidly with distance to the wall. The interaction with the cohesive forces leads to a specific *contact angle* θ between the solid surface and the fluid–fluid interface that depends on the properties of the fluids. The fluid for which $\theta < 90^\circ$ is called the *wetting phase fluid*, the other fluid is called the *non–wetting phase fluid*. In both cases of Fig. 1.7 water is the wetting phase. In the case of three immiscible fluids each fluid is either wetting or non–wetting with respect to the other fluids. E. g. in a water–NAPL–gas system water is typically wetting with respect

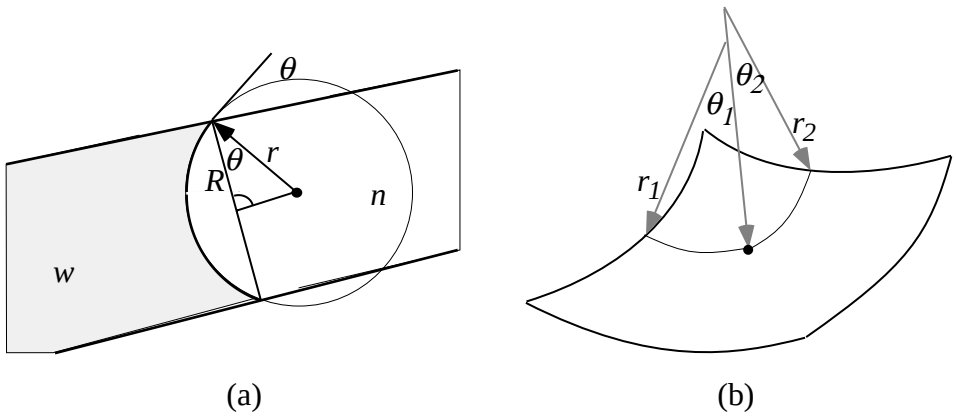


Figure 1.8: Capillary pressure in a tube (a), principal radii of curvature (b).

to both other fluids and NAPL is non–wetting with respect to water and wetting with respect to gas. NAPL is then called the *intermediate* wetting phase.

Surface Tension. The cohesive forces are not balanced at a fluid–fluid interface. Molecules of the wetting phase fluid at the interface experience a net attraction towards the interior of the wetting phase fluid body. This results in the curved form of the interface. In order to move molecules from the interior of the wetting phase to the interface and therefore to enlarge its area work has to be done. The ratio of the amount of work ΔW necessary to enlarge the area of the interface by ΔA is called *surface tension*

$$\sigma = \frac{\Delta W}{\Delta A} \qquad \left[\frac{J}{m^2}\right]. \tag{1.15}$$

1.3.2 Capillary Pressure

The curved interface between a wetting phase w and a non–wetting phase n is maintained by a *discontinuity* in microscopic pressure of each phase. The height of the jump is called *capillary pressure* p_c:

$$p_c = p_n - p_w \geq 0. \tag{1.16}$$

The pressure p_n in the non–wetting phase is larger than the pressure p_w in the wetting phase at the interface (the interface is approached from within the corresponding phase). In order to derive a relation for the capillary pressure we consider a tube with radius diameter $2R$ (R not too large) that is filled with a wetting phase and a non–wetting phase as shown in Fig. 1.8a.

The curved interface has spherical shape with radius r in this case (Bear and Bachmat 1991, p. 335). The radii r and R are related by $R = r\cos\theta$. Now imagine an infinitesimal increase of the radius r by dr. The work required to

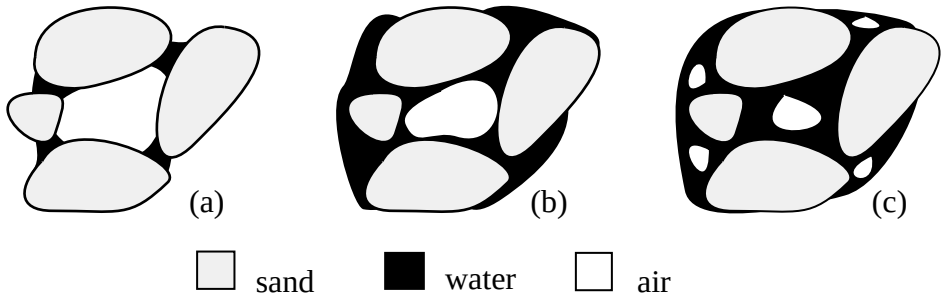


Figure 1.9: Air and water distribution for various amounts of water present. Pendular situation (a), funicular situation (b) and insular air (c).

increase the area of the interface is given by (1.15):

$$\begin{aligned} \Delta W &= \sigma \Delta A = \sigma \left(A(r+dr) - A(r) \right) \\ &= r \left(\frac{1}{2} - \frac{\theta}{\pi} \right) 8\pi r dr + O(dr^2). \end{aligned} \tag{1.17}$$

This work is done by capillary pressure which is assumed to be uniform over the entire interface:

$$\Delta W = F dr = p_c A(r) dr = p_c \left(\frac{1}{2} - \frac{\theta}{\pi} \right) 4\pi r^2 dr. \tag{1.18}$$

Equating these two expressions yields an expression for capillary pressure:

$$p_c = \frac{2\sigma \cos\theta}{R}. \tag{1.19}$$

Surface tension and contact angles are fluid properties whereas R is a parameter of the porous medium. According to (1.19) capillary pressure increases with decreasing pore size diameter.

Similar arguments relate capillary pressure at a point of the interface to surface tension and the principal radii of curvature at this point (also called Laplace's equation):

$$p_c = \sigma \left(\frac{1}{r_1} + \frac{1}{r_2} \right). \tag{1.20}$$

The principal radii of curvature are shown in Fig. 1.8b.

1.3.3 Static Phase Distribution

In this subsection we consider the microscopic spatial distribution of the phases in a two–phase water–air system at rest for various amounts of fluid present in the porous medium (which is assumed to consist of sand grains).

We begin with the situation shown in Fig. 1.9a when only a small amount of water is present in the porous medium. In that case so–called *pendular rings*

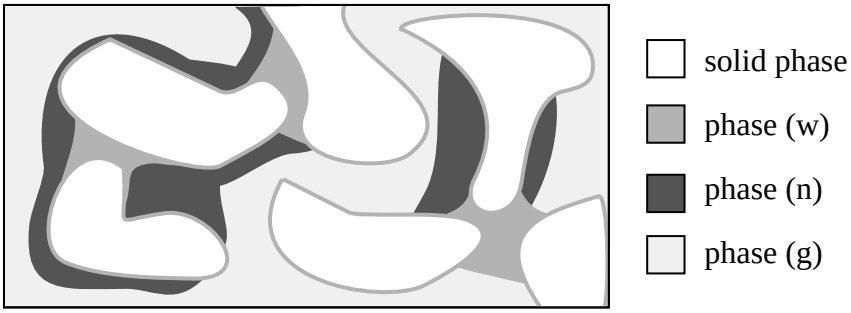


Figure 1.10: Three–phase system.

form around the points of contact of the grains. The pendular rings are disconnected except for a very thin film of water (a few tens of molecules) on the surface of the solid grains. No flow of water is possible in that situation. The water is in the *smallest* pores leading to a large value of capillary pressure according to formula (1.19).

As the amount of water is increased the pendular rings grow until a connected water phase is established and a flow of water is possible. This is the *funicular* situation shown in Fig. 1.9b.

If the amount of water is increased further the air phase becomes disconnected leading to insulated air droplets in the *largest* pores of the porous medium (meaning small capillary pressure). Although no flow is possible in situations (a) and (c) of Fig. 1.9 the amount of water, respectively air can be reduced further by phase transitions, i. e. vaporization and condensation.

1.4 Multiphase Fluid Flow

In this subsection we give the *macroscopic* mathematical model describing multiphase fluid flow in porous media. Each discontinuous phase from the microscopic level is replaced by a continuum on the macroscopic level. We suppose that the void space contains m fluid phases either denoted by greek symbols $\alpha, \beta, \ldots$ or latin symbols $w, n, g, \ldots$ if we want to indicate the wetting phase, non–wetting (NAPL) phase or gaseous phase.

1.4.1 SATURATION

Fig. 1.10 shows a porous medium filled with three fluids (a water phase, a NAPL phase and a gaseous phase). Similar to the void space indicator function we define the phase indicator function

$$\gamma_\alpha(\mathbf{x},t) = \begin{cases} 1 & \mathbf{x} \in \text{phase } \alpha \text{ at time } t \\ 0 & \text{else} \end{cases} \qquad \forall \mathbf{x} \in \Omega. \tag{1.21}$$

Note that the spatial phase distribution now changes with time. For an REV $\Omega_0(\mathbf{x}_0)$ centered at $\mathbf{x}_0$ we define the *saturation* $S_\alpha(\mathbf{x},t)$ of a phase α as

$$S_\alpha(\mathbf{x},t) = \frac{\text{volume of phase } \alpha \text{ in REV}}{\text{volume of void space in REV}} = \frac{\int\limits_{\Omega_0(\mathbf{x}_0)} \gamma_\alpha(\mathbf{x},t)d\mathbf{x}}{\int\limits_{\Omega_0(\mathbf{x}_0)} \gamma(\mathbf{x},t)d\mathbf{x}}. \tag{1.22}$$

Similar remarks about the selection of the REV apply as in the case of the porosity Φ. From the definition of the saturation we obtain immediately

$$\sum_\alpha S_\alpha(\mathbf{x},t) = 1, \quad 0 \le S_\alpha(\mathbf{x},t) \le 1. \tag{1.23}$$

1.4.2 General Form of the Multiphase Flow Equations

Conservation of Mass. Suppose that the porous medium fills the domain $\Omega \subseteq \mathbb{R}^3$. Conservation of mass for each phase α is stated by

$$\frac{\partial(\Phi\rho_\alpha S_\alpha)}{\partial t} + \nabla\cdot\{\rho_\alpha \mathbf{u}_\alpha\} = \rho_\alpha q_\alpha. \tag{1.24}$$

Each phase has its own density ρ_α, saturation S_α, velocity $\mathbf{u}_\alpha$ and source term q_α. Due to the algebraic constraint (1.23) only $m-1$ saturation variables are independent of each other.

Extension of Darcy's Law. As in the single–phase case it can be shown by volume averaging or homogenization techniques that the macroscopic phase velocity can be expressed in terms of the macroscopic phase pressure as

$$\mathbf{u}_\alpha = -\frac{K_\alpha}{\mu_\alpha}(\nabla p_\alpha - \rho_\alpha \mathbf{g}). \tag{1.25}$$

In addition to the assumptions in the single–phase case it has been assumed that the momentum transfer between phases is negligible. The phase permeability K_α, however, depends on the saturation of phase α and can be further decomposed into

$$K_\alpha = k_{r\alpha}(S_\alpha)K, \tag{1.26}$$

i. e. a scalar non–dimensional factor $k_{r\alpha}$ called *relative permeability* and the absolute permeability K which is independent of the fluid. Relation (1.26) is due to (Muskat et al. 1937) and is supported by experimental data, see e. g. (Scheidegger 1974). Theoretical derivations, e. g. in (Whitaker 1986b), suggest that (1.26) may be more complicated in general.

The relative permeability $k_{r\alpha}$ models the fact that the flow paths of fluid α are blocked by the presence of the other phases. It can be considered as a scaling factor and obeys the constraint

$$0 \le k_{r\alpha}(S_\alpha) \le 1. \tag{1.27}$$

Typical shapes of the relative permeability curves will be given in a separate subsection below. Inserting (1.26) into (1.25) we obtain the final form of Darcy's Law for multiphase systems that will be used throughout this book:

$$\mathbf{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K \left(\nabla p_\alpha - \rho_\alpha \mathbf{g}\right). \tag{1.28}$$

The quantity $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$ is often referred to as *mobility*.

Macroscopic Capillary Pressure. In Subs. 1.3.2 it has been shown that the pressure on the microscopic level has a jump discontinuity when passing from one fluid phase to another. The height of the jump is the capillary pressure. This fact is reflected by a *macroscopic capillary pressure* on the macroscopic level

$$p_{c\beta\alpha}(\mathbf{x},t) = p_\beta(\mathbf{x},t) - p_\alpha(\mathbf{x},t) \qquad \forall \beta \neq \alpha. \tag{1.29}$$

The macroscopic capillary pressure $p_{c\beta\alpha}$ will be a function of the phase distribution at point $\mathbf{x}$ and time t:

$$p_{c\beta\alpha}(\mathbf{x},t) = f\left(S_1(\mathbf{x},t), \ldots, S_m(\mathbf{x},t)\right). \tag{1.30}$$

Below we will give some examples of capillary–pressure saturation relationships based on the discussion in Sect. 1.3.

From (1.29) and (1.30) it is evident that only one phase pressure variable can be chosen independently and only $m-1$ capillary pressure–saturation relationships are needed to define the remaining phase pressures. The selection of independent and dependent variables depends on the problem at hand and many examples will be given throughout the text. Before describing specific two– and three–phase models typical shapes of relative permeability and capillary pressure functions will be given.

1.4.3 Capillary Pressure Curves

General Shape. Let us consider a two–phase system with a wetting phase w and a non–wetting phase n. In this case we need a single capillary pressure function $p_c = p_n - p_w$. Initially we assume that the porous medium is filled completely by the wetting phase fluid. When the porous medium is now drained from the bottom with the n-phase coming in from top it is clear from the discussion in Subs. 1.3.3 that the water retreats to smaller and smaller pores with smaller and smaller radii. According to relation (1.19) the capillary pressure at the microscopic fluid–fluid interfaces increases with decreasing pore radius. The (averaged) macroscopic capillary pressure therefore increases with decreasing wetting phase saturation. In general, macroscopic capillary pressure also depends on temperature and fluid composition due to changes in surface tension, but we consider in this work only a dependence $p_c = p_c(S_w)$ in the two–phase case.

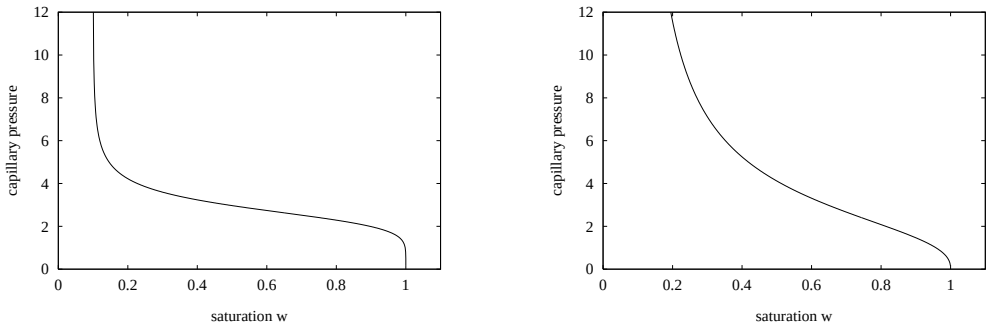
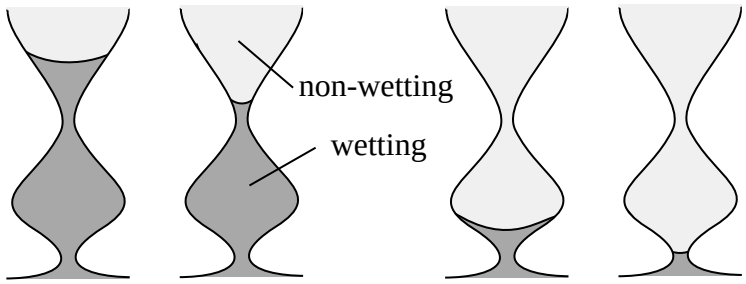


Figure 1.11: Typical shapes of a capillary pressure–saturation function for a poorly graded (left) and a well graded (right) porous medium during drainage.



(a) drainage (b) imbition

Figure 1.12: Ink bottle effect explaining hysteresis in capillary pressure–saturation relationships.

Fig. 1.11 shows two typical capillary pressure–saturation relationships for a porous medium with a highly uniform pore size distribution (left) and a highly non–uniform pore size distribution (right). Both functions are for a drainage cycle.

Entry Pressure. Looking in more detail at Fig. 1.11 we see that at $S_w = 1$ capillary pressure increases rapidly to a value p_d without a noticeable decrease in wetting phase saturation. The value p_d is called *entry pressure* and it is the critical pressure that must be applied for the non–wetting phase to enter the largest pores of the porous medium. A correct treatment of the entry pressure is especially important for heterogeneous porous media.

Hysteresis. The curves in Fig. 1.11 are only valid for a drainage cycle. If the porous medium is subsequently filled again (*imbition*) the capillary pressure–saturation function will be different. In general the $p_c(S)$ relation depends on the complete history of drainage and imbition cycles.

One reason for hysteresis is the *ink bottle effect* illustrated in Fig. 1.12 (after Bear and Bachmat (1991)). Because of the widening and narrowing of the pore channels the same radius, and therefore capillary pressure, occurs for different

elevations. For the same capillary pressure the wetting phase saturation is always higher during drainage than during imbition. For other effects resulting in hysteresis we refer to (Bear and Bachmat 1991; Corey 1994; Helmig 1997).

Residual Saturation. As the reservoir is drained, wetting phase saturation decreases and capillary pressure increases. Finally, the pendular water saturation is reached. The corresponding wetting phase saturation (usually greater than zero) is called *wetting phase residual saturation* S_{wr}. The wetting phase saturation cannot be reduced below residual saturation by pure displacement, however, it can be reduced by phase transition, in this case vaporization. As the residual saturation is approached a large increase in capillary pressure produces practically no decrease in wetting phase saturation. It is this large derivative of the capillary pressure function that will require special care in the numerical solution. The curves in Fig. 1.11 are plotted for a residual saturation $S_{wr} = 0.1$.

On the other hand also the non–wetting phase might have a residual saturation S_{nr} greater than zero as motivated in Subs. 1.3.3 by the insular air droplets. With the residual saturations one can define the *effective saturations* $\bar{S}_\alpha$:

$$\bar{S}_\alpha = \frac{S_\alpha - S_{\alpha r}}{1 - \sum_\beta S_{\beta r}}. \tag{1.31}$$

Obviously we have

$$\sum_\alpha \bar{S}_\alpha = 1, \qquad 0 \le \bar{S}_\alpha \le 1. \tag{1.32}$$

In addition, the residual saturation may depend on position in the case of heterogeneous porous media.

Van Genuchten Capillary Pressure Function. In general there are two possibilities how to obtain capillary pressure–saturation relationships. The first method is direct measurement, for measurement methods we refer to (Corey 1994). The second method is to derive the functional relationship between capillary pressure and saturation from theoretical considerations. Usually these models contain several parameters that are fitted to experimental data.

Here we list the model of Van Genuchten (1980) derived for two–phase water–gas systems. It is written in terms of the effective saturation defined above as

$$p_c(S_w) = \frac{1}{\alpha}\left(\bar{S}_w^{-\frac{1}{m}} - 1\right)^{\frac{1}{n}}. \tag{1.33}$$

The parameter m is often chosen as $m = 1 - \frac{1}{n}$ and therefore only two free parameters n and α remain to be fitted. Typical values of n are in the range $2 \ldots 5$, the α parameter is related to the entry pressure. Fig. 1.13 shows the Van Genuchten function for different values of n and fixed α.

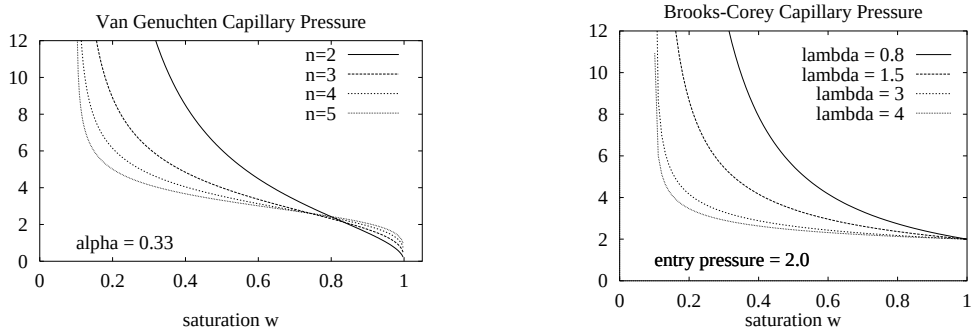


Figure 1.13: Van Genuchten and Brooks–Corey capillary pressure functions for different parameters.

Brooks–Corey Capillary Pressure Function. Another model for two–phase systems is given by Brooks and Corey (1964)

$$p_c(S_w) = p_d \bar{S}_w^{-\frac{1}{\lambda}}. \tag{1.34}$$

with two parameters p_d and λ. p_d is the entry pressure of the porous medium and λ is related to the pore size distribution. A material with a single grain size has a large λ value and a material which is highly non–uniform has a small value of λ, see also Corey (1994). Typical values of λ are in the range $0.2 \ldots 3.0$. Fig. 1.13 shows the Brooks–Corey function for different values of λ and fixed p_d.

Parker Capillary Pressure Function. As an example for three–phase capillary pressure functions we consider the model of Parker et al. (1987). It assumes a wetting phase w, an intermediate wetting phase n and a non–wetting phase g. In the three–phase case we need two capillary pressure functions which we choose as $p_{cnw} = p_n - p_w$ and $p_{cgn} = p_g - p_n$. It is assumed that the function p_{cnw} depends only on S_w and p_{cgn} depends only on $S_w + S_n = 1 - S_g$ in the following way:

$$p_{cnw}(S_w) = \frac{1}{\alpha\beta_{nw}} \left[\left(\bar{S}_w\right)^{\frac{n}{1-n}} - 1 \right]^{\frac{1}{n}}, \tag{1.35a}$$

$$p_{cgn}(S_g) = \frac{1}{\alpha\beta_{gn}} \left[\left(1 - \bar{S}_g\right)^{\frac{n}{1-n}} - 1 \right]^{\frac{1}{n}}. \tag{1.35b}$$

This model is based on the two–phase model of Van Genuchten with the same parameters α and n. The new parameters β_{nw} and β_{gn} are related to the surface tension of the fluid–fluid interfaces:

$$\beta_{gn} = \frac{\sigma_{gw}}{\sigma_{gn}}, \qquad \beta_{nw} = \frac{\sigma_{gw}}{\sigma_{nw}}. \tag{1.36}$$

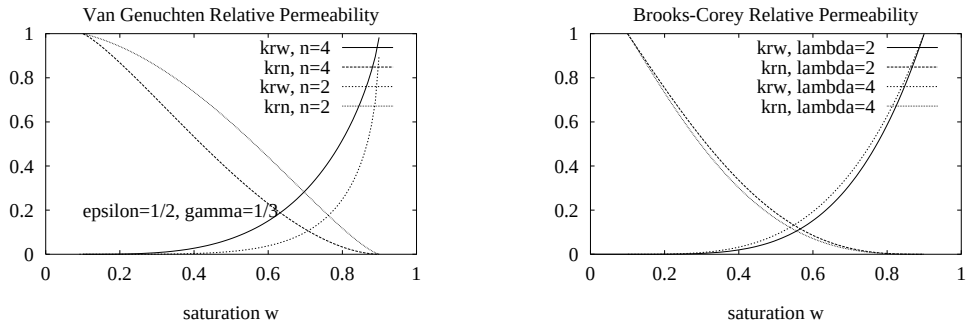


Figure 1.14: Van Genuchten and Brooks–Corey relative permeability functions for different parameters and residual saturations $S_{wr} = S_{nr} = 0.1$.

1.4.4 Relative Permeability Curves

The phase or effective permeability K_α has been defined above as $K_\alpha = k_{r\alpha}K$. In this subsection we review several approaches to define the relative permeability $k_{r\alpha}$. Again there are the two approaches of measurement (see Corey (1994)) and analytical derivation. The analytical approaches use a connection between the capillary pressure–saturation relationship and relative permeability, see Bear and Bachmat (1991) or Helmig (1997). In the two–phase case this leads to the well known functions of Van Genuchten and Brooks–Corey given below.

Van Genuchten Relative Permeability. The Van Genuchten relative permeability functions for a two–phase system with wetting phase *w* and non–wetting phase *n* are written in terms of the residual saturation as

$$k_{rw}(S_w) = \overline{S}_w^{\varepsilon}\left(1 - \left(1 - \overline{S}_w^{\frac{n}{n-1}}\right)^{\frac{n-1}{n}}\right)^2, \tag{1.37a}$$

$$k_{rn}(S_n) = \overline{S}_n^{\gamma}\left(1 - \left(1 - \overline{S}_n\right)^{\frac{n}{n-1}}\right)^{\frac{2(n-1)}{n}} \tag{1.37b}$$

with the form parameters ε and γ that are typically chosen as $\varepsilon = 1/2$ and $\gamma = 1/3$, see Helmig (1997). The parameter n is the same as in the corresponding capillary pressure function of Van Genuchten, i. e. Eq. (1.33). In (1.37) we already used the fact that $m = 1 - \frac{1}{n}$.

Fig 1.14 shows an example for the relative permeability after Van Genuchten. k_{rw} rises slowly for small saturations S_w because the *small* pores are filled first by the wetting phase fluid. When S_w comes close to the maximum saturation k_{rw} is very steep since now the large pores are filled. For k_{rn} we have the opposite situation: The large pores are filled first for small S_n and finally the small pores when S_n is large. Consequently k_{rn} rises faster than k_{rw} for small arguments and slower for large arguments. Relative permeability functions also show hysteresis but this effect is considered to be very small, cf. Corey (1994).

Brooks–Corey Relative Permeability. The Brooks–Corey model for relative permeability in a two–phase system is given by the formulas

$$k_{rw}(S_w) = \bar{S}_w^{\frac{2+3\lambda}{\lambda}}, \tag{1.38a}$$

$$k_{rn}(S_n) = \bar{S}_n^2 \left(1 - \left(1 - \bar{S}_n\right)^{\frac{2+\lambda}{\lambda}}\right). \tag{1.38b}$$

The parameter λ is the same as in the capillary pressure function of Brooks–Corey given by Eq. (1.34). Fig 1.14 shows an example for the relative permeability after Brooks–Corey.

Stone Relative Permeability. As an example of relative permeability–saturation relationships for a three–phase system we consider the model of Stone after Aziz and Settari (1979). Three–phase relative permeabilities are very difficult to measure therefore it has been tried to derive three–phase relative permeabilities from two–phase relative permeabilities. We assume that the three–phase system consist of a wetting phase w, a non–wetting phase g and an intermediate wetting phase n. For simplicity it is further assumed that k_{rw} and k_{rg} depend only on S_w, respectively S_g regardless of the distribution of the other two phases. For the intermediate wetting phase n this is not possible since in a two–phase system $n-w$ phase n fills the large pores and in a system $g-n$ phase n fills the small pores, cf. Bear and Bachmat (1991). Therefore we must have $k_{rn} = k_{rn}(S_w, S_n)$. Using residual saturations the model of Stone defines k_{rn} as follows:

$$k_{rn}(S_w, S_n) = \frac{\bar{S}_n k_{rnw}(S_w) k_{rng}(S_n)}{(1 - S_w)(S_w + S_n)}, \tag{1.39a}$$

$$k_{rnw}(S_w) = \left(1 - \bar{S}_w\right)^{\frac{1}{2}} \left(1 - \bar{S}_w^{\frac{n}{n-1}}\right)^{\frac{2(n-1)}{n}}, \tag{1.39b}$$

$$k_{rng}(S_n) = \bar{S}_n^{\frac{1}{2}} \left(1 - \left(1 - \bar{S}_n^{\frac{n}{n-1}}\right)^{\frac{n-1}{n}}\right)^2. \tag{1.39c}$$

As one can see k_{rnw} considers n to be the non–wetting phase in a $n-w$ system and k_{rng} treats phase n as the wetting phase in a $g-n$ system. The Van Genuchten model with $\varepsilon = \gamma = 1/2$ is used for these two–phase systems. The other two relative permeabilities are defined in correspondence:

$$k_{rw}(S_w) = \bar{S}_w^{\frac{1}{2}} \left(1 - \left(1 - \bar{S}_w^{\frac{n}{n-1}}\right)^{\frac{n-1}{n}}\right)^2, \tag{1.40a}$$

$$k_{rg}(S_g) = \bar{S}_g^{\frac{1}{2}} \left(1 - \left(1 - \bar{S}_g\right)^{\frac{n}{n-1}}\right)^{\frac{2(n-1)}{n}}. \tag{1.40b}$$

For other definitions of three–phase relative permeabilities we refer to Helmig (1997).

1.4.5 Two–Phase Flow Model

We are now in a position to state the complete two–phase flow model. Let the domain $\Omega \subset \mathbb{R}^3$ and time interval $T = (0,T)$ be given. The two–phase problem for phases $\alpha = w,n$ in $\Omega \times T$ then reads

$$\frac{\partial(\Phi\rho_\alpha S_\alpha)}{\partial t} = -\nabla\cdot\{\rho_\alpha \mathbf{u}_\alpha\} + \rho_\alpha q_\alpha, \tag{1.41a}$$

$$\mathbf{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K(\nabla p_\alpha - \rho_\alpha \mathbf{g}), \tag{1.41b}$$

$$S_w + S_n = 1, \tag{1.41c}$$

$$p_n - p_w = p_c(S_w) \tag{1.41d}$$

with initial and boundary conditions

$$S_\alpha(\mathbf{x},0) = S_{\alpha 0}(\mathbf{x}),\ p_\alpha(\mathbf{x},0) = p_{\alpha 0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega, \tag{1.42a}$$

$$S_\alpha(\mathbf{x},t) = S_{\alpha d}(\mathbf{x},t) \text{ on } \Gamma^S_{\alpha d},\quad p_\alpha(\mathbf{x},t) = p_{\alpha d}(\mathbf{x},t) \text{ on } \Gamma^p_{\alpha d}, \tag{1.42b}$$

$$\rho_\alpha \mathbf{u}_\alpha\cdot\mathbf{n} = \phi_\alpha(\mathbf{x},t) \text{ on } \Gamma_{\alpha n}. \tag{1.42c}$$

The boundary conditions (1.42b,1.42c) must be compatible with the algebraic constraints (1.41c,1.41d). Only two variables out of S_w, S_n, p_w and p_n can be chosen as independent unknowns. In the next chapter the advantages and disadvantages of different formulations will be discussed.

In the case of unsaturated groundwater flow the non–wetting (gaseous) phase can be assumed to be at atmospheric pressure, i. e. $p_n = \text{const}$. The wetting phase pressure can then be computed via the capillary pressure function

$$p_w = p_n - p_c(S_w). \tag{1.43}$$

Setting $\Psi = -p_c(S_w)$ and assuming incompressibility of the w–phase we obtain from conservation of mass and Darcy's law for phase w a single equation for Ψ,

$$-\left(p_c^{-1}\right)'(-\Psi)\frac{\partial\Psi}{\partial t} - \nabla\cdot\left\{\frac{k_{rw}(p_c^{-1}(-\Psi))}{\Phi\mu_w} K(\nabla\Psi - \rho_w\mathbf{g})\right\} = q_w/\Phi, \tag{1.44}$$

which is basically Richard's equation from (Richards 1931). This equation is only listed for completeness here and will not be considered further in this work.

1.4.6 Three–Phase Flow Model

In the three–phase case two capillary pressure–saturation functions are required. If we choose, as in the model of Parker, $p_{cnw} = p_n - p_w$ and $p_{cgn} = p_g - p_n$ the capillary pressure between the water and gas phases is given by $p_g - p_w = p_{cnw} + p_{cgn}$. However, if the n–phase is not present in the system, i. e. $S_n = 0$, one

would like to use directly a two–phase capillary pressure function for the water–gas system $p_{cgw} = p_g - p_w$. This situation typically arises in the simulation of a contamination process where the *n*–phase is initially absent. Following Forsyth (1991) we blend between the two– and the full three–phase case in the following way:

$$p_n - p_w = \beta p_{cnw}(S_w) + (1-\beta)p_{cnw}(1), \tag{1.45a}$$
$$p_g - p_n = \beta p_{cgn}(S_g) + (1-\beta)(p_{cgw}(S_w) - p_{cnw}(1)), \tag{1.45b}$$

where

$$\beta = \min(1, S_n/S_n^*). \tag{1.46}$$

This definition of β assumes that $S_{nr} = 0$. S_n^* is a small parameter, e. g. Forsyth and Shao (1991) use $S_n^* = 0.1$. The constant term $p_{cnw}(1)$ in (1.45a) is required in order to represent the entry pressure for the *n*–phase correctly when $S_n = 0$ and S_w near 1. Consequently the term $p_{cnw}(1)$ must be subtracted in the second equation.
The complete three–phase flow model for phases $\alpha = w, n, g$ in $\Omega \times T$ now reads

$$\frac{\partial(\Phi\rho_\alpha S_\alpha)}{\partial t} = -\nabla\cdot\{\rho_\alpha \mathbf{u}_\alpha\} + \rho_\alpha q_\alpha, \tag{1.47a}$$
$$\mathbf{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K(\nabla p_\alpha - \rho_\alpha \mathbf{g}), \tag{1.47b}$$
$$S_w + S_n + S_g = 1, \tag{1.47c}$$
$$p_n - p_w = \beta p_{cnw}(S_w) + (1-\beta)p_{cnw}(1), \tag{1.47d}$$
$$p_g - p_n = \beta p_{cgn}(S_g) + (1-\beta)(p_{cgw}(S_w) - p_{cnw}(1)) \tag{1.47e}$$

with $\beta = \min(1, S_n/S_n^*)$ from above and the initial and boundary conditions

$$S_\alpha(\mathbf{x},0) = S_{\alpha 0}(\mathbf{x}),\ p_\alpha(\mathbf{x},0) = p_{\alpha 0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega, \tag{1.48a}$$
$$S_\alpha(\mathbf{x},t) = S_{\alpha d}(\mathbf{x},t) \text{ on } \Gamma_{\alpha d}^S, \quad p_\alpha(\mathbf{x},t) = p_{\alpha d}(\mathbf{x},t) \text{ on } \Gamma_{\alpha d}^p, \tag{1.48b}$$
$$\rho_\alpha \mathbf{u}_\alpha \cdot \mathbf{n} = \phi_\alpha(\mathbf{x},t) \text{ on } \Gamma_{\alpha n}. \tag{1.48c}$$

Similar to the two–phase case the boundary and initial conditions must be compatible with the algebraic constraints. For the selection of appropriate formulations, i. e. primary and dependent variables we refer to the next chapter.

1.4.7 Compositional Flow

In compositional flows each phase consists of several components. The components (molecular species) are transported within phases and exchanged across phase boundaries (interphase mass transfer). As examples we mention the dissolution of methane in oil or the vaporization (solution) of volatile components

of a NAPL into the gaseous (aqueous) phase. This subsection develops the equations to describe such phenomena in the isothermal case under the assumption of local thermodynamic equilibrium. We assume the general case of m phases and k components.

Component Representation. There are several ways to describe the components within a phase. In Subs. 1.2.3 we already used the *volume fraction* C in the single–phase case. For a component κ in a phase α it reads

$$C_\alpha^\kappa(\mathbf{x},t) = \frac{\text{volume of component } \kappa \text{ in phase } \alpha \text{ in REV}}{\text{volume of phase } \alpha \text{ in REV}}. \tag{1.49}$$

Similarly we can define the *mass fraction* X_α^κ of component κ in phase α:

$$X_\alpha^\kappa(\mathbf{x},t) = \frac{\text{mass of component } \kappa \text{ in phase } \alpha \text{ in REV}}{\text{mass of phase } \alpha \text{ in REV}}. \tag{1.50}$$

Defining the *intrinsic mass density* of component κ in phase α by

$$\rho_\alpha^\kappa(\mathbf{x},t) = \frac{\text{mass of component } \kappa \text{ in phase } \alpha \text{ in REV}}{\text{volume of component } \kappa \text{ in phase } \alpha \text{ in REV}} \tag{1.51}$$

the mass and volume fractions are connected by

$$\rho_\alpha X_\alpha^\kappa = \rho_\alpha^\kappa C_\alpha^\kappa, \tag{1.52}$$

where ρ_α is the density of phase α.
From the definitions above we immediately have

$$\sum_{\kappa=1}^{k} X_\alpha^\kappa = 1, \quad \sum_{\kappa=1}^{k} C_\alpha^\kappa = 1, \qquad \forall \alpha = 1,\dots,m, \tag{1.53}$$

which gives together with (1.52) the relation

$$\rho_\alpha = \sum_{\kappa=1}^{k} C_\alpha^\kappa \rho_\alpha^\kappa. \tag{1.54}$$

Component Mass Balance. Each component κ is transported with its own velocity $\mathbf{u}_\alpha^\kappa$ within phase α. Following Allen et al. (1992) we define the *barycentric phase velocity* as the mass weighted average of all component velocities:

$$\mathbf{u}_\alpha = \sum_{\kappa=1}^{k} X_\alpha^\kappa \mathbf{u}_\alpha^\kappa. \tag{1.55}$$

The deviation of a component's velocity to the mean velocity is then given by

$$\mathbf{w}_\alpha^\kappa = \mathbf{u}_\alpha^\kappa - \mathbf{u}_\alpha. \tag{1.56}$$

Note that the mean velocity is constructed such that

$$\sum_{\kappa=1}^{k} X_\alpha^\kappa \mathbf{w}_\alpha^\kappa = 0. \tag{1.57}$$

Now we can state the equation for conservation of mass for each component κ in a phase α as

$$\frac{\partial(\Phi S_\alpha C_\alpha^\kappa \rho_\alpha^\kappa)}{\partial t} + \nabla \cdot \{\rho_\alpha^\kappa C_\alpha^\kappa \mathbf{u}_\alpha^\kappa\} = r_\alpha^\kappa, \tag{1.58}$$

where r_α^κ is a source/sink term that models the exchange of mass of component κ with the other phases. Using (1.52) and (1.56) we can rewrite the mass balance as

$$\frac{\partial(\Phi S_\alpha \rho_\alpha X_\alpha^\kappa)}{\partial t} + \nabla \cdot \{\rho_\alpha X_\alpha^\kappa \mathbf{u}_\alpha + \mathbf{j}_\alpha^\kappa\} = r_\alpha^\kappa. \tag{1.59}$$

The quantity $\mathbf{j}_\alpha^\kappa = \rho_\alpha X_\alpha^\kappa \mathbf{w}_\alpha^\kappa$, which is the flux produced by the deviation from mean velocity, can be modeled as a diffusive flux analogous to dispersion in single–phase systems:

$$\mathbf{j}_\alpha^\kappa = -D_{pm}^{\kappa,\alpha} \nabla X_\alpha^\kappa. \tag{1.60}$$

However, the approaches for the hydrodynamic dispersion tensor $D_{pm}^{\kappa,\alpha}$ in a multiphase/multicomponent system are even more controversial than in the single–phase case, cf. Helmig (1997, p. 117) or Allen et al. (1992, p. 52). Often the term $\mathbf{j}_\alpha^\kappa$ is simply neglected, see e. g. Peaceman (1977).

For the mean phase velocity it is assumed that the extended multiphase Darcy law

$$\mathbf{u}_\alpha = -\frac{k_{r\alpha}}{\mu_\alpha} K (\nabla p_\alpha - \rho_\alpha \mathbf{g}) \tag{1.61}$$

can be used.

Furthermore we assume that components are only exchanged between phases and no intraphase chemical reactions are taking place. This results in the constraint

$$\sum_{\alpha=1}^{m} r_\alpha^\kappa = 0, \qquad \kappa = 1, \ldots, k, \tag{1.62}$$

for the reaction terms. If the component mass balance (1.59) is summed over all phases the reaction terms cancel out and we obtain the final form (with $\mathbf{j}_\alpha^\kappa = 0$):

$$\sum_{\alpha=1}^{m} \frac{\partial(\Phi S_\alpha \rho_\alpha X_\alpha^\kappa)}{\partial t} + \sum_{\alpha=1}^{m} \nabla \cdot \{\rho_\alpha X_\alpha^\kappa \mathbf{u}_\alpha\} = 0. \tag{1.63}$$

Phase Partitioning. To complete the set of equations we shall restrict ourselves to the isothermal setting and local thermodynamic equilibrium. This means that the flow is slow enough that the partitioning of a component κ across the phases can be determined by equilibrium thermodynamic considerations. Without going into details this yields *algebraic* expressions of the form

$$\frac{X_\alpha^\kappa}{X_\beta^\kappa} = Z_{\kappa\alpha\beta}(T, p_\alpha, p_\beta, X_\alpha^\kappa, X_\beta^\kappa) \tag{1.64}$$

for each $\beta \neq \alpha$. Given one mass fraction X_α^κ the mass fractions of component κ in all other phases can be computed. For a more detailed treatment of thermodynamics we refer to (Allen et al. 1992; Falta 1992; Helmig 1997).

Complete Model. We now show that the equations given are enough to determine all unknown functions. Assuming m phases and k components we have the following unknowns:

Symbol	Description	Count
X_α^κ	mass fractions	km
S_α	phase saturations	m
p_α	phase pressures	m
$\mathbf{u}_\alpha$	mean phase velocities	m
		$(k+3)m$

These unknown functions are determined by the following relations:

Relation	Count
component mass balance summed over phases (1.63)	k
multiphase Darcy law (1.61)	m
capillary pressure–saturation relations	$m-1$
$\sum_{\alpha=1}^m S_\alpha = 1$	1
$\sum_{\kappa=1}^k X_\alpha^\kappa = 1, \quad \forall\alpha$	m
thermodynamic constraints (1.64)	$k(m-1)$
	$(k+3)m$

Note that the number of partial differential equations equals the number of components in the system. All other unknowns are determined by algebraic relations. The particular case of three phases and three components is treated in Forsyth and Shao (1991) and Helmig (1997), non–isothermal flows are considered in Falta (1992) and Helmig (1997). A simplified model with three phases and mass transfer only between the gaseous phase and the oil phase is known as *black oil model*. In the black oil model only the oil phase contains two components, cf. (Peaceman 1977).

Bibliographic Comments

The respective chapters in the books by Allen et al. (1992), Aziz and Settari (1979), Peaceman (1977) and the article by Allen (1985) can be read as an in-

troduction to the field. Bear and Bachmat (1991) and the series of articles by Hassanizadeh and Gray (1979a) and Whitaker (1986a) give a theoretical foundation of the macroscopic single and multiphase flow equations. The books by Corey (1994) and Helmig (1997) give a thorough discussion of the relative permeability and capillary pressure relationships. Compositional flow equations are discussed by Peaceman (1977), Allen et al. (1992) and Helmig (1997). The book edited by Hornung (1997) gives an up–to–date overview of the field of homogenization.

2

Basic Properties of Multiphase Flow Equations

The basic mathematical models for multiphase flow in porous media consist of a set of partial differential equations along with a set of algebraic relations. Typically there are a number of different possibilities to select a set of independent variables with which the remaining unknowns can be eliminated (dependent variables). This results in different mathematical formulations for the same model. The properties of each mathematical formulation depend on the individual problem setup. Moreover, there exist mathematical formulations using new (artificial) unknowns that have favorable mathematical properties. The selection of the proper formulation can strongly influence the behavior of the numerical simulation and is therefore of primary importance.

In this chapter we will almost exclusively consider two–phase immiscible flow. Formulations with primitive variables (i. e. using independent variables present in the mathematical model) and those with artificial variables will be discussed. Of special importance is the treatment of porous media with a discontinuity of media properties like absolute permeability and porosity. The analysis of one–dimensional model problems for both the hyperbolic and the degenerate parabolic case will provide some insight into the complex solution behavior of the two–phase flow model. At the end of the chapter the extension to the three–phase model will be touched briefly. For an introduction to different formulations of the multiphase flow equations see also the books by Peaceman (1977), Chavent and Jaffré (1978), Aziz and Settari (1979) and Helmig (1997).

2.1 Phase Pressure–Saturation Formulation

In this subsection we devise two formulations of the two–phase flow model given in Eqs. 1.41 which are based on “primitive variables”, i. e. variables already present in the model. The type of the resulting system of partial differential equations is determined and its applicability is discussed.

2.1.1 Model Equations Revisited

The model 1.41 consists of two partial differential equations and two algebraic relations for the determination of the four unknowns p_w, p_n, S_w and S_n. In a pressure–saturation formulation one of the pressures and one of the saturations are eliminated using the algebraic constraints.

By substituting

$$S_w = 1 - S_n, \qquad p_n = p_w + p_c(1 - S_n) \tag{2.1}$$

we obtain the (p_w, S_n)–formulation:

$$\frac{\partial(\Phi\rho_w(1-S_n))}{\partial t} = -\nabla\cdot\{\rho_w \mathbf{u}_w\} + \rho_w q_w, \tag{2.2a}$$

$$\mathbf{u}_w = -\frac{k_{rw}(1-S_n)}{\mu_w} K(\nabla p_w - \rho_w \mathbf{g}), \tag{2.2b}$$

$$\frac{\partial(\Phi\rho_n S_n)}{\partial t} = -\nabla\cdot\{\rho_n \mathbf{u}_n\} + \rho_n q_n, \tag{2.2c}$$

$$\mathbf{u}_n = -\frac{k_{rn}(S_n)}{\mu_n} K(\nabla p_w + \nabla p_c(1-S_n) - \rho_n \mathbf{g}). \tag{2.2d}$$

As initial and boundary conditions we may specify

$$S_n(\mathbf{x},0) = S_{n0}(\mathbf{x}),\ p_w(\mathbf{x},0) = p_{w0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega, \tag{2.3a}$$

$$S_n(\mathbf{x},t) = S_{nd}(\mathbf{x},t) \text{ on } \Gamma^S_{nd}, \quad p_w(\mathbf{x},t) = p_{wd}(\mathbf{x},t) \text{ on } \Gamma^p_{wd}, \tag{2.3b}$$

$$\rho_\alpha \mathbf{u}_\alpha\cdot\mathbf{n} = \phi_\alpha(\mathbf{x},t) \text{ on } \Gamma_{\alpha n}. \tag{2.3c}$$

If both fluids are incompressible no initial condition for p_w is required and in order to make p_w uniquely determined the Dirichlet boundary Γ^p_{wd} should be of positive measure.

Similarly we obtain the (p_n, S_w)–formulation by substituting

$$S_n = 1 - S_w, \qquad p_w = p_n - p_c(S_w) \tag{2.4}$$

which yields

$$\frac{\partial(\Phi\rho_n(1-S_w))}{\partial t} = -\nabla\cdot\{\rho_n \mathbf{u}_n\} + \rho_n q_n, \tag{2.5a}$$

$$\mathbf{u}_n = -\frac{k_{rn}(1-S_w)}{\mu_n} K(\nabla p_n - \rho_n \mathbf{g}), \tag{2.5b}$$

$$\frac{\partial(\Phi\rho_w S_w)}{\partial t} = -\nabla\cdot\{\rho_w \mathbf{u}_w\} + \rho_w q_w, \tag{2.5c}$$

$$\mathbf{u}_w = -\frac{k_{rw}(S_w)}{\mu_w} K(\nabla p_n - \nabla p_c(S_w) - \rho_w \mathbf{g}). \tag{2.5d}$$

with initial and boundary conditions given by

$$S_w(\mathbf{x},0) = S_{w0}(\mathbf{x}),\ p_n(\mathbf{x},0) = p_{n0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega, \tag{2.6a}$$

$$S_w(\mathbf{x},t) = S_{wd}(\mathbf{x},t) \text{ on } \Gamma^S_{wd}, \quad p_n(\mathbf{x},t) = p_{nd}(\mathbf{x},t) \text{ on } \Gamma^p_{nd}, \tag{2.6b}$$

$$\rho_\alpha \mathbf{u}_\alpha\cdot\mathbf{n} = \phi_\alpha(\mathbf{x},t) \text{ on } \Gamma_{\alpha n}. \tag{2.6c}$$

A comparison of (2.3) and (2.6) shows that flux type boundary conditions can be specified for both phases in each of the formulations whereas Dirichlet boundary conditions can only be specified (of course) for those variables present in the equations. Note also the structural similarity in both formulations: A code implementing (2.2) can also solve (2.5) by redefining k_{rw}, k_{rn}, p_c and renaming the variables. More intricate differences between the two formulations will be pointed out below.

2.1.2 Type Classification

At first sight both (2.2) and (2.5) look like a system of parabolic partial differential equations. A reformulation reveals, however, that this is not the case. In what follows we restrict ourselves to the incompressible case $\rho_\alpha = \text{const}$, Φ independent of time and p_α. A generalization to the compressible case is given in Subs. (2.2).

Considering first the (p_w, S_n)–formulation we obtain by adding ρ_w^{-1}(2.2a) and ρ_n^{-1}(2.2c) the relation

$$\nabla \cdot \mathbf{u} = q_w + q_n \tag{2.7}$$

where we introduced the *total velocity*

$$\mathbf{u} = \mathbf{u}_w + \mathbf{u}_n. \tag{2.8}$$

From (2.2b) and (2.2d) the total velocity can be written as

$$\mathbf{u} = -\lambda K \left(\nabla p_w + f_n \nabla p_c - \mathbf{G}\right) \tag{2.9}$$

where we introduced the following abbreviations:

$$\lambda_\alpha = k_{r\alpha}/\mu_\alpha \qquad \textit{phase mobility}, \tag{2.10a}$$
$$\lambda = \lambda_w + \lambda_n \qquad \textit{total mobility}, \tag{2.10b}$$
$$f_\alpha = \lambda_\alpha/\lambda \qquad \textit{fractional flow}, \tag{2.10c}$$
$$\mathbf{G} = \frac{\lambda_w \rho_w + \lambda_n \rho_n}{\lambda}\mathbf{g} \qquad \textit{modified gravity}. \tag{2.10d}$$

A set of equations that is equivalent to (2.2a-d) is then given by

$$-\nabla \cdot \{\lambda K \nabla p_w\} = q_w + q_n - \nabla \cdot \left\{\lambda_n p_c' K \nabla S_n + \lambda K \mathbf{G}\right\}, \tag{2.11a}$$
$$\Phi \frac{\partial S_n}{\partial t} + \nabla \cdot \left\{\lambda_n(S_n) K(\rho_n \mathbf{g} - \nabla p_w) + \lambda_n p_c' K \nabla S_n\right\} = q_n. \tag{2.11b}$$

Eq. (2.11a) comes from inserting (2.9) into (2.7) and (2.11b) comes from inserting (2.2d) into (2.2c). Eq. (2.11a) is of elliptic type with respect to the pressure p_w. The type of the second equation (2.11b) is either nonlinear hyperbolic if $p_c' \equiv 0$ or degenerate parabolic if capillary pressure is not neglected. The

diffusion term is degenerate since $\lambda_n(S_n = 0) = 0$. Allowing compressibility of at least one of the fluids would formally turn (2.11a) into a parabolic equation. Since compressibility is typically very small it is still "nearly" elliptic (singularly perturbed).

A similar derivation for the (p_n, S_w)–formulation yields

$$-\nabla \cdot \{\lambda K \nabla p_n\} = q_w + q_n - \nabla \cdot \left\{\lambda_w p_c' K \nabla S_w + \lambda K \mathbf{G}\right\}, \tag{2.12a}$$

$$\Phi \frac{\partial S_w}{\partial t} + \nabla \cdot \left\{\lambda_w(S_w) K(\rho_w \mathbf{g} - \nabla p_n) + \lambda_w p_c' K \nabla S_w\right\} = q_w. \tag{2.12b}$$

2.1.3 Applicability

In order to judge the applicability of both pressure–saturation formulations we consider a weak formulation of the pressure equations (2.11a) and (2.12a). Assuming homogeneous Dirichlet boundary conditions for both pressures and given a saturation S_n or S_w the left hand side of either (2.11a) or (2.12a) defines a $H_0^1(\Omega)$–elliptic bilinear form in the usual way, see (Brenner and Scott 1994) for details. The parameter λ in the bilinear form depends on saturation but is bounded from above and below. In order to get a uniquely determined pressure in $H_0^1(\Omega)$ via the Lax–Milgram theorem the linear functionals given by the right hand sides of (2.11a) and (2.12a)

$$F_n(v) = \int_\Omega q_w + q_n + (\lambda_n p_c' K \nabla S_n + \lambda K \mathbf{G}) \cdot \nabla v dx, \tag{2.13a}$$

$$F_w(v) = \int_\Omega q_w + q_n + (\lambda_w p_c' K \nabla S_w + \lambda K \mathbf{G}) \cdot \nabla v dx \tag{2.13b}$$

must be bounded for all $v \in H_0^1(\Omega)$ and any given saturation which is sufficiently smooth.

Recalling typical shapes of capillary pressure–saturation relationships from Subs. 1.4.3 difficulties can be expected near $\bar{S}_w = 1$ or $\bar{S}_w = 0$ where p_c' can be unbounded. These difficulties are partly compensated by the factor λ_α. In particular we can observe the following:

$$\bar{S}_n \to 1 \qquad \lambda_w p_c' \to 0 \qquad \text{for VG, BC,} \tag{2.14a}$$

$$\bar{S}_w \to 1 \qquad \lambda_n p_c' \to 0 \qquad \text{for VG, BC,} \tag{2.14b}$$

$$\bar{S}_w \to 1 \qquad \lambda_w p_c' < \infty \qquad \text{for BC, } \textit{not} \text{ for VG.} \tag{2.14c}$$

From that we conclude that the (p_w, S_n)–formulation should be used if $\bar{S}_n$ is bounded away from 1 and the (p_n, S_w)–formulation is applicable when $\bar{S}_w$ is bounded away from 1. This holds for both Van Genuchten (VG) and Brooks–Corey (BC) constitutive relations. In the case of Brooks–Corey constitutive relations we see from (2.14a,c) that the (p_n, S_w)–formulation requires no restriction on the range of S_w. However, $\lambda_w p_c'$ might become very large leading to difficulties in the nonlinear solution process.

The argument presented above serves only as a demonstration of the difficulties with phase pressure–saturation formulations. In particular we did not consider at all the properties of the saturation equation. Existence of a weak solution of the system (1.41) with Dirichlet and mixed boundary conditions is shown in (Kroener and Luckhaus 1984). They also assume that $\bar{S}_w$ is bounded away from 0. The next section will provide a formulation that avoids the difficulties associated with the formulations of this subsection. Also most theoretical results for solutions of the two–phase flow problem are based on that formulation.

Finally, we note that (p_w, p_n) is also a possible pair of primary unknowns, called a pressure–pressure formulation. This formulation requires computation of the saturation via inversion of the capillary pressure function $S_w = p_c^{-1}(p_n - p_w)$, which excludes the purely hyperbolic case. Numerically one can also expect difficulties when p_c' is very small. A regularization approach in this case corresponds to artificially adding capillary diffusion to the system. For these reasons we will not consider this formulation in this work.

2.2 Global Pressure Formulation

The global pressure formulation (sometimes also called fractional flow formulation) avoids some of the difficulties associated with the phase pressure formulations introduced in the last section. It is discussed in detail in (Chavent and Jaffré 1978). Parts of the presentation follow the paper Ewing et al. (1995).

2.2.1 Total Velocity

The total velocity has already been introduced in Subs. 2.1.2. Here we will consider the balance equation for total fluid mass in the general case of compressible fluids. Expanding the time derivatives in (1.41a) gives

$$\left\{ \rho_w S_w \frac{\partial \Phi}{\partial t} + \Phi S_w \frac{\partial \rho_w}{\partial t} + \Phi \rho_w \frac{\partial S_w}{\partial t} \right\} + \nabla \cdot \{\rho_w \mathbf{u}_w\} = \rho_w q_w, \tag{2.15a}$$

$$\left\{ \rho_n S_n \frac{\partial \Phi}{\partial t} + \Phi S_n \frac{\partial \rho_n}{\partial t} + \Phi \rho_n \frac{\partial S_n}{\partial t} \right\} + \nabla \cdot \{\rho_n \mathbf{u}_n\} = \rho_n q_n. \tag{2.15b}$$

In order to eliminate the time derivative of the saturations we divide both equations by density, add them and use $S_w + S_n = 1$:

$$\left\{ \frac{\partial \Phi}{\partial t} + \Phi \left(\frac{S_w}{\rho_w} \frac{\partial \rho_w}{\partial t} + \frac{S_n}{\rho_n} \frac{\partial \rho_n}{\partial t} \right) \right\} + \sum_{\alpha = w,n} \rho_\alpha^{-1} \nabla \cdot \{\rho_\alpha \mathbf{u}_\alpha\} = q_w + q_n. \tag{2.16}$$

Applying the product rule to the divergence gives an equation containing the total velocity $\mathbf{u} = \mathbf{u}_w + \mathbf{u}_n$:

$$\frac{\partial \Phi}{\partial t} + \sum_{\alpha = w,n} \rho_\alpha^{-1} \left(\Phi S_\alpha \frac{\partial \rho_\alpha}{\partial t} + \nabla \rho_\alpha \cdot \mathbf{u}_\alpha \right) + \nabla \cdot \mathbf{u} = q_w + q_n. \tag{2.17}$$

The first two terms containing Φ and ρ_α vanish in the incompressible case and we obtain (2.7) again.

Using the extended Darcy–Law (1.41b) and the capillary pressure–saturation relation (1.41d) we can express the total velocity in terms of the non–wetting phase pressure p_n

$$\mathbf{u} = -\lambda K\left(\nabla p_n - f_w \nabla p_c - \mathbf{G}\right) \tag{2.18}$$

with the abbreviations introduced in (2.10).

The phase velocities $\mathbf{u}_\alpha$ can be written in terms of the total velocity *without* referring to the phase pressures using the following observation

$$\lambda_n \mathbf{u}_w - \lambda_w \mathbf{u}_n = \lambda_w \lambda_n K\left(\nabla p_c + (\rho_w - \rho_n)\mathbf{g}\right). \tag{2.19}$$

Exploiting the definition $\mathbf{u} = \mathbf{u}_w + \mathbf{u}_n$ the phase velocities are obtained by:

$$\mathbf{u}_w = f_w \mathbf{u} + \frac{\lambda_n \lambda_w}{\lambda} K\left(\nabla p_c + (\rho_w - \rho_n)\mathbf{g}\right), \tag{2.20a}$$

$$\mathbf{u}_n = f_n \mathbf{u} - \frac{\lambda_w \lambda_n}{\lambda} K\left(\nabla p_c + (\rho_w - \rho_n)\mathbf{g}\right). \tag{2.20b}$$

Note that either λ_w or λ_n is zero for extreme values of saturation.

2.2.2 Global Pressure (Homogeneous Case)

Relation (2.18) would look similar to Darcy's Law if we could find some scalar function $p(\mathbf{x}, t, S_w(\mathbf{x}, t))$ such that

$$\nabla p(\mathbf{x}, t, S_w(\mathbf{x}, t)) = \nabla p_n(\mathbf{x}, t) - f_w(S_w(\mathbf{x}, t)) \nabla p_c(S_w(\mathbf{x}, t)). \tag{2.21}$$

In this case (2.18) turns into

$$\mathbf{u} = -\lambda K\left(\nabla p - \mathbf{G}\right). \tag{2.22}$$

Such a function p will be called *global pressure*. When (2.22) is inserted into (2.17) we obtain an equation for p with a much*weaker* coupling to the saturation (only via λ and $\mathbf{G}$). Moreover, the use of $\mathbf{u}_w$ or $\mathbf{u}_n$ based on $\mathbf{u}$ weakens also the influence of capillary pressure in the saturation equation.

Eq. (2.21) requires that we can write $f_w \nabla p_c$ as the gradient of some scalar function π_w. A necessary condition for this is the interchangeability of partial derivatives, i. e.:

$$\frac{\partial}{\partial x_i}\left\{ f_w \frac{\partial p_c}{\partial x_j} \right\} = \frac{\partial}{\partial x_j}\left\{ f_w \frac{\partial p_c}{\partial x_i} \right\}, \qquad i \neq j. \tag{2.23}$$

This is in general only possible if f_w and p_c are functions of saturation only[1]:

$$f_w = f_w\left(S_w(\mathbf{x}, t)\right), \qquad p_c = p_c\left(S_w(\mathbf{x}, t)\right). \tag{2.24}$$

[1] Two generalizations will be given below

Following (Chavent and Jaffré 1978) we define

$$\pi_w(S) = \int_{S_0}^{S} f_w(\xi) p_c'(\xi) d\xi + \pi_0 \tag{2.25}$$

with S_0, π_0 some constants to be chosen, and set

$$p(\mathbf{x}, t, S_w(\mathbf{x}, t)) = p_n(\mathbf{x}, t) - \pi_w(S_w(\mathbf{x}, t)). \tag{2.26}$$

It can be verified that a global pressure defined in this way obeys (2.21).

We now show how the global pressure is related to the phase pressures. To that end we have to fix the constants S_0, π_0. Let S_{nr} be the non–wetting phase residual saturation and set

$$S_0 = 1 - S_{nr}, \qquad \pi_0 = p_c(1 - S_{nr}), \tag{2.27}$$

i. e.

$$\pi_w(S) = \int_{1-S_{nr}}^{S} f_w(\xi) p_c'(\xi) d\xi + p_c(1 - S_{nr}). \tag{2.28}$$

Note that this integral is well defined for any $S \in [S_{wr}, 1 - S_{nr}]$. For the global pressure we then get for any S_w:

$$\begin{aligned} p(\mathbf{x}, t, S_w) &= p_n - \int_{1-S_{nr}}^{S_w} f_w p_c' d\xi - p_c(1 - S_{nr}) \\ &= p_n + \int_{S_w}^{1-S_{nr}} f_w p_c' d\xi - p_c(1 - S_{nr}) \\ &\leq p_n \end{aligned} \tag{2.29}$$

since $f_w \geq 0, p_c' < 0$ and $p_c(1 - S_{nr}) \geq 0$.

Using $p_n - p_w = p_c(S_w)$ we obtain for the wetting phase pressure:

$$\begin{aligned} p(\mathbf{x}, t, S_w) &= p_n - \pi_w = p_w + p_c(S_w) - \pi_w \\ &= p_w + \int_{1-S_{nr}}^{S_w} p_c'(\xi) d\xi + p_c(1 - S_{nr}) - \int_{1-S_{nr}}^{S_w} f_w p_c' d\xi - p_c(1 - S_{nr}) \\ &= p_w + \int_{1-S_{nr}}^{S_w} (1 - f_w) p_c' d\xi = p_w - \int_{S_w}^{1-S_{nr}} f_n p_c' d\xi \\ &\geq p_w \end{aligned} \tag{2.30}$$

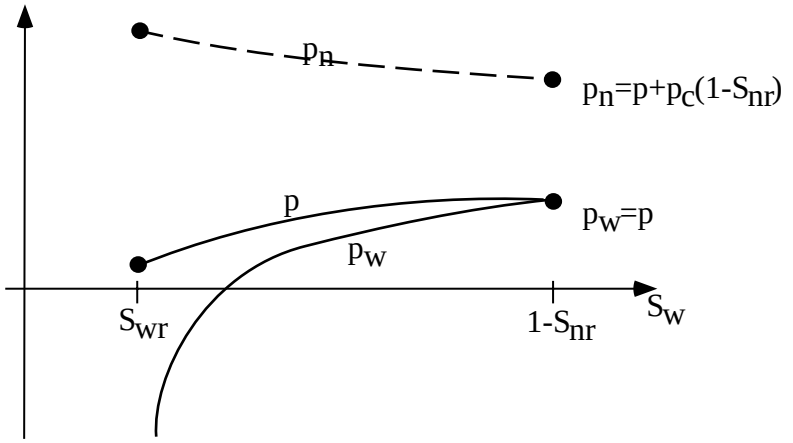


Figure 2.1: Qualitative behavior of global pressure and phase pressures.

since $f_n p_c' \leq 0$. The last two inequalities show that we have

$$p_w \leq p \leq p_n \tag{2.31}$$

for any S_w. If we assume that p is a well behaved function (see discussion below) we get the following extreme cases:

$$S_w = 1 - S_{nr}: \qquad p_w = p, \qquad p_n = p + p_c(1 - S_{nr}) \tag{2.32a}$$

$$S_w = S_{wr}: \qquad p_w = -\infty, \qquad p_n = p + p_c(1 - S_{nr}) + \int_{1-S_{nr}}^{S_{wr}} f_w p_c' d\xi \tag{2.32b}$$

Fig. 2.1 shows the situation graphically.

2.2.3 Complete Set of Equations

We are now able to formulate the complete set of equations for the global pressure–saturation formulation with the unknown functions (p, S_w):

$$\nabla \cdot \mathbf{u} = q_w + q_n - \frac{\partial \Phi}{\partial t} - \sum_{\alpha=w,n} \rho_\alpha^{-1} \left(\Phi S_\alpha \frac{\partial \rho_\alpha}{\partial t} + \nabla \rho_\alpha \cdot \mathbf{u}_\alpha \right), \tag{2.33a}$$

$$\mathbf{u} = -\lambda K (\nabla p - \mathbf{G}), \tag{2.33b}$$

$$\mathbf{u}_w = f_w \mathbf{u} + \lambda_n f_w K (\nabla p_c + (\rho_w - \rho_n)\mathbf{g}), \tag{2.33c}$$

$$\mathbf{u}_n = f_n \mathbf{u} - \lambda_n f_w K (\nabla p_c + (\rho_w - \rho_n)\mathbf{g}), \tag{2.33d}$$

$$\frac{\partial (\Phi \rho_w S_w)}{\partial t} = \rho_w q_w - \nabla \cdot \{\rho_w \mathbf{u}_w\} \tag{2.33e}$$

with the abbreviations introduced in (2.10). In order to avoid any explicit evaluation of the phase pressures p_α in the compressible case we evaluate $\rho_\alpha = \rho_\alpha(p)$

and $\Phi = \Phi(p)$. This is justified since both quantities vary only slowly with pressure and we have $p \approx p_\alpha$ due to the discussion above, cf. Chavent and Jaffré (1978). The boundary conditions are now given in terms of the global pressure and total velocity:

$$S_w(\mathbf{x},0) = S_{w0}(\mathbf{x}),\ p(\mathbf{x},0) = p_0(\mathbf{x}) \qquad \mathbf{x} \in \Omega, \tag{2.34a}$$

$$S_w(\mathbf{x},t) = S_{wd}(\mathbf{x},t) \text{ on } \Gamma_{wd}^S, \quad p(\mathbf{x},t) = p_d(\mathbf{x},t) \text{ on } \Gamma_d^p, \tag{2.34b}$$

$$\rho_w \mathbf{u}_w \cdot \mathbf{n} = \phi_w(\mathbf{x},t) \text{ on } \Gamma_{wn}, \quad \mathbf{u} \cdot \mathbf{n} = U(\mathbf{x},t) \text{ on } \Gamma_n. \tag{2.34c}$$

It should be noted that global pressure and total velocity are mathematical constructs, which makes it difficult to measure the boundary conditions in an experiment. See (Chen et al. 1994) for a discussion of the incorporation of various boundary conditions.

The formulation (2.33) results in a weaker coupling of pressure and saturation equation. Assuming incompressibility and inserting (2.33b) into (2.33a) we obtain for the pressure equation:

$$-\nabla \cdot \{\lambda K \nabla p\} = q_w + q_n - \nabla \cdot \{\lambda K \mathbf{G}\} \tag{2.35}$$

in contrast to (2.11a) and (2.12a). The right hand side now always produces a bounded linear functional for any given S_w and we have $p \in H_0^1(\Omega)$ (assuming Dirichlet boundary conditions). Having p one can compute the phase pressures via (2.29) and (2.30), i. e.:

$$p_\alpha = p + \text{ “correction}_\alpha\text{”} \tag{2.36}$$

where the correction is in general not in a Sobolev space. From (2.20a) and (2.20b) we see that both phase velocities are well defined since $\lambda_n \lambda_w p_c'$ is bounded. This results in a reduction of the nonlinearity (with respect to capillary pressure) of the saturation equation.

Most theoretical results concerning the existence of a solution to the incompressible two–phase flow problem are based on the global pressure formulation. Chavent and Jaffré (1978) show the existence of a solution to certain variational formulations of the incompressible version of (2.33) in the nondegenerate ($\lambda_w f_n p_c' \geq \eta > 0$) and degenerate case. Uniqueness has been shown only in the case of a complete decoupling of pressure and saturation equation (cf. (Chavent and Jaffré 1978)). The solutions of degenerate parabolic equations have very low regularity. (Yotov 1997) points out that $S_w \in L^\infty(0,T;L^1(\Omega))$ and $\frac{\partial S_w}{\partial t} \in L^2(0,T;H^{-1}(\Omega))$. The existence of classical solutions locally in time for the incompressible, elliptic–hyperbolic ($p_c \equiv 0$) two–phase flow problem has been shown by Schroll and Tveito (1997).

2.2.4 Global Pressure for Heterogeneous Media

Subs. 2.2.2 introduced a global pressure function for the case where relative permeability and capillary pressure functions are the same throughout the domain. This subsection introduces a global pressure function in the case where the capillary pressure function varies with position in a special way.

Due to changes in pore diameter the capillary pressure function will vary with porosity and/or absolute permeability. (Leverett 1941) modeled this dependence in the following form

$$p_c(S_w(\mathbf{x},t),\mathbf{x}) = p_{CM}(\mathbf{x})J(S_w(\mathbf{x},t)) \tag{2.37}$$

where J is a normalized capillary pressure function ("J–Leverett function") and $p_{CM}(\mathbf{x}) = \sigma\sqrt{\Phi(\mathbf{x})/k(\mathbf{x})}$ is a scaling factor depending on porosity and absolute permeability. In this subsection we will extend the global pressure formulation to a capillary pressure function of this form with the additional assumption that p_{CM} depends *smoothly* on $\mathbf{x}$. This derivation follows Chavent and Jaffré (1978)

The idea is to find a global pressure function p such that

$$\nabla p = \nabla p_n - f_w \nabla p_c + d(S_w(\mathbf{x},t),x) \tag{2.38}$$

with an easily computable "correction" function d (compare with the original relation (2.21)). We then can replace (2.22) by

$$\mathbf{u} = -\lambda K\left(\nabla p - d(S_w(\mathbf{x},t),x) - \mathbf{G}\right). \tag{2.39}$$

In order to derive d for the special case (2.37) we obtain from (2.25):

$$\pi_w(S,\mathbf{x}) = p_{CM}(\mathbf{x})\left(\int_{1-S_{nr}}^{S} f_w(\xi)J'(\xi)d\xi + J(1-S_{nr})\right). \tag{2.40}$$

With $p = p_n - \pi_w$ we get

$$\begin{aligned}
\nabla p &= \nabla p_n - \nabla \pi_w \\
&= \nabla p_n - \nabla p_{CM}\left(\int_{1-S_{nr}}^{S} f_w(\xi)J'(\xi)d\xi + J(1-S_{nr})\right) \\
&\qquad - p_{CM}\nabla\left(\int_{1-S_{nr}}^{S} f_w(\xi)J'(\xi)d\xi + J(1-S_{nr})\right) \\
&= \nabla p_n - \nabla p_{CM}\left(f_w(S_w)J(S_w) - \int_{1-S_{nr}}^{S_w} f_w' J d\xi\right) \\
&\qquad - p_{CM} f_w \nabla J \\
&= \nabla p_n - f_w \nabla p_c + \nabla p_{CM}\left(\int_{1-S_{nr}}^{S_w} f_w' J d\xi\right).
\end{aligned} \tag{2.41}$$

Comparison with (2.38) shows that we have

$$d(S_w(\mathbf{x},t),\mathbf{x}) = \nabla p_{CM}(\mathbf{x})\left(\int_{1-S_{nr}}^{S_w(\mathbf{x},t)} f_w' J d\xi\right). \tag{2.42}$$

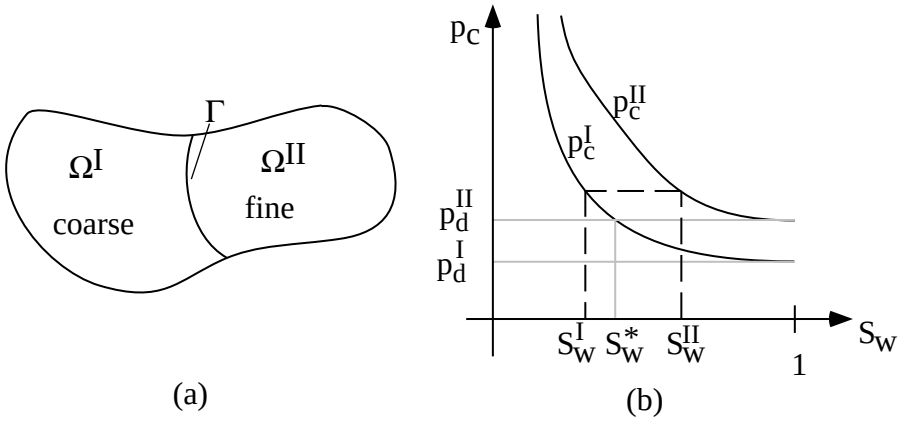


Figure 2.2: A porous medium with a discontinuity.

Note that we can handle the case of Brooks–Corey capillary pressure functions where $J(1-S_{nr}) \neq 0$. In a numerical formulation the integral in the expression for d should be handled analytically.

2.3 Porous Medium with a Discontinuity

In this section we consider a porous medium consisting of a coarse sand in one part of the domain and a fine sand in another part of the domain. On the macroscopic scale this is modeled by a *discontinuity* of porous medium properties at the interface Γ separating the two subdomains.

2.3.1 Macroscopic Model

To fix notation Ω^I is occupied by the coarse material and Ω^{II} is occupied by the fine material (see Fig. 2.2a). The absolute permeability $K(\mathbf{x}) = k(\mathbf{x})\mathbf{I}$ (assumed to be isotropic) will undergo a jump discontinuity

$$k(\mathbf{x}) = \begin{cases} k^I & \mathbf{x} \in \Omega^I \\ k^{II} & \mathbf{x} \in \Omega^{II} \end{cases} \tag{2.43}$$

at the interface Γ. Similarly the porosity may vary from Φ^I in Ω^I to Φ^{II} in Ω^{II}. According to Subs. 2.2.4 there will also be different capillary pressure–saturation relationships in the two subdomains reflecting the change in pore size diameter. Fig. 2.2b shows two typical curves using the Brooks–Corey model.

When the porous medium is initially fully–saturated with water a non–wetting fluid approaching the interface from the coarse sand region Ω^I will only enter the fine sand region Ω^{II} if capillary pressure is large enough for the smaller pores in Ω^{II} to be penetrated. This minimum pressure is called *threshold pressure* or *non–wetting phase entry pressure*, cf. (Bear 1972), and is expressed by p_d in the Brooks–Corey model. Note that $p_d^{II} > p_d^I$ in Fig. 2.2b. This explains the

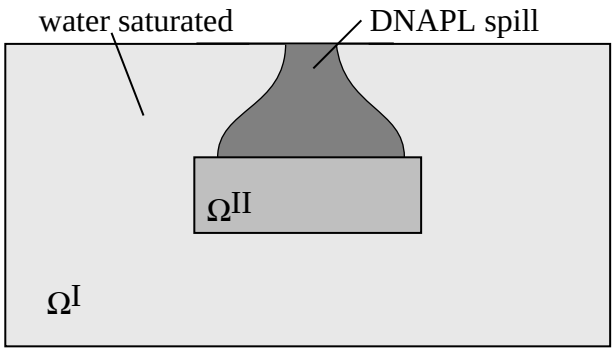


Figure 2.3: Pooling of a DNAPL over a fine sand lens.

"pooling" of a dense nonaqueous phase liquid (DNAPL) over a fine sand lens as shown in Fig. 2.3.

Consider now the situation where both fluids are present at each side of the interface. Let $S_w^I < S_w^*$ (see Fig. 2.2b) be the wetting phase saturation at a point of the interface when approached from Ω^I and S_w^{II} the corresponding saturation when approached from Ω^{II}. From continuity of capillary pressure, (Bear 1972, p. 452), we have $p_c^I(S_w^I) = p_c^{II}(S_w^{II})$ and consequently the wetting phase saturation is *discontinuous* across the interface Γ.

In the next two subsections we develop the mathematical models for porous media with a discontinuity using first a phase pressure formulation and then the global pressure formulation.

2.3.2 Phase Pressure Formulation

We consider a single discontinuity as shown in Fig. 2.2a. The porous medium is initially fully saturated with the wetting phase and we assume that the wetting phase stays always mobile on both sides of the interface. Therefore it is appropriate to choose the (p_w, S_n)–formulation in both subdomains. Each of the two second–order equations will require two conditions at the interface.

Since no mass is lost/produced at the interface we have that

$$\rho_w \mathbf{u}_w \cdot \mathbf{n} \text{ and } \rho_n \mathbf{u}_n \cdot \mathbf{n} \text{ are continuous across } \Gamma, \tag{2.44}$$

where $\mathbf{n}$ is the vector normal to Γ pointing in direction of Ω^{II}.

By analyzing a one–dimensional flow without gravity van Duijn et al. (1995) derived a condition for the wetting phase saturation at the interface which they call the *extended capillary pressure condition*:

$$S_n^{II} = \begin{cases} 0 & S_n^I < 1 - S_w^* \\ 1 - \left(p_c^{II}\right)^{-1}\left(p_c^I(1 - S_n^I)\right) & S_n^I \geq 1 - S_w^* \end{cases} \tag{2.45}$$

where S_n^I and S_n^{II} are the non–wetting phase saturations at a point on Γ when approached from Ω^I and Ω^{II} respectively. S_w^* is the threshold saturation given

by $p_c^I(S_w^*) = p_c^{II}(1)$. In the case of $S_n^I < 1 - S_w^*$ we have $S_n^{II} = 0$ in Ω^{II} and the non–wetting phase does not exist there. Consequently capillary pressure, which is $p_n - p_w$, is undefined in Ω^{II} and need not be continuous across Γ.

Finally, we need a condition for p_w at the interface which is:

$$p_w \text{ is continuous across } \Gamma. \tag{2.46}$$

This follows from the fact that we assumed a mobile wetting phase on both sides of the interface, cf. also (de Neef and Molenaar 1997) where this formulation has been used for theoretical and numerical analysis.

2.3.3 Global Pressure Formulation

We now want to formulate the conditions at the interface when the global pressure formulation with p and S_w as unknowns is used.

In each of the two subdomains Ω^I and Ω^{II} the capillary pressure–saturation relationship is fixed and the equations (2.33) are valid. The interface conditions on Γ will be derived from the (p_w, S_n)–formulation above.

For the flux continuity we obtain in the incompressible case from (2.44) that

$$\mathbf{u}_w \cdot \mathbf{n} \text{ and } \mathbf{u} \cdot \mathbf{n} \text{ are continuous across } \Gamma. \tag{2.47}$$

In the compressible case we have that $\rho_w(p_w)$ is continuous across Γ since p_w is continuous. For the non–wetting phase flux we have that it is either zero if $S_n^I < 1 - S_w^*$ or that p_n and therefore $\rho_n(p_n)$ is continuous if the n–phase is mobile on both sides of the interface. This shows that (2.47) also extends to the compressible case.

The interface condition for S_n from (2.45) is simply rewritten here in terms of S_w:

$$S_w^{II} = \begin{cases} 1 & S_w^I > S_w^* \\ \left(p_c^{II}\right)^{-1}\left(p_c^I(S_w^I)\right) & S_w^I \le S_w^* \end{cases} \tag{2.48}$$

The interface condition for global pressure requires more attention. Since global pressure involves the saturation it will also be discontinuous at the interface in general.

Note that we have the following two equivalent representations of global pressure p from (2.30):

$$p = p_n - \int_1^{S_w} f_w(\xi) p_c'(\xi) d\xi - p_c(1) = p_w + \int_1^{S_w} f_n(\xi) p_c'(\xi) d\xi \tag{2.49}$$

which holds in Ω^I for p_c^I and in Ω^{II} for p_c^{II}. We consider the following two cases:

Case I: $S_w^I > S_w^*$, $S_w^{II} = 1$. Let us first consider the case where the critical saturation is not yet reached and Ω^{II} contains only water. Then we have that p_w is continuous over the interface but p_n is not defined in Ω^{II}. Consequently we use the second representation from (2.49) for the interface condition:

$$p^{II} = p^I - \int_1^{S_w^I} f_n^I(\xi)(p_c^I)'(\xi)d\xi \tag{2.50}$$

for any point on Γ. Note that p is the same on both sides if $S_w^I = 1$.

Case II: $S_w^I \leq S_w^*$, $p_c^I(S_w^I) = p_c^{II}(S_w^{II})$. If the critical saturation is reached we can use the continuity of p_n (which is now defined on both sides) for the interface condition:

$$p^{II} = p^I + \int_1^{S_w^I} f_w^I(\xi)(p_c^I)'(\xi)d\xi - \int_1^{S_w^{II}} f_w^{II}(\xi)(p_c^{II})'(\xi)d\xi + p_c^I(1) - p_c^{II}(1). \tag{2.51}$$

Note that for $S_w^I = S_w^*$ case I and II yield the same jump in global pressure. In case II we could also have used the continuity of p_w for intermediate saturation values. However, if water saturation becomes very small in Ω^I the formulation using p_n behaves better.

2.4 One–dimensional Model Problems

In order to get some insight into the complex behavior of the two–phase flow model it is very helpful to consider one–dimensional model problems. Under additional simplifying assumptions we will derive analytical solutions for the purely hyperbolic case and a case with degenerate capillary diffusion.

2.4.1 One–dimensional Simplified Model

In the case of two incompressible fluids, zero sources and zero gravity the pressure equation of the global pressure formulation (2.33) in one space dimension reduces to

$$\frac{\partial u}{\partial x} = 0, \tag{2.52a}$$

$$u = -\lambda K \frac{\partial p}{\partial x}, \tag{2.52b}$$

in the domain $\Omega = (0, L)$ with boundary conditions

$$u(0,t) = U \geq 0, \qquad p(L,t) = P. \tag{2.53}$$

From (2.52a) together with the boundary condition for u we obtain

$$u(x,t) = U, \tag{2.54}$$

i. e. the total velocity u is *constant* in space and time. For the global pressure p we obtain

$$p(x,t) = P + U \int_x^L \frac{1}{\lambda(S_w(\xi,t))K(\xi)} d\xi \tag{2.55}$$

for a given saturation $S_w(x,t)$. In the case of constant λ (i. e. $\mu_w = \mu_n$, $k_{rw} = 1 - k_{rn}$) and K pressure depends linearly on x.

For the saturation equation we now consider the purely hyperbolic case with vanishing capillary pressure and the degenerate parabolic case.

2.4.2 Hyperbolic Case

In the case of zero capillary pressure we obtain for the saturation equation

$$\frac{\partial S_w}{\partial t} + \frac{U}{\Phi}\frac{\partial}{\partial x} f_w(S_w) = 0 \tag{2.56}$$

where U and Φ are constant. The following boundary and initial conditions are imposed:

$$S_w(0,t) = S, \qquad S_w(x,0) = S_{w0}(x). \tag{2.57}$$

Eq. (2.56) is of nonlinear hyperbolic type and is called “Buckley–Leverett equation”. In order to ease writing we set

$$f(S_w) = \frac{U}{\Phi} f_w(S_w) \tag{2.58}$$

which transforms (2.56) into the standard form

$$\frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x} f(S_w) = 0 \tag{2.59}$$

The solution of this equation is very well understood, we refer to (Renardy and Rogers 1993), (LeVeque 1992) and (Helmig 1997) for a detailed discussion. We only recapitulate the most important facts here without proofs and show applications to different fractional flow functions.

The most prominent feature of hyperbolic conservation laws is that they allow discontinuous solutions called “shocks”. Such a solution does not satisfy the differential equation in the classical sense at all points. Therefore the notion of a generalized (“weak”) solution is introduced that involves some integral form of (2.59). Unfortunately the weak solution need not be unique and one requires additional conditions that select the correct physical solution from all possible weak solutions. Typically, this is done either by the method of “vanishing viscosity” or by stating so–called “entropy conditions”.

The numerical solution of hyperbolic conservation laws inherits the mathematical difficulties mentioned above. Methods have been developed which accurately represent shocks and that converge to the correct physical solution without spurious oscillations. These questions will be considered in a later chapter.

We will compute exact solutions of (2.56) for the so–called Riemann problem. It solves (2.56) in an unbounded domain $\Omega = \mathbb{R}$ with a single discontinuity at $x = 0$ as initial condition:

$$S_{w0}(x) = \begin{cases} S_w^L & x \le 0 \\ S_w^R & x > 0 \end{cases}, \qquad S_w^L > S_w^R. \tag{2.60}$$

S_w^L is called left state and S_w^R is called right state. Since we assume $U \ge 0$ (see above) and $S_w^L > S_w^R$ the non–wetting phase is displaced by the wetting phase.

From the definition of the fractional flow function we obtain

$$f(S_w) = \frac{U}{\Phi} f_w(S_w) = \frac{U}{\Phi} \frac{k_{rw}(S_w)}{k_{rw}(S_w) + \frac{\mu_w}{\mu_n} k_{rn}(1 - S_w)}. \tag{2.61}$$

The form of the solution is governed by the shape of the relative permeability functions and the viscosity ratio. An important quantity is the *frontal mobility ratio* defined as

$$M = \frac{k_{r\alpha}(\text{saturation of } \textit{displacing} \text{ fluid behind the front})}{k_{r\beta}(\text{saturation of } \textit{displaced} \text{ fluid ahead of the front})} \cdot \frac{\mu_\beta}{\mu_\alpha}, \tag{2.62}$$

where α is the displacing fluid and β is the fluid being displaced.

Linear Relative Permeability. We now consider linear relative permeabilities with variable viscosity ratio, i. e. we have

$$k_{rw}(S_w) = S_w, \qquad k_{rn}(S_w) = 1 - S_w \tag{2.63}$$

and the corresponding flow function

$$f_w(S_w) = \frac{S_w}{S_w + \frac{\mu_w}{\mu_n}(1 - S_w)}. \tag{2.64}$$

Fig. 2.4 shows f_w and its derivative for different viscosity ratios μ_n/μ_w.

The method of characteristics applied to the Riemann problem for (2.59) states that we have

$$S_w(x,t) = S_0 \tag{2.65}$$

for all points along the straight line

$$(x,t) \in \left\{ (\hat{x},\hat{t}) | \hat{x} = \hat{t} \cdot f'(S_0) + x_0 \right\} \tag{2.66}$$

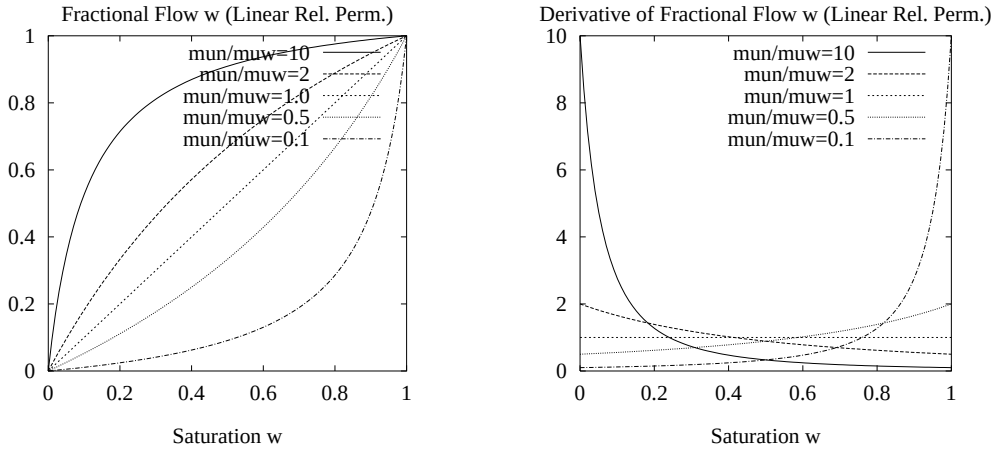


Figure 2.4: Fractional flow function f_w (left) and its derivative (right) for linear relative permeabilities and various viscosity ratios μ_n/μ_w.

with $S_0 = S_{w0}(x_0)$, $x_0 \in \Omega$. Looking at the graph of f we can identify the following three situations, see (Renardy and Rogers 1993) for details.
Case I, $\mu_n > \mu_w$: From Fig. 2.4 we see that $f'(S_w^L) < f'(S_w^R)$ in this case. The solution is continuous for all $t > 0$ and is given by

$$S_w(x,t) = \begin{cases} S_w^L & x/t \le f'(S_w^L) \\ (f')^{-1}(x/t) & f'(S_w^L) < x/t < f'(S_w^R) \\ S_w^R & x/t > f'(S_w^R) \end{cases} . \tag{2.67}$$

This solution is called a *rarefaction wave*.
Case II, $\mu_n = \mu_w$: We have $f'(S_w^L) = f'(S_w^R) = U/\Phi$. Eq. (2.59) is linear in this case and the initial discontinuity is transported through the domain with speed $s = U/\Phi$:

$$S_w(x,t) = \begin{cases} S_w^L & x \le st \\ S_w^R & x > st \end{cases} \tag{2.68}$$

Case II, $\mu_n < \mu_w$: We have $f'(S_w^L) > f'(S_w^R)$ and obtain a shock solution as in case II but with the shock speed s given by the Rankine–Hugoniot condition

$$s = \frac{f(S_w^L) - f(S_w^R)}{S_w^L - S_w^R}. \tag{2.69}$$

The shock obeys the Lax shock criterion which states that all characteristics must enter the shock (in addition to the Rankine–Hugoniot condition). The Lax shock criterion is the entropy condition in this case. Fig. 2.5 shows the characteristic curves for the three different cases discussed above.

Fig. 2.6 shows solution plots for each of the three cases discussed above. The parameters have been set to $U = 3 \cdot 10^{-7}[m/s]$ and $\Phi = 1/5$. The top plot shows

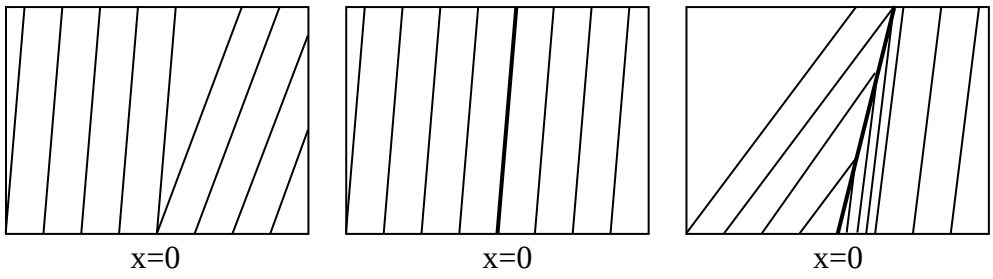


Figure 2.5: Characteristic curves for the three different cases with linear relative permeabilities

the case $\mu_n/\mu_w = 2$ with $S_w^L = 1$ and $S_w^R = 0$. The solution is continuous but not continuously differentiable. The middle plot shows the linear case $\mu_n/\mu_w = 1$ and the bottom plot shows the case $\mu_n/\mu_w = 0.5$. Here the left state has been changed to $S_w^L = 0.9$ in order to show dependence of the shock speed according to (2.69). A comparison of the three plots shows the fast movement of the "front tip" for the rarefaction wave. A typical viscosity ratio for water and oil is $\mu_n/\mu_w = 20$.

S–Shaped Fractional Flow Function. In the case of linear relative permeabilities the function f' is either monotonely decreasing or monotonely increasing (or constant in the linear case). This results in either a rarefaction wave or a shock solution. For typical relative permeabilities used in two–phase flow models, like Brooks–Corey or Van Genuchten functions, we obtain an S–shaped fractional flow function as shown in Fig. 2.7. The resulting solution may have a shock or a rarefaction wave or a combination of both. The case of S–shaped fractional flow functions is treated extensively in (LeVeque 1992) and (Helmig 1997).

We consider a Riemann problem (see Eq. (2.60)) with left and right states $S_w^L > S_w^R$. The value where the derivative of the fractional flow function f_w is maximal is called *inflection point* S_w^I. The solution is obtained by considering the following cases.

Case I, $S_w^R < S_w^L \leq S_w^I$: In that range we have $f_w'(S_w^L) > f_w'(S_w^R)$ as in case III of the last subsection. Therefore we obtain a shock solution

$$S_w(x,t) = \begin{cases} S_w^L & x \leq st \\ S_w^R & x > st \end{cases} \tag{2.70}$$

with the shock speed s given by the Rankine–Hugoniot condition as before:

$$s = \frac{f(S_w^L) - f(S_w^R)}{S_w^L - S_w^R} = \frac{U}{\Phi} \frac{f_w(S_w^L) - f_w(S_w^R)}{S_w^L - S_w^R}. \tag{2.71}$$

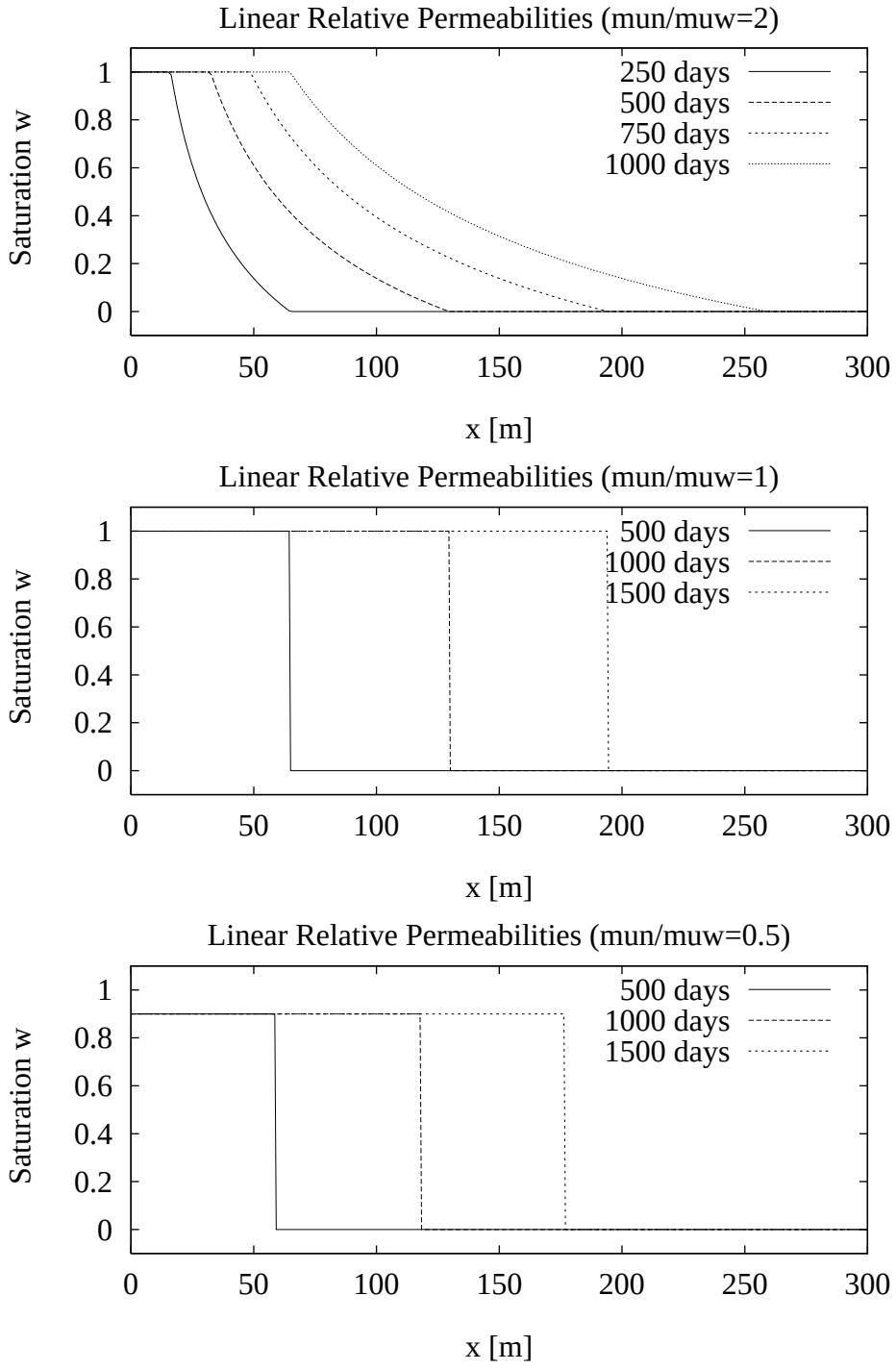


Figure 2.6: Solutions of the Buckley–Leverett problem with linear relative permeability and $\mu_n/\mu_w = 2, 1, 0.5$ (from top).

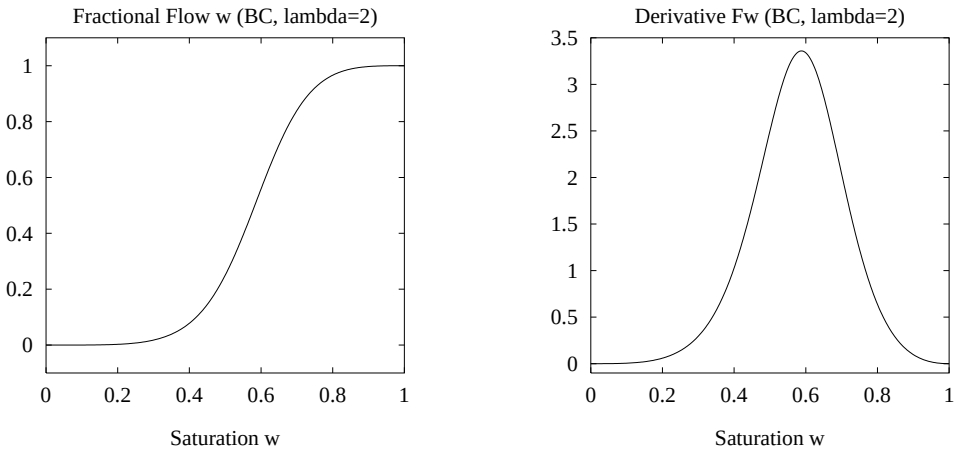


Figure 2.7: Fractional flow function f_w (left) and its derivative (right) for Brooks–Corey relative permeabilities ($\lambda = 2$, $\mu_n/\mu_w = 1$).

Case II, $S_w^I \le S_w^R < S_w^L$: Now we have $f_w'(S_w^L) < f_w'(S_w^R)$ and obtain a rarefaction wave solution:

$$S_w(x,t) = \begin{cases} S_w^L & x/t \le f'(S_w^L) \\ (f')^{-1}(x/t) & f'(S_w^L) < x/t < f'(S_w^R) \\ S_w^R & x/t > f'(S_w^R) \end{cases} . \tag{2.72}$$

For the remaining two cases we define the *tangential point* saturation S_w^T such

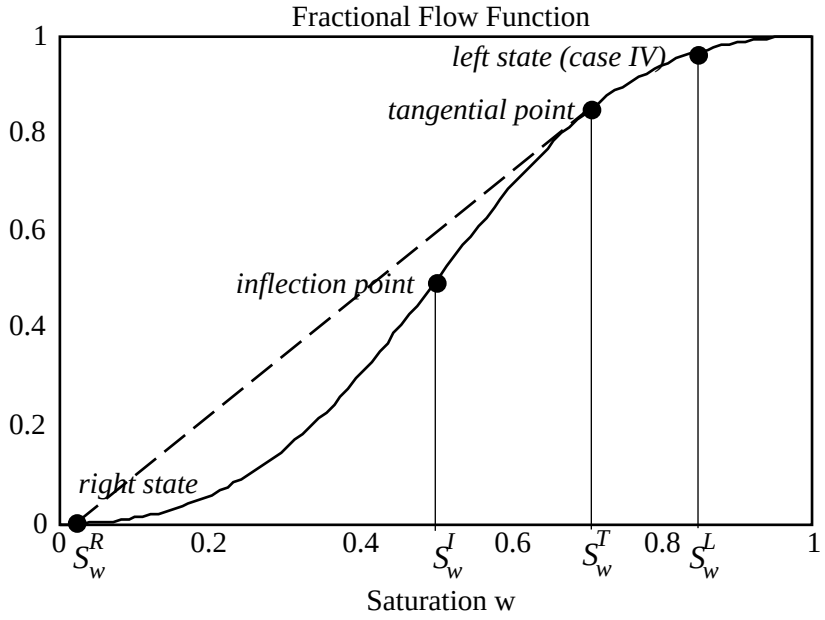


Figure 2.8: Tangential point construction.

that

$$f'_w(S_w^T) = \frac{f_w(S_w^T) - f_w(S_w^R)}{S_w^T - S_w^R}. \tag{2.73}$$

Fig. 2.8 shows the construction of the tangential point graphically.

Case III, $S_w^R < S_w^I < S_w^L \leq S_w^T$: This case is an extension of case I (shock solution) since we still have $f'_w(S_w^L) > f'_w(S_w^R)$ by construction. It is, however, more difficult to show that all characteristics enter the shock.

Case IV, $S_w^R < S_w^I < S_w^T < S_w^L$: The left state is above the tangential point. We obtain a rarefaction wave from the left state to the tangential point and a shock dropping from the tangential point to the right state:

$$S_w(x,t) = \begin{cases} S_w^L & x/t \leq f'(S_w^L) \\ (f')^{-1}(x/t) & f'(S_w^L) < x/t < f'(S_w^T) \\ S_w^R & x/t > f'(S_w^T) \end{cases} . \tag{2.74}$$

Note that the shock speed $s = f'(S_w^T)$ is given by (2.73) and fulfills the Rankine–Hugoniot condition. In fact, (2.73) is constructed in a unique way such that f' is invertible for the rarefaction wave, the shock speed satisfies the Rankine–Hugoniot condition and the characteristics for $x/t > f'(S_w^T)$ enter the shock from below.

Fig. 2.9 shows solution plots for different relative permeabilities and viscosity ratios. The total velocity was $U = 3 \cdot 10^{-7} m/s$, the porosity has been set to $\Phi = 1/5$, the left and right states were $S_w^L = 1$, $S_w^R = 0$. The top plot shows quadratic relative permeabilities ($k_{rw} = S_w^2$, $k_{rn} = (1 - S_w)^2$) with a viscosity ratio of 1, the tangential point is $S_w^T = \sqrt{2}/2$ in this case. The middle and the bottom plot show Brooks–Corey relative permeabilities with a viscosity ratio μ_n/μ_w of 1 (middle) and 100 (bottom). Note that the gradient near $x = 0$ is much larger in this case.

The plots illustrate a problem encountered in the water–flooding technique (secondary recovery) of oil reservoir exploitation: If a more viscous fluid (oil) is displaced by a less viscous fluid (water) the efficiency of the process drops dramatically. In the case of unit viscosity ratio 25% oil remain in the reservoir, whereas 65% oil remain in the reservoir for the high viscosity ratio case.

Moreover, in the multidimensional case the position of the shock front is unstable if the frontal mobility ratio M given by (2.62) is greater than one, see (Bear 1972), (Glimm et al. 1981) and (Glimm et al. 1983) for details. This ultimately leads to the formation of "fingers" of water extending into the oil. For a treatment of the fingering phenomenon from a hydrologists perspective cf. Kueper and Frind (1988). Note that the frontal mobility ratio is (much) smaller than the viscosity ratio since S_w^T (the shock height) decreases with increasing viscosity ratio. For the bottom plot in Fig. 2.9 the frontal mobility ratio is $M = 1.595$.

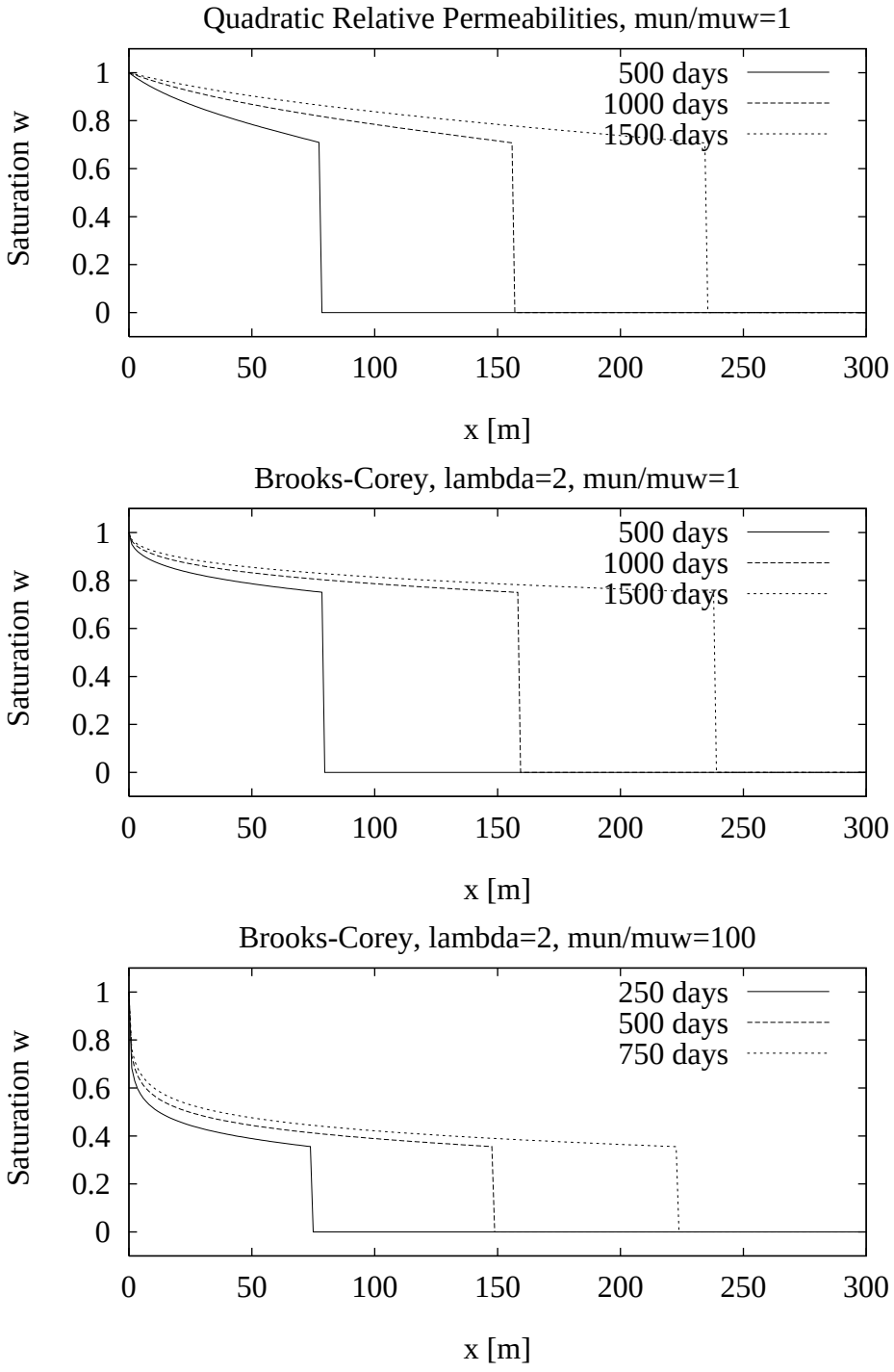


Figure 2.9: Solutions of the Buckley–Leverett problem with quadratic and Brooks–Corey relative permeabilities (see text for details).

2.4.3 PARABOLIC CASE

We now consider the case when capillary forces are present. This case is much more difficult to analyze, even for a one–dimensional model situation. McWhorter and Sunada (1990) gave a quasi–analytical solution for realistic constitutive relations (e. g. Brooks–Corey functions). The discussion here will closely follow this paper.

We restrict ourselves to a "counter–current" flow situation where the total velocity vanishes, this is achieved by setting $U = 0$ in (2.53). From the global pressure formulation (2.33e) we then obtain for the saturation S_w:

$$\Phi \frac{\partial S_w}{\partial t} + \frac{\partial}{\partial x}\left(\lambda_n f_w p_c' K \frac{\partial S_w}{\partial x}\right) = 0 \tag{2.75}$$

(no sources, no gravity). By defining the diffusion coefficient

$$D(S_w(x,t)) = -\frac{\lambda_w \lambda_n}{\lambda_w + \lambda_n} K p_c' \tag{2.76}$$

and the flux function

$$q_w(x,t) = -D(S_w(x,t))\frac{\partial S_w}{\partial x} \tag{2.77}$$

we can rewrite (2.75) as the system

$$\Phi \frac{\partial S_w}{\partial t} = -\frac{\partial}{\partial x} q_w(x,t), \tag{2.78a}$$

$$q_w(x,t) = -D(S_w(x,t))\frac{\partial S_w}{\partial x} \tag{2.78b}$$

The diffusion coefficient D vanishes for extreme values of saturation $S_w = 0$, $S_w = 1$ and is called "doubly degenerate". Eq. (2.78) is solved in the upper right half plane $\Omega = \{(x,t)|x,t > 0\}$ with initial conditions

$$S_w(x,0) = S_\infty, \qquad x > 0 \tag{2.79}$$

and boundary conditions

$$q_w(0,t) = At^{-\frac{1}{2}}, \quad S_w(0,t) = S_0, \quad S_w(\infty,t) = S_\infty, t > 0 \tag{2.80}$$

where $A > 0$ is a constant. A cannot be chosen independently from S_0 and S_∞, a corresponding relation will be derived below but for the time being it is convenient to consider A as an independent parameter.

Using the ansatz

$$S_w(x,t) = S(\lambda(x,t)) = S(xt^{-\frac{1}{2}}) \tag{2.81}$$

with $\lambda(x,t) = xt^{-\frac{1}{2}}$, Eq. (2.78) can be transformed into an ordinary differential equation in the variable λ. Solutions of type (2.81) are called self similar. The transformed equation reads

$$\frac{\lambda\Phi}{2}\frac{d}{d\lambda}S(\lambda) = \frac{d}{d\lambda}q(\lambda), \tag{2.82a}$$

$$q(\lambda) = -D(S(\lambda))\frac{d}{d\lambda}S(\lambda), \tag{2.82b}$$

where S and q are now functions of the independent variable λ. The boundary conditions are transformed into

$$S(0) = S_0, \quad S(\infty) = S_\infty, \quad q(0) = A. \tag{2.83}$$

The boundary condition for $q(0)$ follows from $q(\lambda(x,t)) = q_w(x,t)t^{\frac{1}{2}}$.

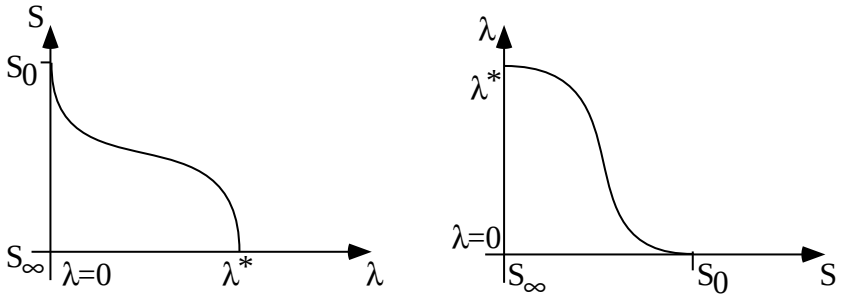


Figure 2.10: $S(\lambda)$ and its inverse function.

$S(\lambda)$ will be a monotonically decreasing function, a typical shape is shown in Fig. 2.10. Moreover, the solution is characterized by a free boundary, i. e. $S(\lambda) = 0$ for all $\lambda \geq \lambda^*$. We can therefore also write λ as a function of the independent variable S. This has the advantage that the domain of definition of the function $\lambda(S)$ is known a priori to be $[S_\infty, S_0]$ and that the position of the free boundary $\lambda^* = \lambda(S_\infty)$ is a result of the computation.

Considering λ as a function of the independent variable S we define a new flux function $\tilde{q}$ depending on saturation:

$$\tilde{q}(S) = q(\lambda(S)). \tag{2.84}$$

By differentiation we obtain

$$\frac{d\tilde{q}}{dS}(\xi) = \frac{dq}{d\lambda}(\lambda(\xi))\frac{d\lambda}{dS}(\xi) \Leftrightarrow \frac{dq}{d\lambda}(\lambda(\xi)) = \frac{d\tilde{q}}{dS}(\xi)\frac{dS}{d\lambda}(\lambda(\xi)). \tag{2.85}$$

Inserting this into (2.82a) we obtain

$$\lambda(\xi) = \frac{2}{\Phi}\frac{d\tilde{q}}{dS}(\xi), \tag{2.86}$$

i. e. given $\tilde{q}(S)$ we obtain $\lambda(S)$ through integration. We now seek an equation for $\tilde{q}$.

Using $\frac{dS}{d\lambda}(\lambda(\xi)) = \left(\frac{d\lambda}{dS}(\xi)\right)^{-1}$ and (2.82b) yields

$$\frac{d\lambda}{dS}(\xi) = -\frac{D(\xi)}{\tilde{q}(\xi)}. \tag{2.87}$$

Differentiating (2.86) and combination with the last equality gives an equation of second order for $\tilde{q}$:

$$\frac{d^2\tilde{q}}{dS^2}(\xi) = -\frac{\Phi}{2}\frac{D(\xi)}{\tilde{q}(\xi)}. \tag{2.88}$$

The boundary conditions for (2.88) are

$$\tilde{q}(S_0) = q(\lambda(S_0)) = q(0) = A, \quad \tilde{q}(S_\infty) = q(\lambda(S_\infty)) = q(\lambda^*) = 0. \tag{2.89}$$

In addition we obtain from (2.86)

$$\frac{d\tilde{q}}{dS}(S_0) = \frac{\Phi}{2}\lambda(S_0) = 0. \tag{2.90}$$

The boundary conditions are not independent of each other. We now relate the constant $A > 0$ in the flux boundary condition to the Dirichlet values S_0 and S_∞. Going back to the original equation

$$\Phi\frac{\partial S_w}{\partial t} + \frac{\partial q_w}{\partial x} = 0 \tag{2.91}$$

in (x,t) coordinates we obtain by integration

$$\Phi\frac{\partial}{\partial t}\int_0^\infty S_w(x,t)dx + q_w(\infty,t) - q_w(0) = 0 \tag{2.92}$$

for any $t > 0$ with the known fluxes $q_w(\infty,t) = 0$, $q_w(0) = At^{-\frac{1}{2}}$. Using $S_w(x,t) = S(xt^{-\frac{1}{2}})$ we obtain by substitution

$$\int_0^\infty S_w(x,t)dx = t^{\frac{1}{2}}\int_0^\infty S(xt^{-\frac{1}{2}})t^{-\frac{1}{2}}dx = t^{\frac{1}{2}}\int_{\lambda(0,t)=0}^{\lambda(\infty,t)=\infty} S(\xi)d\xi. \tag{2.93}$$

Note that the integral is now independent of t. Combining this with (2.92) yields the desired relation

$$A = \frac{\Phi}{2}\int_0^\infty S(\xi)d\xi. \tag{2.94}$$

Since the area under both graphs of Fig. 2.10 is equal we can rewrite this as

$$\begin{aligned} A &= \frac{\Phi}{2}\int_0^{\infty} S(\xi)d\xi = \frac{\Phi}{2}\int_{S_\infty}^{S_0} \lambda(\eta)d\eta \\ &= \frac{\Phi}{2}\int_{S_\infty}^{S_0}\int_{S_0}^{\eta} \frac{d\lambda}{dS}(\xi)d\xi d\eta \\ &= -\frac{\Phi}{2}\int_{S_\infty}^{S_0}\int_{S_0}^{\eta} \frac{D(\xi)}{\tilde{q}(\xi)}d\xi d\eta. \end{aligned} \tag{2.95}$$

Integrating (2.88) twice and using the boundary conditions as well as the expression for A we obtain

$$\tilde{q}(S) = \frac{\Phi}{2}\int_{S_\infty}^{S}\int_{\eta}^{S_0} \frac{D(\xi)}{\tilde{q}(\xi)}d\xi d\eta. \tag{2.96}$$

Finally, using integration by parts, this is transformed into

$$\tilde{q}(S) = \frac{\Phi}{2}\int_{S_\infty}^{S} \frac{(\min(\xi,S)-S_\infty)\,D(\xi)}{\tilde{q}(\xi)}d\xi \tag{2.97}$$

which is an integral equation for $\tilde{q}(S)$. This integral equation is solved numerically by discretizing the interval $[S_\infty, S_0]$ and using a fixed point iteration. From the discrete approximation $\tilde{q}_h$ one can obtain an approximation λ_h by integrating (2.86) numerically. Finally one obtains $S_w(x,t)$ from that by using the similarity transformation. Since the determination of the solution involves the numerical solution of an integral equation it is called "quasi–analytic". However, the fixed point iteration for solving (2.97) converges rapidly and the computation of λ as a function of S gives a precise value for the free boundary and has no problems in representing the large gradients of $S(\lambda)$ near the free boundary.

As an illustration Fig. 2.11 shows the solution $S_w(x,t)$ of Eq. (2.78) for realistic values of the governing parameters: $\Phi = 0.3$, $K = 1\cdot 10^{-10}[m^2]$, $\mu_n/\mu_w = 1$, $S_0 = 1$, $S_\infty = 0$, Brooks–Corey functions with $\lambda = 2$ and $p_d = 5000[Pa]$. Note that the distance of the free boundary from the origin doubles with a four-fold increase of time.

2.5 Three–Phase Flow Formulations

In this subsection we extend the formulations given above to the three phase flow model of Subs. 1.4.6.

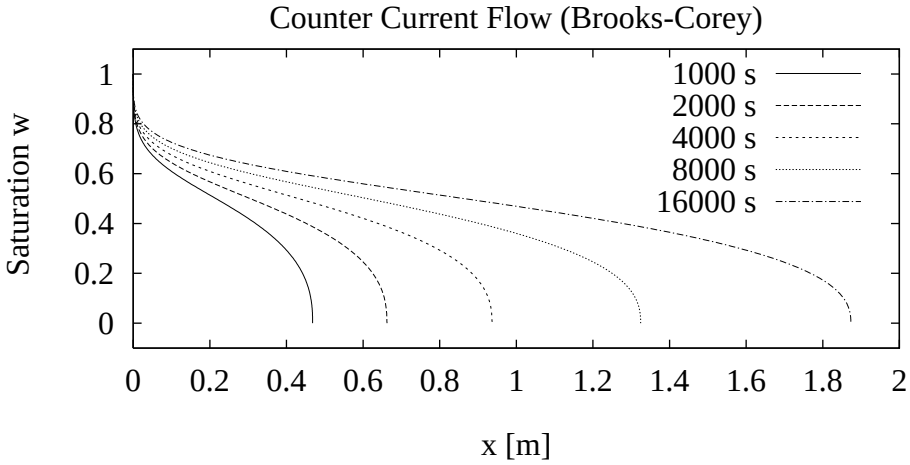


Figure 2.11: Solution of the doubly degenerate parabolic problem.

2.5.1 Phase Pressure–Saturation Formulation

Any of the phase pressures plus two of the saturations can be used as a set of primary variables. For contamination problems the assumption of a continuous water phase is justified making a (p_w, S_n, S_g)–formulation appropriate. It consists of the balance laws

$$\frac{\partial(\Phi\rho_w(1-S_n-S_g))}{\partial t} = -\nabla\cdot\{\rho_w\mathbf{u}_w\} + \rho_w q_w, \tag{2.98a}$$

$$\frac{\partial(\Phi\rho_n S_n)}{\partial t} = -\nabla\cdot\{\rho_n\mathbf{u}_n\} + \rho_n q_n, \tag{2.98b}$$

$$\frac{\partial(\Phi\rho_g S_g)}{\partial t} = -\nabla\cdot\{\rho_g\mathbf{u_g}\} + \rho_g q_g \tag{2.98c}$$

and the phase velocities

$$\mathbf{u}_w = -\frac{k_{rw}(1-S_n-S_g)}{\mu_w}K(\nabla p_w - \rho_w\mathbf{g}), \tag{2.99a}$$

$$\mathbf{u}_n = -\frac{k_{rn}(1-S_n-S_g,S_n)}{\mu_n}K(\nabla p_w + \nabla p_{cnw}(1-S_n-S_g) - \rho_n\mathbf{g}), \tag{2.99b}$$

$$\mathbf{u_g} = -\frac{k_{rg}(S_g)}{\mu_g}K(\nabla p_w + \nabla p_{cnw}(1-S_n-S_g) + \nabla p_{cgn}(1-S_g) - \rho_g\mathbf{g}), \tag{2.99c}$$

where we assumed $k_{rw} = k_{rw}(S_w)$, $k_{rn} = k_{rn}(S_w,S_n)$, $k_{rg} = k_{rg}(S_g)$ and $p_{cnw} = p_{cnw}(S_w)$, $p_{cgn} = p_{cgn}(S_g)$. The blending of capillary pressure curves as in (1.47) can also be used.

The boundary and initial conditions are given by

$$S_n(\mathbf{x},0) = S_{n0}(\mathbf{x}), S_g(\mathbf{x},0) = S_{n0}(\mathbf{x}),\ p_w(\mathbf{x},0) = p_{w0}(\mathbf{x}) \qquad \mathbf{x} \in \Omega, \quad (2.100a)$$

$$S_\alpha(\mathbf{x},t) = S_{\alpha d}(\mathbf{x},t) \text{ on } \Gamma^S_{\alpha d},\ \alpha = n,g\ , \quad p_w(\mathbf{x},t) = p_{wd}(\mathbf{x},t) \text{ on } \Gamma^p_{wd}, \quad (2.100b)$$

$$\rho_\alpha \mathbf{u}_\alpha \cdot \mathbf{n} = \phi_\alpha(\mathbf{x},t) \text{ on } \Gamma_{\alpha n}. \quad (2.100c)$$

As in the two–phase case we see that a formulation based on p_α does not allow $S_\alpha \to 0$. For the p_w–formulation above this means that the *n*-*g* ($S_w = 0$), *n* ($S_w + S_g = 0$) and *g* ($S_w + S_n = 0$) subsystems are excluded.

The type classification of the equations is given in the subsection on the global pressure formulation.

2.5.2 Global Pressure–Saturation Formulation

The global pressure formulation can also be extended to the three–phase case. The presentation closely follows Chavent and Jaffré (1978). As in the case of two phases a total velocity

$$\mathbf{u} = \mathbf{u}_w + \mathbf{u}_n + \mathbf{u}_\mathbf{g} \quad (2.101)$$

can be defined and the same derivation as in Subs. 2.2.1 leads to

$$\frac{\partial \Phi}{\partial t} + \sum_{\alpha=w,n,g} \rho_\alpha^{-1} \left(\Phi S_\alpha \frac{\partial \rho_\alpha}{\partial t} + \nabla \rho_\alpha \cdot \mathbf{u}_\alpha \right) + \nabla \cdot \mathbf{u} = q_w + q_n + q_g. \quad (2.102)$$

From the definition of the phase velocities $\mathbf{u}_\alpha$ we immediately obtain the following expression for the total velocity in terms of p_n and the capillary pressures:

$$\mathbf{u} = -\lambda K \left(\nabla p_n - f_w \nabla p_{cnw} + f_g \nabla p_{cgn} - \sum_\alpha f_\alpha \rho_\alpha \mathbf{g} \right). \quad (2.103)$$

The phase velocities can be expressed without using the phase pressures. With

$$\xi_{wn} = \lambda_w \lambda_n K (\nabla p_{cnw} + (\rho_w - \rho_n)\mathbf{g}), \quad (2.104a)$$

$$\xi_{ng} = \lambda_n \lambda_g K (\nabla p_{cgn} + (\rho_n - \rho_g)\mathbf{g}), \quad (2.104b)$$

$$\xi_{wg} = \lambda_w \lambda_g K (\nabla p_{cnw} + \nabla p_{cgn}(\rho_w - \rho_g)\mathbf{g}) \quad (2.104c)$$

we have

$$\mathbf{u}_w = f_w \mathbf{u} + \lambda^{-1} (\xi_{wn} + \xi_{wg}), \quad (2.105a)$$

$$\mathbf{u}_n = f_n \mathbf{u} + \lambda^{-1} (\xi_{ng} - \xi_{wn}), \quad (2.105b)$$

$$\mathbf{u}_\mathbf{g} = f_g \mathbf{u} + \lambda^{-1} (\xi_{wg} - \xi_{ng}). \quad (2.105c)$$

Note the structural similarity with the two–phase case. Again the singularities of capillary pressure derivatives are compensated by appropriate mobilities.

We now seek a global pressure $p(p_n(\mathbf{x},t), S_w(\mathbf{x},t), S_g(\mathbf{x},t))$ such that

$$\nabla p = \nabla p_n - f_w \nabla p_{cnw} + f_g \nabla p_{cgn} \tag{2.106}$$

where we will assume the general case that p_{cnw} and p_{cgn} may both depend on $S_w(\mathbf{x},t)$ and $S_g(\mathbf{x},t)$. Moreover, we assume that f_α depend only on the saturations S_w, S_g and not on position. Obviously it is the difficult part to find a function $\pi(S_w(\mathbf{x},t), S_g(\mathbf{x},t))$ such that

$$\nabla \pi = -f_w \nabla p_{cnw} + f_g \nabla p_{cgn}. \tag{2.107}$$

The requirement of interchangeability of partial derivatives

$$\frac{\partial}{\partial x_i}\left\{\frac{\partial \pi}{\partial x_j}\right\} = \frac{\partial}{\partial x_j}\left\{\frac{\partial \pi}{\partial x_i}\right\}, \qquad i \neq j, \tag{2.108}$$

leads to

$$\frac{\partial}{\partial S_g}\left\{\frac{\partial \pi}{\partial S_w}\right\} = \frac{\partial}{\partial S_w}\left\{\frac{\partial \pi}{\partial S_g}\right\}, \tag{2.109}$$

which in turn leads to

$$\frac{\partial f_g}{\partial S_g}\frac{\partial p_{cgn}}{\partial S_w} - \frac{\partial f_w}{\partial S_g}\frac{\partial p_{cnw}}{\partial S_w} = \frac{\partial f_g}{\partial S_w}\frac{\partial p_{cgn}}{\partial S_g} - \frac{\partial f_w}{\partial S_w}\frac{\partial p_{cnw}}{\partial S_g}. \tag{2.110}$$

Eq. (2.110) is called the *total differential condition* in (Chavent and Jaffré 1978). It states that the relative permeabilities and the capillary pressure functions cannot be chosen independently of each other in the three–phase global pressure formulation. This may be a severe restriction of this formulation. Chavent and Jaffré (1978) give a numerical procedure for constructing relative permeabilities and capillary pressure functions from given ones that fulfill (2.110).

The function π is then defined by

$$\begin{aligned}\pi(s_w, S_g) = &\int_1^{S_w}\left\{-f_w(\xi,0)\frac{\partial p_{cnw}}{\partial S_w}(\xi,0) + f_g(\xi,0)\frac{\partial p_{cgn}}{\partial S_w}(\xi,0)\right\}d\xi \\ &+\int_0^{S_g}\left\{-f_w(S_w,\xi)\frac{\partial p_{cnw}}{\partial S_{wg}}(S_w,\xi) + f_g(S_w,\xi)\frac{\partial p_{cgn}}{\partial S_g}(S_w,\xi)\right\}d\xi\end{aligned} \tag{2.111}$$

A tedious calculation shows that $p = p_n + \pi$ fulfills (2.106) when (2.110) is satisfied. Thus we can write the total velocity as

$$\mathbf{u} = -\lambda K\left(\nabla p - \sum_\alpha f_\alpha \rho_\alpha \mathbf{g}\right). \tag{2.112}$$

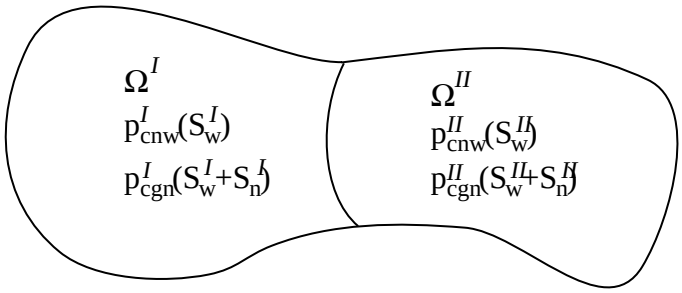


Figure 2.12: Three–phase capillary pressure functions at a porous medium discontinuity.

Insertion of (2.112) into (2.102) gives an elliptic equation for p if the fluids are incompressible or a parabolic equation if at least one of the fluids is compressible. The remaining equations for the two saturations are given by

$$\frac{\partial(\Phi\rho_w S_w)}{\partial t} = -\nabla\cdot\{\rho_w \mathbf{u}_w\} + \rho_w q_w, \tag{2.113a}$$

$$\frac{\partial(\Phi\rho_n S_g)}{\partial t} = -\nabla\cdot\{\rho_g \mathbf{u}_\mathbf{g}\} + \rho_g q_g, \tag{2.113b}$$

with $\mathbf{u}_w, \mathbf{u}_\mathbf{g}$ given by (2.105a,2.105c). If capillary pressure effects are neglected (2.113a,2.113b) is a first order system of conservation laws. It turns out that the hyperbolicity of that system depends on the shape of the relative permeabilities. Chavent and Jaffré (1978) found that the system is not hyperbolic for all values of S_w, S_g when the often used relative permeabilities of (Stone 1973) are used. The modified relative permeability functions of Chavent and Jaffré (1978) satisfying (2.110) results in a hyperbolic system for the two saturation variables when capillary pressure is neglected.

Boundary and initial conditions for the three–phase global pressure formulation are a straightforward extension from the two–phase case.

2.5.3 Media Discontinuity

As in the two–phase case we consider a porous medium that is composed of two subdomains Ω^I and Ω^{II} where different sets of capillary pressure–saturation relationships are valid. This situation and the notation is shown in Fig. 2.12.

We assume for simplicity the capillary pressure functions of Parker et al. (1987) given by (1.35) where $p_{cnw} = p_{cnw}(S_w)$, $p_{cgn} = p_{cgn}(S_w + S_n)$ and $p_{cnw}, p_{cgn} \in [0,\infty)$. Moreover, we restrict our interest to the phase pressure formulation of Subs. 2.5.1. In that case we get from the continuity of p_{cnw} the relation

$$p^I_{cnw}(S^I_w) = p^{II}_{cnw}(S^{II}_w) \tag{2.114}$$

from which S^{II}_w can be obtained for a given S^I_w. The continuity of p_{cgn} leads to

$$p^I_{cgn}(S^I_w + S^I_n) = p^{II}_{cgn}(S^{II}_w + S^{II}_n) \tag{2.115}$$

where S_w^{II} is already fixed and S_w^I, S_n^I are assumed known. Since p_{cgn}^{II} is strictly decreasing and $S_n^{II} \geq 0$ the following condition is required to hold for the capillary pressure functions:

$$p_{cgn}^{II}\left(\left(p_{cnw}^{II}\right)^{-1}\left(p_{cnw}^{I}(S_w^I)\right)\right) \geq p_{cgn}^{I}(S_w^I + S_n^I), \quad \forall S_w^I, S_n^I. \tag{2.116}$$

If this relation holds then (2.115) yields the saturation S_n^{II}. The remaining conditions at the interface are the continuity of normal fluxes $\rho_\alpha \mathbf{u}_\alpha \cdot \mathbf{n}$ and the continuity of p_w if a mobile wetting phase is always assumed on both sides of the interface.

3

Fully Implicit Finite Volume Discretization

3.1 Introduction

3.1.1 Numerical Difficulties in Simulation

The numerical solution of the coupled nonlinear two–phase flow system (1.41) in its various formulations is a formidable task. For the problems arising in the simulation of hydrocarbon recovery processes, Ewing (1983) gives an excellent overview. Four major numerical difficulties are identified in that paper which are listed in the following.

Transport-dominated parabolic problems. In reservoir simulation transport is often the dominating physical process. The saturation equation (2.33e) is almost hyperbolic. Centered differences or standard Galerkin finite element methods applied to the first order terms are second order accurate but yield oscillatory numerical approximations if the solution is not smooth enough. Upwind stabilizations lead to monotone numerical solutions but the approximation is only first order accurate, sharp fronts tend to be smeared out and the numerical solution is sensitive to grid orientation. This difficulty has lead to the development of various types of "characteristic methods".

Time–stepping procedures. The multiphase flow equations are coupled systems of nonlinear, time–dependent partial differential equations. Various degrees of implicitness in the discretization and coupling in the nonlinear solver are possible with the different methods. Stability and robustness on the one hand must be balanced with accuracy and computational efficiency on the other hand. With fully implicit, fully coupled methods large systems of nonlinear algebraic equations have to be solved.

Accurate fluid velocities. The coupling of the pressure equation (2.33a) and (2.33b) to the saturation equation (2.33c,2.33e) is only through the total velocity $\mathbf{u}$. While the pressure p may vary considerably, the total velocity is a relatively smooth function. Computing p via a standard conforming finite element method and evaluating the velocity (2.33b) through numerical differentiation results in a velocity $\mathbf{u}$ that is less accurate. The effect is pronounced with the presence of abrupt changes in permeability or viscosity. The mixed finite element method directly approximates $\mathbf{u}$ and is able to produce a better approximation (especially of the flux $\mathbf{u} \cdot \mathbf{n}$), cf. Durlofsky (1994) for a numerical comparison.

Viscous fingering. This is more a modeling issue. The phenomenon of viscous fingering comes from instabilities on the microscopic level which are not modeled by the macroscopic equations. However, the macroscopic equations are unstable when the frontal mobility ratio is greater than one. Numerical solutions exhibit finger–like phenomena (even with homogeneous parameters) but these are triggered by numerical errors, depend on the mesh and do not model the underlying physics. The macroscopic effects of viscous fingering could be included in the model by a varying (anisotropic) permeability field and longitudinal dispersion effects (at least in miscible displacement). More generally the problem of representing effects from a smaller scale in a model on a larger scale is termed "upscaling".

For the simulation of infiltration and remediation problems two additional difficulties can be mentioned.

Degenerate parabolic problems. Infiltration and remediation problems are often simulated on a smaller scale, e. g. the VEGAS experimental facility described in Kobus (1996) has a size of 15 by 10 by 7 $[m^3]$, and counter–current flow situations exist where the total velocity is small. In these cases capillary pressure is important, which adds a degenerate diffusion term to the saturation equation. Numerical methods must be able to accurately follow the resulting free boundary.

Media discontinuities, entry pressure effects. It has been shown in Sect. 2.3 that, under certain conditions, the saturation is discontinuous at a medium discontinuity and that infiltration of a low permeable lens is only possible if a certain critical saturation is reached. A numerical simulation must accurately represent this condition. There are many numerical methods that would allow an immediate penetration of such a lens as has been demonstrated in Helmig (1997).

3.1.2 Overview of Numerical Schemes

The first numerical simulator for incompressible two–phase flow in porous media has been described by Douglas Jr., Peaceman, and Rachford Jr. (1959) about 40 years ago. Since then many different methods have been devised. In the incompressible case the pressure equation is elliptic and a fully explicit treatment is not possible, cf. (Peaceman 1977). Therefore any numerical method for the multiphase flow problem has to solve systems of algebraic equations but various degrees of implicitness and coupling are possible.

Many of the newer numerical schemes for the two–phase flow problem focus on the accurate and efficient treatment of the advection dominated saturation equation. Typically these methods use the global pressure formulation which naturally leads to a sequential solution process: From a given saturation at time level t^n the pressure p at time t^n is computed from (2.33a,b) (incompressible case) which is a linear equation in p with coefficients depending on saturation.

Then saturation at time level t^{n+1} is computed with a frozen velocity field. If no iteration of this procedure within a time step is performed a time step restriction must be obeyed. If the saturation equation is treated with an explicit method the whole time–stepping procedure is named IMPES for Implicit pressure explicit saturation.

If the saturation equation is assumed to be advection–dominated then standard methods of finite difference, element or volume type do not perform well. They either show nonphysical oscillations or numerical diffusion and grid orientation sensitivity. Due to the nonlinearity of the fractional flow function (self sharpening effect: velocity behind the shock is greater than in front of the shock) it is not so much the spatial truncation error but the temporal truncation error of the backward Euler scheme that causes the smearing of fronts. The use of higher order time discretizations such as Crank–Nicolson or BDF(2) results in severe time step restrictions due to lack of stability.

More successful methods therefore attempt to treat spatial and temporal derivatives in combination. This can be done e. g. by canceling temporal truncation errors with spatial truncation errors as in the *Taylor–Galerkin method* of Donea (1984) or by considering the characteristics of the hyperbolic part. The methods of the last class, so–called *characteristic methods*, have gained a lot of interest in the last 15 years and two methods will be treated in more detail now.

The *modified method of characteristics (MMOC)* has been introduced by Douglas Jr. and Russel (1982) for a scalar, linear advection–diffusion equation in one space dimension. The main idea is to interpret temporal derivative and advective part $c(x)\partial u/\partial t + b(x)\partial u/\partial x$ together as a directional derivative in the characteristic direction $\tau(x) = (b(x),\ c(x))^T$, which is then discretized by a backward difference quotient. The value at the "foot" of the characteristic is interpolated from solution at the preceding time level. The diffusive part is treated implicitly with standard methods. The resulting error estimate contains a term $\|\partial^2 u/\partial\tau^2\|$ instead of $\|\partial^2 u/\partial t^2\|$ which is much smaller in the advection–dominated case. The method has been extended to miscible displacement (linear advection term!) in two space dimensions by Russel (1985). It is shown that very large time steps (Courant number significantly greater than one) can be taken with very good accuracy. The drawbacks are the difficulty of handling boundary conditions that are not of Dirichlet type and its inability to conserve mass. The latter problem has been overcome (for a special case) in Douglas Jr. et al. (1997) where the method is extended to the incompressible two–phase flow problem. It is shown in that paper that the mass balance error of the standard MMOC is considerable even on very fine meshes. In the nonlinear case the non–conservativeness results in a wrong approximation of the front position. It should also be noted that MMOC for nonlinear hyperbolic problems cannot use the long time steps possible in the linear case. A proposition to overcome that problem is made in Espedal and Ewing (1987).

The *Eulerian–Lagrangian localized adjoint method (ELLAM)* has been introduced in Celia et al. (1990) and provides a framework where many characteristic

methods can be derived from. The main improvement is that the resulting methods are locally mass conservative and that all types of boundary conditions can be treated. Another advantage is that advective and diffusive part are treated in combination and not separately via an operator splitting approach. The idea of the method is to use a weighted residual formulation of the equation in space–time and to choose the weight functions such that they have local support and solve the homogeneous adjoint equations in the interior of each space–time element exactly. Treatment of all types of boundary conditions is possible but is relatively complicated already in one space dimension. Multi–dimensional formulations are mentioned in Celia (1994) and Binning and Celia (1994). The method has been applied to transport of nuclear waste contamination in Ewing et al. (1994). Its application to multiphase flow is outlined in Ewing (1991) and mentioned in Binning and Celia (1994) but no numerical results were presented.

A certain disadvantage of both the MMOC and the ELLAM method is that they are primarily designed for linear hyperbolic problems. There are methods, however, that directly use the knowledge about the nonlinear hyperbolic conservation law (cf. Subs 2.4.2).

In the *front tracking method*, see Glimm et al. (1983) and Risebro and Tveito (1991), the solution is, in one space dimension, represented by a piecewise constant function and the coefficient functions are replaced by piecewise linears. The Riemann problems at each discontinuity can be solved analytically (cf. Subs 2.4.2) and shock collisions have to be resolved. The multi–dimensional extension follows a tensor product approach. Capillary diffusion, if present, is included via operator splitting. An improved operator splitting that allows large time steps has recently been introduced by Hvistendahl Karlsen et al. (1997). Another *nonlinear characteristic method* for two–phase flow has been developed by Mulder and Meyling (1993) where it is combined with local mesh refinement. A disadvantage of these methods is that they generally do not conserve mass.

The aim of the methods discussed so far is to allow large time steps (Courant number significantly greater than one) while maintaining good accuracy. In order to facilitate that a fair amount of work is necessary per time step, at least in the nonlinear case (recomputation of weight functions in ELLAM, Riemann solver and shock collision resolution in front tracking and nonlinear characteristic methods). Another well known approach to solve nonlinear conservation laws is to use higher order explicit finite volume schemes, cf. (LeVeque 1992), which have a Courant number limitation but can be quickly evaluated. For two–phase flow without capillary pressure such a method is presented in Helmig and Huber (1996), capillary pressure has been included explicitly in Durlofsky (1993). Dawson (1991) analyses a higher order Godunov method combined with a mixed finite element method for the diffusive part for a one–dimensional scalar model problem.

All methods that focus on the solution of the advection–dominated saturation equation rely on a decoupling of pressure and saturation equation. For “difficult

nonlinearities" this may result in a severe time step restriction, see Ewing (1983) or Gundersen and Langtangen (1997), although a detailed comparison does not seem to be available. The amount of coupling between pressure and saturation equation heavily depends on the formulation that is used, it is certainly much weaker in the global pressure formulation. On the other hand it is generally agreed upon that spatial variability of the permeability, porosity and constitutive relations (especially capillary pressure) makes a problem "more difficult".

With respect to robustness, i. e. the ability to solve a wide range of problems, a fully implicit and fully coupled treatment of the governing equations is most reliable. In that case an implicit time discretization is applied (e. g. backward Euler) where all spatial derivatives are evaluated at the new time level. The resulting system of nonlinear algebraic equations is then solved with a (quasi–) Newton method.

The fully implicit/fully coupled approach has been combined with virtually all discretization methods. Many different variants exist to incorporate the necessary upstream weighting into the standard methods. In the conforming finite element method higher order test functions, e. g. quadratic or cubic polynomials, are used, see Heinrich et al. (1977). Applications to two–phase as well as multi–component non–isothermal flow can be found in Helmig (1997) and Emmert (1997). Further possibilities are the streamline diffusion method of Brooks and Hughes (1982) and the control–volume finite element approach of Forsyth (1991). The latter method conserves mass locally and is applied to three–phase/three component flow in Forsyth and Shao (1991).

Finite volume methods (sometimes called "integrated finite differences" in the groundwater literature) are also very popular due to their mass conservation and monotonicity properties. The method is used in the cell centered form on structured meshes already in Peaceman (1977) and on unstructured meshes in the widely used TOUGH2 simulator of Pruess (1991). A good overview and a theoretical treatment of various methods for a model problem has been given recently in Michev (1996).

A rather new development is the use of the mixed finite element method in combination with fully implicit/fully coupled techniques in Dawson et al. (1997).

For a detailed comparison of several fully implicit methods and other techniques we refer to Helmig (1997).

3.1.3 Approach taken in this Work

Domain of Application. In this work we are interested in the simulation of infiltration and remediation problems on scales that are small in comparison to those in oil reservoir simulation. Also, countercurrent flow and flow over low permeability lenses is important in this type of application. Capillary diffusion is important in these cases. Especially the treatment of entry pressure effects

at media discontinuities is necessary to accurately simulate the phenomena of lateral spreading and entrapment of DNAPL.

Furthermore, compressible fluids, e. g. water–gas systems are of interest in the simulation of enhanced remediation processes such as soil vapor extraction or in the security assessment of underground waste repositories. Although these particular applications require more sophisticated models (compositional, non–isothermal, fractured, ...) a method that can simulate water–gas systems is certainly a necessary requirement.

Finally, practical application of subsurface models requires the ability to handle very complex geometries. One should not at all underestimate this point but rather include it in the decision process for the numerical method.

Numerical Requirements. The properties of *stability and consistency* are, of course, of fundamental importance for any numerical simulation. Additionally, a simulation software that is used in an engineering environment must be *robust* in the sense that it is stable, accurate and computationally efficient for a wide range of problems. This certainly requires a compromise and there may be better methods for special cases.

Furthermore, we require that the method *conserves mass locally* in order to get correct shock positions and to be able to follow small concentrations.

The *monotonicity property* (nonoscillating solutions) is also of primary importance since the governing nonlinearities are only defined for saturation values between zero and one. This property becomes even more important in compositional flows.

Complex geometries can be handled in different ways. Ultimately, we believe that only *unstructured meshes* are able to handle the needs in this direction since they can be generated fully automatically from CAD input.

Outline of Solution Procedure. The numerical requirements lead to the selection of a rather "traditional" scheme. In space a *vertex centered finite volume method* with upstream weighting of mobilities is used. For the time discretization either *backward Euler, Crank–Nicolson or BDF(2)* are used. The time–stepping strategy is *fully implicit/fully coupled* for a maximum of robustness. This method conserves mass locally (BDF(2) with restrictions), it can be used on fully unstructured, multi–element type meshes and produces monotone solutions even on highly distorted meshes.

The approach outlined so far is applied either to a *phase pressure–saturation formulation* or to the *global pressure formulation*. Media discontinuities are handled either by the fully upwinding procedure as in Helmig and Huber (1998) or by an incorporation of the *interface conditions* into the discretization as in de Neef and Molenaar (1997).

The fully implicit discretization produces a large system of nonlinear algebraic equations to be solved per time step. The fully coupled solution procedure uses an *inexact Newton* method for its solution. The inexactness of the Newton

method refers to an inexact solution of the linear systems within the Newton method. Global convergence is achieved by an appropriate *line search* procedure. The quadratic convergence of the Newton method enables one to solve the nonlinear systems very accurately which is necessary to ensure local conservation of mass.

A main objective of this work is to show that the linear systems arising within the Newton method can be solved efficiently with a *multigrid* method. A further reduction in computation time is achieved by a *data parallel implementation* of the simulator following the ideas of Bastian (1996).

The implementation of the simulator is based on the PDE software toolbox UG described in Bastian et al. (1997). Specifically, the parallelization is mostly hidden in the general purpose UG library and is not specific to the two–phase flow simulator.

3.2 Stationary Advection–Diffusion Equation

In this section we describe the vertex centered finite volume method for a stationary linear advection–diffusion equation on general unstructured meshes and introduce the necessary notation along the way.
The equation for concentration C is given by

$$\nabla \cdot \mathbf{j} = q \qquad \text{in } \Omega, \tag{3.1a}$$

$$\mathbf{j}(C) = \mathbf{r}(\mathbf{x})C - D(x)\nabla C, \tag{3.1b}$$

$$C = C_d(\mathbf{x}) \qquad \text{on } \Gamma_d, \tag{3.1c}$$

$$\mathbf{j} \cdot \mathbf{n} = J(\mathbf{x}) \qquad \text{on } \Gamma_n, \tag{3.1d}$$

with Ω a polyhedral domain in $\mathbb{R}^d$, $d = 2,3$. Both, Dirichlet and Neumann (flux) type boundary conditions will be treated. The flow field $\mathbf{r}(\mathbf{x})$ and the symmetric positive definite tensor $D(\mathbf{x})$ are assumed to be given and depend only on position.

Eq. (3.1) is discretized on an unstructured mesh $E_h = \{e_1, \ldots, e_K\}$ consisting of elements e_i. The index h indicates the mesh width, e. g. the diameter of the largest element. Triangular and quadrilateral elements are used in 2D while tetrahedra, pyramids, prisms and hexahedra are used in 3D. It is assumed that quadrilateral faces in 3D are planar. Different types of elements can be mixed provided the mesh is admissible, i. e. E_h covers Ω and the intersection of two different elements is either empty, a common vertex, edge or face of the two elements. The set of vertices is denoted by $V = \{v_1, \ldots, v_N\}$, the location of vertex v_i is $\mathbf{x}_i$ and the barycenter of element e_k is denoted by $\mathbf{x}^k$. Furthermore, $V(k)$ denotes the set of all indices i where v_i is a corner of element e_k and conversely $E(i)$ is the set of all indices k such that $i \in V(k)$.

The finite volume method needs an additional mesh that is called secondary or dual mesh. In the vertex centered variant to be described here this mesh is

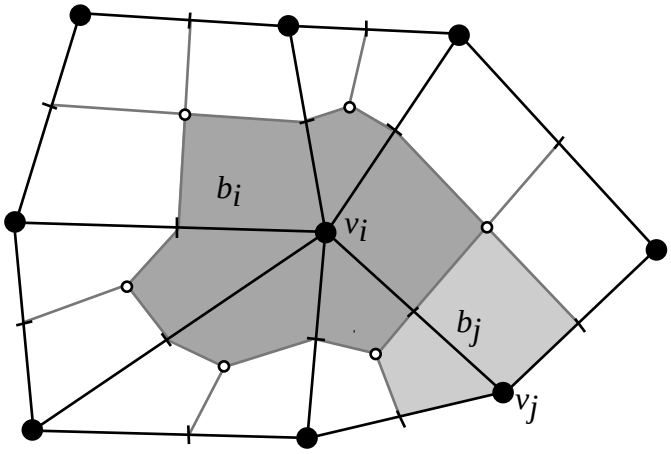


Figure 3.1: Construction of secondary mesh in 2D.

constructed from the primary mesh E_h by the following procedure: Element barycenters are connected to edge midpoints in 2D or to face barycenters in 3D. Face barycenters in 3D are then connected to edge midpoints. Examples of this construction are shown in Fig. 3.1 for the 2D case.

The secondary mesh $B_h = \{b_1, \ldots, b_N\}$ consists of polyhedral regions b_i called *boxes* or *control volumes*. Each control volume b_i is naturally associated with vertex v_i in the primary mesh. Interior vertices are approximately in the center of their associated control volume while boundary vertices are at the boundary of their control volume (see e. g. vertex v_j in Fig. 3.1). Note that the construction of the secondary mesh is not subject to an angle condition and can be carried out in the same way for all element types listed above. For other variants of the finite volume method we refer to (Michev 1996).

It is convenient to define the following index sets

$$I = \{1, \ldots, N\}, \qquad I_d = \{i \in I \mid \mathbf{x}_i \notin \Gamma_d\}. \tag{3.2}$$

Based on the primary and secondary mesh we can define two finite dimensional function spaces. $V_h \subseteq H^1(\Omega)$ is the standard conforming finite element space defined as

$$V_h = \left\{v \in C^0(\bar{\Omega}) \mid v \text{ (multi–) linear on } t \in E_h\right\} \tag{3.3}$$

and W_h is a non–conforming space defined as

$$W_h = \left\{w \in L^2(\Omega) \mid w \text{ constant on each } b \in B_h\right\} \tag{3.4}$$

Finite element functions, e. g. $C_h \in V_h$ are typically denoted with a subscript h. In order to incorporate the Dirichlet boundary conditions we will frequently make use of the following subspaces of V_h and W_h:

$$V_{hd} = \{v \in V_h \mid v(\mathbf{x}_i) = C_d(\mathbf{x}_i), i \in I \setminus I_d\} \tag{3.5}$$

and

$$W_{hd} = \{w \in W_h \mid w(\mathbf{x}_i) = 0, i \in I \setminus I_d\}. \tag{3.6}$$

Note that Dirichlet boundary conditions are directly incorporated into V_{hd}.

V_h and W_h are generated by the usual nodal basis functions given by

$$\forall i, j \in I, \varphi_i \in V_h: \qquad \varphi_i(\mathbf{x}_j) = \delta_{ij} \tag{3.7}$$

and

$$\forall i, j \in I, \psi_i \in W_h: \qquad \psi_i(\mathbf{x}_j) = \delta_{ij}. \tag{3.8}$$

Every finite element function $C_h \in V_h$ is identified with a vector $\mathbf{C} \in \mathbb{R}^N$ by the mapping $P: \mathbb{R}^N \to V_h$ in the usual way:

$$P(\mathbf{C}) = C_h, \qquad C_h(\mathbf{x}) = \sum_{i \in I} \mathbf{C}_i \varphi_i(\mathbf{x}). \tag{3.9}$$

We are now in a position to state the discrete vertex centered finite volume problem:

Find $C_h \in V_{hd}$ such that

$$A_h(C_h, w_h) = Q_h(w_h) \qquad \forall w_h \in W_{hd}, \tag{3.10}$$

where the forms A_h and Q_h are given by

$$A_h(C_h, w_h) = \sum_{i \in I} w_h(\mathbf{x}_i) \int_{\partial b_i \cap \Omega} \mathbf{j}(C_h) \cdot \mathbf{n} \, ds, \tag{3.11a}$$

$$Q_h(w_h) = \sum_{i \in I} w_h(\mathbf{x}_i) \left[\int_{b_i} q(\mathbf{x}) \, d\mathbf{x} - \int_{\partial b_i \cap \Gamma_n} J(\mathbf{x}) \, ds \right], \tag{3.11b}$$

with $\mathbf{n}$ the outer unit normal to b_i. This weak form follows from

$$\begin{aligned} \int_\Omega w_h \nabla \cdot \mathbf{j}(C_h) \, d\mathbf{x} = & \\ &= \sum_{i \in I} w_h(\mathbf{x}_i) \int_\Omega \psi_i \nabla \cdot \mathbf{j}(C_h) \, d\mathbf{x} \\ &= \sum_{i \in I} w_h(\mathbf{x}_i) \int_{b_i} \nabla \cdot \mathbf{j}(C_h) \, d\mathbf{x} \\ &= \sum_{i \in I} w_h(\mathbf{x}_i) \left[\int_{\partial b_i \cap \Omega} \mathbf{j}(C_h) \cdot \mathbf{n} \, ds + \int_{\partial b_i \cap \Gamma_n} J(\mathbf{x}) \, ds \right] \end{aligned} \tag{3.12}$$

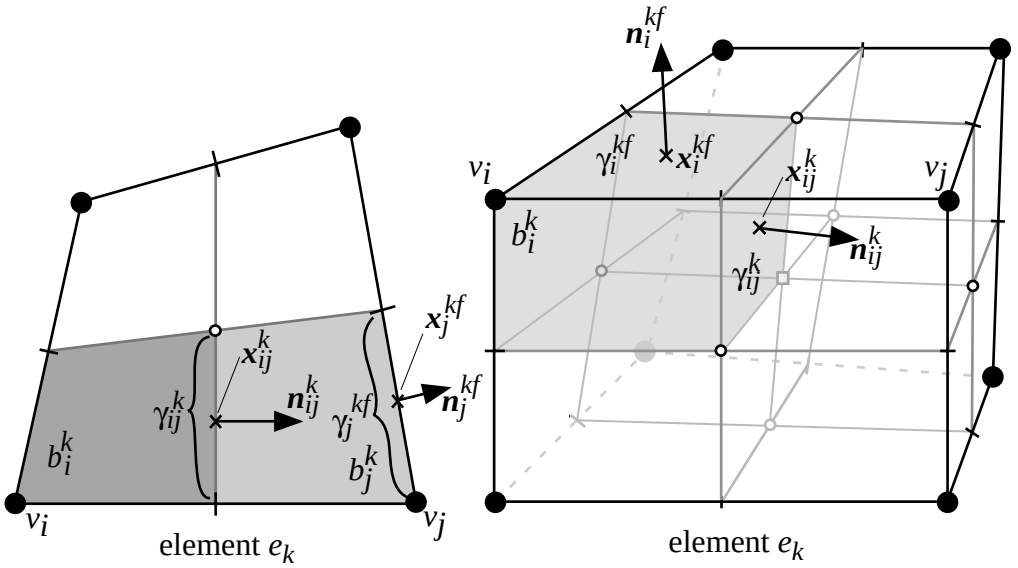


Figure 3.2: Intersection of a control volume with an element.

and

$$\begin{aligned}\int_\Omega w_h q(\mathbf{x})\,d\mathbf{x} &= \\ &= \sum_{i\in I} w_h(\mathbf{x}_i)\int_\Omega \psi_i q(\mathbf{x})\,d\mathbf{x} \\ &= \sum_{i\in I} w_h(\mathbf{x}_i)\int_{b_i} q(\mathbf{x})\,d\mathbf{x}\end{aligned} \tag{3.13}$$

Using the basis function representation the weak formulation is equivalent to the algebraic problem:

Find $\mathbf{C}\in\mathbb{R}^N$, $P(\mathbf{C})\in V_{hd}$ such that

$$\mathbf{A}(\mathbf{C}) = \mathbf{Q} \tag{3.14}$$

with

$$\mathbf{A}_i = A_h(P(\mathbf{C}),\psi_i),\ \mathbf{Q}_i = Q_h(\psi_i), \qquad i\in I_d. \tag{3.15}$$

Clearly the vector valued mapping $\mathbf{A}$ is linear here and (3.14) is a system of linear equations but since all problems to be discussed below will be nonlinear we will consider $\mathbf{A}$ as vector valued mapping from $\mathbb{R}^N$ to $\mathbb{R}^{|I_d|}$. Note that the degrees of freedom related to Dirichlet vertices are fixed in $\mathbf{C}$ through the requirement $P(\mathbf{C})\in V_{hd}$.

It remains to describe the evaluation of A_h and Q_h for the special test functions ψ_i. For this we need some further notation related to the control volumes. The complicated structure of the secondary mesh becomes feasible by considering

the intersection of a single control volume b_i with an element e_k of the primary mesh as illustrated in Fig. 3.2. The intersection of b_i with e_k is called *sub–control volume* and is denoted by b_i^k. The part of the control volume boundary ∂b_i lying within element e_k consists of straight line segments in 2D and quadrilateral (planar) faces in 3D which are called *sub–control volume faces*. Each sub–control volume face can be associated with an edge of the primary mesh and is therefore denoted by γ_{ij}^k (the sub–control volume face in e_k associated with the edge (v_i, v_j)). The unit normal vector to γ_{ij}^k pointing out of b_i is denoted by $\mathbf{n}_{ij}^k$. The normal vector is constant since the sub–control volumes faces are planar (since the faces of the primary mesh are assumed to be planar). The barycenter of sub–control volume face γ_{ij}^k is denoted by $\mathbf{x}_{ij}^k$.

If part of ∂b_i^k coincides with the boundary of the domain Ω these *boundary sub–control volume faces* are denoted by γ_i^{kf} with outer normal $\mathbf{n}_i^{kf}$ and barycenter $\mathbf{x}_i^{kf}$. The superscript f denotes the face (edge in 2D) of element e_k that is part of the boundary. Note that there may be more than one boundary sub–control volume face per sub–control volume (e. g. vertex v_i in Fig. 3.2 right would have three boundary sub–control volume faces if the domain Ω is the single hexahedron).

With this notation we have

$$A_h(P(\mathbf{C}), \psi_i) = \int\limits_{b_i \cap \Omega} \mathbf{j}(C_h) \cdot \mathbf{n} \, ds = \sum_{k,j} \int\limits_{\gamma_{ij}^k} \mathbf{j}(C_h) \cdot \mathbf{n}_{ij}^k \, ds \approx \sum_{k,j} J_{ij}^k. \tag{3.16}$$

The *numerical flux* J_{ij}^k over sub–control volume face γ_{ij}^k is computed by

$$J_{ij}^k = \left\{ [C_h]_{ij}^k \, \mathbf{r}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k - \sum_{m \in I} \mathbf{C}_m \nabla \varphi_m(\mathbf{x}_{ij}^k) D(\mathbf{x}_{ij}^k) \mathbf{n}_{ij}^k \right\} \operatorname{meas}(\gamma_{ij}^k). \tag{3.17}$$

where the midpoint rule has been used to evaluate the surface integral. The evaluation of C_h at the integration point $\mathbf{x}_{ij}^k$ in the advective part is done as follows

$$[C_h]_{ij}^k = (1-\beta) C_h(\mathbf{x}_{ij}) + \beta \cdot \begin{cases} \mathbf{C}_i & \mathbf{r}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \geq 0 \\ \mathbf{C}_j & \text{else} \end{cases}. \tag{3.18}$$

For $\beta = 1$ we obtain the *fully upwinding* method whereas $\beta = 0$ corresponds to central differencing. The factor β is fixed in our application but could, in general, be chosen depending on the local Peclet number (resulting in modified upwind schemes, see Michev (1996)) or the smoothness of the solution (resulting in limiter methods).

For the evaluation of the diffusion tensor $D(\mathbf{x}_{ij}^k)$ several choices exist. In order to get optimal error estimates in the L^2–norm one has to set

$$D(\mathbf{x}_{ij}^k) = D^k, \tag{3.19}$$

where D^k is constant on each element and the entries are volume averages over the element e_k, cf. (Bey 1997). On the other hand one–dimensional homogenization of $\partial j/\partial x = 0$, $j = -D(x)\partial u/\partial x$ in $(0,L)$ leads to

$$j = \left(\frac{1}{L} \int_0^L \frac{1}{D(x)} dx \right)^{-1} \frac{u(0) - u(L)}{L}, \tag{3.20}$$

i. e. the average diffusion coefficient is computed as a harmonic mean value. This suggests to associate a permeability value with every control volume of the secondary mesh and to set (in the scalar case):

$$D(\mathbf{x}_{ij}^k) = \frac{2}{\frac{1}{D(\mathbf{x}_i)} + \frac{1}{D(\mathbf{x}_j)}}. \tag{3.21}$$

This ad hoc definition can be made more rigorous in the case of cell centered finite volume schemes on Voronoi meshes, see (Michev 1996).

With these definitions we have in all cases that the numerical fluxes fulfill

$$J_{ij}^k = -J_{ji}^k, \tag{3.22}$$

which ensures local conservation of mass over control volumes.

From (3.18) one can conclude that the fully upwinding discretization ($\beta = 1$) of the advective flux always leads to positive diagonal and negative offdiagonal entries in the stiffness matrix, regardless of any condition on the mesh. Under reasonable assumptions on $\mathbf{r}$ the discretization of the advective part leads to an M–matrix and therefore obeys a discrete maximum principle, see (Bey 1997). The discretization of the diffusive part yields an M–matrix only under certain assumptions on the mesh, e. g. for triangular elements in 2D the sum of the two angles opposite of an edge must be less than or equal π.

Finally, the linear form Q_h is also approximated using the midpoint rule:

$$\begin{aligned} Q_h(\psi_i) &= \int_{b_i} q(\mathbf{x})\, d\mathbf{x} - \int_{\partial b_i \cap \Gamma_n} J(\mathbf{x})\, ds \\ &\approx \sum_k q(\mathbf{x}_i)\,\mathrm{meas}(b_i^k) - \sum_{\gamma_i^{kf} \cap \Gamma_n} J(\mathbf{x}_i^{kf})\,\mathrm{meas}(\gamma_i^{kf}). \end{aligned} \tag{3.23}$$

The convergence properties of the vertex centered finite volume method for the stationary advection–diffusion problem have been investigated by several authors. The most comprehensive treatment can be found in the recent work of Bey (1997). For the diffusion–dominated case (no upwinding) one can show optimal error estimates in the H^1 and L^2–norms (only Dirichlet boundary conditions), i. e. $O(h)$ and $O(h^2)$ convergence, respectively, if $u \in H^2(\Omega)$. In the advection–dominated case one has $O(h)$ convergence when the fully upwinding procedure is used. The advection–dominated case is also investigated in Michev (1996) where some modified upwinding schemes are defined. Since these schemes are difficult to extend to the two–phase flow equations we do not consider them here.

3.3 Phase Pressure–Saturation Formulation (*PPS*)

In this section we apply the vertex centered finite volume method of the previous section to the two–phase flow equations in phase pressure–saturation formulation with p_w and S_n as unknowns. The resulting discretization scheme is referred to as *PPS* in the rest of this work.

The equations to be solved are given by (compare to Eqs. (2.2)):

$$\frac{\partial(\Phi\rho_w(1-S_n))}{\partial t} + \nabla\cdot\{\rho_w\mathbf{u}_w\} - \rho_w q_w = 0, \tag{3.24a}$$

$$\mathbf{u}_w = \lambda_w\mathbf{v}_w,\ \mathbf{v}_w = -K(\nabla p_w - \rho_w\mathbf{g}), \tag{3.24b}$$

$$\frac{\partial(\Phi\rho_n S_n)}{\partial t} + \nabla\cdot\{\rho_n\mathbf{u}_n\} - \rho_n q_n = 0, \tag{3.24c}$$

$$\mathbf{u}_n = \lambda_n\mathbf{v}_n,\ \mathbf{v}_n = -K(\nabla p_w + \nabla p_c - \rho_n\mathbf{g}). \tag{3.24d}$$

in $(0,T)\times\Omega$, Ω a polyhedral domain in $\mathbb{R}^N$, $d = 2,3$. Boundary conditions are given by

$$p_w(\mathbf{x},t) = p_{wd}(\mathbf{x},t) \text{ on } \Gamma_{wd} \qquad \rho_w\mathbf{u}_w\cdot\mathbf{n} = \phi_w(\mathbf{x},t) \text{ on } \Gamma_{wn} \tag{3.25a}$$

$$S_n(\mathbf{x},t) = S_{nd}(\mathbf{x},t) \text{ on } \Gamma_{nd} \qquad \rho_n\mathbf{u}_n\cdot\mathbf{n} = \phi_n(\mathbf{x},t) \text{ on } \Gamma_{nn} \tag{3.25b}$$

and initial conditions

$$p_w(\mathbf{x},0) = p_{w0}(\mathbf{x}),\ S_n(\mathbf{x},0) = S_{n0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega. \tag{3.26}$$

We will consider the general case where the coefficients may have the following dependencies ($\alpha = w,n$):

$$\mathbf{g} \text{ constant}, \tag{3.27a}$$

$$q_\alpha = q_\alpha(\mathbf{x},t), \tag{3.27b}$$

$$p_c = p_c(\mathbf{x},S_w),\ k_{r\alpha} = k_{r\alpha}(\mathbf{x},S_\alpha), \tag{3.27c}$$

$$\rho_\alpha = \rho_\alpha(p_\alpha),\ \mu_\alpha = \mu_\alpha(p_\alpha),\ \Phi = \Phi(\mathbf{x},p_w,p_n). \tag{3.27d}$$

The incompressible case, where ρ_w,ρ_n,Φ are constant, is also included. We will discretize Eqs. (3.24) first in space leaving the time variable continuous. Suitable time discretizations will be derived in a later section.

Following the derivation for the linear advection–diffusion equation we define index sets

$$I_{wd} = \{i\in I \mid \mathbf{x}_i\notin\Gamma_{wd}\}, \qquad I_{nd} = \{i\in I \mid \mathbf{x}_i\notin\Gamma_{nd}\} \tag{3.28}$$

as well as subsets of the finite element space V_h

$$V_{whd}(t) = \{v\in V_h \mid v(\mathbf{x}_i) = p_{wd}(\mathbf{x}_i,t), i\in I\setminus I_{wd}\}, \tag{3.29a}$$

$$V_{nhd}(t) = \{v\in V_h \mid v(\mathbf{x}_i) = S_{nd}(\mathbf{x}_i,t), i\in I\setminus I_{nd}\} \tag{3.29b}$$

and the test space W_h:

$$W_{whd} = \{w \in W_h \mid w(\mathbf{x}_i) = 0, i \in I \setminus I_{wd}\}, \tag{3.30a}$$
$$W_{nhd} = \{w \in W_h \mid w(\mathbf{x}_i) = 0, i \in I \setminus I_{nd}\}. \tag{3.30b}$$

Note that spaces $V_{\alpha hd}$ depend on time t. The corresponding weak formulation of the two–phase flow problem is then given by:

Find $p_{wh}(t) \in V_{whd}(t)$, $S_{nh}(t) \in V_{nhd}(t)$ such that for $\alpha = w, n$

$$\begin{aligned} &\frac{\partial}{\partial t} M_{\alpha h}(p_{wh}(t), S_{nh}(t), w_{\alpha h}) + A_{\alpha h}(p_{wh}(t), S_{nh}(t), w_{\alpha h}) \\ &\quad + Q_{\alpha h}(t, p_{wh}(t), S_{nh}(t), w_{\alpha h}) = 0 \qquad w_{\alpha h} \in W_{\alpha hd}, 0 < t < T. \end{aligned} \tag{3.31}$$

with the accumulation terms (the time argument is omitted for ease of writing)

$$M_{wh}(p_{wh}, S_{nh}, w_{wh}) = \sum_{i \in I} w_{wh}(\mathbf{x}_i) \int_{b_i} \Phi \rho_w (1 - S_{nh}) \, d\mathbf{x}, \tag{3.32a}$$

$$M_{nh}(p_{wh}, S_{nh}, w_{nh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \int_{b_i} \Phi \rho_n S_{nh} \, d\mathbf{x}, \tag{3.32b}$$

the internal flux terms

$$A_{wh}(p_{wh}, S_{nh}, w_{wh}) = \sum_{i \in I} w_{wh}(\mathbf{x}_i) \int_{\partial b_i \cap \Omega} \rho_w \mathbf{u}_w \cdot \mathbf{n} \, ds, \tag{3.33a}$$

$$A_{nh}(p_{wh}, S_{nh}, w_{nh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \int_{\partial b_i \cap \Omega} \rho_n \mathbf{u}_n \cdot \mathbf{n} \, ds, \tag{3.33b}$$

and the source, sink and boundary flux terms

$$Q_{wh}(t, p_{wh}, S_{nh}, w_{wh}) = \sum_{i \in I} w_{wh}(\mathbf{x}_i) \left[\int_{\partial b_i \cap \Gamma_{wn}} \phi_w \, ds - \int_{b_i} \rho_w q_w \, d\mathbf{x} \right], \tag{3.34a}$$

$$Q_{nh}(t, p_{wh}, S_{nh}, w_{wh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \left[\int_{\partial b_i \cap \Gamma_{nn}} \phi_n \, ds - \int_{b_i} \rho_n q_n \, d\mathbf{x} \right]. \tag{3.34b}$$

Writing (3.31) in terms of coefficient vectors leads to a system of ordinary differential equations (ODE) or, more precisely, to a system of differential algebraic equations in the incompressible case as will be discussed below:

For $0 < t < T$ find $\mathbf{p}_w(t) \in \mathbb{R}^N$, $P(\mathbf{p}_w(t)) \in V_{whd}(t)$ and $\mathbf{S}_n(t) \in \mathbb{R}^N$, $P(\mathbf{S}_n(t)) \in V_{nhd}(t)$ such that for $\alpha = w, n$:

$$\frac{\partial}{\partial t} \mathbf{M}_\alpha(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{A}_\alpha(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{Q}_\alpha(t, \mathbf{p}_w(t), \mathbf{S}_n(t)) = 0. \tag{3.35}$$

where the components are given by (time argument is suppressed)

$$\mathbf{M}_{\alpha,i}(\mathbf{p}_w, \mathbf{S}_n) = M_{\alpha h}(P(\mathbf{p}_w), P(\mathbf{S}_n), \psi_i), \tag{3.36a}$$
$$\mathbf{A}_{\alpha,i}(\mathbf{p}_w, \mathbf{S}_n) = A_{\alpha h}(P(\mathbf{p}_w), P(\mathbf{S}_n), \psi_i), \tag{3.36b}$$
$$\mathbf{Q}_{\alpha,i}(t, \mathbf{p}_w, \mathbf{S}_n) = Q_{\alpha h}(t, P(\mathbf{p}_w), P(\mathbf{S}_n), \psi_i) \tag{3.36c}$$

for all $i \in I_{\alpha d}$ and $\alpha = w, n$.

It remains to declare the precise evaluation of the quantities given in (3.36). All nonlinearities are evaluated at vertices and then interpreted as finite element functions, i. e. given $\mathbf{p}_w(t)$ and $\mathbf{S}_n(t)$ we have (time argument is suppressed):

$$p_{ch} = P(\mathbf{p_c}), \qquad \mathbf{p}_{\mathbf{c},i} = p_c(\mathbf{x}_i, 1 - \mathbf{S}_{n,i}), \tag{3.37}$$
$$p_{nh} = P(\mathbf{p}_n), \qquad \mathbf{p}_{n,i} = \mathbf{p}_{w,i} + \mathbf{p}_{\mathbf{c},i}, \tag{3.38}$$
$$\rho_{\alpha h} = P(\rho_\alpha), \qquad \rho_{\alpha,i} = \rho_\alpha(\mathbf{p}_{\alpha,i}), \tag{3.39}$$
$$\mu_{\alpha h} = P(\mu_\alpha), \qquad \mu_{\alpha,i} = \mu_\alpha(\mathbf{p}_{\alpha,i}), \tag{3.40}$$
$$\Phi_h = P(\Phi), \qquad \Phi_i = \Phi(\mathbf{x}_i, \mathbf{p}_{w,i}, \mathbf{p}_{n,i}), \tag{3.41}$$
$$k_{rwh} = P(\mathbf{k}_{rw}), \qquad \mathbf{k}_{rw,i} = k_{rw}(\mathbf{x}_i, 1 - \mathbf{S}_{n,i}), \tag{3.42}$$
$$k_{rnh} = P(\mathbf{k}_{rn}), \qquad \mathbf{k}_{rn,i} = k_{rn}(\mathbf{x}_i, \mathbf{S}_{n,i}), \tag{3.43}$$
$$\lambda_{\alpha h} = P(\lambda_\alpha), \qquad \lambda_{\alpha,i} = \mathbf{k}_{\mathbf{r}\alpha,i}/\mu_{\alpha,i}. \tag{3.44}$$

We begin with the accumulation terms which are approximated as

$$\begin{aligned} M_{wh}(p_{wh}, S_{nh}, \psi_i) &= \int_{b_i} \Phi_h \rho_{wh}(1 - S_{nh})\, d\mathbf{x} = \\ &\approx \sum_{k \in E(i)} \Phi_i \rho_{w,i}(1 - \mathbf{S}_{n,i}) \operatorname{meas}(b_i^k) \end{aligned} \tag{3.45}$$

and

$$\begin{aligned} M_{nh}(p_{wh}, S_{nh}, \psi_i) &= \int_{b_i} \Phi_h \rho_{nh} S_{nh}\, d\mathbf{x} = \\ &\approx \sum_{k \in E(i)} \Phi_i \rho_{n,i} \mathbf{S}_{n,i} \operatorname{meas}(b_i^k) \end{aligned} \tag{3.46}$$

The use of the midpoint rule corresponds to the mass lumping approach in the finite element method.

For the interior fluxes in the wetting phase we obtain

$$\begin{aligned} A_{wh}(p_{wh}, S_{nh}, \psi_i) &= \int_{\partial b_i \cap \Omega} \rho_{wh} \mathbf{u}_w \cdot \mathbf{n}\, ds = \\ &= \sum_{k,j} \int_{\gamma_{ij}^k} \rho_{wh} \lambda_{wh} \mathbf{v}_w \cdot \mathbf{n}\, ds \\ &\approx \sum_{k,j} \rho_{wh}(\mathbf{x}_{ij}^k)\, [\lambda_{wh}]_{ij}^k\, \mathbf{v}_{wh}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \operatorname{meas}(\gamma_{ij}^k) \end{aligned} \tag{3.47}$$

with the *directional part* of the Darcy velocity given by

$$\mathbf{v}_{wh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k)\left[\sum_{m\in I}\left(\mathbf{p}_{w,m}\nabla\varphi_m(\mathbf{x}_{ij}^k) - \rho_{w,m}\varphi_m(\mathbf{x}_{ij}^k)\mathbf{g}\right)\right] \tag{3.48}$$

(note that absolute permeability is evaluated at element barycenters) and the upwind evaluation of the mobility given by

$$[\lambda_{wh}]_{ij}^k = (1-\beta)\lambda_{wh}(\mathbf{x}_{ij}^k) + \beta\cdot\begin{cases}\lambda_{w,i} & \mathbf{v}_{wh}(\mathbf{x}_{ij}^k)\cdot\mathbf{n}_{ij}^k \geq 0\\ \lambda_{w,j} & \text{else}\end{cases}. \tag{3.49}$$

In the same way we obtain for the non–wetting phase

$$\begin{aligned} A_{nh}(p_{wh},S_{nh},\psi_i) &= \int\limits_{\partial b_i\cap\Omega}\rho_{nh}\mathbf{u}_n\cdot\mathbf{n}\,ds = \\ &= \sum_{k,j}\int\limits_{\gamma_{ij}^k}\rho_{nh}\lambda_{nh}\mathbf{v}_n\cdot\mathbf{n}\,ds \\ &\approx \sum_{k,j}\rho_{nh}(\mathbf{x}_{ij}^k)\,[\lambda_{nh}]_{ij}^k\,\mathbf{v}_{nh}(\mathbf{x}_{ij}^k)\cdot\mathbf{n}_{ij}^k\,\mathrm{meas}(\gamma_{ij}^k)\end{aligned} \tag{3.50}$$

with

$$\mathbf{v}_{nh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k)\left[\sum_{m\in I}\left(\mathbf{p}_{n,m}\nabla\varphi_m(\mathbf{x}_{ij}^k) - \rho_{n,m}\varphi_m(\mathbf{x}_{ij}^k)\mathbf{g}\right)\right] \tag{3.51}$$

and

$$[\lambda_{nh}]_{ij}^k = (1-\beta)\lambda_{nh}(\mathbf{x}_{ij}^k) + \beta\cdot\begin{cases}\lambda_{n,i} & \mathbf{v}_{nh}(\mathbf{x}_{ij}^k)\cdot\mathbf{n}_{ij}^k \geq 0\\ \lambda_{n,j} & \text{else}\end{cases}. \tag{3.52}$$

Finally, the sources/sinks and boundary fluxes are evaluated as

$$\begin{aligned} Q_{wh}(t,p_{wh},S_{nh},\psi_i) &= \int\limits_{\partial b_i\cap\Gamma_{wn}}\phi_w\,ds - \int\limits_{b_i}\rho_{wh}q_w\,d\mathbf{x} \\ &\approx \sum_{\gamma_i^{kf}\cap\Gamma_{wn}}\phi_w(\mathbf{x}_i^{kf},t)\,\mathrm{meas}(\gamma_i^{kf}) - \sum_k\rho_{w,i}q_w(\mathbf{x}_i,t)\,\mathrm{meas}(b_i^k)\end{aligned} \tag{3.53}$$

and

$$\begin{aligned} Q_{nh}(t,p_{wh},S_{nh},\psi_i) &= \int\limits_{\partial b_i\cap\Gamma_{nn}}\phi_n\,ds - \int\limits_{b_i}\rho_{nh}q_n\,d\mathbf{x} \\ &\approx \sum_{\gamma_i^{kf}\cap\Gamma_{nn}}\phi_n(\mathbf{x}_i^{kf},t)\,\mathrm{meas}(\gamma_i^{kf}) - \sum_k\rho_{n,i}q_n(\mathbf{x}_i,t)\,\mathrm{meas}(b_i^k)\end{aligned} \tag{3.54}$$

respectively.

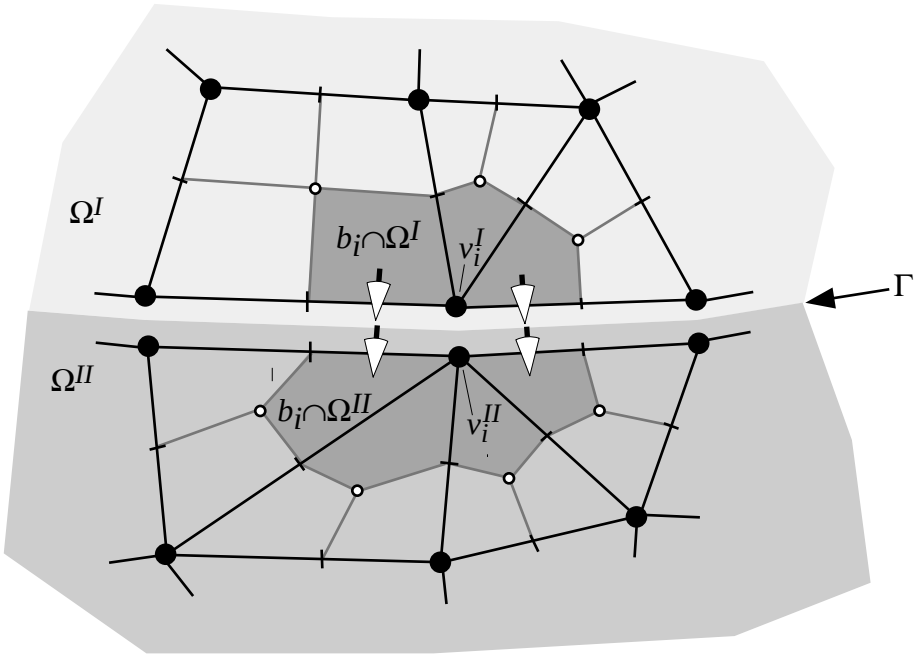


Figure 3.3: Control volume at the boundary of two subdomains.

3.4 Interface Condition Formulation (*PPSIC*)

In this section we incorporate the interface conditions at media discontinuities developed in Section 2.3 into the the vertex centered finite volume method. The resulting method is referred to as the *PPSIC* method.

The idea is as follows. Assume a domain consisting of two subdomains Ω^I and Ω^{II} with interface Γ. Let Ω be meshed in such a way that the interface Γ is resolved by mesh edges in 2D and faces in 3D. In order to develop the discrete equations imagine the two subdomains Ω^I, Ω^{II} to be separated and all vertices and corresponding degrees of freedom on the interface Γ to be duplicated. This situation is illustrated for vertex v_i in Fig. 3.3. Now (virtually) apply the vertex centered finite volume method separately in each of the two subdomains with flux type boundary conditions at the interface Γ. We now incorporate the interface conditions developed in Section 2.3. Since p_w is continuous at Γ the two degrees of freedom for p_w in vertex v_i on either side can be identified. From the extended capillary pressure continuity (2.45) we can compute S_n in v_i^{II} from the value of S_n in v_i^I which reduces the degrees of freedom for an interface vertex back to two. If we sum the discrete mass balance equation for phase α over control volumes $b_i \cap \Omega^I$ and $b_i \cap \Omega^{II}$ the normal fluxes over the edges (faces) at the interface cancel (indicated by the arrows in Fig. 2.3) out due to condition (2.44). Forgetting about the separation of Ω^I and Ω^{II} we are thus left with the standard balance equation over control volume b_i where the fluxes over control volume faces are evaluated in a special way.

In the development of the *PPS* method in Sect. 3.3 all quantities were assumed

to be continuous and have been evaluated at mesh vertices. This is not appropriate here since saturation (and all quantities derived from it) may be discontinuous at element boundaries. Furthermore we assume that spatial dependence, e. g. of porosity, may be discontinuous at element boundaries as well. All quantities are therefore evaluated as (multi-) linear functions on each element that do not have to be globally continuous.

Let the degrees of freedom be given as vectors $\mathbf{p}_w(t) \in \mathbb{R}^N$ and $\mathbf{S}_n(t) \in \mathbb{R}^N$ as before. Since p_w is globally continuous the evaluation restricted to the element e_k is the same as before:

$$p_{wh}|_{e_k}(\mathbf{x}) = \sum_{m \in V(k)} \mathbf{p}_{w,m} \varphi_m(\mathbf{x}) \tag{3.55}$$

where $V(k)$ are the indices of vertices of e_k and $\mathbf{x} \in \bar{e_k}$.

As an auxiliary vector we define $\mathbf{p}_{cmin}$ as

$$\mathbf{p}_{cmin,i} = \min_{k \in E(i)} p_c(\mathbf{x}^k, 1 - \mathbf{S}_{n,i}) \tag{3.56}$$

where $E(i)$ are the indices of all elements having vertex v_i as a corner and $\mathbf{x}^k$ is the barycenter of element e_k. In (3.56) we evaluate the capillary pressure function in all elements adjacent to vertex v_i for the saturation given there and compute the minimum value.

Using $\mathbf{p}_{cmin,i}$ we can compute the saturation S_n at vertex v_i *with respect to element* e_k via the extended capillary pressure condition as follows:

$$\hat{S}_{n,i,k} = \begin{cases} \mathbf{S}_{n,i} & \text{if } p_c(\mathbf{x}^k, 1 - \mathbf{S}_{n,i}) = \mathbf{p}_{cmin,i} \\ 0 & \mathbf{p}_{cmin,i} < p_c(\mathbf{x}^k, 1) \\ 1 - S & \text{where } S \text{ solves } p_c(\mathbf{x}^k, S) = \mathbf{p}_{cmin,i} \end{cases} . \tag{3.57}$$

Note that this definition also allows more than two subdomains meeting at vertex v_i. The evaluation of saturation with respect to e_k for any $\mathbf{x} \in \bar{e_k}$ (includes the corners !) is then given by

$$S_{nh}|_{e_k}(\mathbf{x}) = \sum_{m \in V(k)} \hat{S}_{n,m,k} \varphi_m(\mathbf{x}). \tag{3.58}$$

We are now in a position to state the evaluation of quantities depending on saturation:

$$p_{ch}|_{e_k}(\mathbf{x}) = \sum_{m \in V(k)} p_c(\mathbf{x}^k, 1 - \hat{S}_{n,m,k}) \varphi_m(\mathbf{x}) \tag{3.59}$$

$$p_{nh}|_{e_k}(\mathbf{x}) = \sum_{m \in V(k)} \left(\mathbf{p}_{w,m} + p_c(\mathbf{x}^k, 1 - \hat{S}_{n,m,k}) \right) \varphi_m(\mathbf{x}) \tag{3.60}$$

$$\rho_{nh}|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} \rho_n(\, p_{nh}|_{e_k}(\mathbf{x}_m))\varphi_m(\mathbf{x}) \tag{3.61}$$

$$\mu_{nh}|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} \mu_n(\, p_{nh}|_{e_k}(\mathbf{x}_m))\varphi_m(\mathbf{x}) \tag{3.62}$$

$$k_{rwh}|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} k_{rw}(\mathbf{x}^k, 1-\hat{S}_{n,m,k})\varphi_m(\mathbf{x}) \tag{3.63}$$

$$k_{rnh}|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} k_{rn}(\mathbf{x}^k, \hat{S}_{n,m,k})\varphi_m(\mathbf{x}) \tag{3.64}$$

$$\lambda_{\alpha h}|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} \frac{k_{r\alpha h}|_{e_k}(\mathbf{x}_m)}{\mu_{\alpha h}|_{e_k}(\mathbf{x}_m)}\varphi_m(\mathbf{x}) \tag{3.65}$$

$$\Phi_h|_{e_k}(\mathbf{x}) = \sum_{m\in V(k)} \Phi\left(\mathbf{x}^k, \mathbf{p}_{w,m},\, p_{nh}|_{e_k}(\mathbf{x}_m)\right)\varphi_m(\mathbf{x}) \tag{3.66}$$

Note that the positional argument is always the barycenter of the element to catch the dependence on subdomains correctly. The definition of ρ_w and μ_w is the same as in the *PPS* method since p_w is continuous.

The approximation of the dual forms $M_{\alpha h}$, $A_{\alpha h}$ and $Q_{\alpha h}$ is the same as in (3.45) through (3.54) with evaluation of coefficients replaced by their element–wise counterparts defined above, we give M_{nh} as an example:

$$\mathbf{M}_{n,i}(\mathbf{p}_w, \mathbf{S}_n) \approx \sum_{k\in E(i)} \Phi_h|_{e_k}(\mathbf{x}_i)\, \rho_{nh}|_{e_k}(\mathbf{x}_i)\, \hat{S}_{n,i,k}\ \mathrm{meas}(b_i^k) \tag{3.67}$$

3.5 Global Pressure with Total Velocity (*GPSTV*)

The purpose of this section is to apply the vertex centered finite volume method to the *incompressible* two–phase flow problem in global pressure/total velocity formulation. This formulation has the advantage that both extreme values of saturation can be treated in the domain Ω. Furthermore, the pressure and saturation equations are less coupled and should therefore be easier to solve.

The continuous problem is given by

$$\nabla\cdot\mathbf{u} - q_w - q_n = 0, \tag{3.68a}$$

$$\mathbf{u} = \lambda\mathbf{v},\ \mathbf{v} = -K\left(\nabla p - \mathbf{G}\right), \tag{3.68b}$$

$$\frac{\partial(\Phi\rho_n S_n)}{\partial t} + \nabla\cdot\{\rho_n\mathbf{u}_n\} - \rho_n q_n = 0, \tag{3.68c}$$

$$\mathbf{u}_n = \lambda_n\mathbf{v}_n,\ \mathbf{v}_n = -K\left(\nabla p + f_w\nabla p_c - \rho_n\mathbf{g}\right). \tag{3.68d}$$

with boundary conditions

$$p(\mathbf{x},t) = p_d(\mathbf{x},t) \text{ on } \Gamma_{wd} \qquad \mathbf{u}\cdot\mathbf{n} = U(\mathbf{x},t) \text{ on } \Gamma_{wn} \tag{3.69a}$$

$$S_n(\mathbf{x},t) = S_{nd}(\mathbf{x},t) \text{ on } \Gamma_{nd} \qquad \rho_n\mathbf{u}_n\cdot\mathbf{n} = \phi_n(\mathbf{x},t) \text{ on } \Gamma_{nn} \tag{3.69b}$$

and initial conditions

$$S_n(\mathbf{x},0) = S_{n0}(\mathbf{x}) \qquad \mathbf{x} \in \Omega. \tag{3.70}$$

The coefficient functions are supposed to have the following properties:

$$\rho_\alpha,\ \mu_\alpha,\ \mathbf{g} \text{ constant}, \tag{3.71a}$$
$$q_\alpha = q_\alpha(\mathbf{x},t),\ \Phi = \Phi(\mathbf{x}) \tag{3.71b}$$
$$p_c = p_c(S_w),\ k_{r\alpha} = k_{r\alpha}(S_\alpha). \tag{3.71c}$$

The definition of the index sets I_{wd}, I_{nd} and discrete function spaces $V_{\alpha hd}(t)$, $W_{\alpha hd}$ carries over from Sect. 3.3 in the obvious way. The weak formulation is now given by:

Find $p_h(t) \in V_{whd}(t)$, $S_{nh}(t) \in V_{nhd}(t)$ such that

$$\frac{\partial}{\partial t} M_{nh}(S_{nh}(t), w_{nh}) + A_{nh}(p_h(t), S_{nh}(t), w_{nh}) + Q_{nh}(t, w_{nh}) = 0 \tag{3.72a}$$
$$A_{wh}(p_h(t), S_{nh}(t), w_{wh}) + Q_{wh}(t, w_{wh}) = 0 \tag{3.72b}$$

for all $w_{\alpha h} \in W_{\alpha hd}$ and $0 < t < T$ with the accumulation term given by

$$M_{nh}(S_{nh}, w_{nh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \int_{b_i} \Phi \rho_n S_{nh}\, d\mathbf{x}, \tag{3.73}$$

the internal flux terms given by

$$A_{nh}(p_h, S_{nh}, w_{nh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \int_{\partial b_i \cap \Omega} \rho_n \mathbf{u}_n \cdot \mathbf{n}\, ds, \tag{3.74a}$$

$$A_{wh}(p_h, S_{nh}, w_{wh}) = \sum_{i \in I} w_{wh}(\mathbf{x}_i) \int_{\partial b_i \cap \Omega} \mathbf{u} \cdot \mathbf{n}\, ds, \tag{3.74b}$$

and the source, sink and boundary flux terms

$$Q_{nh}(t, w_{wh}) = \sum_{i \in I} w_{nh}(\mathbf{x}_i) \left[\int_{\partial b_i \cap \Gamma_{nn}} \phi_n\, ds - \int_{b_i} \rho_n q_n\, d\mathbf{x} \right], \tag{3.75a}$$

$$Q_{wh}(t, w_{wh}) = \sum_{i \in I} w_{wh}(\mathbf{x}_i) \left[\int_{\partial b_i \cap \Gamma_{wn}} U\, ds - \int_{b_i} q_w + q_n\, d\mathbf{x} \right]. \tag{3.75b}$$

Writing (3.72) in terms of coefficient vectors leads to a system of ordinary differential equations supplemented by a set of algebraic equations (constraints):

For $0 < t < T$ find $\mathbf{p}(t) \in \mathbb{R}^N$, $P(\mathbf{p}(t)) \in V_{whd}(t)$ and $\mathbf{S}_n(t) \in \mathbb{R}^N$, $P(\mathbf{S}_n(t)) \in V_{nhd}(t)$ such that

$$\frac{\partial}{\partial t} \mathbf{M}_n(\mathbf{S}_n(t)) + \mathbf{A}_n(\mathbf{p}(t), \mathbf{S}_n(t)) + \mathbf{Q}_n(t) = 0 \tag{3.76a}$$
$$\mathbf{A}_w(\mathbf{p}(t), \mathbf{S}_n(t)) + \mathbf{Q}_w(t) = 0 \tag{3.76b}$$

with components given in the obvious way (see Sect. 3.3).

Since we do not consider porous media with discontinuities here the evaluation of coefficient functions is done at vertices with subsequent finite element interpolation as described in the *PPS* method.

The approximation of M_{nh}, Q_{nh} and Q_{wh} is straightforward (see Sect. 3.3). In the A_{nh}–term the velocity is now written in terms of global pressure:

$$\begin{aligned} A_{nh}(p_{wh}, S_{nh}, \psi_i) &= \int\limits_{\partial b_i \cap \Omega} \rho_n \mathbf{u}_n \cdot \mathbf{n} \, ds = \\ &= \sum_{k,j} \int\limits_{\gamma_{ij}^k} \rho_n \lambda_{nh} \mathbf{v}_n \cdot \mathbf{n} \, ds \\ &\approx \sum_{k,j} \rho_n \, [\lambda_{nh}]_{ij}^k \, \mathbf{v}_{nh}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \, \mathrm{meas}(\gamma_{ij}^k) \end{aligned} \tag{3.77}$$

with

$$\mathbf{v}_{nh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k) \left[\sum_{m \in I} \left(\mathbf{p}_m + f_{wh}(\mathbf{x}_{ij}^k) \mathbf{p}_{c,m} \right) \nabla \varphi_m(\mathbf{x}_{ij}^k) - \rho_n \mathbf{g} \right] \tag{3.78}$$

and

$$[\lambda_{nh}]_{ij}^k = (1-\beta)\lambda_{nh}(\mathbf{x}_{ij}^k) + \beta \cdot \begin{cases} \lambda_{n,i} & \mathbf{v}_{nh}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \geq 0 \\ \lambda_{n,j} & \text{else} \end{cases} . \tag{3.79}$$

Note that the velocity $\mathbf{v}_{nh}(\mathbf{x}_{ij}^k)$ used to evaluate the upwind switch still contains f_w. A "central" evaluation of f_w in (3.78) seems to work ok since the problem is diffusion dominated if ∇p_c is the dominant term in $\mathbf{v}_n$.

Upwinding for the total mobility in the A_{wh}–term is done via separate upwinding of the phase mobilities. Therefore we evaluate the wetting phase velocity (direction) at the integration point

$$\mathbf{v}_{wh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k) \left[\sum_{m \in I} \left(\mathbf{p}_m - f_{nh}(\mathbf{x}_{ij}^k) \mathbf{p}_{c,m} \right) \nabla \varphi_m(\mathbf{x}_{ij}^k) - \rho_w \mathbf{g} \right] \tag{3.80}$$

and the corresponding integration point value of wetting phase mobility

$$[\lambda_{wh}]_{ij}^k = (1-\beta)\lambda_{wh}(\mathbf{x}_{ij}^k) + \beta \cdot \begin{cases} \lambda_{w,i} & \mathbf{v}_{wh}(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \geq 0 \\ \lambda_{w,j} & \text{else} \end{cases} . \tag{3.81}$$

The A_{wh}–term is evaluated as

$$\begin{aligned} A_{wh}(p_{wh}, S_{nh}, \psi_i) &= \int_{\partial b_i \cap \Omega} \mathbf{u} \cdot \mathbf{n} \, ds = \\ &= \sum_{k,j} \int_{\gamma_{ij}^k} \lambda_h \mathbf{v} \cdot \mathbf{n} \, ds \\ &\approx \sum_{k,j} [\lambda_h]_{ij}^k \, \mathbf{v}_h(\mathbf{x}_{ij}^k) \cdot \mathbf{n}_{ij}^k \, \mathrm{meas}(\gamma_{ij}^k) \end{aligned} \tag{3.82}$$

with the integration point value of total mobility given by

$$[\lambda_h]_{ij}^k = [\lambda_{wh}]_{ij}^k + [\lambda_{nh}]_{ij}^k \tag{3.83}$$

and the directional part of the total velocity given by

$$\mathbf{v}_h(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k) \left[\sum_{m \in I} \mathbf{p}_m \nabla \varphi_m(\mathbf{x}_{ij}^k) - \frac{\rho_w [\lambda_{wh}]_{ij}^k + \rho_n [\lambda_{nh}]_{ij}^k}{[\lambda_h]_{ij}^k} \mathbf{g} \right] \tag{3.84}$$

3.6 Global Pressure with Total Flux (*GPSTF*)

In contrast to the last section we now wish to apply the vertex centered finite volume discretization to the *compressible* two phase flow problem in global pressure formulation. Unfortunately, Eq. (2.33) is not in conservative form and the finite volume technique cannot be applied to the term $\nabla \rho_\alpha \cdot \mathbf{u}_\alpha$. We therefore propose a formulation with a global pressure that uses the *total flux* instead of the total velocity. The continuous problem is given by

$$\frac{\partial \left(\Phi \rho_w (1 - S_n) + \Phi \rho_n S_n \right)}{\partial t} + \nabla \cdot \mathbf{j} - \rho_w q_w - \rho_n q_n = 0, \tag{3.85a}$$

$$\mathbf{j} = \rho_w \mathbf{u}_w + \rho_n \mathbf{u}_n, \tag{3.85b}$$

$$\mathbf{u}_w = \lambda_w \mathbf{v}_w, \; \mathbf{v}_w = -K \left(\nabla p - f_n \nabla p_c - \rho_w \mathbf{g} \right), \tag{3.85c}$$

$$\mathbf{u}_n = \lambda_n \mathbf{v}_n, \; \mathbf{v}_n = -K \left(\nabla p + f_w \nabla p_c - \rho_n \mathbf{g} \right), \tag{3.85d}$$

$$\frac{\partial \left(\Phi \rho_n S_n \right)}{\partial t} + \nabla \cdot \{ \rho_n \mathbf{u}_n \} - \rho_n q_n = 0. \tag{3.85e}$$

$$\tag{3.85f}$$

With global pressure being defined by (2.21), capillary pressure is now not completely eliminated from the pressure equation, however, its influence is reduced compared to the phase pressure formulation since it is always multiplied by the product $\lambda_w \lambda_n$ which vanishes for extreme values of saturation. Furthermore, the quantity $\rho_w \lambda_w + \rho_n \lambda_n$ does vary less than $\lambda_w + \lambda_n$ in the case of water–gas systems (the variation in viscosity is partly compensated by the variation in density).

Boundary conditions for (3.85) are given by

$$p(\mathbf{x},t) = p_d(\mathbf{x},t) \text{ on } \Gamma_{wd} \qquad \mathbf{j}\cdot\mathbf{n} = J(\mathbf{x},t) \text{ on } \Gamma_{wn} \tag{3.86a}$$

$$S_n(\mathbf{x},t) = S_{nd}(\mathbf{x},t) \text{ on } \Gamma_{nd} \qquad \rho_n \mathbf{u}_n\cdot\mathbf{n} = \phi_n(\mathbf{x},t) \text{ on } \Gamma_{nn} \tag{3.86b}$$

and initial conditions by

$$p(\mathbf{x},0) = p(\mathbf{x}),\ S_n(\mathbf{x},0) = S_{n0}(\mathbf{x}) \qquad \mathbf{x}\in\Omega. \tag{3.87}$$

The coefficient functions are supposed to have the following properties:

$$\mu_\alpha,\ \mathbf{g} \text{ constant}, \tag{3.88a}$$

$$q_\alpha = q_\alpha(\mathbf{x},t), \tag{3.88b}$$

$$p_c = p_c(S_w),\ k_{r\alpha} = k_{r\alpha}(S_\alpha), \tag{3.88c}$$

$$\rho_\alpha = \rho_\alpha(p),\ \Phi = \Phi(\mathbf{x},p). \tag{3.88d}$$

With the standard notation introduced in Sect. 3.3 we have the weak formulation:

Find $p_h(t)\in V_{whd}(t)$, $S_{nh}(t)\in V_{nhd}(t)$ such that for $\alpha = w,n$

$$\begin{aligned}\frac{\partial}{\partial t}M_{\alpha h}(p_h(t),S_{nh}(t),w_{\alpha h}) + A_{\alpha h}(p_h(t),S_{nh}(t),w_{\alpha h})\\ + Q_{\alpha h}(t,p_h(t),w_{\alpha h}) = 0 \qquad w_{\alpha h}\in W_{\alpha hd}, 0<t<T.\end{aligned} \tag{3.89}$$

with

$$M_{wh}(p_h,S_{nh},w_{wh}) = \sum_{i\in I} w_{wh}(\mathbf{x}_i)\int_{b_i}\Phi\left(\rho_w(1-S_{nh}) + \rho_n S_{nh}\right)\,d\mathbf{x}, \tag{3.90a}$$

$$A_{wh}(p_h,S_{nh},w_{wh}) = \sum_{i\in I} w_{wh}(\mathbf{x}_i)\int_{\partial b_i\cap\Omega}(\rho_w\mathbf{u}_w + \rho_n\mathbf{u}_n)\cdot\mathbf{n}\,ds, \tag{3.90b}$$

$$Q_{wh}(t,p_h,w_{wh}) = \sum_{i\in I} w_{wh}(\mathbf{x}_i)\left[\int_{\partial b_i\cap\Gamma_{wn}} J\,ds - \int_{b_i}(\rho_w q_w + \rho_n q_n)\,d\mathbf{x}\right], \tag{3.90c}$$

and the other terms as in Sect. 3.3. The system of ODE also has the same structure as in the *PPS* method.

The evaluation of the forms (3.90) for a test function ψ_i is done as follows:

$$M_{wh}(p_h,S_{nh},\psi_i) \approx \sum_k \Phi_i\left(\rho_{w,i}(1-\mathbf{S}_{n,i}) + \rho_{n,i}\mathbf{S}_{n,i}\right)\text{meas}(b_i^k), \tag{3.91a}$$

$$A_{wh}(p_h,S_{nh},\psi_i) \approx \sum_{k,j}\sum_{\alpha=w,n}\rho_{\alpha h}(\mathbf{x}_{ij}^k)\,[\lambda_{\alpha h}]_{ij}^k\,\mathbf{v}_{\alpha h}(\mathbf{x}_{ij}^k)\cdot\mathbf{n}_{ij}^k\,\text{meas}(\gamma_{ij}^k), \tag{3.91b}$$

$$Q_{wh}(t,p_h,\psi_i) \approx \sum_{\gamma_i^{kf}\cap\Gamma_{wn}} J(\mathbf{x}_i^{kf},t)\,\text{meas}(\gamma_i^{kf}) - \sum_k\sum_\alpha \rho_{\alpha,i}q_\alpha(\mathbf{x}_i,t)\,\text{meas}(b_i^k) \tag{3.91c}$$

with

$$\mathbf{v}_{wh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k)\left[\sum_{m\in I}\left(\mathbf{p}_m - f_{nh}(\mathbf{x}_{ij}^k)\mathbf{p}_{c,m}\right)\nabla\varphi_m(\mathbf{x}_{ij}^k) - \rho_{wh}(\mathbf{x}_{ij}^k)\mathbf{g}\right], \quad (3.92a)$$

$$\mathbf{v}_{nh}(\mathbf{x}_{ij}^k) = -K(\mathbf{x}^k)\left[\sum_{m\in I}\left(\mathbf{p}_m + f_{wh}(\mathbf{x}_{ij}^k)\mathbf{p}_{c,m}\right)\nabla\varphi_m(\mathbf{x}_{ij}^k) - \rho_{nh}(\mathbf{x}_{ij}^k)\mathbf{g}\right] \quad (3.92b)$$

and

$$[\lambda_{\alpha h}]_{ij}^k = (1-\beta)\lambda_{\alpha h}(\mathbf{x}_{ij}^k) + \beta\cdot\begin{cases}\lambda_{\alpha,i} & \mathbf{v}_{\alpha h}(\mathbf{x}_{ij}^k)\cdot\mathbf{n}_{ij}^k \geq 0\\ \lambda_{\alpha,j} & \text{else}\end{cases}. \quad (3.93)$$

3.7 Implicit Time Discretization

We now describe some implicit time discretization schemes that are used to derive fully discrete schemes from the semi–discrete equations given above.

Let the time interval $(0,T)$ be subdivided into discrete steps

$$0 = t^0, t^1, \ldots, t^n, t^{n+1}, \ldots, t^M = T \quad (3.94)$$

that are not necessarily equidistant. The evaluation of any quantity at time level t^n is denoted by a superscript n (not to be mixed up with subscript n, which denotes the non–wetting phase). E. g. we have $p_{wh}(t^n) = p_{wh}^n$, $V_{whd}(t^n) = V_{whd}^n$ or $\mathbf{S}_n(t^n) = \mathbf{S}_n^n$. The notation for a time step is

$$\Delta t^n = t^{n+1} - t^n. \quad (3.95)$$

3.7.1 One Step θ-Scheme

The one step θ–scheme (see e. g. (Rannacher 1994; Helmig 1997)) applied to the semi–discrete system (3.35) yields:
For $n = 0, 1, \ldots, M-1$ find $\mathbf{p}_w^n, \mathbf{S}_n^n$ such that for $\alpha = w, n$

$$\mathbf{M}_\alpha^{n+1} - \mathbf{M}_\alpha^n + \Delta t^n\theta\left(\mathbf{A}_\alpha^{n+1} + \mathbf{Q}_\alpha^{n+1}\right) + \Delta t^n(1-\theta)\left(\mathbf{A}_\alpha^n + \mathbf{Q}_\alpha^n\right) = 0, \quad (3.96)$$

with $\mathbf{M}_\alpha^{n+1} = \mathbf{M}_\alpha(\mathbf{p}_w^n, \mathbf{S}_n^n)$, etc. In case of the semi–discrete system (3.76) it would read

$$\mathbf{M}_n^{n+1} - \mathbf{M}_n^n + \Delta t^n\theta\left(\mathbf{A}_n^{n+1} + \mathbf{Q}_n^{n+1}\right) + \Delta t^n(1-\theta)\left(\mathbf{A}_n^n + \mathbf{Q}_n^n\right) = 0, \quad (3.97a)$$

$$\mathbf{A}_w^{n+1} + \mathbf{Q}_w^{n+1} = 0, \quad (3.97b)$$

i. e. the time discretization is only applied to the saturation equation and the pressure equation should be satisfied at the new time level.

For $\theta = 1$ we obtain the first order accurate backward Euler scheme and for $\theta = 1/2$ the Crank–Nicolson scheme which is second order accurate in time. However, the Crank–Nicolson scheme has only weak damping properties, cf. Rannacher (1988), which may cause stability problems.

3.7.2 Backward Difference Formula

The second order backward difference formula, BDF(2), has superior damping properties when compared to the Crank–Nicolson scheme (see (Rannacher 1988)) and is a standard method for stiff ODE problems, see e. g. Hairer and Wanner (1991). BDF(2) is a two step scheme requiring the solution at two preceding time levels. In our scheme the solution at t^1 is simply computed with the one step θ–scheme from above. Starting with the second time step the scheme reads:
For $n = 1, 2, \ldots, M-1$ find $\mathbf{p}_w^n, \mathbf{S}_n^n$ such that for $\alpha = w, n$

$$\sum_{k=-1}^{1} a_{n,k}\mathbf{M}_\alpha^{n+k} + \Delta t^n \left(\mathbf{A}_\alpha^{n+1} + \mathbf{Q}_\alpha^{n+1}\right) = 0, \tag{3.98}$$

with the coefficients given by

$$a_{n,1} = \frac{\Delta t^{n-1} + 2\Delta t^n}{\Delta t^{n-1} + \Delta t^n},\ a_{n,0} = -\frac{\Delta t^{n-1} + \Delta t^n}{\Delta t^{n-1}}, \tag{3.99a}$$

$$a_{n,-1} = \frac{(\Delta t^n)^2}{(\Delta t^{n-1})^2 + \Delta t^{n-1}\Delta t^n}. \tag{3.99b}$$

The application to the semi–discrete system (3.76) is done as in the one step θ–scheme.

3.7.3 Differential Algebraic Equations

With the global pressure formulation *GPSTV* for the incompressible case we obtained the semi–discrete problem

$$\frac{\partial}{\partial t}\mathbf{M}_n(\mathbf{S}_n(t)) + \mathbf{A}_n(\mathbf{p}(t), \mathbf{S}_n(t)) + \mathbf{Q}_n(t) = 0 \tag{3.100}$$

$$\mathbf{A}_w(\mathbf{p}(t), \mathbf{S}_n(t)) + \mathbf{Q}_w(t) = 0 \tag{3.101}$$

cf. Eq.(3.76). This system is in the form of a system of *differential algebraic equations* (DAE), i. e. a system of ODE supplemented with a set of algebraic constraints. More specific, it is a system of DAE with index 1 since the constraint equation can always be solved for $\mathbf{p}(t)$ when $\mathbf{S}_n(t)$ is given. Furthermore, it is said to be in *explicit* form since the constraint equation is given separately.

With the other three schemes *PPS*, *PPSIC* and *GPSTF* we obtained semi–discrete systems of the form

$$\frac{\partial}{\partial t}\mathbf{M}_w(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{A}_w(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{Q}_w(t, \mathbf{p}_w(t), \mathbf{S}_n(t)) = 0$$

$$\frac{\partial}{\partial t}\mathbf{M}_n(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{A}_n(\mathbf{p}_w(t), \mathbf{S}_n(t)) + \mathbf{Q}_n(t, \mathbf{p}_w(t), \mathbf{S}_n(t)) = 0$$

with a time derivative in both equations. This system can be formally rewritten in the form

$$\underbrace{\begin{pmatrix} M_{ww} & M_{wn} \\ M_{nw} & M_{nn} \end{pmatrix}}_{M} \begin{pmatrix} \frac{\partial \mathbf{p}_w(t)}{\partial t} \\ \frac{\partial \mathbf{S}_n(t)}{\partial t} \end{pmatrix} + \begin{pmatrix} \mathbf{A}_w(\mathbf{p}_w, \mathbf{S}_n) + \mathbf{Q}_w(t, \mathbf{p}_w, \mathbf{S}_n) \\ \mathbf{A}_n(\mathbf{p}_w, \mathbf{S}_n) + \mathbf{Q}_n(t, \mathbf{p}_w, \mathbf{S}_n) \end{pmatrix} = 0$$

with the (solution–dependent) submatrices given by

$$(M_{\alpha w})_{ij} = \frac{\partial \mathbf{M}_{\alpha \mathbf{w},i}}{\partial \mathbf{p}_{w,j}}, \qquad (M_{\alpha n})_{ij} = \frac{\partial \mathbf{M}_{\alpha \mathbf{n},i}}{\partial \mathbf{S}_{n,j}}.$$

In the incompressible case this results into a system of DAE in *implicit* form which is characterized by M being a singular matrix. This has some consequences for the time discretization schemes. Necessary properties for the general case can be found in Hairer and Wanner (1991). We will only show here that the two schemes defined above correctly treat the implicit constraint when applied to the incompressible two–phase flow problem.

Let us assume that $I_{wd} = I_{nd}$, i. e. at a boundary vertex both components either have Dirichlet or flux boundary conditions. In the incompressible case (ρ_α, Φ constant) we obtain the following equations for the one step θ–scheme:

$$M_{wn}\left(\mathbf{S}_n^{n+1} - \mathbf{S}_n^n\right) + \Delta t^n \theta \left(\mathbf{A}_w^{n+1} + \mathbf{Q}_w^{n+1}\right) + \Delta t^n (1-\theta)\left(\mathbf{A}_w^n + \mathbf{Q}_w^n\right) = 0, \tag{3.102a}$$

$$M_{nn}\left(\mathbf{S}_n^{n+1} - \mathbf{S}_n^n\right) + \Delta t^n \theta \left(\mathbf{A}_n^{n+1} + \mathbf{Q}_n^{n+1}\right) + \Delta t^n (1-\theta)\left(\mathbf{A}_n^n + \mathbf{Q}_n^n\right) = 0, \tag{3.102b}$$

where the matrices M_{wn} and M_{nn} are diagonal and of the same size with entries independent of the solution. By eliminating $\mathbf{S}_n^{n+1} - \mathbf{S}_n^n$ from this system we obtain the relation

$$\begin{aligned} &\theta\left[M_{wn}^{-1}\left(\mathbf{A}_w^{n+1} + \mathbf{Q}_w^{n+1}\right) - M_{nn}^{-1}\left(\mathbf{A}_n^{n+1} + \mathbf{Q}_n^{n+1}\right)\right] \\ &\quad + (1-\theta)\left[M_{wn}^{-1}\left(\mathbf{A}_w^n + \mathbf{Q}_w^n\right) - M_{nn}^{-1}\left(\mathbf{A}_n^n + \mathbf{Q}_n^n\right)\right] = 0. \end{aligned} \tag{3.103}$$

The expression in square brackets is, in case of the *PPS*–method, a discrete version of the constraint equation (2.11a). As can be seen, the constraints at the new time level and the old time level occur in equation (3.103). If $\theta = 1$, i. e. the backward Euler method, the constraint is always fulfilled at the new time level. If $0 < \theta < 1$ we can state that the constraint equation at the new time level is satisfied if it has been satisfied at the old time level. It is therefore important for the Crank–Nicolson scheme to start with a pressure field that satisfies the constraint equation. Since we do not want to rewrite the DAE in explicit form we simply use one step of backward Euler in the very first time step to make the pressure field fulfill the constraint equation.

Since only spatial terms of the new time level are needed in the BDF(2) scheme the implicit constraint is always satisfied as in the backward Euler scheme. Unfortunately some favorable schemes, such as the fractional step θ–scheme, see (Rannacher 1994), cannot be applied directly to a system of implicit DAE since they do not satisfy the implicit constraint.

3.7.4 Global Conservation of Mass

Any finite–volume scheme has the property of conserving mass locally and globally. It is therefore important that this property is not destroyed by the time discretization scheme. We will shortly illustrate here that the one step θ–scheme (together with the finite volume discretization in space) conserves mass globally even for variable time steps. Unfortunately, the BDF(2) scheme suffers from mass balance errors when the time step size is changed.

In order to verify global conservation of mass of a discrete scheme in the time–dependent case we have to show that the sum over all discrete equations and time levels has the form

$$\left\{ \begin{array}{l} \text{Total mass} \\ \text{in } \Omega \text{ at } t^M \end{array} \right\} - \left\{ \begin{array}{l} \text{Total mass} \\ \text{in } \Omega \text{ at } t^0 \end{array} \right\} + \left\{ \begin{array}{l} \text{Sum of sources,} \\ \text{sinks, boundary fluxes} \end{array} \right\} = 0. \tag{3.104}$$

Indeed, for the one step θ–scheme we obtain for $\alpha = w, n$:

$$\sum_{i \in I_\alpha} \left(\mathbf{M}^M_{\alpha,i} - \mathbf{M}^0_{\alpha,i} \right) + \sum_{n=0}^{M-1} \sum_{i \in I_\alpha} \Delta t^n \left[\theta \mathbf{Q}^{n+1}_{\alpha,i} + (1-\theta) \mathbf{Q}^n_{\alpha,i} \right] = 0, \tag{3.105}$$

which is of the required form.

For a fixed size of the time step $\Delta t^n = \Delta t$ the BDF(2) scheme leads to

$$\sum_{i \in I_\alpha} \left[\frac{3}{2} \mathbf{M}^M_{\alpha,i} - \frac{1}{2} \mathbf{M}^{M-1}_{\alpha,i} - \frac{1}{2} (\mathbf{M}^1_{\alpha,i} + \mathbf{M}^0_{\alpha,i}) \right] + \sum_{n=0}^{M-1} \sum_{i \in I_\alpha} \Delta t \mathbf{Q}^{n+1}_{\alpha,i} = 0, \tag{3.106}$$

where one step of backward Euler has been used for the very first time step. The "fancy" approximation of the initial mass comes from the two step nature of BDF(2). If the time step size is allowed to vary then the accumulation terms at intermediate time steps do not cancel out since $\sum_{k=-1}^{1} A_{n+k,k} \neq 0$ in general. This is a consequence of BDF(2) being a difference scheme in time, whereas the one step θ–scheme comes from an integral formulation in time with trapezoidal rule for the spatial terms.

3.8 Validation of the Numerical Model

In this section we compare numerical computations and (quasi–) analytic solutions for the two one–dimensional model problems analyzed in Sections 2.4.2 and 2.4.3. The aim here is to show that the numerical solution converges towards the exact solution and to determine the experimental order of convergence.

3.8.1 HYPERBOLIC CASE

The incompressible two–phase problem without capillary pressure is solved in the domain $\Omega = (0,300[m]) \times (0,75[m])$ and the time interval $(0,1500[d])$ with the following parameters:

fluids:
$\rho_w = \rho_n = 1000[kg/m^3]$
$\mu_w = \mu_n = 0.001[Pa\ s]$

rock:
$\Phi = 0.2$
$K = 10^{-7}[m^2]$

residual saturation:
$S_{wr} = S_{nr} = 0$

relative permeability:
Brooks–Corey, $\lambda = 2.0$

capillary pressure:
$p_c \equiv 0$

boundary conditions:
$\phi_\alpha = 0$ for $y = 0$ and $y = 75[m]$
$p_n = 2 \cdot 10^5[Pa]$, $S_w = 1$ for $x = 0$
$\phi_n = 3 \cdot 10^{-4}[kg/(ms^2)]$, $S_w = 0$ for $x = 300$

initial conditions:
$S_w(\mathbf{x},0) = 0$ for $\mathbf{x} \in \Omega$

The domain Ω is discretized with $K \times 1$ quadrilateral elements, where $K = 32, 64, \ldots, 512$. Since no capillary diffusion is present all methods introduced above essentially behave the same, therefore the *PPS* scheme with (p_n, S_w) as primary unknowns has been selected. In order to enable a quantitative comparison the L^p–norm of the error in the saturation variable,

$$||S_w - S_{wh}||_{L^p} = \left(\int_\Omega |S_w - S_{wh}|^p \, d\mathbf{x} \right)^{\frac{1}{p}}, \tag{3.107}$$

is computed for $p = 1,2$ at the final time $T = 1500[d]$. With the parameters given above the velocity of the front is $v \approx 1.84 \cdot 10^{-6}[m/s]$. A spatial resolution of 64 elements and a temporal resolution of 64 time steps (equidistant) therefore corresponds to a Courant number $C = v\Delta t/\Delta x \approx 0.8$.

Table 3.1 shows the error norms of the saturation for the final time $T = 1500[d]$ using either backward Euler or Crank–Nicolson time–stepping. Both methods used fully upwinding of the mobilities ($\beta = 1$) and a fixed Courant number of 0.8. The convergence rate r is determined as

$$r = \log \left(\frac{||S_w - S_{wh}||_{L^p}}{||S_w - S_{w2h}||_{L^p}} \right) / \log \frac{1}{2}. \tag{3.108}$$

The optimal approximation order of a step function with the ansatz space V_h given here is $O(h)$ in the L^1–norm and $O(h^{\frac{1}{2}})$ in the L^2–norm. The table shows that these approximation orders are almost reached.

Figure 3.4 shows the numerical solutions in comparison to the analytic solution. The top and middle plots show the solutions corresponding to Table 3.1 above. As can be seen, the Crank–Nicolson scheme gives a much better shock resolution. The bottom plot gives results of the backward Euler scheme with

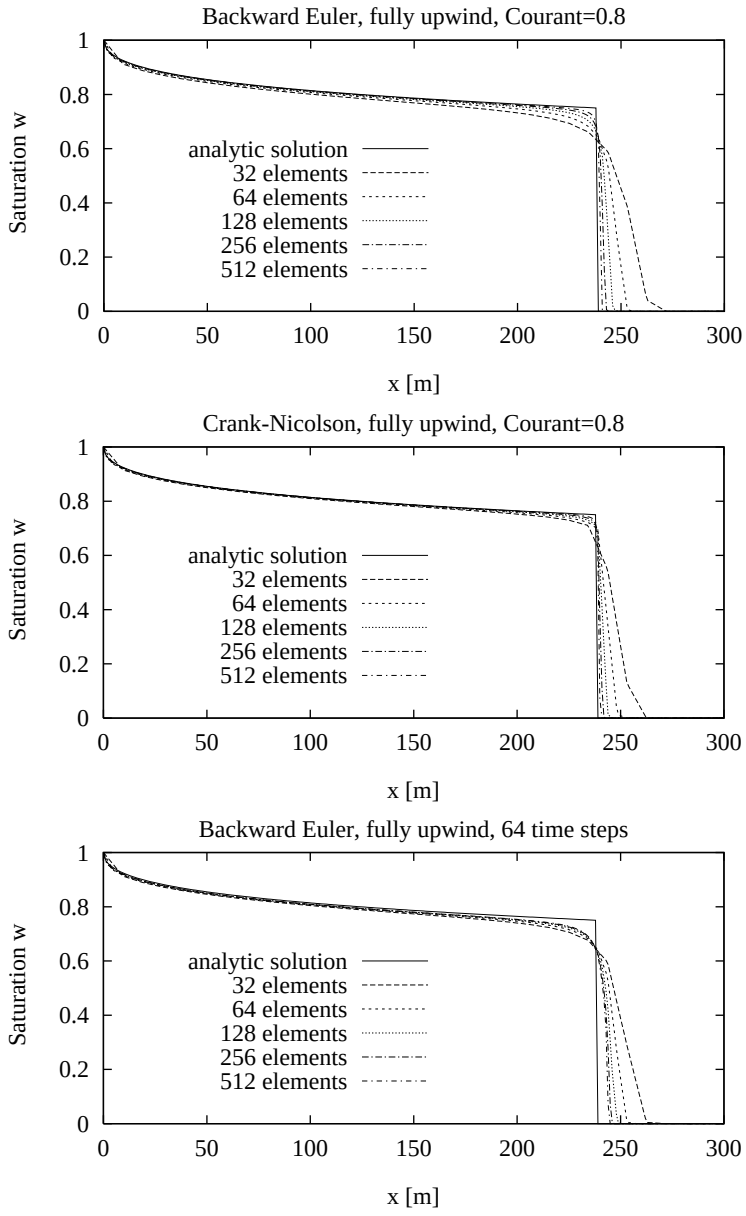


Figure 3.4: Numerical solution of the Buckley–Leverett problem: Backward Euler with Courant number 0.8 (top), Crank–Nicolson with Courant number 0.8 (middle) and backward Euler with fixed number of time steps (bottom).

Table 3.1: Experimental order of convergence for the Buckley–Leverett problem with Brooks–Corey relative permeability and Courant number 0.8.

Method	space	time	L^1		L^2	
	elements	steps	error	rate	error	rate
backward	32	32	$1.54 \cdot 10^1$		$2.21 \cdot 10^0$	
Euler,	64	64	$8.86 \cdot 10^0$	0.80	$1.67 \cdot 10^0$	0.40
fully	128	128	$5.06 \cdot 10^0$	0.81	$1.26 \cdot 10^0$	0.41
upwind	256	256	$2.86 \cdot 10^0$	0.82	$9.44 \cdot 10^{-1}$	0.42
	512	512	$1.61 \cdot 10^0$	0.83	$7.03 \cdot 10^{-1}$	0.43
Crank–	32	32	$9.23 \cdot 10^0$		$1.77 \cdot 10^0$	
Nicolson,	64	64	$5.14 \cdot 10^0$	0.84	$1.34 \cdot 10^0$	0.40
fully	128	128	$2.94 \cdot 10^0$	0.81	$1.01 \cdot 10^0$	0.41
upwind	256	256	$1.68 \cdot 10^0$	0.81	$7.54 \cdot 10^{-1}$	0.42
	512	512	$9.59 \cdot 10^{-1}$	0.81	$5.50 \cdot 10^{-1}$	0.46

fully upwinding and a fixed time step size of $\Delta t = 1500[d]/64$ while the spatial mesh size varies. It can be seen that there is very little improvement in solution quality above a Courant number of 1.6 which corresponds to 128 elements, i. e. errors coming from spatial and temporal discretization are balanced for a Courant number of about 1. Although the backward Euler scheme is unconditionally stable it is not reasonable to take large time steps from an approximation point of view. It should be noted, however, that the shock resolution (for Courant 1) is quite good due to the nonlinearity of the advection term (so–called "self–sharpening effect"). The Crank–Nicolson scheme becomes unstable for a Courant number exceeding 1, the BDF(2) scheme requires even a Courant number below $1/2$ for the problem here!

We conclude that the implicit schemes presented above converge towards the exact solution with rates that can be expected for this type of problem. However, they are not very efficient for the purely hyperbolic case discussed in this subsection. We have chosen implicit schemes since we are interested in the case where capillary diffusion is important. This case is discussed next.

3.8.2 Parabolic Case

The two–phase flow problem is solved in the domain $\Omega = (0, 1.6[m])^2$ and the time interval $(0, 8000[s])$. The parameters are chosen as follows:

Table 3.2: Experimental order of convergence for the McWhorter problem with Brooks–Corey relative permeability and capillary pressure.

Method	space	time	L^1		L^2	
	elements	steps	error	rate	error	rate
backward	32	12	$8.45 \cdot 10^{-2}$		$9.13 \cdot 10^{-2}$	
Euler,	64	24	$5.04 \cdot 10^{-2}$	0.75	$6.19 \cdot 10^{-2}$	0.56
fully	128	48	$2.93 \cdot 10^{-2}$	0.78	$4.03 \cdot 10^{-2}$	0.62
upwind	256	96	$1.71 \cdot 10^{-2}$	0.78	$2.57 \cdot 10^{-2}$	0.65
	512	192	$1.00 \cdot 10^{-2}$	0.77	$1.61 \cdot 10^{-2}$	0.67
backward	32	12	$2.56 \cdot 10^{-2}$		$4.05 \cdot 10^{-2}$	
Euler,	64	24	$1.33 \cdot 10^{-2}$	0.94	$2.44 \cdot 10^{-2}$	0.73
central	128	48	$7.21 \cdot 10^{-3}$	0.88	$1.45 \cdot 10^{-2}$	0.75
differences	256	96	$4.22 \cdot 10^{-3}$	0.77	$8.56 \cdot 10^{-3}$	0.76
	512	192	$2.74 \cdot 10^{-3}$	0.62	$4.99 \cdot 10^{-3}$	0.78

fluids:
$\rho_w = \rho_n = 1000[kg/m^3]$
$\mu_w = \mu_n = 0.001[Pa\ s]$

rock:
$\Phi = 0.3$
$K = 10^{-10}[m^2]$

residual saturation:
$S_{wr} = S_{nr} = 0$

relative permeability:
Brooks–Corey, $\lambda = 2.0$

capillary pressure:
Brooks–Corey with $\lambda = 2$ and $p_d = 5000[Pa]$

boundary conditions:
$\phi_\alpha = 0$ for $y = 0$ and $y = 1.6[m]$
$p_n = 2 \cdot 10^5[Pa]$, $S_w = 1$ for $x = 0$
$\phi_n = 0$, $S_w = 0$ for $x = 1.6[m]$

initial conditions:
$S_w(\mathbf{x}, 0) = 0$ for $\mathbf{x} \in \Omega$

The parameters correspond to the example given at the end of Subs. 2.4.3. The domain Ω is discretized with $K \times 2$ quadrilateral elements, where $K = 32, 64, \ldots, 512$. The number of time steps varies from 12 to 192 (time steps are equidistant).

Table 3.2 lists the L^1 and L^2–norms of the error in the saturation variable at the final time $T = 8000[s]$. The *PPS* method with (p_n, S_w) as primary unknowns has been used with backward Euler time–stepping and fully upwinding ($\beta = 1$) as well as central differencing of mobilities ($\beta = 0$). The solutions for both variants are shown graphically in Figure 3.5 top and middle. Since the problem is diffusion–dominated central differencing can be used which leads to a better approximation in the smooth parts of the solution. The rates, however, are about equal due to the lack of regularity in the solution. The bottom plot in Figure 3.5 shows the numerical solution when the number of time steps is fixed to 24 and only the spatial mesh size is varied. As can be seen, very large time steps can be taken. It should be noted that the free boundary moves very fast

at the beginning of the simulation since we have that $S(x,t) = S(xt^{-\frac{1}{2}})$. An explicit scheme would require excessively small time steps at the beginning. This behavior also suggests that the time step size should be chosen adaptively.

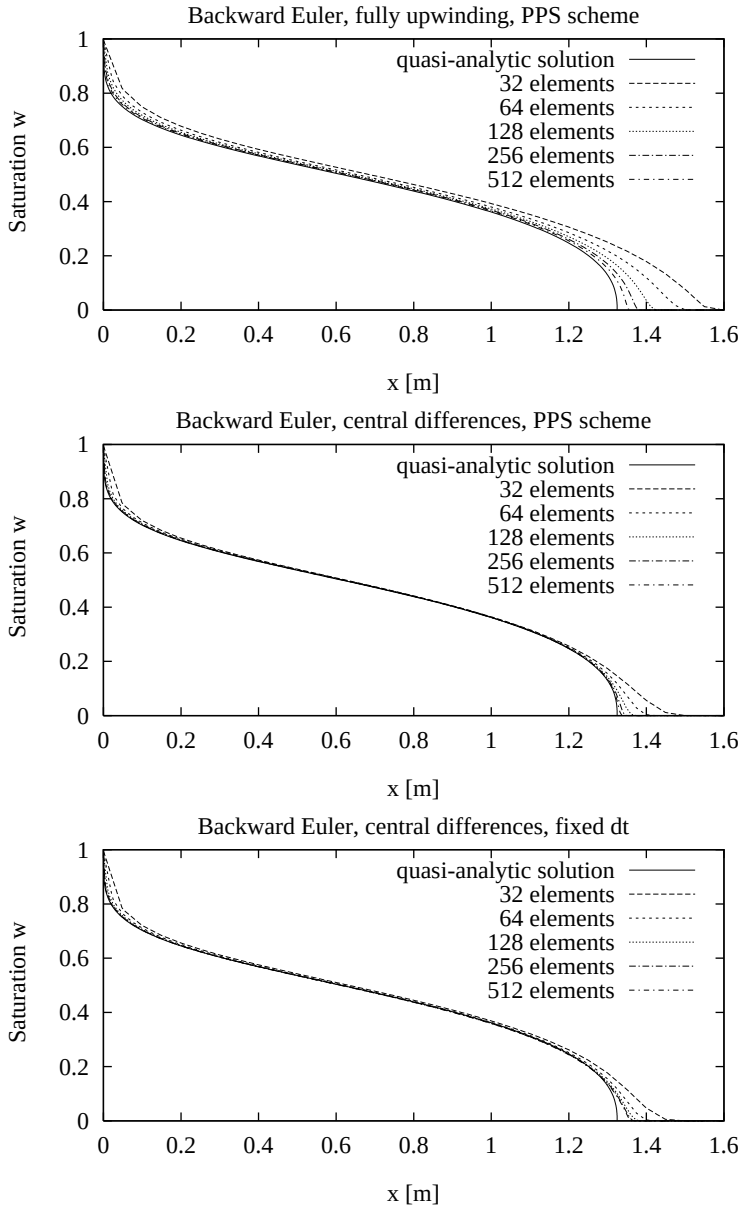


Figure 3.5: Numerical solution of the McWhorter problem: Backward Euler with fully upwinding and fixed $\Delta t/\Delta x$ (top), backward Euler with central differencing and fixed $\Delta t/\Delta x$ (middle) and backward Euler with fixed number of time steps (bottom).

4

Solution of Algebraic Equations

This chapter concentrates on the resolution of the algebraic equations arising within each time step of the fully implicit/fully coupled solution procedure.

After a description of the multigrid mesh structure the inexact Newton method will be reviewed shortly. Then we will turn our attention to the resolution of the linear systems arising within each Newton step. The main objective of this chapter is the construction of an appropriate multigrid method for these systems. Finally, the last section of this chapter is devoted to the parallel implementation of the multigrid solver.

4.1 Multigrid Mesh Structure

The nonlinear and linear solvers to be described in this chapter utilize a multigrid mesh structure to accelerate the solution process. This multigrid mesh structure denoted by

$$E_0, E_1, \ldots, E_J \tag{4.1}$$

is constructed from an intentionally coarse mesh E_0 (generated by hand or an initial mesh generator) by *regular* subdivision of each element. Figure 4.1 illustrates the subdivision process for all six element types. The stable refinement of tetrahedra is based on the method of Bey (1995).

The set of vertices belonging to mesh E_l is written as V_l. The number of elements on level l is denoted by K_l and the number of vertices by N_l. In the

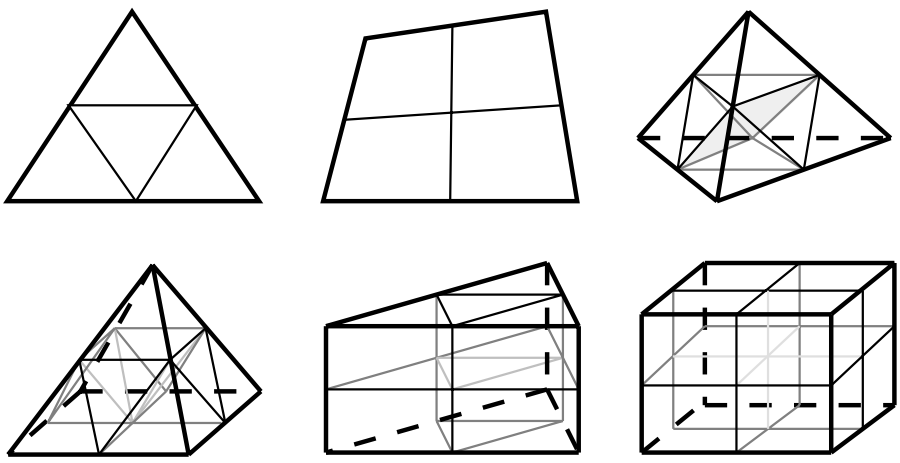


Figure 4.1: Regular refinement rules.

multigrid case the mesh size index h is replaced by the level index l. Moreover, the mesh size index is omitted where not absolutely necessary.

Local mesh refinement is also possible. In that case we prefer conforming meshes without hanging nodes. This is achieved by introducing additional *irregular* refinement rules. Elements produced by irregular refinement rules are not allowed to be refined. If further refinement is required they are replaced by regularly refined elements, for details we refer to (Bank, Sherman, and Weiser 1983; Bey 1995) in the sequential case and (Bastian 1996; Lang 1999) for a parallel implementation.

4.2 Inexact Newton Method

4.2.1 ALGORITHM

The discrete schemes derived in Chapter 3 all lead to a large set of nonlinear algebraic equations

$$\mathbf{F}(\mathbf{z}) = \mathbf{0} \tag{4.2}$$

to be solved per time step. The vector $\mathbf{z}$ contains pressure and saturation unknowns in the following ordering

$$\mathbf{z} = (\mathbf{p}_{w,1}, \ldots, \mathbf{p}_{w,N}, \mathbf{S}_{n,1}, \ldots, \mathbf{S}_{n,N})^T \tag{4.3}$$

which is referred to as *equation–wise ordering*. The vector function $\mathbf{F}$ has components

$$\mathbf{F} = (\mathbf{F}_{w,1}, \ldots, \mathbf{F}_{w,N}, \mathbf{F}_{n,1}, \ldots, \mathbf{F}_{n,N})^T \tag{4.4}$$

which, e. g. in the case of the *PPS*–method and a one step θ–scheme are given by

$$\mathbf{F}_\alpha = \mathbf{M}_\alpha^{n+1} - \mathbf{M}_\alpha^n + \Delta t^n \theta \left(\mathbf{A}_\alpha^{n+1} + \mathbf{Q}_\alpha^{n+1}\right) + \Delta t^n (1-\theta)\left(\mathbf{A}_\alpha^n + \mathbf{Q}_\alpha^n\right), \tag{4.5}$$

see Eq. (3.96). Actually, those coefficients in $\mathbf{z}$ corresponding to Dirichlet boundary conditions are not unknown and the number of nonlinear equations is reduced correspondingly. In the implementation (and description of it) it is more convenient to keep these components as "unknowns" and to extend $\mathbf{F}$ by an appropriate number of trivial equations.

The linearization (Jacobian) A of $\mathbf{F}$ at the linearization point $\mathbf{z}$ is the matrix with entries

$$(A(\mathbf{z}))_{ij} = \frac{\partial \mathbf{F}_i}{\partial \mathbf{z}_j}(\mathbf{z}). \tag{4.6}$$

The entries of the Jacobian are either computed analytically or by numerical differentiation:

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{z}_j}(\mathbf{z}) = \frac{\mathbf{F}_i(\mathbf{z} + \Delta \mathbf{z}_j \mathbf{e}_j) - \mathbf{F}_i(\mathbf{z})}{\Delta \mathbf{z}_j} + O(\Delta \mathbf{z}_j) \tag{4.7}$$

with $\mathbf{e}_j$ the j–th unit vector, $\Delta\mathbf{z}_j = \varepsilon(1+|\mathbf{z}_j|)$ and $\varepsilon \in [10^{-8}, 10^{-6}]$.

We are now in a position to state the inexact Newton algorithm.

ALGORITHM 4.1 The following algorithm **inewton** solves the nonlinear system $\mathbf{F}(\mathbf{z}) = \mathbf{0}$ to accuracy ε_{nl} starting from the initial guess $\mathbf{z}$.

```
      inewton ( F, z, ε_nl )
      {
(1)       κ = 0; z^0 = z;
(2)       while ( ||F(z^κ)||_2 ≥ ε_nl ||F(z^0)||_2 ) {
(3)           Choose ε_lin^κ ∈ (0,1];
              Find s^κ such that
(4)               ||F(z^κ) + A(z^κ)s^κ||_2 ≤ ε_lin^κ ||F(z^κ)||_2;
(5)           Choose λ^κ ∈ (0,1];
(6)           z^{κ+1} = z^κ + λ^κ s^κ;
(7)           κ = κ + 1;
          }
      }
```

Superscript κ denotes the iteration index and $\|.\|_2$ is the Euclidean vector norm.

Two strategies for the selection of the initial guess are available. The first strategy simply uses the converged value of the preceding time step. The second strategy uses the multigrid hierarchy to compute a better initial guess. The nonlinear problem $\mathbf{F}(\mathbf{z}_0) = \mathbf{0}$ is solved on the coarsest mesh using the value of the preceding time step restricted to the coarsest mesh (straight injection is used here) as initial guess. Then $\mathbf{z}_0$ is interpolated to mesh level 1 to be used as initial guess and the process is repeated until the finest level is reached. In the linear case this procedure is called *nested iteration* or the *full multigrid procedure*. Nested iteration is especially effective in the case of large time steps. The auxiliary nonlinear coarse grid problems need not be solved as accurately as the fine grid equations.

Steps (3),(4) in algorithm **inewton** compute an approximation of the Newton update $\mathbf{s}^\kappa$ which is the solution of the linear system

$$A(\mathbf{z}^\kappa)\mathbf{s}^\kappa = -\mathbf{F}(\mathbf{z}^\kappa). \tag{4.8}$$

The accuracy ε_{lin}^κ, also called a *forcing term*, required in the solution of this linear system is chosen as

$$\varepsilon_{lin}^\kappa = \begin{cases} \varepsilon_0 & \kappa = 0 \\ \min\left(\varepsilon_0, \left(\frac{\|\mathbf{F}(\mathbf{z}^\kappa)\|_2}{\|\mathbf{F}(\mathbf{z}^{\kappa-1})\|_2}\right)^2\right) & \kappa > 0 \end{cases}. \tag{4.9}$$

This choice allows for an inaccurate solution in the first Newton steps while ensuring quadratic convergence in the final steps. For a comparison of forcing

term strategies we refer to (Eisenstat and Walker 1996). The safety factor ε_0 should not be chosen too large in the problems considered here. This is due to the fact that the convergence of the linear solver may not be monotone in the sense that all saturation values are in the interval $[0,1]$. We typically use $\varepsilon_0 = 10^{-4}$ in the numerical computations reported below.

Since Newton's method converges only in a sufficiently close neighborhood of the solution a damping strategy is needed to achieve global convergence. Step (5) implements a simple line search strategy where the damping factor λ^κ is chosen as the largest value in the set $\{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ such that

$$\|\mathbf{F}(\mathbf{z}^\kappa + \lambda^\kappa \mathbf{s}^\kappa)\|_2 \leq \left(1 - \frac{1}{4}\lambda^\kappa\right) \|\mathbf{F}(\mathbf{z}^\kappa)\|_2. \tag{4.10}$$

For a theoretical motivation of this strategy we refer to (Braess 1992).

4.2.2 Linearized Operator for *PPS*–Scheme

In order to get more insight into the structure of the Jacobian system for the fully coupled two–phase flow problem we consider it as a discretization of the linearized continuous equations.

We set

$$p_w = \tilde{p}_w + \delta p_w, \qquad S_n = \tilde{S}_n + \delta S_n, \tag{4.11}$$

where $\tilde{p}_w, \tilde{S}_n$ is the linearization point. A system of *linear* partial differential equations for an approximation of the updates $\delta p_w, \delta S_n$ is obtained by using Taylor expansion of the nonlinearities and ignoring all terms that are more than linear in the updates. For the *PPS*–method these equations are given in the incompressible case by

$$\begin{pmatrix} -\nabla\cdot\{\lambda_w K\nabla\delta p_w\} & -\frac{\partial(\Phi\delta S_n)}{\partial t} - \nabla\cdot\{\mathbf{w}_w\delta S_n\} \\ -\nabla\cdot\{\lambda_n K\nabla\delta p_w\} & \frac{\partial(\Phi\delta S_n)}{\partial t} + \nabla\cdot\{\mathbf{w}_n\delta S_n + \lambda_n p_c' K\nabla\delta S_n\} \end{pmatrix} = \text{rhs}. \tag{4.12}$$

with the velocities

$$\mathbf{w}_w = -\lambda_w' K\left(\nabla\tilde{p}_w - \rho_w\mathbf{g}\right), \tag{4.13a}$$

$$\mathbf{w}_n = -\lambda_n' K\left(\nabla\tilde{p}_n - \rho_n\mathbf{g}\right) + \lambda_n K\nabla(p_c'). \tag{4.13b}$$

All coefficient functions in (4.12) are to be evaluated at the linearization point $\tilde{p}_w, \tilde{S}_n$. From the definition of $\mathbf{F}$, $\mathbf{z}$ and the Jacobian A it is evident that the Jacobian has a 2×2 block structure

$$A = \begin{pmatrix} A_{ww} & A_{wn} \\ A_{nw} & A_{nn} \end{pmatrix}. \tag{4.14}$$

The 2×2 structure in (4.14) directly corresponds to that in (4.12), i. e. for $h \to 0$, A_{ww} is a discretization of the term $-\nabla\cdot\{\lambda_w K\nabla\delta p_w\}$, etc. . From this correspondence we can deduce some qualitative properties of the linear system to be solved in each Newton step.

The case $\tilde{S}_n = 1$. For $\tilde{S}_n = 1$ in Ω we have $k_{rw} = 0$ and consequently the whole block A_{ww} vanishes. Of course, selected rows of A_{ww} vanish if $\tilde{S}_n = 1$ locally. In this case point–wise iterative schemes cannot be applied.

Variability of coefficients . Remembering that typical shapes of the relative permeability functions are e. g. $k_{rn}(S_n) = S_n^4$ and that the solution may have steep gradients (even shocks) we see that the coefficients in all blocks vary strongly with spatial position. The absolute permeability K may be anisotropic and also strongly variable in magnitude with spatial position. Finally the function p_c' in the *nn*–block depends on the solution and therefore on position.

Convection vs. diffusion . The *nn*–block is the discretization of a time–dependent convection–diffusion operator. Depending on the parameters it may be either convection or diffusion dominated. The *ww*–block is always the discretization of an elliptic operator. This corresponds directly to the characterization of the two–phase flow equations as a coupled system of an elliptic and a parabolic/hyperbolic equation.

4.3 Multigrid Solution of Linear Systems

4.3.1 Introduction

This section treats the resolution of large and sparse systems of linear equations

$$A\mathbf{z} = \mathbf{b} \tag{4.15}$$

where in our application A is the Jacobian arising in the fully coupled Newton solution of the two–phase flow problem and $\mathbf{b}$ is the nonlinear defect. For illustrative purposes we will also frequently refer to the case where A is the discretization of a linear scalar model problem of the form $\nabla \cdot \{\mathbf{r}C - D\nabla C\}$.

Let N be the dimension of the system (4.15). Direct resolution of (4.15) with Gaussian elimination requires $O(N^3)$ arithmetical operations, see e. g. (Golub and Van Loan 1989). Taking the sparsity structure into account (i. e. avoiding fill–in and computation with zero elements) the operation count can be reduced for two–dimensional problems to $O(N^2)$ for banded Gaussian elimination or $O(N^{1.5})$ for nested disection. The corresponding numbers for three–dimensional problems are $O(N^{2.33})$ and $O(N^2)$, see e. g. (Axelsson and Barker 1984). In two space dimensions direct methods are very efficient up to several thousand unknowns. In three dimensions, however, direct resolution quickly becomes infeasible.

For large problems (we will handle millions of unknowns) iterative methods are the only choice. Starting with an initial guess $\mathbf{z}^0$, iterative methods for the resolution of (4.15) produce a sequence of iterates $\mathbf{z}^1, \mathbf{z}^2, \ldots$ that (hopefully) converges to the exact solution $\mathbf{z}$.

In the case of *relaxation methods* the idea is to split the matrix A into

$$A = M - N \tag{4.16}$$

where M should be a approximation of A that is easy to invert. The iteration is then given by

$$\mathbf{z}^{\mu+1} = \mathbf{z}^{\mu} + M^{-1}(\mathbf{b} - A\mathbf{z}^{\mu}). \tag{4.17}$$

The quantity $\mathbf{d}^{\mu} = \mathbf{b} - A\mathbf{z}^{\mu}$ is called the *defect* in step μ. Typical choices for M are the diagonal of A (Jacobi method) or the lower triangle of A (Gauß-Seidel method). Another popular choice is

$$A = LU - N \tag{4.18}$$

where L and U are lower and upper triangular matrices derived from A by *incomplete* LU decomposition without introduction of additional fill–in. Thus L and U have the same sparsity pattern as A. A measure of the speed of convergence of an iterative method is given by

$$\|\mathbf{z} - \mathbf{z}^{\mu+1}\| \leq \rho \|\mathbf{z} - \mathbf{z}^{\mu}\| \tag{4.19}$$

with a suitable norm $\|.\|$, e. g. Euclidean norm, maximum norm or energy norm if A is symmetric positive definite. In order to reduce the error by a factor of ε at most $\mu = \lceil \log\varepsilon / \log\rho \rceil$ steps are required.

For methods of type (4.17) the convergence factor typically has the form

$$\rho = 1 - O(h^2) \tag{4.20}$$

leading to a fourfold increase in the number of iterations to achieve a fixed error reduction when the mesh size h is halfened. Simple modifications of the basic scheme (4.17) (the SOR method) are able to reduce the convergence factor to

$$\rho = 1 - O(h) \tag{4.21}$$

but rely on a problem–dependent parameter that is, in general, not known. Classical textbooks for relaxation methods are (Varga 1962; Young 1971), a newer source containing many more methods is the excellent monograph by Hackbusch (1994). The arithmetical complexity of methods with property (4.20) is $O(N^2)$ in two and $O(N^{1.67})$ in three dimensions. Methods with property (4.21) have a complexity of $O(N^{1.5})$ and $O(N^{1.33})$ respectively. The convergence factor ρ certainly depends on the type of problem to be solved. Convergence of these basic iterative methods can be shown for symmetric positive definite matrices, diagonally dominant matrices or so–called M–matrices, see e. g. (Hackbusch 1994). Unfortunately non of these theories is able to ensure the convergence for the Jacobian systems arising in the fully coupled Newton solution of the two–phase flow problem.

Another large class of methods for the solution of (4.15) are *Krylov subspace methods*. The basic idea is to choose the correction $\mathbf{s}^\mu$ ($\mu > 1$) to the initial guess $\mathbf{z}^0$ from the Krylov subspace $\{\mathbf{d}^0, A\mathbf{d}^0, \ldots, A^{\mu-1}\mathbf{d}^0\}$ in such a way that the error $\mathbf{e}^\mu = \mathbf{z} - \mathbf{z}^0 - \mathbf{s}^\mu$ is minimized in some way, e. g. in the energy norm in case of the conjugate gradient method. A good description of these algorithms is given in (Barrett et al. 1994). The methods can be accelerated substantially by using a preconditioner, which is a basic iterative method as discussed above or the multigrid method to be discussed below. By using optimally damped SSOR the computational complexity of such methods can be as low as $O(N^{1.25})$ and $O(N^{1.17})$ in two and three space dimensions, cf. (Axelsson and Barker 1984). For unsymmetric matrices A the minimization over the Krylov subspace cannot be done as cheaply as in the symmetric case. Several methods are known, each sacrificing another property, see (Barrett et al. 1994) for details. We will use the BiCGSTAB method of Van der Vorst (1992) as an accelerator for the multigrid method in this work.

The third class of iterative methods to be mentioned here is the class of multigrid (or multilevel) methods. By studying the convergence behavior of the basic iterative scheme (4.17) applied to the model problem $-\Delta C = q$ one observes that highly oscillating errors are damped much more quickly than slowly oscillating errors. The slow convergence stated in (4.20) is due to these low frequency error components. The idea is now to combine a basic iterative method with a so–called coarse grid correction which reduces the low frequency error components effectively. Details of this procedure will be given below. For an introduction to multigrid methods we refer to (Hackbusch 1985; Wesseling 1992; Briggs 1987).

For elliptic model problems it can be shown that the convergence factor ρ of the multigrid method is *independent* of the mesh size h. The computational complexity is therefore $O(N)$ and thus optimal. As most other iterative methods, multigrid does not converge for arbitrary matrices A. Rigorous convergence proofs are available for elliptic model problems, possibly with low order perturbations, see (Hackbusch 1985; Xu 1992; Bramble 1993) or systems like the Stokes equation, see (Verfürth 1988; Wittum 1990).

Unlike the other methods discussed above the multigrid method needs auxiliary matrices $A_0, A_1, \ldots, A_{J-1}$ in addition to the system matrix $A = A_J$. The construction of these auxiliary matrices will be discussed below. From an implementation point of view the interface of the linear solver with the discretization part of the computer program is much more involved when multigrid methods are used.

For the specific case of the two–phase problem multigrid methods have been applied to the solution of the pressure equation within a decoupled (IMPES) type of approach, see e. g. (Scott 1985; Dendy Jr. 1987). This can be considered a “standard” application of multigrid since only a scalar elliptic problem has to be solved (although with possibly strongly varying or anisotropic coefficients).

Multigrid applied to the fully implicit/fully coupled type of approach has been studied in (Brakhagen and Fogwell 1990; Molenaar 1995). Both investigations

have been restricted to the incompressible case on structured meshes in two space dimensions. In the remaining part of this section we will describe the components of our multigrid algorithm in detail.

4.3.2 Standard Multigrid Algorithm

We now describe the standard method when A is the discretization of a scalar linear and elliptic model problem, e. g. $\nabla \cdot \{\mathbf{r}C - D\nabla C\} = q$. Let a hierarchy of meshes $\{E_l\}_{l=0}^{J}$ as described in Sect. 4.1 be given. The discretized equations on each mesh level are then given by

$$A_l \mathbf{z}_l = \mathbf{b}_l, \qquad l = 0, \ldots, J. \tag{4.22}$$

The dimension of these systems is N_l. Furthermore we need *grid transfer operators* R_l, P_l which are linear mappings of appropriate dimension:

$$R_l : \quad \mathbb{R}^{N_l} \rightarrow \mathbb{R}^{N_{l-1}} \qquad \text{(Restriction)} \tag{4.23a}$$

$$P_l : \quad \mathbb{R}^{N_{l-1}} \rightarrow \mathbb{R}^{N_l} \qquad \text{(Prolongation)} \tag{4.23b}$$

Finally let S denote any of the relaxation methods discussed above. S is called a *smoother* in multigrid notation. We are now able to formulate the standard multigrid algorithm.

Algorithm 4.2 The following algorithm **mgc** executes a single iteration of the standard multigrid method with finest level l applied to the current iterate $\mathbf{z}_l$.

mgc (l, $\mathbf{z}_l$, $\mathbf{b}_l$)
{
  if (l == 0) $\mathbf{z}_0 = A_0^{-1}\mathbf{b}_0$;
  else {
    Apply ν_1 *iterations of* S *to* $A_l\mathbf{z}_l = \mathbf{b}_l$;
    $\mathbf{d}_l = \mathbf{b}_l - A_l\mathbf{z}_l$;
    $\mathbf{d}_{l-1} = R_l\mathbf{d}_l$;
    $\mathbf{s}_{l-1} = 0$;
    for ($g = 1, \ldots, \gamma$) **mgc**($l-1$,$\mathbf{s}_{l-1}$,$\mathbf{d}_{l-1}$);
    $\mathbf{s}_l = P_l\mathbf{s}_{l-1}$;
    $\mathbf{z}_l = \mathbf{z}_l + \mathbf{s}_l$;
    Apply ν_2 *iterations of* S *to* $A_l\mathbf{z}_l = \mathbf{b}_l$;
  }
}

The parameters ν_1, ν_2 are the number of pre– and postsmoothing steps. Typically they are in the range $1, \ldots, 3$. The parameter γ controls the cycle form. We will only use $\gamma = 1$, called a V–cycle, in the numerical experiments below.

The canonical way to define the prolongation operator (matrix) P_l is via finite element interpolation:

$$(P_l \mathbf{s}_{l-1})_i = \sum_{j=1}^{N_{l-1}} \mathbf{s}_{l-1,j} \varphi_{l-1,j}(\mathbf{x}_i), \tag{4.24}$$

where $\varphi_{l-1,j}$ is the finite element basis function corresponding to vertex j on level $l-1$. Since the support of the basis functions is local P_l is a very sparse rectangular matrix. The standard choice for the restriction operator R_l is

$$R_l = P_l^T \tag{4.25}$$

in the case of a finite element or finite volume discretization. For the coarse grid matrices A_l, $l < J$, two standard choices exist. They are either computed by discretization of the continuous problem (which we assumed up to now) or via the *Galerkin* coarse grid operator approach as

$$A_{l-1} = R_l A_l P_l. \tag{4.26}$$

Various advantages and disadvantages of these standard components will now be discussed in more detail.

4.3.3 Robustness

The convergence rate of the standard multigrid method applied to the model problem $-\Delta C = q$ can be shown to be independent of the mesh size parameter h. However, when applied to the more complicated model problem $\nabla \cdot \{\mathbf{r}C - D\nabla C\} = q$ it is not independent of the coefficients $\mathbf{r}$ and D. A multigrid method is considered to be robust if it converges independent of other "bad" parameters in addition to the mesh size h.

Three types of scalar model problems are typically discussed in this respect:

$$-\nabla \cdot \{d(\mathbf{x})\nabla C\} = q, \qquad d \text{ discontinuous with position}, \tag{4.27a}$$

$$-\nabla \cdot \{D(\mathbf{x})\nabla C\} = q, \qquad D \text{ anisotropic tensor}, \tag{4.27b}$$

$$\nabla \cdot \{\mathbf{r}C - \varepsilon\nabla C\} = q, \qquad \|\mathbf{r}\| \gg \varepsilon, \text{ dominating convection}. \tag{4.27c}$$

Most work of multigrid practitioners is concerned with making the method work with one or more of these problems. With a few exceptions these methods are motivated heuristically and no rigorous proofs are available. We will now give a short overview of the different approaches.

Problems of type (4.27a) are called *interface problems*. The diffusion coefficient d is supposed to be discontinuous by orders of magnitude across internal boundaries of the domain. If these internal boundaries are resolved by the coarsest mesh multigrid converges well and almost optimal convergence estimates are available in the two–dimensional case, cf. (Bramble et al. 1991). In three space

dimensions the situation is more involved and multigrid convergence may deteriorate for certain coefficient distributions, see Dryja et al. (1996) for details.

In many practical situations the discontinuities of the diffusion coefficient are not aligned with coarse grid edges (faces). In this case the standard multigrid algorithm with discretized coarse grid operators does not converge. Several remedies have already been developed in (Alcouffe et al. 1981; Kettler 1982), see also (Hackbusch 1985). These approaches use specially designed prolongation operators constructed from the stiffness matrix A_l and the Galerkin coarse grid operator. For newer approaches we refer to (Wagner, Kinzelbach, and Wittum 1997) who use a Schur–Complement coarse grid operator and (Molenaar 1994).

Problem (4.27b) with $D = \mathrm{diag}(1, \varepsilon)$ is called the *anisotropic model problem*. It belongs to the class of singular perturbation problems since the type of the equation changes from elliptic to parabolic when $\varepsilon = 0$. The convergence rate of standard multigrid with point–wise Jacobi or Gauß–Seidel smoothing quickly deteriorates when ε gets smaller or larger than one. One remedy is to use block–line smoothers or a modified ILU smoother with appropriate ordering of the unknowns, for theoretical results see (Wittum 1989). Another remedy is to prevent mesh coarsening in the direction of weak coupling (e. g. the *y*–direction if $\varepsilon \ll 1$) which is called *semi–coarsening*. The robust smoother approach is hardly extendable to three space dimensions since the solution of two–dimensional subproblems is required within the smoother. Semi–coarsening works also in the three–dimensional case and is extended to unstructured meshes in the context of algebraic multigrid methods (see below).

Finally, problem (4.27c) is may be the most challenging of all. In case the flow field $\mathbf{r}$ has no recirculation zones the problem with $\varepsilon = 0$ (pure convection) results in a lower triangular matrix if the unknowns are ordered properly and an appropriate upwind discretization is used. Various techniques have been devised to construct robust smoothers in the case with recirculation zones, see (Hackbusch 1997; Rentz–Reichert 1996; Hackbusch and Probst 1997; Bey 1997; Bey and Wittum 1997). Alternatively, one can try to improve the coarse grid correction. A crucial property in this respect is to inherit the stability of the fine grid matrix (achieved through an upwind discretization) to the coarse grid problems. This requires carefully constructed prolongation and restriction operators in connection with the Galerkin approach. Standard prolongation and restriction does not work. Recently, robust methods with improved coarse grid correction have been suggested in (Reusken 1995a; Reusken 1996).

A class of multigrid methods that aim at solving all of the problems (4.27a–4.27c) are the *algebraic multigrid methods*. They are very attractive from a practical point of view since only the fine grid problem is required as input. The pioneering work in this direction is (Ruge and Stüben 1987). Agglomeration type multigrid methods have been developed in (Vaněk, Mandel, and Brezina 1996; Braess 1995; Raw 1996). New approaches based on incomplete LU factorizations have been presented by (Reusken 1995b; Bank and Wagner 1998).

4.3.4 SMOOTHERS FOR SYSTEMS

We want to apply the multigrid method to the Jacobian system arising from the fully implicit discretization of the two–phase flow problem. According to (4.14) the system matrix A has a 2×2 block structure in the equation–wise ordering. Since some or all rows of the A_{ww}–block may vanish point–wise smoothers are not applicable.

Well–defined smoothers are obtained by using the *point–block ordering* where all unknowns corresponding to a vertex of the mesh are grouped together. This may be written as

$$\bar{\mathbf{z}} = (\mathbf{p}_{w,1}, \mathbf{S}_{n,1}, \ldots, \mathbf{p}_{w,N}, \mathbf{S}_{n,N})^T = Q\mathbf{z}, \tag{4.28}$$

where Q is a permutation matrix performing the reordering. The equivalent, transformed system of equations is then written as

$$\bar{A}\bar{\mathbf{z}} = \bar{\mathbf{b}} \tag{4.29}$$

with $\bar{A} = QAQ^T$, $\bar{\mathbf{z}} = Q\mathbf{z}$ and $\bar{\mathbf{b}} = Q\mathbf{b}$. The permuted matrix $\bar{A}$ has a $N \times N$ block structure

$$\bar{A} = \begin{pmatrix} \bar{A}_{11} & \ldots & \bar{A}_{1N} \\ \vdots & & \vdots \\ \bar{A}_{N1} & \ldots & \bar{A}_{NN} \end{pmatrix}, \tag{4.30}$$

where each block is 2×2.

Dirichlet boundary conditions are treated by replacing the corresponding row of the linear system by a trivial equation. Thus $\bar{A}$ has always dimension $2N$. It turns out that the diagonal blocks $\bar{A}_{ii}$ are always regular except at boundary vertices where a boundary condition of the following form is prescribed:

$$\rho_w \mathbf{u}_w \cdot \mathbf{n} = \phi_w, \qquad S_n(\mathbf{x}, t) = 1 \tag{4.31}$$

(this assumes (p_w, S_n) as unknowns). Obviously, if $S_n = 1$ the flux of the wetting phase over this boundary is zero and cannot be prescribed. Therefore boundary conditions of type (4.31) do not occur.

As a smoother in our multigrid procedure we use block variants of the Jacobi, Gauß–Seidel and ILU iterations with respect to the blocking given in (4.30). As has been indicated, the Jacobi and Gauß–Seidel schemes are always well defined but convergence of all schemes and existence of the ILU decompositions cannot be proven in general for the matrices given here.

A general approach for the construction of smoothers for systems of equations are the transforming smoothers of Wittum (1990). With the point–block diagonal matrix

$$\bar{D} = \operatorname{diag}(\bar{A}_{11}, \ldots, \bar{A}_{NN}) \tag{4.32}$$

one could use

$$\hat{A} = Q^T \bar{D}^{-1} Q \tag{4.33}$$

as a left transformation. Point–wise iteration could then be applied to the transformed system $\hat{A}A$. The resulting smoothers are very similar to the point–block smoothers defined above, in fact Jacobi and Gauß–Seidel variants are identical. This type of smoother is used as preconditioner in (Dawson et al. 1997).

Finally we note that $\hat{A}A$ becomes block triangular when $A_{ww} = 0$ showing the effectiveness of the transformation in this case. In the numerical experiments below only point–block smoothers will be used.

4.3.5 Truncated Restriction

High spatial variability or even discontinuity of the absolute permeability tensor often occurs in single and multiphase flow applications. Furthermore, the relative permeability and capillary pressure functions also give rise to high spatial variability of the coefficients of the second order terms as has been discussed in Subs. 4.2.2. In view of the discussion on interface problems in Subs. 4.3.3 one should choose carefully designed grid transfer operators in connection with a Galerkin or Schur–Complement coarse grid operator. It is, however, not clear how the stability of the coarse grid matrices can be ensured in case of the fully coupled solution of the two–phase flow problem, especially in the hyperbolic case. All approaches for interface problems mentioned above were only concerned with scalar problems.

We were therefore interested in using the discretized equations on the coarse grids for stability reasons. Then, however, the standard multigrid method cannot handle large permeability variations that are not aligned with the coarsest grid. In order to understand this behavior we consider the following simple model problem in one space dimension

$$-\frac{d}{dx}\left(d(x)\frac{dC}{dx}\right) = q \qquad \text{in } \Omega = (0,1) \tag{4.34a}$$

$$C = 0 \qquad \text{on } \partial\Omega \tag{4.34b}$$

with the diffusion coefficient d given by

$$d(x) = \begin{cases} 1 & x < \theta \\ \varepsilon \ll 1 & \text{else} \end{cases} . \tag{4.35}$$

A finite volume discretization of (4.34) on an equidistant mesh of size h yields the tridiagonal system

$$-\frac{d_{i-\frac{1}{2}}}{h}\mathbf{z}_{i-1} + \frac{d_{i-\frac{1}{2}} + d_{i+\frac{1}{2}}}{h}\mathbf{z}_i - \frac{d_{i+\frac{1}{2}}}{h}\mathbf{z}_{i+1} = hq_i, \quad 0 < hi < 1, \tag{4.36}$$

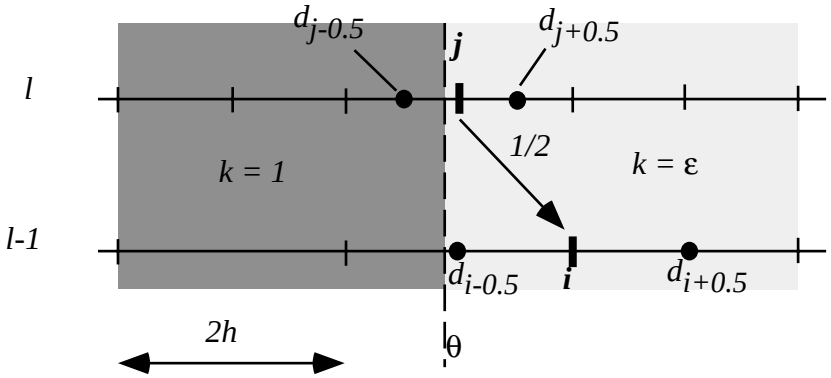


Figure 4.2: One–dimensional interface problem.

where $d_{i-\frac{1}{2}}$, $d_{i+\frac{1}{2}}$ denotes pointwise evaluation of (4.35) half way between grid points.

Let us now consider a standard two–grid algorithm with discretized coarse grid operator for solving (4.36). Specifically, we consider vertices j on the fine grid and i on the coarse grid that happen to lie in the vicinity of the interface as shown in Fig. 4.2.

The situation shown in the figure is such that for the fine mesh vertex j we have $d_{j-\frac{1}{2}} = 1$, $d_{j+\frac{1}{2}} = \varepsilon$ and for the coarse mesh vertex i we have $d_{i-\frac{1}{2}} = d_{i+\frac{1}{2}} = \varepsilon$. During coarse grid correction a defect of order $O(1)$ is computed at fine mesh vertex j which is restricted with factor $1/2$ to the right hand side of the coarse grid equation i. The coarse grid solve then essentially computes a correction of order $O(\varepsilon^{-1})$ at vertex i which results in the divergence of the standard multigrid algorithm for sufficiently small ε.

The proposed remedy is simple: We just have to prevent the restriction of the defect from a vertex with large diagonal entry to a vertex with small diagonal entry. In the following, we devise a purely algebraic way to do this. The result will be a modified restriction operator to be used in the multigrid algorithm.

We denote the system (4.36) on mesh level l as usual by $A_l \mathbf{z}_l = \mathbf{b}_l$, $l = 0, \ldots, J$. By $D_l = \mathrm{diag}(A_l)$ we denote the diagonal of A_l. Suppose that we scale the equations on each mesh level from the left with D_l^{-1} and denote the result by

$$\tilde{A}_l \mathbf{z}_l = \tilde{\mathbf{b}}_l, \qquad \tilde{A}_l = D_l^{-1} A_l, \quad \tilde{\mathbf{b}}_l = D_l^{-1} \mathbf{b}_l. \tag{4.37}$$

A two–level coarse grid correction with standard components applied to the original equations $A_l \mathbf{z}_l = \mathbf{b}_l$ can be written in terms of the diagonally scaled equations as

$$\mathbf{z}_l^{\mathrm{new}} = \mathbf{z}_l^{\mathrm{old}} + P_l \tilde{A}_l^{-1} \tilde{R}_l \left(\tilde{\mathbf{b}}_l - \tilde{A}_l \mathbf{z}_l^{\mathrm{old}} \right) \tag{4.38}$$

with

$$\tilde{R}_l = D_{l-1}^{-1} R_l D_l. \tag{4.39}$$

The entries of the "new" restriction operator $\tilde{R}_l$ are given by

$$(\tilde{R}_l)_{ij} = (R_l)_{ij} \frac{(A_l)_{jj}}{(A_{l-1})_{ii}} \tag{4.40}$$

and reflect exactly the difficulties with division by ε as discussed above. We therefore propose to replace $\tilde{R}_l$ by a *truncated* version $\hat{R}_l$ given as

$$(\hat{R}_l)_{ij} = (R_l)_{ij} \cdot \min\left(cut, \frac{(A_l)_{jj}}{(A_{l-1})_{ii}}\right), \tag{4.41}$$

where *cut* is some user supplied parameter. We have to ensure that (4.41) does not spoil the multigrid convergence rate in the case of constant coefficients. A quick calculation shows that in this case $(A_l)_{jj}/(A_{l-1})_{ii} \leq 1$ at interior vertices if the order of the differential operator is not larger than the space dimension. Thus in all cases of interest for us the standard multigrid method is retained for constant coefficients if $cut \geq 1$. Since $(A_l)_{jj}/(A_{l-1})_{ii}$ may be larger than 1 when restricting from an interior vertex to a vertex at a Neumann boundary we choose $cut = 2$ in all the examples below. Numerical experiments confirm that the precise value of *cut* is not important as long as it is smaller than 5.

The implementation of the multigrid method with truncated restriction is straightforward. In a preprocessing step the truncated restriction operator $\hat{R}_l$ is computed and stored for all levels. Then the system matrices on all levels and the right hand side on the finest level are scaled by D_l^{-1} from the left. Now multigrid cycles are performed with the standard restriction replaced by $\hat{R}_l$. We will call this method the diagonally scaled/truncated restriction multigrid algorithm or DSTR–MG.

The DSTR–MG method has been developed on a purely heuristic basis. It is plain e. g. that if the spatial variations of the coefficients are on the order of the mesh size almost all entries of the restriction are truncated, which will result in a poor coarse grid correction. Thus the coarse grid size should be chosen with respect to the problem to be solved.

We will now illustrate the behavior of the method with two scalar examples in 2D. Applications to the fully coupled two–phase flow problem are given in Chapter 7. The model problem $-\nabla \cdot \{d(\mathbf{x})\nabla C\} = 0$ is solved in the unit square with Dirichlet boundary conditions left and right and Neumann boundary conditions at top and bottom. The coefficient distribution for both examples is shown in Fig. 4.3. Note that the cell size in example 2 is $\pi/15$. The model problem is discretized with a vertex centered finite volume scheme on a sequence of equidistant quadrilateral meshes with $h_0 = 1/2$. The diffusion coefficient is evaluated at the barycenter of each element. Table 4.1 shows results for a 10^{-8} reduction of the residual in the euclidean norm starting with initial guess zero. For comparison an algebraic multigrid method similar to the one given in (Braess 1995) is included. Both methods are used as a preconditioner in a Krylov subspace method and the number of preconditioner evaluations is reported. As a

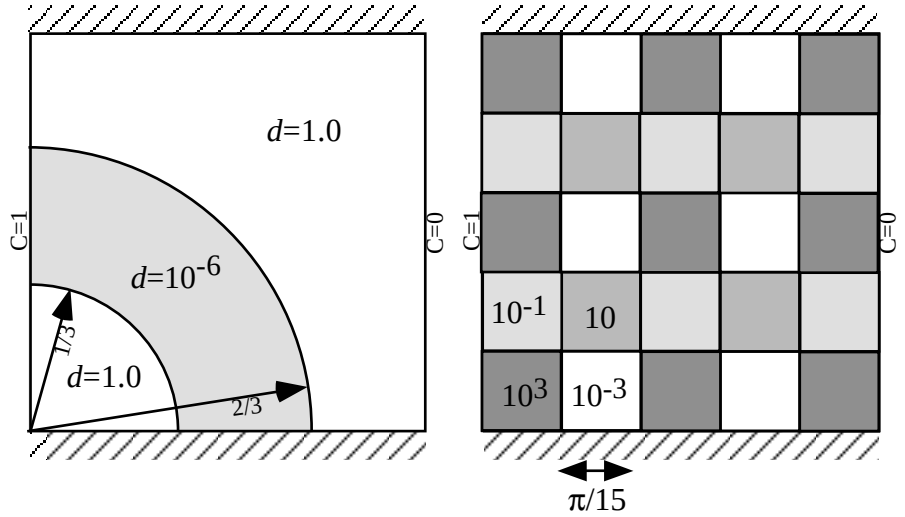


Figure 4.3: Coefficient distribution for the two example problems.

smoother either a symmetric Gauß–Seidel method or an Incomplete factorization is used as indicated in the table. The number of pre– and postsmoothing steps was 2 in all cases ($\nu_1 = \nu_2 = 2$).

For example 1 the DSTR–MG method exhibits standard multigrid performance. The convergence rate is about 0.1 and it can be used without Krylov method. The algebraic multigrid method shows an iteration count proportional to the number of levels J. For the more difficult example 2 both methods show an $O(J)$ behavior. The algebraic multigrid method converges faster in this case. The convergence behavior of the algebraic multigrid method is only slightly worse when compared to example 1. It should be noted that standard multigrid with discretized coarse grid operator does not converge for both examples with or without Krylov acceleration.

It remains to extend the DSTR–MG method to systems of equations. We

Table 4.1: Multigrid performance for two interface problems.

	Example 1			Example 2		
	no krylov	BiCGSTAB	CG	BiCGSTAB	BiCGSTAB	CG
	DSTR	DSTR	AMG	DSTR	DSTR	AMG
h^{-1}	SGS(2,2)	SGS(2,2)	SGS(2,2)	SGS(2,2)	ILU(2,2)	SGS(2,2)
16	8	6	6	21	13	7
32	8	6	8	23	17	8
64	8	7	10	29	21	11
128	8	7	12	31	21	14
256	9	6	15	34	24	17

consider the system to be in point–block ordering. In the derivation the diagonal matrix D_l is then replaced by the point–block diagonal $\bar{D}_l$ from (4.32). The 2×2 block structure carries over naturally to the restriction matrices giving

$$\tilde{\bar{R}}_l = \bar{D}_{l-1}^{-1} \bar{R}_l \bar{D}_l \tag{4.42}$$

in analogy to (4.39). $\bar{R}_l$ is the component–wise standard restriction. The individual 2×2 blocks of $\tilde{\bar{R}}_l$ are given by

$$(\tilde{\bar{R}}_l)_{ij} = (R)_{ij} (\bar{D}_{l-1})_{ii}^{-1} (\bar{D}_l)_{jj} \tag{4.43}$$

where we used the fact that $(\bar{R}_l)_{ij} = (R)_{ij} I_{2\times 2}$ with $(R)_{ij}$ the *scalar* component of standard restriction in the non–system case. Following the idea above $\tilde{\bar{R}}_l$ is now replaced by a truncated version defined as

$$\left((\tilde{\bar{R}}_l)_{ij} \right)_{\alpha\beta} = (R)_{ij} \cdot \max \left(0, \min \left(cut, \left((\bar{D}_{l-1})_{ii}^{-1} (\bar{D}_l)_{jj} \right)_{\alpha\beta} \right) \right) \tag{4.44}$$

for $\alpha, \beta = 1, 2$. Note that entries are truncated from above by *cut* and from below by zero. Note also that $(\tilde{\bar{R}}_l)_{ij}$ is, in general, a full 2×2 matrix.

4.3.6 Additional Remarks

The multigrid algorithm, including the truncated restriction, can be applied to problems discretized on locally refined meshes. Since adaptivity and local mesh refinement is not used in this work we refer to (Bastian 1996) for notes on the implementation of multigrid on adaptively refined meshes.

Multigrid can also be applied directly to discretizations of nonlinear partial differential equations, see (Hackbusch 1985, Chap. 9). In this so–called *nonlinear multigrid method* the smoother is replaced by an iterative scheme for the nonlinear problem and a nonlinear coarse grid problem is set up. There are three reasons why did not try to use this method here:

- Nonlinear smoothers are inefficient to implement in most unstructured mesh codes since they require to reassemble a single row (or a small set of rows) of the Jacobian at a time.
- Nonlinear smoothers are typically restricted to Jacobi or Gauß–Seidel type schemes. The robust smoother methodology is not (yet) as developed as in the linear case.
- Nonlinear multigrid is more expensive with respect to computation time when compared to Newton–multigrid, at least for the type of problems we are interested in, see (Molenaar 1995) for a comparison of both methods.

5

Parallelization

Computing time requirements for time–dependent three–dimensional nonlinear problems are still enormous. Field scale models with fine geometrical detail require on the order of millions of mesh elements. In that respect linear solvers with optimal complexity become increasingly important since all other components of a simulator scale linearly with mesh size.

The first section of this chapter describes a data parallel implementation of the multigrid solver which is based on a suitable decomposition of the multigrid hierarchy into as many parts as processors are available. The construction of such decompositions will be the subject of the second section in this chapter.

5.1 Parallelization of the Solver

5.1.1 Introduction

In order to increase the size of tractable problems to the range of millions of mesh elements the use of parallel computer architectures is mandatory. In this section we will therefore introduce a data parallel implementation of the Newton–multigrid solver. Even with a multigrid solver the linear system solver typically requires more than 60% of total computation time and is therefore the important part to parallelize.

The parallel solution of linear systems arising from the discretization of (preferably elliptic) partial differential equations is an area of active research for many years. The most successful methods are domain decomposition and multigrid methods. An excellent introduction to both methods with respect to parallelization is given in (Smith et al. 1996), detailed parallel implementations are given in (Van de Velde 1993).

Provided a suitable smoothing iteration is chosen all components of the standard multigrid method are inherently parallel. Thus a parallel implementation can be based on mapping the mesh data structure to the processors. Since the multigrid algorithm is not modified during the parallelization process (strictly true only for Jacobi smoothing) optimal convergence properties of the multigrid method are not harmed. This does not apply, however, for many multigrid methods that are robust against additional bad parameters such as anisotropy or dominating convection. It turns out that typical robust smoothers like line smoothers or methods based on incomplete LU factorization are hardly parallelizable.

Domain decomposition (DD) methods on the other hand are specifically designed for parallel computation. A whole new body of theory had to be developed to show the near optimality of these methods, see (Dryja and Widlund

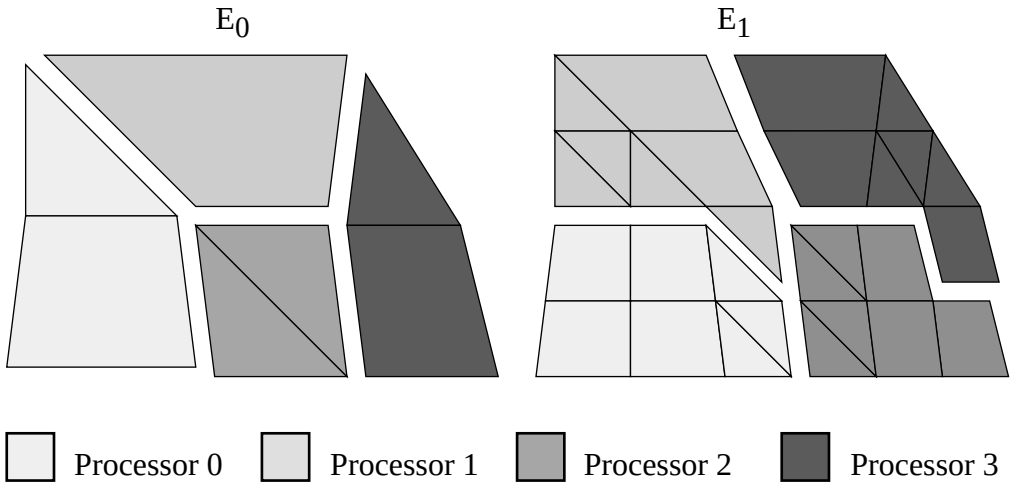


Figure 5.1: Mapping two grid levels to four processors.

1990; Xu 1992). With respect to robustness DD methods typically suffer from the same problems as do standard multigrid methods. Direct comparisons of DD and multigrid methods are rare but a sophisticated comparison is available from Heise and Jung (1995). They found a data parallel multigrid implementation to be consistently faster by a factor $2 \ldots 5$ when compared to a non–overlapping DD method (with coarse grid space) in two space dimensions. This is mostly due to the better convergence properties of the multigrid method, the favorable parallelization properties of DD (fewer and shorter messages) cannot be utilized on contemporary parallel computers with their "fat" processing nodes and fast communication networks.

5.1.2 Data Decomposition

Our data parallel multigrid implementation is based on a suitable mapping of the hierarchical mesh data structure $\{E_l | l = 0, \ldots, J\}$ to the set of processors $P = \{1, \ldots, P\}$ denoted formally by mapping functions

$$m_l : E_l \to P, \qquad l = 0, \ldots, J. \tag{5.1}$$

In principle an element $e \in E_l$ can be mapped to any available processor. Different mappings will, of course, result in realizations with varying efficiency. Selection of a set of mappings which give a high efficiency is called the *load balancing problem*. In most parts of this section we are concerned with the implementation of the multigrid components for arbitrary mappings m_l, but we will comment on the load balancing problem later on. Fig. 5.1 shows an example where a mesh hierarchy with two levels is mapped to four processors.

Since the mesh construction is hierarchical we can associate with each $e \in E_l$, $l > 0$, an element $f(e) \in E_{l-1}$ such that e originated from refinement of element $f(e)$. $f(e)$ is called the *father element* of e. Furthermore we denote by $V_l(e)$ the vertices of element e and by $NB_l(e)$ the neighboring elements of any $e \in E_l$.

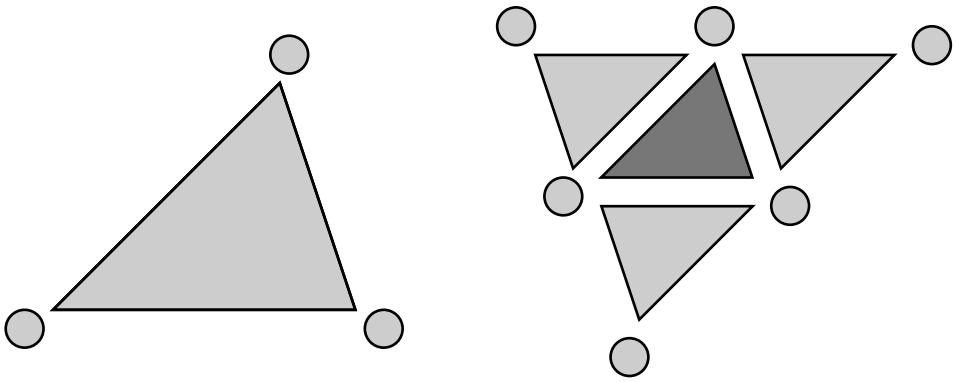


Figure 5.2: Context of an element.

Suppose that element $e \in E_l$ is assigned to processor $p = m_l(e)$. In order to implement the numerical algorithms described in this work a set of additional elements and vertices related to e, called its *context*, have to be stored by processor p. In detail the context of element e consists of

- the vertices $V_l(e)$,
- the neighboring elements $NB_l(e)$ together with their vertices
- and the father element $f(e)$ together with its vertices.

Fig. 5.2 shows the context of a single element. Thus the elements on level l stored by processor p are given by

$$E_l^{(p)} = \left\{ e \in E_l \;\middle|\; \begin{array}{ll} & m_l(e) = p \\ \text{or} & m_l(n) = p \text{ with } n \in NB_l(e) \\ \text{or} & m_{l+1}(s) = p \text{ with } f(s) = e \end{array} \right\} \tag{5.2}$$

and the vertices stored by processor p are given by

$$V_l^{(p)} = \left\{ v \in V_l \;\middle|\; \exists e \in E_l^{(p)} : v \in V_l(e) \right\} \quad . \tag{5.3}$$

It is clear that no additional storage for the context is necessary if neighboring elements and father elements are mapped to the same processor. It is the aim of the load balancing procedure to find mappings $\{m_l\}$ such that each processor has about the same number of elements on each mesh level while minimizing the additional storage (and computation) needed for the overlap. The overlapping decomposition of the mesh data structure is sufficient to implement a variety of numerical algorithms including error estimators and multigrid methods.

We can now proceed to describe the data parallel multigrid implementation. We assume for ease of presentation that degrees of freedom are associated only

with the vertices of the mesh. The general case with additional degrees of freedom in edges, faces and elements is also possible, even with local mesh refinement, cf. (Wieners 1997; Lang 1999). Under this assumption the degrees of freedom on each level form a vector $\mathbf{z}_l \in \mathbb{R}^{N_l}$ on each level. The description of the multigrid components is based on the definition of various projections from $\mathbb{R}^{N_l}$ to $\mathbb{R}^{N_l}$.

Projection $H_l^{(p)}$ is a linear map from $\mathbb{R}^{N_l} \to \mathbb{R}^{N_l}$ that picks out the components of a vector that correspond to vertices of elements assigned to processor p:

$$\left(H_l^{(p)}\mathbf{z}_l\right)_i = \begin{cases} (\mathbf{z}_l)_i & \exists e \in E_l : m_l(e) = p \wedge v_i \in V_l(e) \\ 0 & else \end{cases} . \tag{5.4}$$

The projection $V_l^{(p)}$ picks out components of a vector that correspond to corners of elements which are fathers of elements on the fine level that are mapped to p. Additionally, the projection is zero at a vertex if it is already a corner of a level-l-element that is mapped to p. Formally it is given by

$$\left(V_l^{(p)}\mathbf{z}_l\right)_i = \begin{cases} (\mathbf{z}_l)_i - (H_l^{(p)}\mathbf{z}_l)_i & \exists e \in E_{l+1} : m_{l+1}(e) = p \wedge v_i \in V_l(f(e)) \\ 0 & else \end{cases} . \tag{5.5}$$

Note also that neighboring elements are not included in the definitions of the two projections since they play only a rôle in the evaluation of error estimators but not in the solver components.

Projections $H_l^{(p)}$ and $V_l^{(p)}$ define subspaces of $\mathbb{R}^{N_l}$ via

$$H_l^{(p)} = \{\mathbf{x} \in \mathbb{R}^{N_l} \,|\, \exists \mathbf{y} : \mathbf{x} = H_l^{(p)}\mathbf{y}\}, \quad V_l^{(p)} = \{\mathbf{x} \in \mathbb{R}^{N_l} \,|\, \exists \mathbf{y} : \mathbf{x} = V_l^{(p)}\mathbf{y}\}. \tag{5.6}$$

By construction we have $H_l^{(p)} \cap V_l^{(p)} = \{\mathbf{0}\}$. Furthermore, $\{H_l^{(p)} | p \in P\}$ is an overlapping subspace decomposition. With the help of the "picking function" $p_l^* : \{1, \ldots, N_l\} \to P$ given by

$$p_l^*(i) = p \Leftrightarrow \forall p, q \in P : \left(H_l^{(p)}\mathbf{e}_i = \mathbf{e}_i \wedge H_l^{(q)}\mathbf{e}_i = \mathbf{e}_i \wedge p \neq q\right) \Rightarrow p < q, \tag{5.7}$$

$\mathbf{e}_i$ being the i-th unit vector, we define also a non–overlapping decomposition by

$$\left(Q_l^{(p)}\mathbf{z}_l\right)_i = \begin{cases} (\mathbf{z}_l)_i & p_l^*(i) = p \\ 0 & else \end{cases} \tag{5.8}$$

and its corresponding subspace $Q_l^{(p)}$. Finally, we define

$$I_l^{(p)} = H_l^{(p)} + V_l^{(p)}, \qquad I_l^{(p)} = H_l^{(p)} + V_l^{(p)}, \tag{5.9}$$

which gives us the inclusion

$$Q_l^{(p)} \subseteq H_l^{(p)} \subseteq I_l^{(p)}. \tag{5.10}$$

We denote by $\{\mathbf{z}_l^{(p)} \in X_l^{(p)} | p \in P\}$ a decomposition of a vector $\mathbf{z}_l \in \mathbb{R}^{N_l}$ where $X_l^{(p)}$ is any of $Q_l^{(p)}$, $H_l^{(p)}$ or $I_l^{(p)}$.

Since the $Q_l^{(p)}$ are non–overlapping we have the *unique* representation of any vector as

$$\mathbf{z}_l = \sum_{p \in P} Q_l^{(p)} \mathbf{z}_l. \tag{5.11}$$

In addition the projections $Q_l^{(p)}$ are orthogonal with respect to the euclidean inner product which gives us

$$\|\mathbf{z}_l\|_2^2 = \sum_{p \in P} \|Q_l^{(p)} \mathbf{z}_l\|_2^2, \tag{5.12}$$

i. e. the global norm can be computed by summing local norms.

We say that a decomposition $\{\mathbf{z}_l^{(p)} \in X_l^{(p)} | p \in P\}$ of $\mathbf{z}_l$ has the *X–summation property* if

$$\mathbf{z}_l = \sum_{p \in P} \mathbf{z}_l^{(p)}. \tag{5.13}$$

Since $H_l^{(p)}$ and $I_l^{(p)}$ are overlapping the corresponding decompositions are not unique.

For any vector $\mathbf{z}_l \in \mathbb{R}^{N_l}$ a decomposition defined by

$$\mathbf{z}_l^{(p)} = H_l^{(p)} \mathbf{z}_l, \qquad \mathbf{z}_l^{(p)} = I_l^{(p)} \mathbf{z}_l \tag{5.14}$$

is called *H–consistent* or *I–consistent* respectively.

A similar notation can be introduced for matrices. A decomposition $\{A_l^{(p)} : X_l^{(p)} \to X_l^{(p)} | p \in P\}$ of $A_l \in \mathbb{R}^{N_l \times N_l}$ has the X–summation property if

$$A_l = \sum_{p \in P} A_l^{(p)} \qquad . \tag{5.15}$$

5.1.3 Parallel Multigrid Algorithm

The aim is to decompose all operations of the sequential algorithm into local computations in the subspaces $Q_l^{(p)}$, $H_l^{(p)}$ and $I_l^{(p)}$ with corresponding communication operations providing the global coupling. We begin by deriving every single step and then combine all steps into the complete parallel multigrid cycle.

$\underline{\mathbf{d}_l = \mathbf{b}_l - A_l\mathbf{z}_l}$; *Defect Computation.* In finite element and finite volume computations on unstructured meshes the stiffness matrix (or Jacobian in the nonlinear case) is assembled element by element. The summation over all elements $\{e \in E_l | m_l(e) = p\}$ in each processor p naturally results in matrices $\{A_l^{(p)} : H_l^{(p)} \to H_l^{(p)} | p \in P\}$ and right hand sides $\{\mathbf{b}_l^{(p)} \in H_l^{(p)} | p \in P\}$ that have the H–summation property. This can be done without any communication.

Therefore we have

$$\begin{aligned} \mathbf{d}_l &= \mathbf{b}_l - A_l\mathbf{z}_l \\ &= \sum_{p\in P} H_l^{(p)}\mathbf{b}_l^{(p)} - \sum_{p\in P} H_l^{(p)}A_l^{(p)}H_l^{(p)}\mathbf{z}_l \end{aligned} \tag{5.16}$$

where we inserted projections to indicate the subspaces. If we introduce the H–consistent decomposition $\{\mathbf{z}_l^{(p)} = H_l^{(p)}\mathbf{z}_l | p \in P\}$ of the vector $\mathbf{z}_l$ we get

$$\mathbf{d}_l = \sum_{p\in P} H_l^{(p)} \left(\mathbf{b}_l^{(p)} - A_l^{(p)}\mathbf{z}_l^{(p)}\right) = \sum_{p\in P} H_l^{(p)}\mathbf{d}_l^{(p)}. \tag{5.17}$$

In summary we have

- Given $\{\mathbf{b}_l^{(p)}\}$, $\{A_l^{(p)}\}$ with H–summation property
- and $\{\mathbf{z}_l^{(p)}\}$ H–consistent
- $\{\mathbf{d}_l^{(p)} = \mathbf{b}_l^{(p)} - A_l^{(p)}\mathbf{z}_l^{(p)}\}$ can be computed locally without communication and
- $\{\mathbf{d}_l^{(p)}\}$ has H–summation property.

$\underline{\mathbf{s}_l = M_l^{-1}\mathbf{d}_l}$; *Approximate Solve*. In order to arrive at local computations some restrictions on M_l are necessary. Clearly M_l can only be inverted without communication if it is block diagonal with respect to the subspaces $Q_l^{(p)}$, resulting in a block–Jacobi type smoother for the multigrid method. Given A_l we set

$$M_l^{(p)} = Q_l^{(p)}A_l^{(p)}Q_l^{(p)}, \qquad M_l = \sum_{p\in P} M_l^{(p)}. \tag{5.18}$$

Assuming that $\{\mathbf{s}_l^{(p)} = Q_l^{(p)}\mathbf{s}_l\}$ and $\{\mathbf{d}_l^{(p)} = Q_l^{(p)}\mathbf{d}_l\}$ are unique decompositions of $\mathbf{s}_l$ and $\mathbf{d}_l^{(p)}$ we get

$$\begin{aligned} & M_l\mathbf{s}_l = \mathbf{d}_l \\ \Leftrightarrow \quad & \sum_{p\in P} Q_l^{(p)}A_l^{(p)}Q_l^{(p)}Q_l^{(p)}\mathbf{s}_l = \sum_{p\in P} Q_l^{(p)}\mathbf{d}_l \\ \Leftrightarrow \quad & M_l^{(p)}\mathbf{s}_l^{(p)} = \mathbf{d}_l^{(p)}, \quad \forall p \in P. \end{aligned} \tag{5.19}$$

In summary we have

- If M_l is a block diagonal matrix w. r. t. the subspaces $Q_l^{(p)}$
- and the defect is provided in unique form $\{\mathbf{d}_l^{(p)} = Q_l^{(p)}\mathbf{d}_l\}$
- a correction $\{\mathbf{s}_l^{(p)} = Q_l^{(p)}\mathbf{s}_l\}$ in unique form can be computed locally.

$\underline{\mathbf{z}_l = \mathbf{z}_l + \mathbf{s}_l;\ \textit{Update}}$. We assume that $\{\mathbf{z}_l^{(p)} = H_l^{(p)}\mathbf{z}_l^{(p)}\}$ is in H–consistent form as required by the defect computation step. Applying $H_l^{(p)}$ to both sides of the update equation

$$H_l^{(p)}\mathbf{z}_l = H_l^{(p)}\mathbf{z}_l + H_l^{(p)}\mathbf{s}_l \tag{5.20}$$

we see that $\mathbf{s}_l$ is also required in H–consistent form to enable a local computation of the update step.

$\underline{\mathbf{d}_{l-1} = R_l\mathbf{d}_l;\ \textit{Restriction}}$. From an element–wise consideration the following identity can be derived:

$$R_l H_l^{(p)}\mathbf{d}_l = I_{l-1}^{(p)} R_l H_l^{(p)}\mathbf{d}_l, \quad \forall p \in P\,. \tag{5.21}$$

Assuming that $\{\mathbf{d}_l^{(p)}\}$ is a decomposition with the H–summation property we get

$$\begin{aligned}\mathbf{d}_{l-1} = R_l\mathbf{d}_l &= R_l \sum_{p\in P} H_l^{(p)}\mathbf{d}_l^{(p)} \\ &= \sum_{p\in P} R_l H_l^{(p)}\mathbf{d}_l^{(p)} = \sum_{p\in P} I_{l-1}^{(p)} R_l H_l^{(p)}\mathbf{d}_l \\ &= \sum_{p\in P} R_l^{(p)}\mathbf{d}_l^{(p)}\end{aligned} \tag{5.22}$$

with $R_l^{(p)} = I_{l-1}^{(p)} R_l H_l^{(p)}$.

In summary we have that

- $\{\mathbf{d}_l^{(p)}\}$ with H–summation property
- can be restricted to $\{\mathbf{d}_{l-1}^{(p)}\}$ locally without communication
- where $\{\mathbf{d}_{l-1}^{(p)}\}$ has the I–summation property.

$\underline{\mathbf{s}_l = P_l\mathbf{s}_{l-1};\ \textit{Prolongation}}$. Again, an element–wise consideration yields the following relation for the prolongation step:

$$H_l^{(p)} P_l \mathbf{s}_{l-1} = H_l^{(p)} P_l I_{l-1}^{(p)}\mathbf{s}_{l-1}, \quad \forall p \in P\,. \tag{5.23}$$

Provided that

- $\{\mathbf{s}_{l-1}^{(p)} = I_{l-1}^{(p)}\mathbf{s}_{l-1}\}$ is I–consistent,
- it can be interpolated locally to $\{\mathbf{s}_l^{(p)} = P_l^{(p)}\mathbf{s}_{l-1}^{(p)}\}$ with $P_l^{(p)} = H_l^{(p)}P_l I_{l-1}^{(p)}$ and
- resulting $\{\mathbf{s}_l^{(p)}\}$ is H–consistent.

We are now in a position to formulate the parallel version of the multigrid cycle by concatenating the steps discussed in detail above. The different consistency requirements of the individual steps will naturally lead to the necessary communication operations.

ALGORITHM 5.1 The following algorithm **pmgc** implements one cycle of the standard multigrid method in parallel. It works on decompositions of the current iterate $\{\mathbf{z}_l^{(p)}\}$ and the right hand side $\{\mathbf{b}_l^{(p)}\}$ which are assumed to possess H–consistency and H–summation property, respectively, on entry. All statements preceded by $\forall p : \ldots$ are assumed to be executed in parallel.

pmgc (l, $\{\mathbf{z}_l^{(p)}\}$, $\{\mathbf{b}_l^{(p)}\}$)
{
(1) **if** (l == 0) $\mathbf{z}_0 = A_0^{-1}\mathbf{b}_0$;
else {
(2) **for** $(m = 1, \ldots, \nu_1)$ { // presmoothing
(3) $\forall p : \mathbf{d}_l^{(p)} = \mathbf{b}_l^{(p)} - A_l^{(p)}\mathbf{z}_l^{(p)}$; // $\{A_l^{(p)}\}$, $\{\mathbf{b}_l^{(p)}\}$ H–sum
(4) **Hsum_to_Q**$(\{\mathbf{d}_l^{(p)}\})$; // communication
(5) $\forall p : \mathbf{s}_l^{(p)} = {M_l^{(p)}}^{-1}\mathbf{d}_l^{(p)}$; // $M_l^{(p)} = Q_l^{(p)}A_l^{(p)}Q_l^{(p)}$
(6) **Q_to_Hcons**$(\{\mathbf{s}_l^{(p)}\})$; // communication
(7) $\forall p : \mathbf{z}_l^{(p)} = \mathbf{z}_l^{(p)} + \omega\mathbf{s}_l^{(p)}$; // H–cons. update
}
(8) $\forall p : \mathbf{d}_l^{(p)} = \mathbf{b}_l^{(p)} - A_l^{(p)}\mathbf{z}_l^{(p)}$; // $\{\mathbf{d}_l^{(p)}\}$ H–sum
(9) $\forall p : \mathbf{d}_{l-1}^{(p)} = R_l^{(p)}\mathbf{d}_l^{(p)}$; // $R_l^{(p)} = I_{l-1}^{(p)}R_l H_l^{(p)}$
(10) **Isum_to_Hsum**$(\{\mathbf{d}_{l-1}^{(p)}\})$; // communication
(11) $\forall p : \mathbf{s}_{l-1}^{(p)} = 0$; // H–consistent
for $(g = 1, \ldots, \gamma)$
(12) **pmgc**$(l-1, \{\mathbf{s}_{l-1}^{(p)}\}, \{\mathbf{d}_{l-1}^{(p)}\})$; // recursive call
(13) **Hcons_to_Icons**$(\{\mathbf{s}_{l-1}^{(p)}\})$; // communication
(14) $\forall p : \mathbf{s}_l^{(p)} = P_l^{(p)}\mathbf{s}_{l-1}^{(p)}$; // $P_l^{(p)} = H_l^{(p)}P_l I_{l-1}^{(p)}$
(15) $\forall p : \mathbf{z}_l^{(p)} = \mathbf{z}_l^{(p)} + \mathbf{s}_l^{(p)}$; // update
(16) **for** $(m = 1, \ldots, \nu_2)$ // postsmoothing
// same as (3) – (7) above
}
}

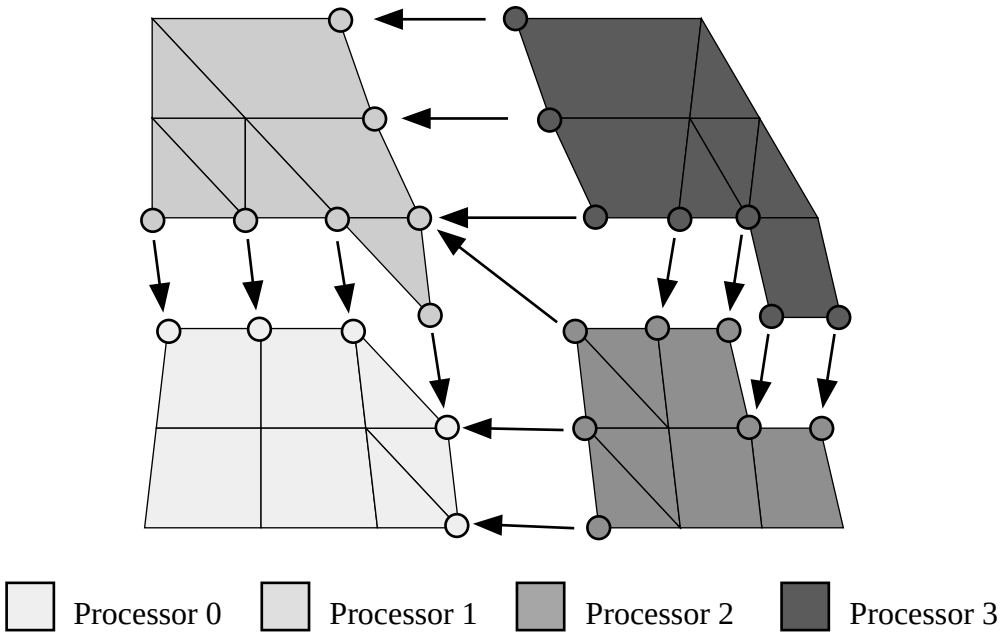


Figure 5.3: Flow of information in **Hsum_to_Q** communication.

Upon entry the solution is assumed to be in H–consistent form and the right hand side in H–summation form. The local defect computed in step (3) of algorithm **pmgc** also has H–summation property as has been discussed above. The defect needed in the subsequent local solve step (5) has to be in unique form. Therefore a communication operation of the form

Hsum_to_Q ($\{\mathbf{d}_l^{(p)}\}$)
{
$\quad \mathbf{d}_l^{(p)} = Q_l^{(p)} \sum_{q \in P} \mathbf{d}_l^{(q)};$
}

has to be inserted in step (4). This communication requires every processor to send the data not belonging to its subspace $Q_l^{(p)}$ to another processor. Fig. 5.3 illustrates the flow of information for the fine mesh in Fig. 5.1.

The local solve in step (5) yields a correction that is unique but the update in step (7) requires a H–consistent correction. The communication operation

Q_to_Hsum ($\{\mathbf{s}_l^{(p)}\}$)
{
$\quad \mathbf{s}_l^{(p)} = H_l^{(p)} \sum_{q \in P} \mathbf{s}_l^{(q)};$
}

performs this transformation. The flow of information is exactly reverse to that given in Fig. 5.3. The parallel multigrid implementation requires *two* communi-

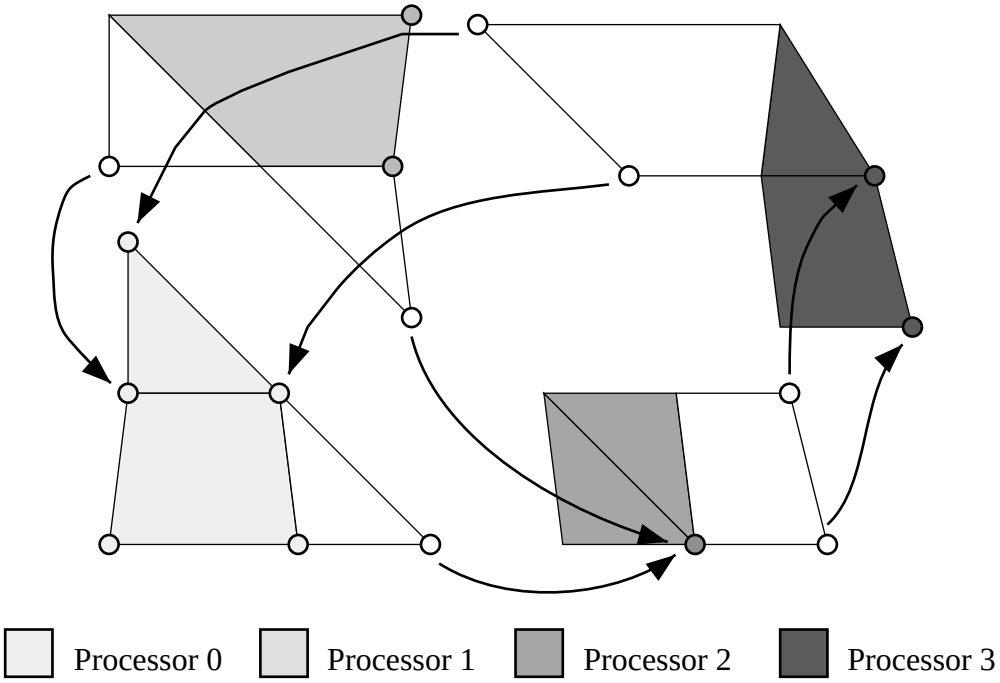


Figure 5.4: Flow of information in **Isum_to_Hsum** communication.

cation operations per smoothing step. This is a consequence of the small overlap in the data partitioning. With a more generous overlap where for a given vertex all surrounding elements are stored on the same processor each smoothing step can be implemented with one communication operation.

We now proceed to the coarse grid correction. As has been discussed above, the local restriction in step (9) of algorithm **pmgc** results in $\{\mathbf{d}_{l-1}^{(p)}\}$ which has I–summation property. However, H–summation property is required in the recursive call in step (12). A communication on the coarse mesh is inserted in step (10) to perform this transformation:

Isum_to_Hsum ($\{\mathbf{d}_{l-1}^{(p)}\}$)
{
$$\mathbf{d}_{l-1}^{(p)} = \left(\mathbf{d}_{l-1}^{(p)} - V_{l-1}^{(p)}\mathbf{d}_{l-1}^{(p)}\right) + Q_{l-1}^{(p)} \sum_{q\in P} V_{l-1}^{(q)}\mathbf{d}_{l-1}^{(q)};$$
}

Note that only the part $V_{l-1}^{(p)}\mathbf{d}_{l-1}^{(p)}$ of the local defect in each processor is redistributed. Fig. 5.4 shows the flow of information for the two level example from Fig. 5.1.

Note that $V_{l-1}^{(p)} \neq \{\mathbf{0}\}$ can only occur if $m_{l-1}(f(e)) \neq m_l(e)$ for some element $e \in E_l$ with $m_l(e) = p$. If $m_{l-1}(f(e)) = m_l(e)$ for all elements e then *no* communication is necessary in the restriction (and prolongation). This is a consequence

of processing the defect in H–summation form. It is a necessary requirement for the implementation of additive multigrid methods, cf. (Bastian 1996).

Finally, the recursive call of the multigrid cycle in step (12) of **pmgc** results in a correction that is H–consistent but I–consistency is required as a prerequisite in the prolongation step (14). The corresponding communication operation is formally given by

Hcons_to_Icons ($\{\mathbf{s}_{l-1}^{(p)}\}$)
{
$$\mathbf{s}_{l-1}^{(p)} = \mathbf{s}_{l-1}^{(p)} + V_{l-1}^{(p)} \sum_{q \in P} Q_{l-1}^{(q)} \mathbf{s}_{l-1}^{(q)};$$
}

and the flow of information is exactly reverse to that shown in Fig. 5.4. No communication is necessary if all elements are mapped to the same processor as their father element.

Algorithm **pmgc** is still in abstract mathematical formulation. In the actual implementation the different subspaces are replaced by vector spaces of appropriate dimension and corresponding mappings of local to global indices. For more details we refer to (Bastian 1996; Lang 1999).

In addition algorithm **pmgc** requires a preprocessing phase where the matrices $\{M_l^{(p)}\}$ are constructed from $\{A_l^{(p)}\}$ obtained from discretization. This requires a communication operation similar to **Hsum_to_Q** since $\{A_l^{(p)}\}$ has H–summation property and $\{M_l^{(p)}\}$ is unique. If the truncated restriction from Subs. 4.3.5 is used another local communication is required in the setup phase.

Assembling of the stiffness matrix $\{A_l^{(p)}\}$ and the right hand side $\{\mathbf{b}_l^{(p)}\}$ can typically be done without communication provided a H–consistent decomposition $\{\mathbf{z}_l^{(p)}\}$ of the current solution is available in the nonlinear case on each level. The *PPSIC* method from Section 3.4 requires a local communication to compute the smallest capillary pressure in each vertex.

The parallel multigrid method is only part of the global solution algorithm. The time–stepping procedures, inexact Newton scheme and Krylov subspace methods can be parallelized using the same data partitioning with the guiding principle that right hand sides are stored in H–summation mode (or unique if norms are to be computed) and solution vectors are stored in H–consistent form.

Finally, we note that algorithm **pmgc** can be extended to the case of adaptively refined meshes . The description is, however, rather tedious and we refer to (Bastian 1996; Lang 1999) for details.

5.2 Load Balancing

This section is devoted to the problem of partitioning a multigrid hierarchy in such a way that load balance is obtained in each computational phase and communication required in the smoother and the intergrid transfer is kept small. We

begin by stating four related abstract graph partitioning problems where two of them are new. Then we will illustrate how these graph partitioning problems are used to solve the load balancing problem for possibly locally refined multigrid hierarchies. After reviewing shortly the work that has been done on standard graph partitioning we will describe two algorithms which can be used to solve the new graph partitioning problems. The algorithms will be based on an approach known as the multilevel graph partitioner.

5.2.1 Graph Partitioning Problems

This section defines four related graph partitioning problems which are utilized in the solution of load balancing problems for parallel unstructured (hierarchical) mesh applications.

k-way Graph Partitioning Problem. The input quantities for the k-way graph partitioning problem are an undirected graph $G = (N, A), A \subseteq N \times N$, a number $2 \leq k \in \mathbb{N}$ and weights for vertices and edges: $w : N \to \mathbb{N}$ and $w : A \to \mathbb{N}$. The total weight of all vertices is then $W = \sum_{n \in N} w(n)$. Let $\pi : N \to \{0, \ldots, k-1\}$ be a function associating a number in the range $0 \ldots k-1$ with each vertex. π is called a *partition map* and the subset $N(i) = \{n \in N | \pi(n) = i\}$ is called a *partition*. The subset $X = \{(n, n') \in A | \pi(n) \neq \pi(n')\}$ is called an *edge separator*.

A partition map π is called a solution of the k-way graph partitioning problem if the following two properties hold:

$$(i) \qquad \sum_{n \in N(i)} w(n) \leq \delta W/k \ \ \forall i \qquad \text{(Balance condition)}, \tag{5.24a}$$

$$(ii) \qquad \sum_{a \in X} w(a) \quad \text{is minimal} \qquad \text{(Minimal separator weight)}. \tag{5.24b}$$

The first condition ensures that the weight of each partition (the work) is balanced, whereby a load imbalance δ is allowed. A reasonable value for δ is $1.0 \ldots 1.1$. The second condition ensures that the weight associated with the separator edges (modeling communication cost) is minimized.

k-way Graph Repartitioning Problem. The k-way graph repartitioning problem is a variation of the k-way graph partitioning problem where an *initial partition map* π^0 is supplied in addition. The corresponding partitioning $N^0(i) = \{n \in N | \pi^0(n) = i\}$ may be arbitrary. In order to satisfy the balance condition some vertices have to change partitions. The cost associated with moving a vertex is given by the vertex size function $s : N \to \mathbb{N}$.

The output partition map π to be computed is required to satisfy the balance condition (i), the edge separator weight minimization (ii) and the migration cost minimization condition

$$(iii) \qquad \sum_{n \in \{m \in N | \pi(m) \neq \pi^0(m)\}} s(n) \quad \text{is minimal} \quad \text{(Minimal migration cost)} \tag{5.25}$$

Clearly separator weight and migration cost cannot be minimized simultaneously. Priority has to be given to one or the other or a combined objective function has to be formed. The heuristic algorithms to be described below will not exactly minimize either separator weight or migration cost but will rather keep them small.

The vertex migration cost (iii) is the total migration cost. Alternatively, one could also minimize the maximum migration cost associated with all vertices going in and out of *one* partition.

Constrained k-way Graph Partitioning Problem. This problem is a variation of the k-way graph partitioning problem that is useful for load balancing hierarchical meshes as will be shown in the next subsection.

In the constrained version of the graph partitioning problem the vertex set N is divided into two disjoint subsets $N = N' \cup N''$, $N' \cap N'' = \emptyset$. The vertices $n \in N'$ are assumed to be already assigned to their partition, i. e. $\pi(n)$ is fixed on those vertices and is *not* subject to change. N' is called the set of constrained vertices or simply the constraint and N'' is called the set of free or unconstrained vertices. The definitions of a partition and total weight naturally carry over to the subsets:

$$N'(i) = \{n \in N' | \pi(n) = i\}, \;\; N''(i) = \{n \in N'' | \pi(n) = i\}, \tag{5.26a}$$

$$W' = \sum_{n \in N'} w(n), \;\; W'' = \sum_{n \in N''} w(n). \tag{5.26b}$$

A partition map π is a solution of the constrained k-way partitioning problem if it provides a balanced partitioning of the free vertices N'' in the following way:

$$(i') \qquad \sum_{n \in N''(i)} w(n) \leq \delta W''/k \;\; \forall i \qquad \text{(Constrained balance)} \tag{5.27}$$

together with a minimization of the edge cut weight (ii). Note that the balance condition (i') is restricted only to the free vertices. The weight of the constrained vertices is not considered at all. The separator weight, however, includes all inter-partition edges, even those incident on constrained vertices. Since the partition of any constrained vertex cannot be changed the cost associated with the edges $X' = \{(n,n') \in A | \pi(n) \neq \pi(n') \wedge n, n' \in N'\}$ is a fixed contribution to the separator weight and could have been excluded in the definition.

Fig. 5.5 gives an illustration of the constrained k-way graph partitioning problem. In typical applications only a subset of the free vertices is connected to the set of constrained vertices. Edges incident only on constrained vertices have been excluded since they do not influence the solution.

Constrained k-way Repartitioning Problem. The constrained k-way partitioning problem can be extended to the case of repartitioning. As in the unconstrained version we supply an initial partition map π^0. The solution of the constrained k-way repartitioning problem has to satisfy conditions (i') and (ii) and

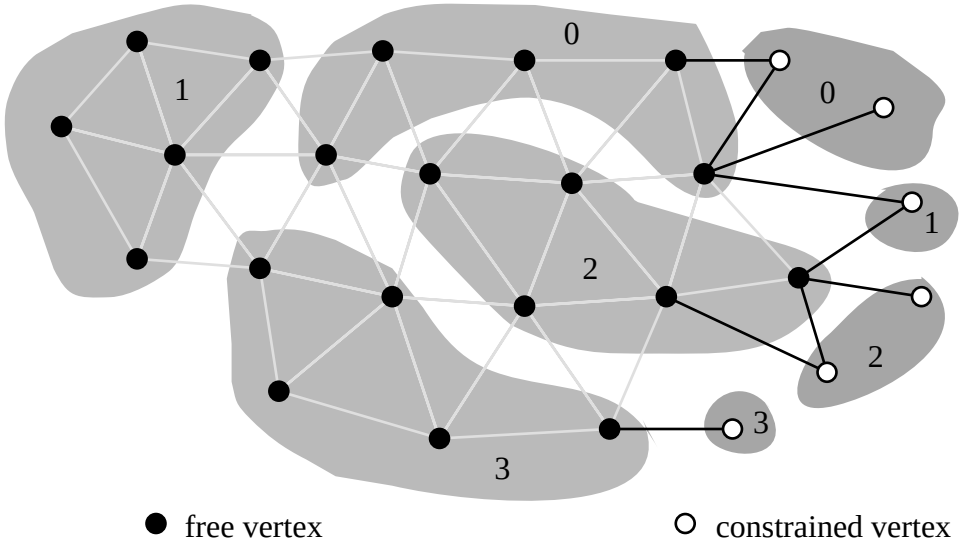


Figure 5.5: Illustration of the constrained k-way partitioning problem

in addition should minimize the migration cost which is now

$$(iii') \qquad \sum_{n \in \{m \in N'' | \pi(m) \neq \pi^0(m)\}} s(n) \quad \text{(Minimal migration cost)}, \tag{5.28}$$

since the constrained vertices are not assumed to change partitions.

5.2.2 Application to Mesh–Based Parallel Algorithms

In this subsection we consider how the abstract graph partitioning problems defined above can be utilized to solve the load balancing problem for a variety of mesh–based parallel applications such as the numerical simulator developed in this work. In particular we consider the class of unstructured hierarchical meshes as they have been defined in Section 4.1 including the possibility of local mesh refinement.

Non–Hierarchical Meshes. To start with, we consider an unstructured mesh in two or three space dimensions. Multiple element types are allowed but the mesh is assumed to be non–hierarchical, i. e. it exists of exactly one level E_0. Since our parallel solver described in Section 5.1 is based on a decomposition of the element set the load balancing problem amounts to solving a k-way graph partitioning problem with $N = E_0$, $k = P$ and the edge set $A = \{(e, e') |\ e \text{ and } e' \text{ are neighboring elements }\}$. The weight associated with each graph vertex (mesh element) can be used to balance multiple element types (e. g. one could make a quadrilateral twice as expensive as a triangle to roughly balance the matrix–vector operations) or types of computationally different elements (as for example in some computational mechanics problems). The edges of the input graph usually are assigned unit weight.

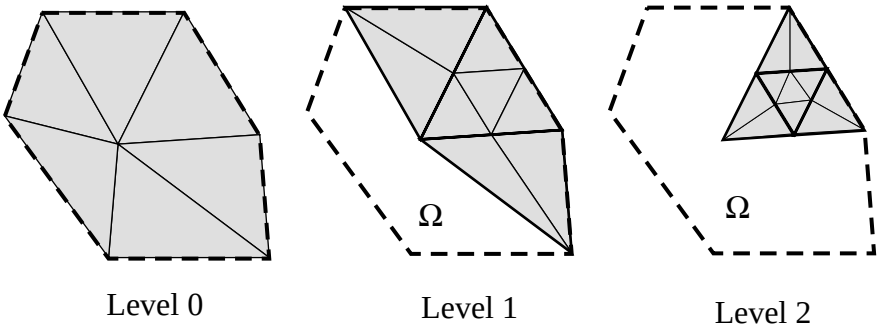


Figure 5.6: A locally refined hierarchical mesh.

We now consider the case of an adaptively refined non–hierarchical mesh. Although this form of mesh modification is not possible in our code we include it here for completeness. Assume that the mesh has been mapped to $k = P$ processors before refinement. Now the mesh is modified by *replacing* each of the elements to be refined by a set of smaller elements covering the same volume (area) of the original element. In a parallel version of the mesh refinement algorithm it is natural that the newly created elements are stored in the same processor as the element being replaced. Thus we obtain the initial partition map π^0 of a k–way graph repartitioning problem.

Locally Refined Hierarchical Meshes. A locally refined hierarchical mesh consists of a sequence of unstructured meshes $E_0, E_1, \ldots, E_J$ where each E_l, $l > 0$, is constructed from E_{l-1} by refining not necessarily all elements of E_{l-1} according to certain refinement rules. The construction is termed hierarchical since each element $e \in E_l$, $l > 0$, is associated with exactly one element $f(e) \in E_{l-1}$ (its father) such that e originated from refinement of $f(e)$. In contrast to Section 4.1 we allow that not all elements of some level E_{l-1} are refined and therefore E_l need not cover the whole domain Ω. Fig. 5.6 shows a locally refined mesh hierarchy with three levels in two dimensions.

The load balancing problem for a locally refined hierarchical mesh can be reduced to *a sequence* of constrained k-way partitioning problems as follows. The parallel multigrid algorithm developed above uses local communication between neighboring partitions on each mesh level in the smoother. This requires the work on each mesh level to be balanced over *all* processors. In typical applications (that require a parallel computer) it can be assumed that work increases exponentially with mesh levels making it most effective to have a good partitioning on the finest mesh levels. We therefore start with balancing the finest mesh level E_J first by solving a standard k-way partitioning problem as in the non–hierarchical case. Now consider the next coarser mesh level E_{J-1}. In the downward phase of a multigrid V–cycle the transfer of residuals from E_J to E_{J-1} essentially requires transfer of data from each element $e \in E_J$ to its father element $f(e) \in E_{J-1}$. The parallel version of it requires a communication whenever e and $f(e)$ are not assigned to the same processor. It is clear that the partitioning

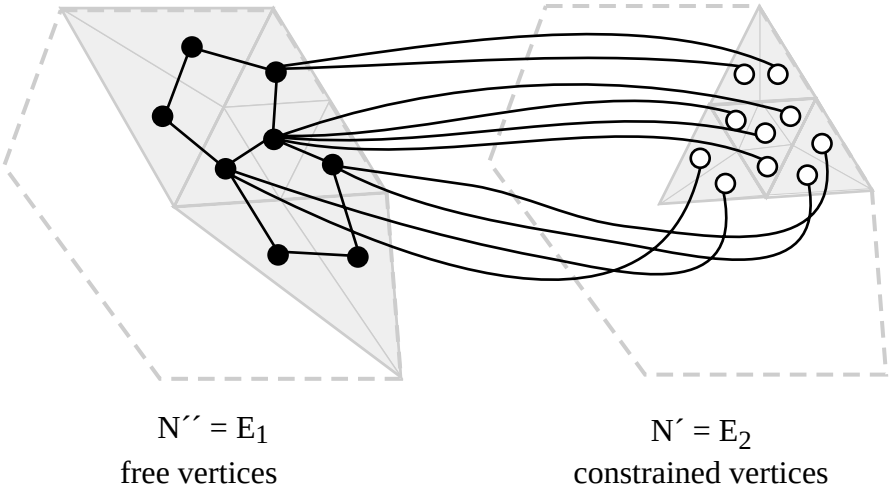


Figure 5.7: Constrained *k*-way partitioning problem input graph obtained from two consecutive multigrid levels.

of E_{J-1} must be related to that of E_J in order to minimize communication requirements in the grid transfer operation. Note that an unrelated decomposition of E_J and E_{J-1} may very well lead to the situation where data must be send from every fine grid element to every coarse grid element although each level itself may have a low separator weight.

The load balancing problem for E_{J-1} can be modeled by a constrained *k*-way partitioning problem by setting $G = (N, A)$, $N = N' \cup N''$ with $N' = E_J$, $N'' = E_{J-1}$ and the edge set

$$(e,e') \in A \Leftrightarrow \begin{cases} (1) & e,e' \in E_{J-1} \text{ and } e,e' \text{ are neighbors, or} \\ (2) & e = f(e') \vee e' = f(e) \end{cases} . \qquad (5.29)$$

Since smoothing is done more often than grid transfers the graph edges corresponding to condition (1) should have a higher weight than those originating from condition (2), e. g. 4 and 1 if $\nu_1 = \nu_2 = 2$ in the multigrid method. As an example, Fig. 5.7 shows the input graph for the constrained *k*-way partitioning problem that is used for partitioning level 1 from Fig. 5.6.

Obviously the same situation is encountered recursively for all coarser grid levels leading to the following general procedure which is called the *incremental mapping strategy*:

Solve k-way graph partitioning problem for finest mesh E_J;
for $l = J-1$ **downto** 0 {
  Solve constrained k-way partitioning problem with
  $N' = E_{l+1}$, $N'' = E_l$ and edge set from above;
}

In a parallel adaptive code the hierarchical mesh structure is distributed to the processors and modified in parallel. Refinement and coarsening, i. e. deletions

of previous refinements, may lead to modifications in all mesh levels except the coarsest. Naturally, newly created elements are stored in the same processor as their father element, cf. (Bastian 1996; Lang 1999) for details. In this context the partitioning should take migration cost into account leading to the *incremental remapping strategy*:

Solve k-way graph repartitioning problem for finest mesh E_J*;*
for $l = J-1$ **downto** 0 $\{$
  Solve constrained k-way repartitioning problem with
  $N' = E_{l+1}$, $N'' = E_l$ *and edge set from above;*
$\}$

Application–Dependent Clustering Schemes. In the process of constructing an input graph for a graph partitioning problem from a given finite element mesh (hierarchy) one need not associate a vertex of the input graph with every individual mesh element but one could associate it with a whole group of elements. We call this an application–dependent clustering scheme since it is handled outside the partitioners. Application–dependent clustering can considerably reduce the size of an input graph allowing a sequential solution with negligible cost (compared to the computation phase of the parallel algorithm), moreover it can often be done in parallel.

The hierarchical mesh construction described above allows several natural clustering strategies. E. g. , one can group together all elements on level l that have a common ancestor on level $\max(0, l-d)$ for some integer $d > 0$. A second strategy would cluster all elements on level l that have a common ancestor on level $l - (l \mod d)$ for $d > 0$ (an element is considered to be its own ancestor). The neighbor and father–son relations of the elements in the hierarchical mesh carry over to the clusters in the natural way. The second clustering strategy has the advantage of producing a particularly simple father–son relationship for all clusters in the range of levels $m \cdot d \ldots (m+1) \cdot d - 1$ for any $m \geq 0$: Every cluster has exactly one son. This construction has been used in Bastian (1996) and Bastian (1998) to derive a load balancing method for a multigrid hierarchy where within an incremental mapping strategy every coarse grid cluster is assigned to the same processor as its only son cluster. The remaining coarse grid clusters that do not have a son (have not been refined) are partitioned with a standard k-way graph partitioner. However, this partitioning step does not take into account edges connecting clusters that have a son to those that do not have a son. Moreover, when partitioning level $m \cdot d - 1$, $m > 0$, the father–son relation has simply been ignored in Bastian (1998) since a cluster on level $m \cdot d - 1$ can have up to 4^d son clusters in 2D and 8^d in 3D. Therefore, the algorithms based on the solution of the constrained k–way partitioning problem as outlined above are able to take more data dependencies into account than the algorithms given in Bastian (1998).

5.2.3 Review of Partitioning Methods

The k-way graph partitioning problem is considered to be a difficult combinatorial problem. Even the case $k = 2$ has been shown to be NP–complete, (Garey, Johnson, and Stockmeyer 1976), meaning that no polynomial time algorithm is likely to be found to solve this problem. Therefore emphasis has been laid on developing heuristic algorithms that can find a good solution in reasonable time.

The most well known of the early heuristics is that by Kernighan and Lin (1970). It is designed to iteratively improve an initial (random) load balanced bisection of the graph (i. e. a partitioning with $k = 2$). k-way partitionings are obtained by recursive application of the procedure. An efficient implementation of the Kernighan–Lin (KL) algorithm has been given by Fiduccia and Mattheyses (1982). In the 1980s a number of heuristics have been developed (Bokhari 1981; Fox 1986; Sadayappan and Ercal 1987) that identified the problem with (unstructured) finite element and sparse matrix computations on parallel computers. In the early 1990s the recursive spectral bisection method (Pothen et al. 1990; Williams 1990; Hendrickson and Leland 1992) emerged as a method that can find very good partitions (especially in combination with KL improvement) but which is somewhat expensive (it involves the computation of an eigenvector of a sparse matrix related to the input graph). Shortly afterwards the multilevel recursive bisection method has been introduced by Hendrickson and Leland (1993b). This method matches or improves the quality of recursive spectral bisection while having linear time complexity in its recent k-way variant (Karypis and Kumar 1995). With the development of the multilevel partitioning method the k-way graph partitioning problem is considered to practically solved. State-of-the-art implementations are available as free software libraries, the most well known being JOSTLE (`http://www.gre.ac.uk/~c.walshaw/jostle`) and METIS (`http://www-users.cs.umn.edu/~karypis/metis`). Even parallel versions are available (Karypis and Kumar 1996; Walshaw, Cross, and Everett 1997).

Most recently focus shifted towards the development of algorithms to solve the (unconstrained) repartitioning problem. Early attempts (Walshaw and Berzins 1993; Van Driesche and Roose 1995) tried to modify the spectral bisection algorithm, meanwhile the multilevel approach in combination with diffusion methods (Cybenko 1989) proved to be more successful (Schloegel, Karypis, and Kumar 1997; Walshaw, Cross, and Everett 1997).

In comparison, load balancing for adaptively refined hierarchical meshes has very seldomly been considered in the literature. In de Keyser and Roose (1991) and de Keyser and Roose (1992) an incremental mapping strategy is described that proceeds from fine to coarse meshes and remaps parts of the coarse grid by use of a cost function that models inter– and intra–grid communication. However, their grids were not truly local, i. e. every grid level covered the whole domain Ω. The work of Bastian (1993), Bastian (1996) and Bastian (1998) makes use of optimal–complexity multigrid methods and describes load balanc-

ing strategies for multiplicative and additive multigrid (which have different synchronization behavior) based on special clustering strategies. Klaas, Niekamp, and Stein (1994) implemented a parallel adaptive method with a hierarchical basis solver (a variant of additive multigrid). They used Cuthill–McKee ordering with subsequent blockwise column partitioning of the stiffness matrix for load balancing. Recently, Griebel and Zumbusch (1998) proposed to use space–filling curves for load balancing in an adaptive additive multigrid solver.

A particular problem in data parallel multigrid methods is the treatment of the very coarsest grids where the number of elements may not be large in comparison to the number of processors (or even less). In our implementation we are able to choose an appropriate number of processors for each mesh level separately.

5.2.4 Multilevel Schemes for Constrained k-way Graph (Re-) Partitioning

Introduction. In this subsection we extend the multilevel partitioning approach of Hendrickson and Leland (1993b), Karypis and Kumar (1995), Schloegel, Karypis, and Kumar (1997) and Walshaw and Cross (1998) to the k-way graph partitioning and repartitioning problems *with constrained vertices*.

In true multigrid fashion we first describe a two–level method. The two–level method first constructs a "coarser" version of the input graph by collapsing small groups of vertices into clusters which then form the vertices of the coarser graph. This process is very similar to the coarsening phase in aggregation–type algebraic multigrid methods for solving systems of linear equations.

Then the (re-) partitioning problem is solved for the coarser graph where it is less expensive. Now the coarse partitioning can be interpolated back to the finer graph in a canonical way by using the clustering. The partitioning of the fine graph can be further improved by employing an iterative improvement procedure, usually some variant of the KL algorithm or some simpler greedy method.

We obtain the multilevel method from the two–level method by applying the idea recursively for the coarse grid problem. Below each of the components of the multilevel partitioner is described in detail. We first concentrate on the constrained k-way partitioning problem and then move on to the repartitioning problem.

Coarsening Phase for Constrained Partitioning. The aim is to construct a sequence of "coarser" graphs $G_1, G_2, \ldots, G_J$ (J being the coarsest) with a decreasing number of vertices from a given input graph G_0.

Given an intermediate Graph $G_i = (N_i, A_i)$ the coarser graph $G_{i+1} = (N_{i+1}, A_{i+1})$ is constructed by collapsing vertices of N_i into so–called clusters. Each vertex of the coarse graph then uniquely corresponds to a set of vertices in the fine graph. This correspondence is described formally by the

cluster map $c_i : N_i \to N_{i+1}$. The cluster $C_i(n)$ of a vertex $n \in N_i$ is then the set $C_i(n) = \{n' \in N_i | c_i(n') = c_i(n)\}$.

The construction of the clusters is as follows. The constrained vertices $N_i' \subseteq N_i$ are clustered according to their partition assignment, i. e. for any $n \in N_i'$ we have $C_i(n) = \{n' \in N_i' | \pi_i(n') = \pi_i(n)\}$. These clusters make up the set of constrained vertices N_{i+1}' on the coarser level. For the clustering of the free vertices $N_i'' \subseteq N_i$ we first construct a *maximal independent set* M_i of N_i'', i. e. a subset of N_i'' such that no two vertices are joined by an edge and no vertex can be added without violating this condition. Good maximal independent sets can be constructed by greedy procedures. The use of a maximal independent set of the vertices produces faster coarsening than the maximal matching–based procedures normally used in multilevel partitioners. Then initially each vertex of the maximal independent set is assigned to its own cluster, the remaining vertices $N_i'' \setminus M_i$ are left unassigned. By doing so we will construct at least $|M_i|$ ($|.|$: number of elements in a set) different clusters which will have an average weight $\overline{W}_i = W''/|M_i|$. Furthermore, we define two gain functions that will be used in the heuristics below. Let $n, m \in N_i''$ be two neighboring vertices, i. e. $(n, m) \in A_i$ and m is already assigned to a cluster, then

$$\text{connectivity}(n, m) = \sum_{a \in \{(p,p') \in A_i | p = n \wedge p' \in C_i(m)\}} w_i(a) \tag{5.30}$$

sums the weights of all edges connecting n to the cluster of m. The second gain function measures connectivity with respect to a constrained vertex. In addition to vertices $n, m \in N_i''$ from above assume that $o \in N_i'$ is also a neighbor of n, then

$$\text{constraint–connectivity}(n, m, o) = \sum_{a \in \{(p,p') \in A_i | p \in N_i' \wedge \pi_i(p) = \pi_i(o) \wedge (p' = n \vee p' \in C_i(m))\}} w_i(a) \tag{5.31}$$

sums the weights of all edges that connect n and the cluster of m to the constrained vertices assigned to partition $\pi_i(o)$. Fig. 5.8 illustrates these definitions. With the given edge weights we get $\text{connectivity}(n, m) = 4 + 4 = 8$ and $\text{constraint–connectivity}(n, m, o) = 1 + 2 + 2 = 5$.

The remaining vertices $N_i'' \setminus M_i$ are assigned to clusters by scanning them in random order and applying the following heuristics:

1. Let n be the vertex to be assigned next. Check that n has at least one neighbor $o \in N_i'$ else go to 2. By construction of the maximal independent set n has at least one neighbor $m \in N_i''$ that already has been assigned to a cluster. If adding n to the cluster of m does not exceed the average weight $\overline{W}_i$ then do this. If more than one possible pair (o, m) exists choose the one which maximizes $\text{constraint–connectivity}(n, m, o)$ and adding n to the cluster of m does not exceed the average weight $\overline{W}_i$.

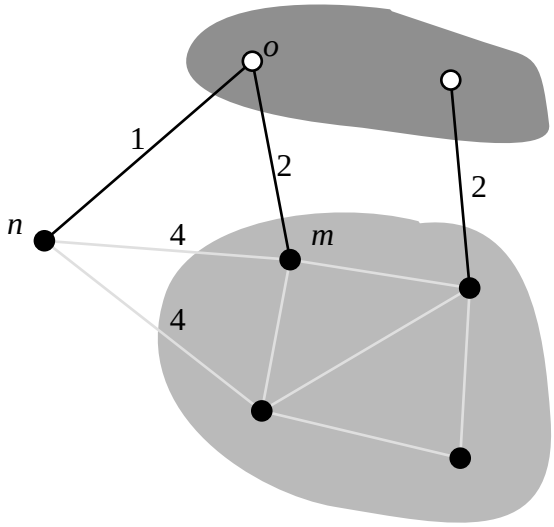


Figure 5.8: Illustration of connectivity and constraint–connectivity.

2. If none of the above applies, check all neighbors $m \in N_i''$ of n that already have been assigned to a cluster and choose the one that maximizes connectivity(n,m) and adding n to the cluster of m does not yield a cluster that has more than average weight $\bar{W}_i$.

3. If non of the above applies then n will be assigned to a new cluster.

After all vertices have been assigned to clusters the edge set, the weight functions and the partition assignment for constrained vertices of the coarse graph $G = (N_{i+1}, A_{i+1})$ are constructed as follows:

$$(u,u') \in A_{i+1} \Leftrightarrow \exists (n,n') \in A_i : c_i(n) = u \wedge c_i(n') = u', \tag{5.32a}$$

$$w_{i+1}(u) = \sum_{n \in \{n' \in N_i | c_i(n') = u\}} w_i(n), \tag{5.32b}$$

$$w_{i+1}\left((u,u')\right) = \sum_{(n,n') \in \{(m,m') \in A_i | c_i(m) = u \wedge c_i(m') = u'\}} w_i\left((n,n')\right) \tag{5.32c}$$

$$\pi_{i+1}(u) = j \Leftrightarrow \exists n \in N_i' : c_i(n) = u \wedge \pi_i(n) = j \tag{5.32d}$$

This ends the description of the coarsening step. Coarsening is applied recursively until a given number of vertices has been reached or the size of the graph cannot be sufficiently reduced. The target number of vertices is a small number (3 to 10) times k.

Coarsest Problem Solve. For the coarsest graph $G_J = (N_J, A_J)$ a constrained k-way graph partitioning problem has to be solved. We will do this in two steps. First a standard k-way graph partitioning problem is solved for the subgraph $G_J'' = (N_J'', A_J \cap N_J'' \times N_J'')$ consisting of the free vertices. This will result in a reasonable clustering of vertices but the assignment of partition numbers possibly will not be optimal. Think of the example shown in Fig. 5.5 but with the

partition numbers arbitrarily permuted. Therefore we will try to improve the partition number assignment with a KL–like algorithm in the second step.

Assume that a partition map π_J with corresponding partitions $N''_J(i)$, $0 \leq i < k$ has been computed. The elementary operation of our iterative improvement procedure will swap the partition number assignments of partitions i and $j \neq i$, i. e. it will redefine the partition map in the following way:

$$\text{swap}(i,j): \quad \forall n \in N''_J(i) : \pi_J(n) = j, \quad \forall n \in N''_J(j) : \pi_J(n) = i. \tag{5.33}$$

For any two partition numbers $0 \leq i, j < k$ we define

$$\text{constraint–sep–weight}(i,j) = \sum_{a \in \{(p'',p') \in A_J \mid p'' \in N''_J(i) \wedge p' \in N'_J(j)\}} w_J(a) \tag{5.34}$$

as the sum of weights of all edges connecting a vertex in partition i of the free vertices with any vertex in partition j of the constrained vertices. Possible candidates for partition i to be swapped with are

$$\text{swap–candidates}(i) = \{j \neq i \mid \text{constraint–sep–weight}(i,j) > 0\}. \tag{5.35}$$

The gain in total separator weight for swapping partition i with any $j \in \text{swap–candidates}(i)$ is given by

$$\begin{aligned}\text{swap–gain}(i,j) = {} & \text{constraint–sep–weight}(i,j) + \text{constraint–sep–weight}(j,i) \\ & - \text{constraint–sep–weight}(i,i) - \text{constraint–sep–weight}(j,j).\end{aligned} \tag{5.36}$$

A positive gain means an improvement in total cost.

The iterative improvement procedure consists of a number of iterations. Within each iteration a sequence of tentative swaps is constructed as follows:

1. Scan all pairs $(i,j) \in \{(a,b) \mid 0 \leq a < k \wedge b \in \text{swap–candidates}(a)\}$ and for each value $z \in \mathbb{Z}$ set up a list of all pairs with $\text{swap–gain}(i,j) = z$.

2. Select a pair (i_{max}, j_{max}) from the list with highest gain value, append it to the sequence of swaps and remove all remaining pairs $(i_{max}, .)$ from the lists. Do the swap, recompute all gain values and rearrange the lists. Note that the highest obtainable gain may be negative. Repeat until all the lists are empty (this will be after $k-1$ swaps).

3. Now the sequence of moves is reexamined. Let g_i be the gain obtained in the i-th swap of the sequence. Choose l such that $\sum_{i=1}^{l} g_i$ is maximal and positive. Restore the state of the partition map that has been obtained after the first l swap operations. If $l = 0$ then no improvement is possible and the algorithm ends, otherwise do another iteration.

Assuming that $|\text{swap–candidates}(i)|$ is bounded for all i one iteration of the algorithm above can be implemented with run–time proportional to k by using the bucket sorting idea described in (Fiduccia and Mattheyses 1982).

Projection Step. The first operation in the refinement phase is to transfer a partitioning from a coarse graph to the partitioning of the fine graph. This is simply accomplished by setting

$$\pi_i(n) = \pi_{i+1}(c_i(n)) \tag{5.37}$$

Iterative Improvement. Consider a graph G_i, $0 \leq i \leq J$ together with its partition map π_i that has been obtained by solving the coarsest level problem or that has been interpolated from a coarser graph. Due to nonuniform vertex weights it may be that the partitioning obtained so far does not satisfy the load balancing condition. In addition one might be able to improve the separator weight by moving vertices from one partition to another. We will now describe an iterative improvement procedure that tries to improve separator weight and load balance simultaneously. The algorithm follows the ideas presented in the work of Walshaw and Cross (1998). Previous algorithms tried to improve load balance and separator weight separately, see (Schloegel, Karypis, and Kumar 1997), but this does not seem to be necessary. Since the improvement procedure does not involve graphs on different levels we omit the level index in the following.

The algorithm to be developed now is again of KL–type with hill climbing ability. We begin by defining the local separator weight of vertex $n \in N''$ with respect to a partition i as

$$\text{local–sep–weight}(n,i) = \sum_{a \in \{(m,m') \in A | m = n \wedge \pi(m') = i\}} w(a), \tag{5.38}$$

i. e. the sum of weights of all edges that connect vertex n with a vertex in partition i. Note that the neighboring vertices include free vertices *and* constrained vertices! The elementary step in the optimization algorithm consists of moving a vertex $n \in N''$ from partition $i = \pi(n)$ to another partition $j \neq i$. The gain in separator weight associated with this move is

$$\text{move–gain}(n,j) = \text{local–sep–weight}(n,j) - \text{local–sep–weight}(n,\pi(n)) \tag{5.39}$$

The gain is positive if the separator weight will be smaller after the move. A vertex $n \in N''$ is only considered to be moved to one of its candidate partitions given by

$$\begin{aligned} &\text{move–candidates}(n) = \\ &\qquad \{j \neq \pi(n) | \exists (m,m') \in A : m = n \wedge m' \in N'' \wedge \pi(m') = j\}. \end{aligned} \tag{5.40}$$

All vertices n with $\text{move–candidates}(n) \neq \emptyset$ are called *border vertices*.

The weight of partition i in the free vertices is $W''(i) = \sum_{n \in N''(i)} w(n)$. We say that partition i is overweight if $W''(i) > T$ where $T = \delta W''/k$ is the target weight of a partition.

The iterative improvement procedure requires that the vertices N'' form a connected subgraph of G. This may not be the case in our application since each grid level of a locally refined mesh hierarchy need not cover the whole domain Ω. Therefore, if $(N'', A \cap (N'' \times N''))$ has non–connected components, additional edges with *weight zero* are introduced to ensure connectedness prior to optimization.

The optimization procedure consists of a number of iterations. Each iteration constructs a sequence of moves where each move transfers a vertex from its current partition to another partition. A vertex may only be transferred once in an iteration. We now describe the details of a single iteration:

1. Initialization. In order to reduce run–time only a limited set of vertices and destination partitions is considered in a single iteration, see (Schloegel, Karypis, and Kumar 1997; Walshaw and Cross 1998). In particular we set up a list of all pairs (n, j) where n is a border vertex and $j \in$ move–candidates(n).

2. Selection. Take the pair (n, j) from the list which maximizes move–gain(n, j). If several pairs have the same move–gain value take the one with *smallest* vertex weight $w(n)$ if move–gain$(n, j) \geq 0$ and *largest* vertex weight $w(n)$ if move–gain$(n, j) < 0$. This strategy maximizes the gain over several moves, see (Walshaw and Cross 1998).

3. Acceptance. Moving vertex n from partition $i = \pi(n)$ to partition j is accepted if one of the following conditions hold:

   (a) $\max_{0 \leq l < k} W''(l) > T$ and $W''(j) + w(n) < W''(i)$, or

   (b) $\max_{0 \leq l < k} W''(l) \leq T$ and $W''(j) + w(n) \leq T$.

   The first condition always accepts a move if global balance has not been reached yet and load balance is improved. If global balance has been reached the second condition accepts moves that do not violate the load balance condition. Remove pair (n, j) from the list of pairs to be considered. If (n, j) has been accepted then go to 4. If the list of pairs is empty then go to 5 else go to 2.

4. Confirmation and hill climbing. The algorithm has the ability to tentatively accept also a negative gain. If the current partition map π is "better" (see below) than a partition map $\bar{\pi}$ previously considered as "best"partition then it is confirmed to be the new best partition and the list of recent moves is cleared. If the current partition map is not better than the best partition map obtained so far then the last move (n, j) is appended to the list of recent moves. The current partition π is considered to be better than the previous best partition $\bar{\pi}$ if one of the following conditions holds:

(a) The separator weight associated with π is smaller than that of $\bar{\pi}$. Note that every individual move maintains or improves load balance.

(b) The separator weight is maintained but load balance is improved in the sense that the maximum weight of any partition has been decreased.

(c) The previous best partition $\bar{\pi}$ did not satisfy the load balancing condition and load balance is improved with π. Note that in this case we accept also an increase in separator weight.

If the list of pairs is empty then go to 5 else go to 2.

5. Undo recent moves. The end of an iteration has been reached. Undo all moves that are stored in the list of recent moves since they did not lead to an improvement in the sense of 4.

Iterations are executed until no improvement can be made or a prescribed number of iterations has been reached. The algorithm can be implemented with run–time proportional to $|N''|$ if the vertex degree of the input graphs is bounded, see (Walshaw and Cross 1998).

Multilevel Method for Constrained k-way Partitioning. We are now in a position to state the complete multilevel algorithm for solving a constrained k-way graph partitioning problem:

ALGORITHM 5.2 Multilevel method for constrained k-way graph partitioning problem.

Input: Graph $G = (N, A)$, $k > 1$, π *on* N' *and weights* w;
Set $G_0 = G$; $i = 0$;
while (G_i *not coarse enough*) {
  Coarsen G_i *to* G_{i+1} ;
  $i = i + 1$;
}
$J = i$; *Solve constrained k-way partitioning problem for* G_J;
Iteratively improve partitioning of G_J;
for $i = J - 1$ **downto** 0 {
  Project partitioning from G_{i+1} *to* G_i;
  Iteratively improve partitioning of G_i;
}

Extension to Repartitioning. The components of the multilevel algorithm given above can be readily extended to the case of repartitioning.

In the coarsening phase only vertices that are assigned to the same initial partition can be merged into a cluster. This allows a unique extension of the initial

partition map π^0 to the coarser graph. Moreover, coarsening can be done in parallel if desired. The coarse graph solve can be omitted since we can simply set $\pi_J = \pi_J^0$. Load balance is subsequently achieved through the use of the iterative improvement procedure described above. Data migration cost is implicitly kept low through the diffusion process (data will only be moved if load balance is improved). If the initial partitioning is not too much out of balance, load balance will be achieved quickly on the few coarsest graphs and the finer graphs will only be used to improve the partition quality.

6

UG: A Framework for Unstructured Grid Computations

The discretization schemes and solution algorithms described in previous chapters have been implemented in the partial differential equations (PDE) toolbox UG. In this chapter we take a somewhat broader look at the problem of writing a simulation software package.

The numerical solution of PDE problems on unstructured grids using parallel computers leads to an increase in software complexity of several orders of magnitude when compared to a sequential, structured mesh code. Consequently, the design of simulation software with respect to code reuse over problem domains is of great importance.

In the following we review the steps of the PDE solution process with respect to parallel computing and discuss the modular structure of the UG software toolbox. The object–oriented design of the numerical algorithms is discussed in some detail to give the reader an impression how new components can be incorporated into the UG framework.

Development of UG started in 1990 at the IWR, University of Heidelberg and proceeded at the ICA III, University of Stuttgart, from 1994. Meanwhile it consists of several hundred thousand lines of source code and has reached a rather mature state. The construction of such a large software package was only possible through the engagement of a large number of people (see author list on (Bastian, Birken, Lang, Johannsen, Neuß, Rentz-Reichert, and Wieners 1997)) and a very cooperative and unselfish style of work over the past years.

6.1 The PDE Solution Process

The numerical solution of partial differential equations involves a sequence of related steps starting with geometric modeling and ending with the visualization of the results as shown in Fig. 6.1. Arrows in the figure indicate the flow of control, links in gray are optional. Although the steps are the same for structured and unstructured grids as well as sequential and parallel computation, programming effort can vary from almost nothing to man–years, as e. g. in mesh generation.

In the following we comment each of the basic building blocks from Fig. 6.1:

Geometric Modeling. Holds a representation of the (three–dimensional) body in which the PDE is to be solved. Access to the representation must include methods to find points in the interior, on (internal and external) surfaces and on manifolds where two or more surfaces intersect.

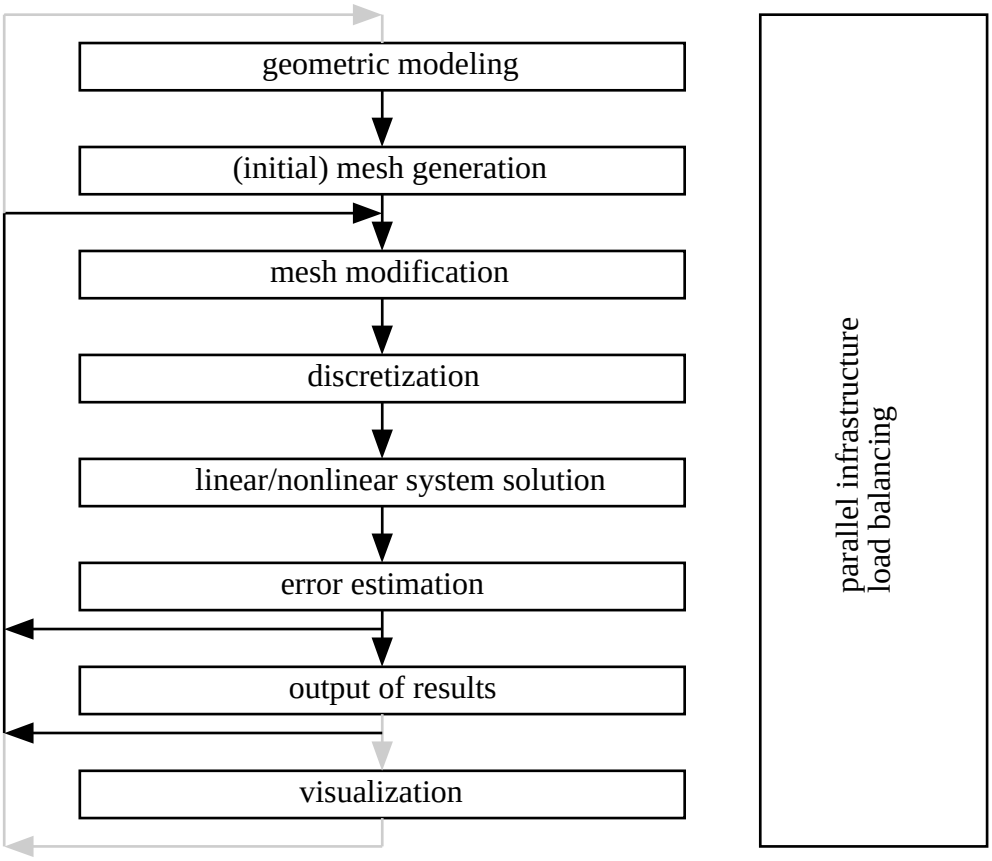


Figure 6.1: Basic building blocks of the PDE solution process.

Creation of the geometric model might be done with CAD software or special tools (e. g. generating internal surfaces from borehole data in a porous medium). In a parallel environment the geometric model might be duplicated on each processor if it is small enough, otherwise it has to be distributed together with the mesh data.

(Initial) Mesh Generation. Constructs a volume mesh approximating the domain given by the geometric model. Small details, e. g. a well or a tiny region of highly conductive material, must be resolved by the mesh if they are critical for the solution of the PDE. Other parameters to be controlled are mesh quality (angle condition), mesh size and anisotropy. In the parallel case load balancing/domain decomposition is notoriously difficult for this step.

Mesh Modification. Given a mesh, the purpose of this step is to construct a new mesh that is finer in some regions and possibly coarser in other regions of the domain without doing a complete remesh. The regions are indicated by the error estimator. A very effective way to do this is the hierarchical approach where individual elements of the given mesh are subdivided according to certain rules. Coarsening is achieved by recombination of previously subdivided elements. This results in local operations and a reasonably data–parallel implementation is possible, see Bastian (1996) or Jones and Plassmann (1997). Other techniques based on point insertion/deletion and mesh smoothing are also possible. Mesh modification requires dynamic load redistribution in order to balance the load after the refinement step.

Discretization. Sets up a finite–dimensional approximation of the differential equation. Operations are typically trivially parallel on element level. Difficulties in load balance might arise if different types of equations are to be solved in subregions or if elements require internal calculations (like in elastoplasticity).

(Non-)Linear System Solution. Large systems in 3D are typically solved with iterative solvers. It is important to maintain a low iteration count independent of the size of the mesh and the number of processors (and possibly other parameters). Multilevel and domain decomposition methods (often) have this property. Communication is required for every node that is stored on more than one processor. See Smith, Bjørstad, and Gropp (1996) for a good introduction.

Error Estimation/Refinement Strategy. Determine how accurately the discrete solution approximates the differential equation. Provide information where the mesh has to be refined or coarsened. Operations are typically parallel on element level requiring at most access to data in neighboring elements.

Output of Results. Store geometry/mesh/solution information to a disk file for subsequent restart or visualization. Huge amounts of data are pro-

duced by parallel computations necessitating the use of clever file formats (suppress redundant information) and parallel file I/O.

Visualization. Huge amounts of data are produced from the simulation of time–dependent processes on fine meshes. Although sequential visualization software can be improved to handle fairly large data sets (e. g. about five million nodes in GRAPE on a workstation with 1 GB of memory), ultimately also the rendering process will have to be parallelized.

The components of the PDE solution process

- need access to one or even several of the distributed data structures (geometric model, unstructured mesh, matrices and vectors) and
- are used in combination with each other: The solution drives the modification of the mesh in adaptive methods, visualization might be done during computation or the solution might change the geometric model.

At full scale this requires the incorporation of all components into an integrated environment. In order to ease the interaction between the components and to allow reuse of code for the different distributed data structures it is convenient to provide an abstraction such as a "distributed object" and operations for communicating among objects as well as mapping and migrating objects. This "parallel infrastructure" is drawn as a vertical box in Fig. 6.1 since it is intended to support all components.

6.2 Aims of the UG Project

No research group today possesses the fully integrated parallel PDE environment envisioned in the previous subsection. Due to lack in man–power and expertise we concentrated on the mesh modification, solver and parallel infrastructure parts. Distributed visualization has been implemented for its value as a debugging aid (see (Lampe 1997)) and parallel file I/O has been added as part of a project aimed at a production code, see (Fein 1998).

The main objectives of the UG project were:

- Research in numerical algorithms, especially
  - Robust multigrid methods on unstructured, locally refined meshes.
  - Parallel multigrid algorithms.
  - Solution of various PDE systems, a list of currently implemented problems is shown in Fig. 6.2.
- Research in software design

DIFFUSION EQUATION
- Linear conforming P1
- Quadratic conforming P2
- Linear non-conforming CR
- mixed RT0,RT1
- mixed BDM

LINEAR ELASTICITY
- Linear conforming P1
- Quadratic conforming P2
- Non-conforming (Falk)
- Stabilized BDM

ELASTOPLASTICITY
- Linear conforming P1
- Quadratic conforming P2

BIHARMONIC EQUATION
- Morley
- Argyris Element

NONLINEAR CONVECTION–DIFFUSION
- Finite–Volume
- Control–Volume FE

STOKES
- Taylor–Hood Element

NAVIER–STOKES
- Finite–Volume stabilized
- stationary–instationary
- compressible–incompressible
- laminar, turbulent ($k-\varepsilon$)

DENSITY DRIVEN FLOW
- Finite–Volume

MULTI–PHASE FLOW
- Finite–Volume
- Global Pressure
- Transition Conditions
- Fractured Porous Media
- Multicomponent Flow

Figure 6.2: PDE problems and discretizations currently implemented in the UG toolbox.

- Design of numerical algorithms such that they can be reused, composed in many ways and implemented with limited knowledge of the whole software.
- Design of a ‘parallel infrastructure’ that can manage a complex distributed data structure in a general way.
- Code should be portable from Macintosh to parallel supercomputer.

6.3 The UG Toolbox

This subsection first describes the modular structure of the UG software. Then some of the modules are described in more detail.

6.3.1 MODULAR STRUCTURE

The UG software is structured into several layers shown in Fig. 6.3. We will browse through the layers from bottom to top.

The Dynamic Distributed Data (DDD) layer provides the parallel infrastructure for creating and maintaining the distributed unstructured mesh data structure. It uses the Parallel Processor Interface (PPIF, a set of message passing

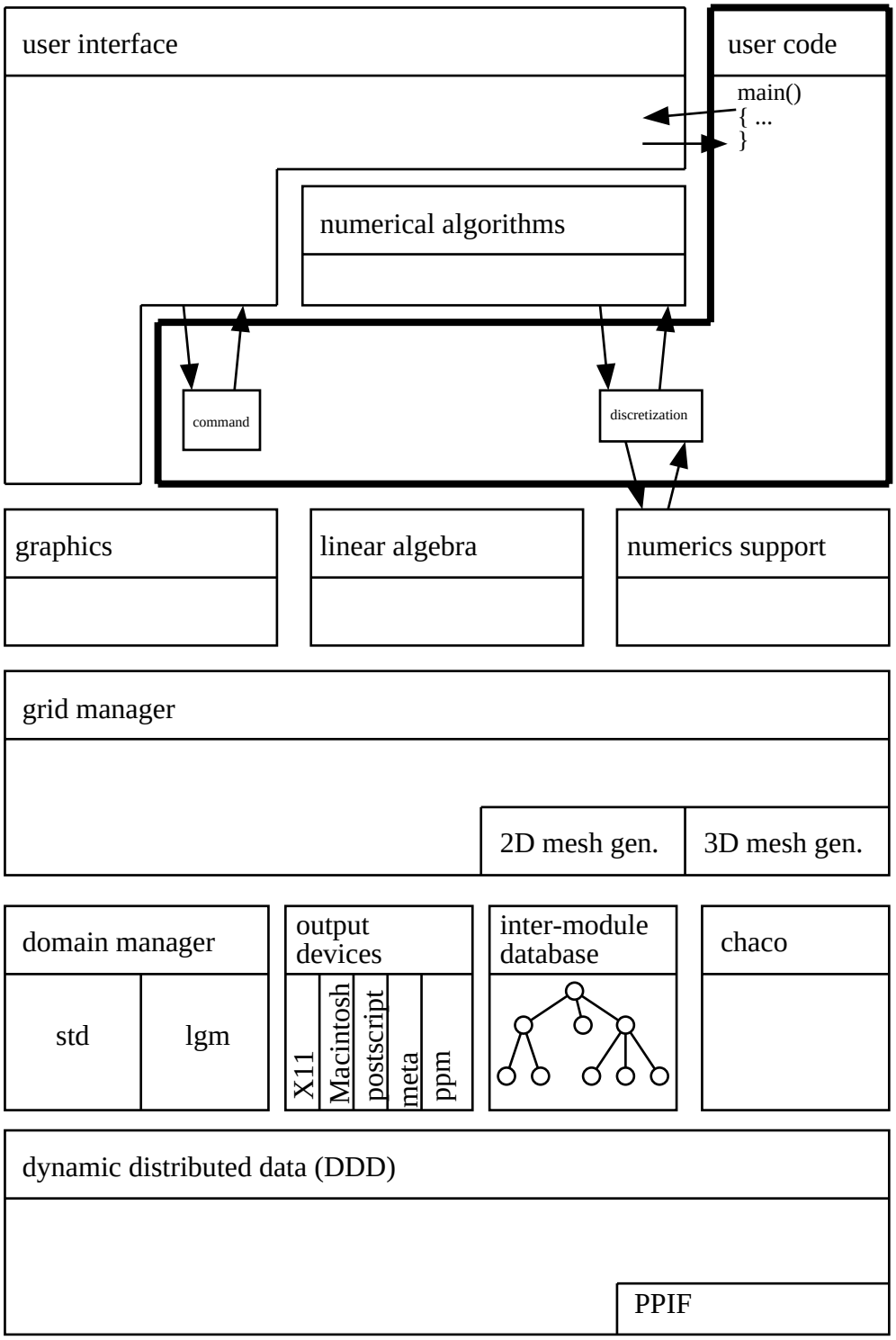


Figure 6.3: Modular structure of UG.

functions which have been implemented on top of PVM, MPI and many vendor specific message passing systems) for portability to many platforms. DDD is described in more detail below.

The next layer provides basic sequential functionality. The domain manager offers an abstract geometry interface to the grid manager. Two different implementations of this interface are available, the standard domain and the linear geometric model (both are described in more detail below). The output devices module offers a portable graphics interface which is implemented for X11, Macintosh, postscript and other formats. The inter–module database is used by modules to exchange data with each other in a standardized way. Finally, the graph partitioner CHACO, see (Hendrickson and Leland 1993a), has been included for use in the load balancing routines (DDD does not include the partitioning step, this has to be supplied by the code using DDD).

The grid manager module is responsible for creation and modification of the unstructured mesh data structure. Creation of initial meshes is done sequentially by 2D/3D advancing front mesh generators. The 3D mesh generator has been contributed by J. Schöberl, Linz, and is described in Schöberl (1997).

On top of the grid manager we have the graphics module enabling 2D and 3D visualization of meshes and solutions on planar cuts. Parallel 3D hidden surface removal is included, see (Lampe 1997). Graphical output can be sent to any output device. The linear algebra module provides kernels for sparse matrix–vector operations and iterative solvers. Numerics support includes useful functionality for many finite volume and finite element discretizations.

The numerical algorithms module provides a large variety of numerical methods such as linear solvers, nonlinear solvers, time–stepping schemes etc. From the point of view of the application programmer UG provides a framework (see (Gamma, Helm, Johnson, and Vlissides 1995)) for building specialized simulator applications. The numerical algorithms are implemented in a set of classes which can be used directly or from which the application programmer can inherit in order to add new components or to replace existing ones. In the implementation of his new classes the programmer can use functionality offered by other UG modules (e. g. numerics support) in the traditional form of subroutine libraries. The object oriented design of the numerical algorithms is described in detail in Subs. 6.4.

At initialization time the user application instantiates various objects to be used by the framework (such as geometry description or boundary conditions) and passes control to the user interface module. Numerical algorithm objects are typically instantiated from interpreted script files for flexible control of the solution process.

UG has been implemented in the C programming language. Most of its design follows the modular programming style, except the numerical algorithms which have been designed with object oriented methods.

6.3.2 DYNAMIC DISTRIBUTED DATA

The DDD layer provides the parallel infrastructure to create and maintain the distributed unstructured mesh data structure as well as the distributed sparse matrices and vectors. The underlying idea of DDD is that an arbitrary data structure (such as an unstructured mesh) can be identified with a directed graph where each node corresponds to an object (e. g. a vertex or an element) and each edge in the graph corresponds to a reference (pointer) from one object to another.

For the purpose of parallel processing we want to assign parts of the graph to different processors. Since we aim at distributed memory architectures a processor can only store an edge if it has also been assigned the two corresponding nodes (no pointers to objects in another processor's memory are possible). In order for each edge to be stored in at least one processor some nodes have to be stored in several processors, resulting in an overlapping decomposition of the graph. Different forms of overlap are possible and are determined by the needs of the application. Fig. 6.4 shows an example. Part (a) of the figure shows a simple graph that is to be distributed. Parts (b) and (c) show two different possibilities of overlapping decompositions. The overlap arises naturally in many data–parallel algorithms (often called "ghost cells") and also allows the sequential code to be reused on each processor. Objects that are stored on several processors are called *distributed objects* in DDD notation.

Data–parallel algorithms typically require information to be exchanged among different copies of a distributed object in order to maintain consistency. Since many objects may reside on a processor, DDD provides means to exchange data for whole sets of objects shared by pairs of processors. Lists of references to such sets of shared objects (called "interfaces" in DDD notation) are kept sorted by globally unique identification numbers of objects in order to quickly implemented the necessary gather/scatter operations to and from message buffers. Fig. 6.4(d) shows an example of an interface list.

The most powerful feature of DDD is its ability to dynamically migrate object copies from one processor to another while *automatically* updating the references to neighboring objects and the corresponding interface lists. E. g. one could move node 4_1 in Fig. 6.4(b) from processor 1 to processor 2. In order not to loose edge $(4_1, 2)$, one must include a copy of node 2 and edge $(4_1, 2)$ into the transfer. The result would be the distribution shown in Fig. 6.4(c). DDD would automatically figure out that a copy of object 4 already exists on processor 2 (i. e. 4_2), it would create a copy of object 2 on processor 2 and insert (correctly translated) pointers between objects 4 and 2 on processor 2. Finally, it would adjust the interface lists accordingly to enable subsequent communication. Note that it is the responsibility of the application to ensure that no reference is lost during a transfer operation.

DDD objects correspond to individual vertices or elements of the mesh data structure. All operations are designed to handle hundreds of thousands of ob-

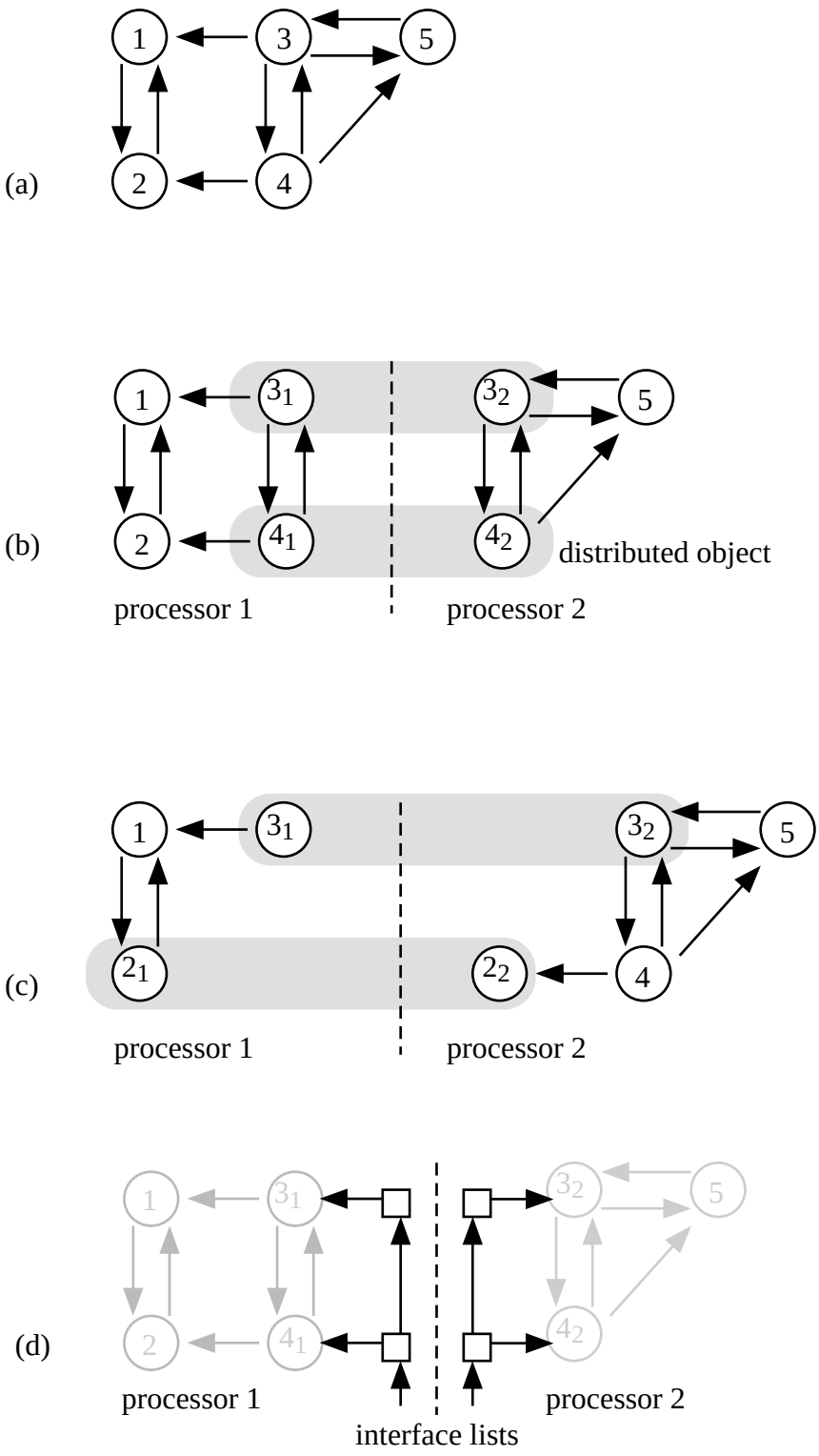


Figure 6.4: Concepts of Dynamic Distributed Data.

jects per processor efficiently. Memory overhead is 12 bytes in each object and an additional 12 bytes for each remote copy of an object. DDD only stores information about local objects and copies of these local objects on other processors, no component of DDD has global information about all objects.

DDD has been developed by Klaus Birken in his thesis, see (Birken 1998). The underlying concepts of DDD have been extracted from the first parallel version of UG described in Bastian (1996), see also (Birken and Bastian 1994).

6.3.3 GEOMETRY DEFINITION

The domain manager, see Fig. 6.3, provides an abstract geometry interface to the grid manager. This allows the grid manager to operate without knowledge of the actual representation of the geometry. Two implementations of the domain manager interface are currently available: The "standard domain" and the "linear geometric model" (LGM).

A standard domain consists of a piecewise description of the surface (boundary) of the domain. Each part of the surface (called a boundary segment) is given by a mapping of a parameter space (e. g. $[0,1]^2$) to $\mathbb{R}^3$. The mapping is supplied in the form of a suitable C-function. Consistency of boundary segments at intersections (which are points or one–dimensional manifolds) has to be ensured by the user. The standard domain interface is well suited to describe simple geometric forms like a cube, a sphere, a torus or a cylinder.

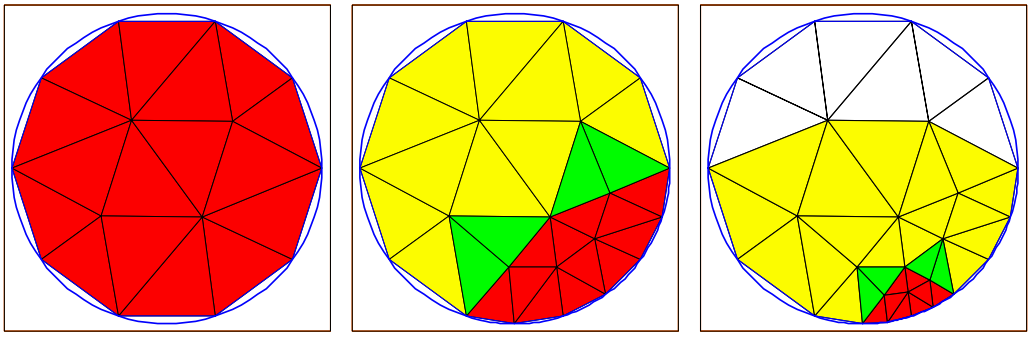
In the linear geometric model a domain is also defined by a piecewise description of the boundary. Each boundary segment, however, is represented as an unstructured triangular mesh in 3D space. The LGM is useful for domains with highly irregular surface allowing no parameterization.

Boundary condition information has to be supplied consistently with the geometry information, therefore it is also accessed through the abstract domain manager interface. This also makes the discretization code independent of the domain model.

If the standard domain has been used for the geometry definition then boundary conditions are given separately for each boundary segment using the same parameterization. In the linear geometric model boundary conditions are evaluated with respect to global coordinates (i. e. 3D space) for each boundary segment.

6.3.4 HIERARCHICAL MESH DATA STRUCTURE

The central idea of UG's approach to scalability is the use of a hierarchical mesh data structure. It is assumed that the geometry is simple enough to be duplicated on each processor and that a reasonable initial mesh can be constructed that is much coarser than the final mesh that is used to compute the solution of the differential equation. Thus it is possible to generate an initial mesh sequentially which is then distributed to (a subset of) the processors for (adaptive) refinement in parallel.



level 0 level 1 level 2

Figure 6.5: Three consecutive grid levels.

The mesh refinement is an extension of the algorithms of Bank, Sherman, and Weiser (1983) (2D, triangles) and Bey (1995) (3D, tetrahedra) to multiple element types (triangles and quadrilaterals in 2D, tetrahedra, pyramids, prisms and hexahedra in 3D). An efficient data–parallel implementation is enabled through a level–wise formulation (only elements of one grid level at a time are modified) and the use of a complete set of rules (there is a refinement rule for any possible refinement of edges and faces of an element), see (Bastian 1996) and (Lang 1999).

Besides in mesh generation, the hierarchical mesh structure is also of central importance to other steps of the PDE solution process: It is used to define a hierarchy of finite element spaces to be used in the multigrid solver, it can be used to obtain good initial guesses in the nonlinear solver (nested iteration) and it is useful for reduction of the complexity of the load balancing problem, see Bastian (1998). Furthermore, the hierarchical structure allows for tremendous savings in the size of output files, see (Fein 1998), and can be used for an efficient parallel solution of the 3D hidden surface problem, see (Lampe 1997).

We will now briefly consider the data structure used to represent the hierarchical mesh. It is described in more detail in (Bastian, Birken, Lang, Johannsen, Neuß, Rentz-Reichert, and Wieners 1997).

The `MULTIGRID` data type represents a complete hierarchical unstructured mesh consisting of several grid levels. A multigrid hierarchy with three levels is shown in Fig. 6.5. The six white elements on level 2 are *not* stored, i. e. grid levels need not cover the whole domain. Each grid level is accessible via the `GRID` data type which is an aggregate type holding elements (the `ELEMENT` data type), vertices (`NODE` and `VERTEX` data types) and edges (`LINK` and `EDGE` data type).

The mesh topology is given by the references between its components. The `ELEMENT` data type provides access to its corners of type `NODE` and to its neighboring `ELEMENT`s, see Fig. 6.6(a). The `NODE` data type stores a single linked list of references to neighboring `NODE`s as shown in Fig. 6.6(b). Each list element is of type `LINK` and two `LINK` objects are combined to form an `EDGE`.

Objects on successive grid levels are connected by the references shown in Fig. 6.6(c,d). Bidirectional references are stored between an element and each of its siblings. Each `NODE` has references to the corresponding `NODE` structure in the finer and coarser levels (or to an `EDGE` or `ELEMENT` if the `NODE` did not exist on the coarser level.

The same structure is also used in three space dimensions. *No* data type representing a face of the mesh exists in the three–dimensional version. Quantities related to faces (such as degrees of freedom) are referenced from each of both elements directly.

If an `ELEMENT` or `NODE` happens to be on the boundary of the domain it is supplemented by the corresponding boundary information. Topological information local to each of the six element types currently implemented is provided in a uniform way in order to being able to write code that is independent of the element type.

Typical operations on the data structure include browsing, tagging elements for refinement/coarsening and mesh modification.

6.3.5 Sparse Matrix–Vector Data Structure

In finite element or finite volume methods the solution of a PDE problem is approximated in a finite–dimensional function space equipped with a local basis. This means that any function in that space is determined locally on each element by degrees freedom related to that element and its faces, edges or vertices. Two elements share degrees of freedom at common vertices, edges and faces. Fig. 6.7 shows the degree of freedom layout for some finite element spaces. E. g. the $P_2 - P_1$ Taylor–Hood element (shown right in Fig. 6.7) for the Navier–Stokes equation approximates each component of the velocity vector with an elementwise quadratic function and the pressure with an elementwise linear function. In total this requires three numbers per node and two numbers per edge to represent the solution.

Typically, the whole solution process requires several solutions and/or right hand sides to be stored. Therefore the grid manager allows a variable number of floating point values to be associated with each geometric location (node, edge, face, element) at run–time. Note that degrees of freedom forming for example the solution vector are not stored in one big array but rather all floating point values related to a geometric location are stored in a small block. This prevents the use of efficient, array–based matrix–vector operations but on the other hand enables easy addition/deletion of degrees of freedom as the mesh is refined/coarsened. Furthermore it allows the direct use of DDD for the parallelization of matrix–vector operations. The efficiency issue is somewhat relaxed in the case of several degrees of freedom per location (systems of PDE), see (Neuß 1999) but it is likely that the sparse matrix data structure will be redesigned in future versions of the software.

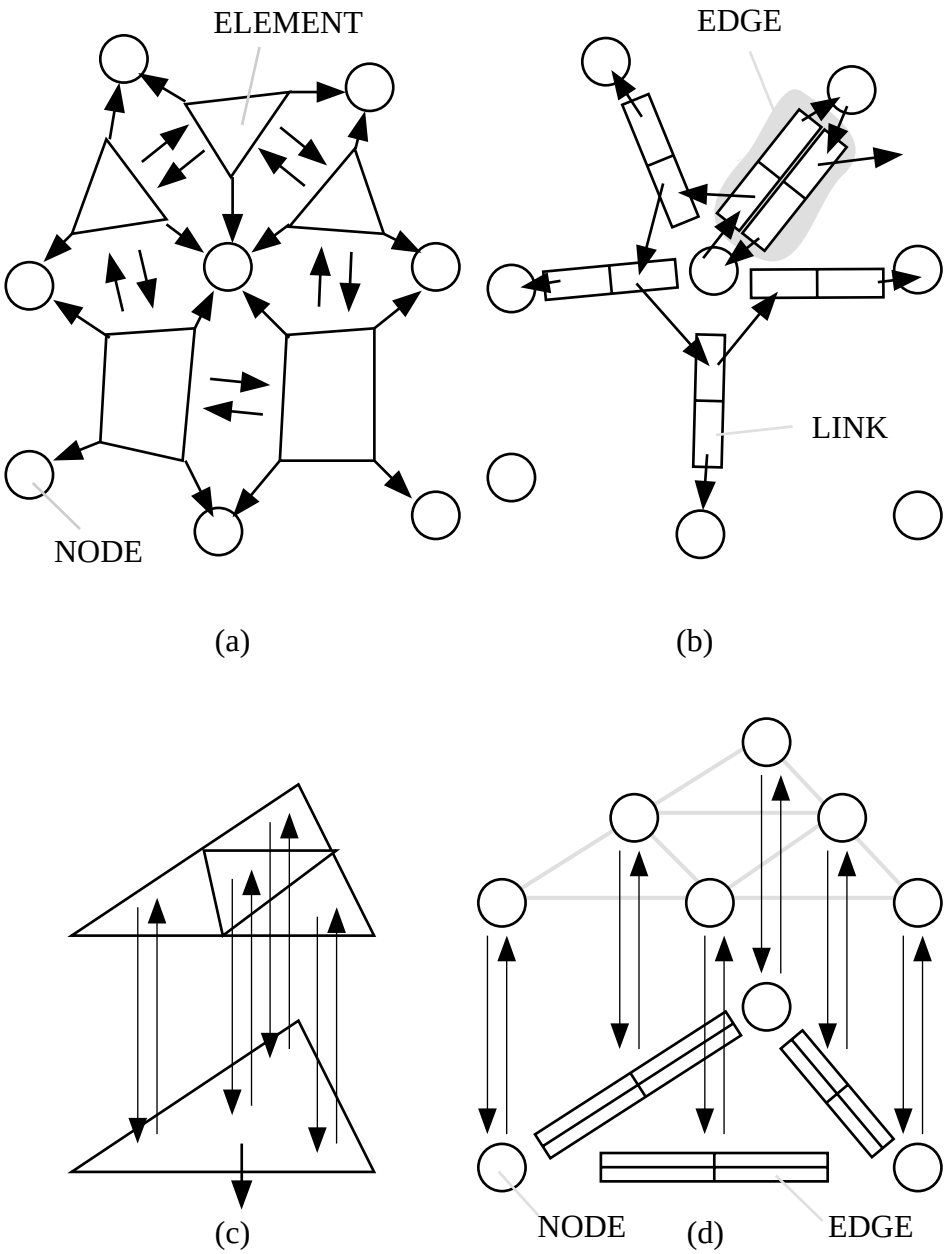
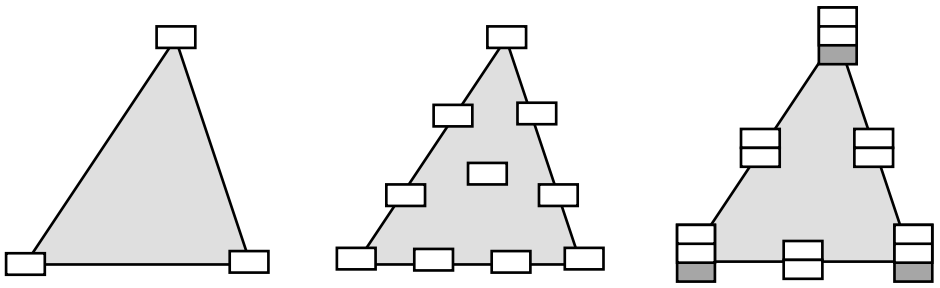


Figure 6.6: UG unstructured mesh data structure.



P_1: linear on triangle, globally continuous

P_3: cubic on triangle, globally continuous

P_2-P_1: velocity quadratic, pressure linear

Figure 6.7: Degrees of freedom for some finite element spaces.

Matrix entries are collected in blocks that couple all degrees of freedom in a geometric location with those in another geometric location. Each location stores a list of all matrix blocks coupling this location with other locations (i. e. a block compressed row storage scheme).

The `VECDATA_DESC` data type describes a collection of floating point values in one or several geometric locations to be treated as a single entity by the numerical algorithms. For the matrices a similar `MATDATA_DESC` data type exists. BLAS (basic linear algebra subroutines) level 1 and 2 routines as well as kernels for iterative methods operating on the `VECDATA_DESC` and `MATDATA_DESC` structures are available.

6.3.6 Discretization Support

Computation of stiffness matrices and right hand sides in the finite element or finite volume method requires local, element–wise calculations. Many components of these calculations can be reused across problem domains. The discretization support module provides:

- Various kinds of shape functions along with their derivatives for different types of elements.
- Transformation from local to global coordinate system.
- Quadrature formulae of varying order for all element types.
- A tool constructing the secondary mesh in the vertex–centered finite volume method.

6.3.7 Command Line Interface

In order to use UG, the application programmer has to write a `main()` function that initializes UG, registers application supplied code or data with the UG framework and passes control to UG's user interface.

Then the user can type commands interactively or execute sequences of commands from a script file. The set of available commands can be extended easily by writing an appropriate C–function and registering it as a new command with the command interpreter.

6.4 Object–Oriented Design of Numerical Algorithms

6.4.1 Class Hierarchy

The solution of nonlinear, time–dependent problems involves several cooperating numerical algorithms. E. g. an implicit time discretization requires the solution of a system of nonlinear algebraic equations per time step. Solving that by Newton's method requires the solution of a system of linear equations per iteration. As a linear solver one might consider a Krylov subspace method which requires a preconditioner, e. g. multigrid. A multigrid iteration needs a smoothing iteration, grid transfer operators and a coarse grid solver which in turn might be another preconditioned Krylov method or an algebraic multigrid scheme if the coarse grid is not small enough to solve the equations exactly. The Newton scheme may also require an interpolation scheme to transfer an initial guess from coarse to fine grid. In the adaptive case an error estimator is required. Even more complex scenarios can be imagined when using decoupled solution strategies for systems of PDEs.

The "numerical procedures" in UG have been designed to support this kind of flexible composition of solver components. In particular we wanted to have the following:

- Components should be reusable across problem domains. E. g. the time–stepping code should be the same regardless of the PDE to be solved.
- Components should not use outside knowledge. E. g. the nonlinear solver should not know whether it solves a nonlinear problem within a time–step or a stationary problem.
- The components should be configurable from script file to be able to quickly test different configurations.

In order to achieve these goals the numerical algorithms have been realized as a class hierarchy. The class diagram is shown in Figs. 6.8 and 6.9. Classes are denoted by rectangular boxes having the class name at the top. Classes with names in italics denote abstract classes, a class name in regular text denotes concrete classes. A line with a triangle denotes class inheritance, a regular arrow denotes usage (reference) of a class.

Abstract classes are used to define an interface, i. e. a set of functions with certain parameters and intended functionality. Functions of abstract classes are

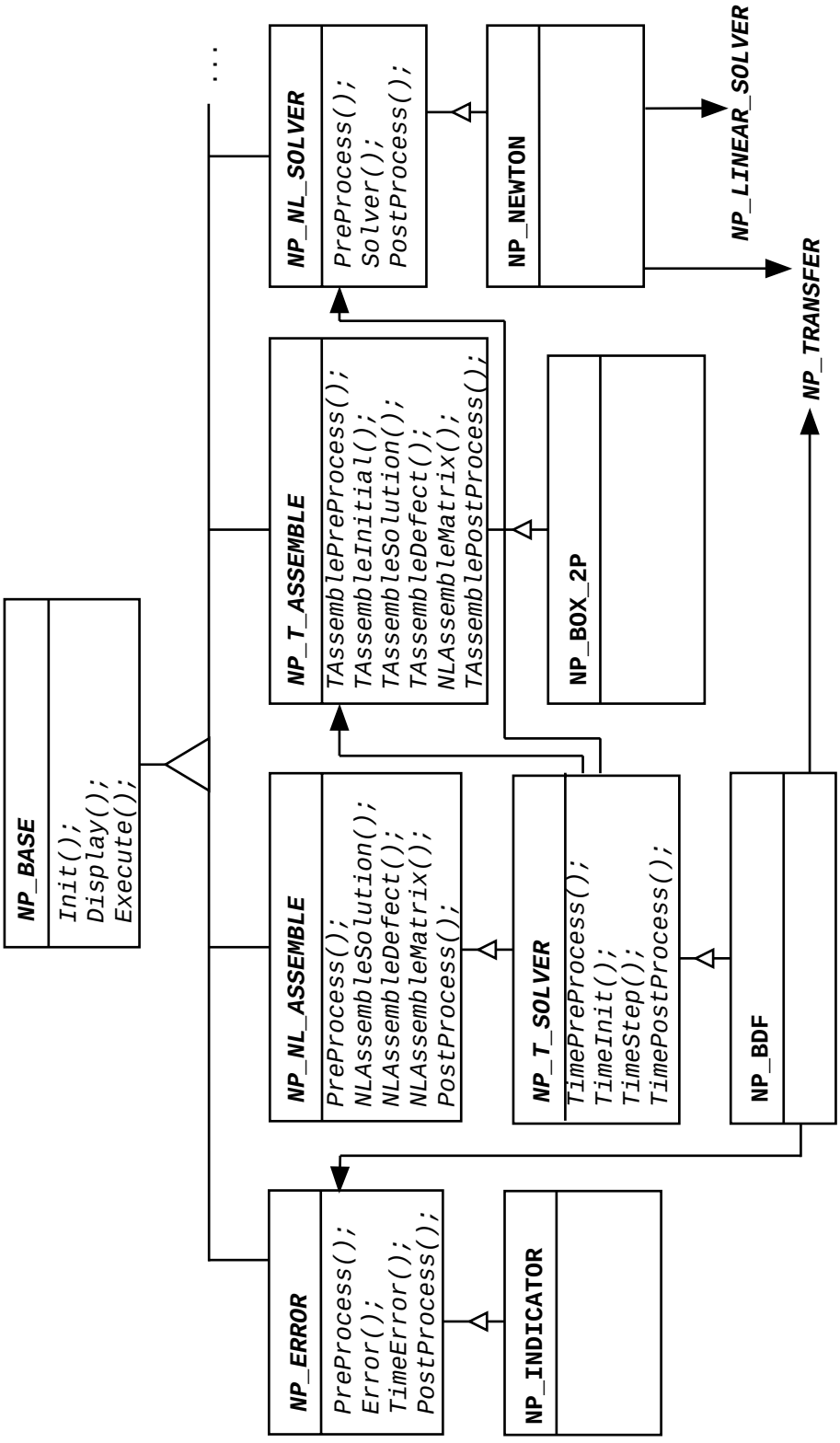


Figure 6.8: Numerical algorithms class diagram: Assemble, time–stepping and nonlinear solver classes.

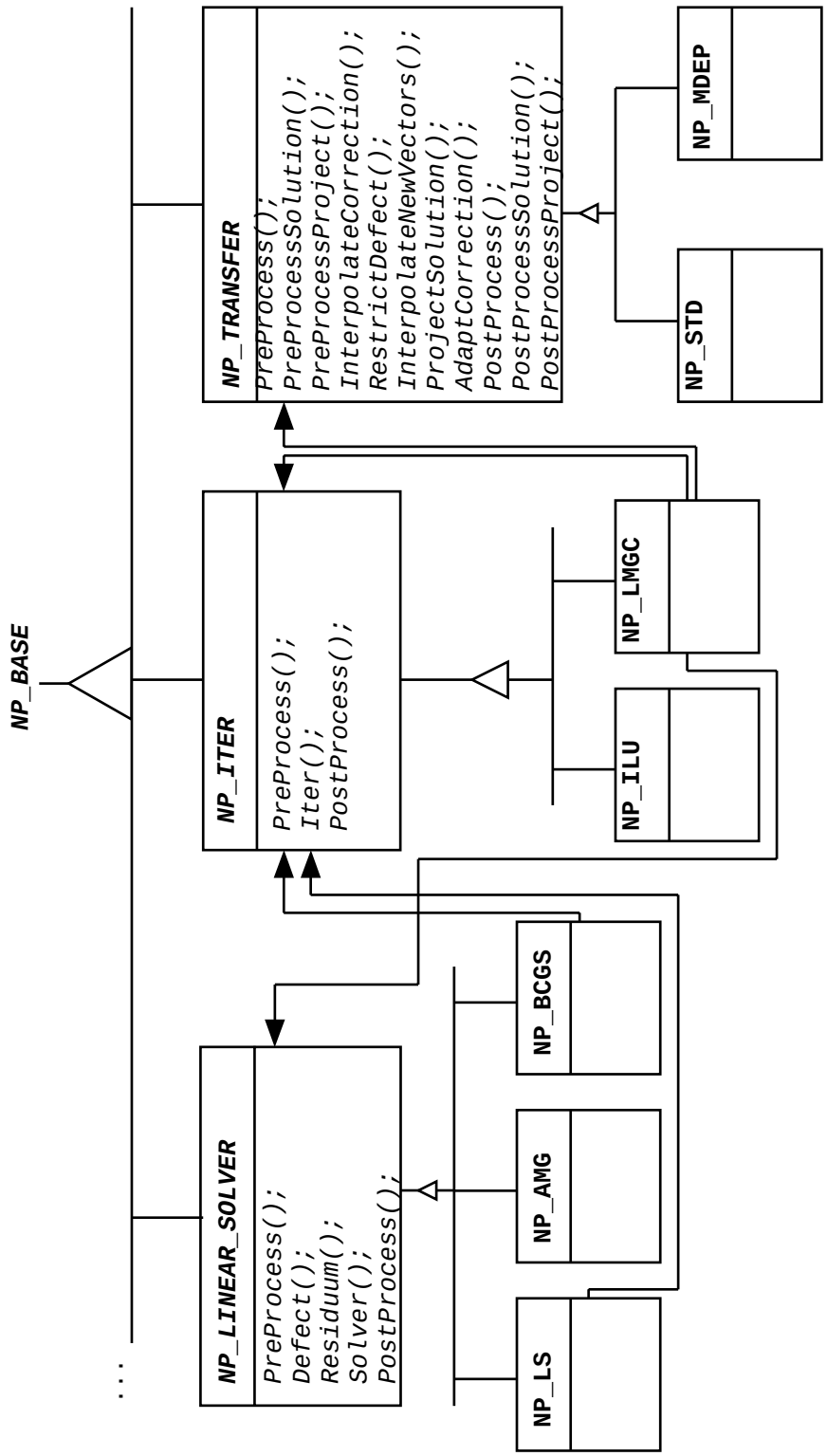


Figure 6.9: Numerical algorithms class diagram: Linear solvers and grid transfers.

virtual and are written in italic font. Concrete classes are derived from abstract classes and implement the interface given by the abstract base class. Typically there are several different implementations of an abstract interface that can be substituted at run–time (polymorphism). Classes can use other classes to implement their methods.
All numerical algorithms are derived from the abstract base class *NP_BASE*

NP_BASE
*int Init (int argc , char **argv);* *int Display ();* *int Execute (int argc, char **argv);*
MULTIGRID *mg; int status;

having three virtual member functions *Init()*, *Display()* and *Execute()* realizing the script file interface for the numerical component. *Init()* will be called by the command `npinit` and is used to set parameters of an object (such as the number of smoothing steps in a multigrid cycle). The *Display()* function is called by the `npdisplay` command and prints the current settings of an object. The *Execute()* member function is called by the `npexecute` command and triggers execution of a numerical algorithm (such as computing one time step). The *NP_BASE* class has two variables: A reference to the `MULTIGRID` data structure the object is supposed to work on and a `status` variable indicating whether the object is executable.

A few words about implementation may be in order here since UG is written in C, not in C++. Classes are implemented as structs containing function pointers. E. g. , *NP_BASE* is implemented as:

```
struct np_base {

    /* data */
    MULTIGRID *mg;
    int status;

    /* functions */
    int (*Init) (struct np_base *, int, char **);
    int (*Display) (struct np_base *);
    int (*Execute) (struct np_base *, int, char **);
};
typedef struct np_base NP_BASE;
```

Note that every member function receives a pointer to the object as first parameter (the `this` pointer). All function pointers are included in every instance of a class. A virtual function table has been omitted since memory requirements are not critical.

Inheritance is implemented by including the "base class" in the "derived class":

```
struct derived_class {
    struct base_class base;
    ...
};
```

We are now in a position to consider some classes in more detail.

6.4.2 Interaction of Time–Stepping Scheme, Nonlinear Solver and Discretization

Let us consider the solution of a system of ordinary differential equations (ODEs) in the form

$$\frac{d(\mathbf{m}(\mathbf{y}(t)))}{dt} = \mathbf{f}(t,\mathbf{y}(t)), \qquad \mathbf{y}:\mathbb{R}\to\mathbb{R}^N, \mathbf{y}(t^0)=\mathbf{y}^0. \tag{6.1}$$

Solving (6.1) with an implicit Euler scheme leads to the following nonlinear algebraic system to be solved in time step $n = 0,1,\ldots$:

$$\mathbf{F}_{IE}(\mathbf{y}^{n+1}) = \mathbf{m}(\mathbf{y}^{n+1}) - \Delta t\mathbf{f}(t^{n+1},\mathbf{y}^{n+1}) - \mathbf{m}(\mathbf{y}^n) = 0. \tag{6.2}$$

The second order backward difference formula and the Crank–Nicolson scheme lead to

$$\mathbf{F}_{BDF2}(\mathbf{y}^{n+1}) = \mathbf{m}(\mathbf{y}^{n+1}) - \frac{2}{3}\Delta t\mathbf{f}(t^{n+1},\mathbf{y}^{n+1}) - \frac{4}{3}\mathbf{m}(\mathbf{y}^n) - \frac{1}{3}\mathbf{m}(\mathbf{y}^{n-1}) = 0 \tag{6.3}$$

and

$$\mathbf{F}_{CN}(\mathbf{y}^{n+1}) = \mathbf{m}(\mathbf{y}^{n+1}) - \frac{\Delta t}{2}\mathbf{f}(t^{n+1},\mathbf{y}^{n+1}) - \mathbf{m}(\mathbf{y}^n) - \frac{\Delta t}{2}\mathbf{f}(t^n,\mathbf{y}^n) = 0, \tag{6.4}$$

respectively. We assume that the general form of the nonlinear system occurring in an implicit solution of (6.1) is

$$\mathbf{F}(\mathbf{y}^{n+1}) = \sum_{k=k_0}^{1} \left[\alpha_{n,k}\mathbf{m}(\mathbf{y}_{n+k}) + \beta_{n,k}\mathbf{f}(t_{n+k},\mathbf{y}_{n+k})\right] = 0 \tag{6.5}$$

with $\alpha_{n,1}$ normalized to 1.0. In order to decouple the problem–dependent part from the time–stepping scheme and the nonlinear solver the user basically has to provide two functions. The first function does one step of (6.5):

$$\mathbf{d} = \mathbf{d} + \alpha\mathbf{m}(\mathbf{y}) + \beta\mathbf{f}(t,\mathbf{y}) \tag{6.6}$$

for given α, β, t and $\mathbf{y}$. The second function is required to provide some linearization $J \in \mathbb{R}^{N\times N}$ of (6.5), e. g. the full linearization

$$(J)_{ij} = \frac{\partial \mathbf{m}_i}{\partial \mathbf{y}_j} - \beta_{n,1}\frac{\partial \mathbf{f}_i(t,\mathbf{y})}{\partial \mathbf{y}_j} \tag{6.7}$$

where $\mathbf{y}$ is the current iterate at time t. From a mathematical point of view this interface is general enough to allow a number of different time–stepping schemes such as those mentioned above but also the fractional step-θ–scheme and diagonally implicit Runge–Kutta methods. The linearization method (full Newton, Picard) and the way to compute the Jacobian (numerical, analytical) is completely up to the application and is not part of the interface.

In the code the interface to the time–dependent PDE problem is defined in the class *NP_T_ASSEMBLE*:

NP_T_ASSEMBLE
TAssemblePreProcess(from,to,t^{n+1}*,*t^n*,*t^{n-1}*,* $\mathbf{y}^{n+1}$*,*$\mathbf{y}^n$*,*$\mathbf{y}^{n-1}$*);*
TAssembleInitial(from,to,t^0*,*$\mathbf{y}^0$*);*
TAssembleSolution(from,to,t*,*$\mathbf{y}$*);*
TAssembleDefect(from,to,t*,*α*,*β*,*$\mathbf{y}$*,*$\mathbf{d}$*,*J*);*
TAssembleMatrix(from,to,t*,*β*,*$\mathbf{y}$*,*$\mathbf{d}$*,*$\mathbf{v}$*,*J*);*
TAssemblePostProcess(from,to,t^{n+1}*,*t^n*,*t^{n-1}*,* $\mathbf{y}^{n+1}$*,*$\mathbf{y}^n$*,*$\mathbf{y}^{n-1}$*);*

TAssemblePreProcess() and *TAssemblePostProcess()* are called at the beginning and end of each time step. Parameters *from* and *to* denote the range of grid levels the function should operate on. Other parameters are given in the mathematical notation. In the code, time values are of type *double* and vectors and matrices are of type VECDATA_DESC and MATDATA_DESC.

TAssembleInitial() fills the initial values of the ODE problem into the vector $\mathbf{y}$. *TAssembleSolution()* inserts Dirichlet boundary conditions at time t into the given solution vector $\mathbf{y}$ (required after calculation of an initial guess). *TAssembleDefect()* directly corresponds to (6.6). The linearization matrix may already be computed by *TAssembleDefect()* if this is more efficient (e. g. when using a fixed–point iteration with a nonlinearity of the form $A(\mathbf{y})\mathbf{y}$). Finally, the member function *TAssembleMatrix()* is used to calculate the linearization (6.7) (if not already done) and sets up the system of linear equations $J\mathbf{v} = \mathbf{d}$.

The discretization interface for *stationary* nonlinear problems of the form

$$\mathbf{F}(\mathbf{y}) = 0 \tag{6.8}$$

is given by the class *NP_NL_ASSEMBLE*:

NP_NL_ASSEMBLE
*PreProcess(from,to,***y***);* *NLAssembleSolution(from,to,***y***);* *NLAssembleDefect(from,to,***y***,***d***,J);* *NLAssembleMatrix(from,to,***y***,***d***,***v***,J);* *PostProcess(from,to,***y***);*

The interface is very similar to that in the time–dependent case with *NLAssembleSolution()* setting the Dirichlet boundary conditions in a given vector, *NLAssembleDefect()* computing $\mathbf{d} = \mathbf{F}(\mathbf{y})$ and *NLAssembleMatrix()* setting up the linear system. A nonlinear solver from class *NP_NL_SOLVER* expects an object of type *NP_NL_ASSEMBLE* as an argument to its *Solver()* member function:

*NP_NL_SOLVER::Solver(...,NP_NL_ASSEMBLE *problem, ...);*

The interaction between the time–stepping scheme, defined in class *NP_T_SOLVER* (see Fig. 6.8), and the nonlinear solver is as follows: *NP_T_SOLVER* is derived from *NP_NL_ASSEMBLE* and uses an object of type *NP_T_ASSEMBLE* to implement the *NP_NL_ASSEMBLE* interface for the nonlinear problem to be solved in a time step.

When the time–stepping scheme calls the nonlinear solver it passes *itself* as the *problem* parameter. When the nonlinear solver then executes a member function of the *problem* object, control will return to the time–stepping scheme which has all the information available in order to compute the defect and jacobian. Hence, the nonlinear solver does not need to know whether it solves a nonlinear problem within a time step.

6.4.3 Linear Solvers

The purpose of the linear solver is to solve a system of linear equations $A\mathbf{x} = \mathbf{b}$ to a given tolerance. The basic idea here is to split this task into a class *NP_LINEAR_SOLVER* that executes iterations given by class *NP_ITER* and checks convergence.

NP_LINEAR_SOLVER may be a simple loop (implemented by concrete class NP_LS or one of several Krylov subspace methods. Various implementations of the *NP_ITER* interface are available ranging from exact solvers (converging in one “iteration”) and single grid iterations to the multigrid method. The multigrid scheme uses grid transfers from the *NP_TRANSFER* class. The class diagram of the solver objects is given in Fig. 6.9.

6.4.4 Configuration from Script File

Fig. 6.10 shows part of a script file configuring a set of solver components for the solution of a nonlinear time–dependent problem. The npcreate command

```
npcreate transfer $c transfer;
npinit transfer $x sol $S 2.0;

npcreate box $c box2p;
npinit box $alphaw 1.0 $alphan 1.0 $inc 1.0E-8;

npcreate ilu $c ilu;
npinit ilu $damp n 1.0:1.0;

npcreate lu $c ex;
npinit lu $damp n 1.0:1.0;

npcreate basesolver $c ls;
npinit basesolver $abslimit 1E-10 $red 1.0E-3 $m 50 $I lu $display no;

npcreate lmgc $c lmgc;
npinit lmgc $S ilu ilu basesolver $T transfer $n1 2 $n2 2 $g 1 $b 0;

npcreate mgs $c bcgs;
npinit mgs $abslimit 1E-10 $m 40 $I lmgc $display red;

# nonlinear solver numproc to be used by time solver
npcreate newton $c newton;
npinit newton $abslimit 1E-10 $red 1.0E-5 $T transfer $S mgs
              $rhoreass 0.8 $lsteps 6 $maxit 50 $line 1 $linrate 0
              $lambda 1.0 $divfac 1.0E100 $linminred 0.0001
              $display red;

# the time solver
npcreate ts $c bdf;
npinit ts $y sol $A box $S newton $T transfer
          $baselevel 0 $order 1 $predictorder 0 $nested 0
          $dtstart 1.0 $dtmin 1.0 $dtmax 1.0 $dtscale 1.0
          $rhogood 0.01 $display red;

npexecute ts $pre $init;
step=0; steps=100;
repeat {
    step=step+1;
    npexecute ts $bdf1;
    if (step==steps) break;
}
```

Figure 6.10: Script file to configure numerical procedures.

instantiates a new object of the class given by the `$c` option. The `npinit` command sets the parameters of the named object. E. g the first two lines create and configure an instant of class `NP_TRANSFER`. Objects get references to other (already existing) objects as parameters, e. g. the initialization of the object `basesolver` (of class *`NP_LINEAR_SOLVER`*) contains a reference to object `lu` (of class *`NP_ITER`*) in its `$I` option. Correctness of the types is checked internally. The last object `ts` to be created is the time–stepping scheme. `ts` has references to the discretization object `box`, the nonlinear solver object `newton` and the grid transfer operator object `transfer`. The setting of initial values is done by the `npexecute ts $pre $init` command and the calculation of one time step is done by the `npexecute ts $bdf1` command in the repeat–loop.

The control of a simulation per script file is very convenient for the user. Parameters and solver components can be changed quickly or file output/graphical display can be added at the end of each time step.

6.5 Related Work and Conclusions

There exist several frameworks aimed at “Parallel Scientific Computing” in general such as the POOMA, see (POOMA Home Page 1998), and POET, see (POET Home Page 1998), software developed at Los Alamos National Laboratory and Sandia National Laboratory, respectively. These packages provide abstractions for data–parallel computations consisting of a number of communicating objects. POOMA offers three so–called “Global Data Types” which are N–dimensional arrays, banded matrices (those arising from finite difference schemes on structured meshes) and a general particle class. The techniques, however, seem to be suited only for rather coarse grained objects. POET, e. g., maintains a global data structure mapping data to processors. This is not acceptable on the level of individual vertices or elements of an unstructured mesh.

Parallel software for unstructured mesh computations is developed at several places. The work done at the SCOREC center at Rensselaer Polytechnic institute, see (SCOREC Home Page 1998), may be the most complete approach to an integrated environment for parallel unstructured grid computations. Several parallel mesh generators have been developed and complex PDE problems can be solved with adaptive finite element methods. In contrast to UG it does not use a hierarchical mesh structure, however, algebraic multigrid methods are available for fast solution of linear systems.

Diffpack, see (Diffpack Home Page 1998), developed at SINTEF and the University of Oslo emphasizes object–oriented design for code reuse. Parallelism and multi–level methods have been added recently, see (Cai 1998).

The FUDOP code, see (Mitchell 1998), features a new parallel multigrid method for adaptively refined meshes. FUDOP can refine, partition and redistribute in parallel and currently supports two–dimensional, triangular meshes. It uses hierarchical mesh refinement based on bisection.

Sumaa3d, see (Sumaa3d Home Page 1998), developed at Argonne National Laboratory offers sequential mesh generation as well as parallel mesh refinement and linear solvers. Despite its name, the parallel mesh refinement seems to be implemented only for two–dimensional, triangular meshes. It uses a hierarchical mesh structure with bisection refinement.

PadFEM, see (PadFEM Home Page 1998), developed at the University of Paderborn currently implements 2D/3D sequential mesh generation, parallel refinement of 2D triangular meshes and domain decomposition solvers. Diffusion based dynamic load redistribution algorithms have been developed.

The PETSc toolkit, see (Balay, Gropp, McInnes, and Smith 1997), provides parallel solvers for sets of linear and nonlinear equations as well as unconstrained minimization problems. It offers several efficient sparse matrix–vector formats on which the solvers operate. It does not include any mesh data structure or redistribution capability. These must be supplied by the application code.

This overview of software for unstructured grid computations is not intended to be complete. Nevertheless there are very few codes combining three–dimensional mesh generation/adaptive mesh refinement, dynamic redistribution capability and scalable numerical methods in a single environment. Capabilities needed for production type codes such as parallel file I/O and distributed visualization are virtually non–existing.

Due to lack of man–power and expertise probably no single group of researchers will ever have the fully–integrated parallel adaptive PDE software package. It is therefore mandatory to define standardized interfaces for the PDE software components such that each group can contribute modules from its area of expertise and use the modules of other groups in the remaining areas. Module interfaces should be flexible enough to allow competing implementations concentrating on different aspects such as speed, memory requirements or generality. Algorithms and data structures should be decoupled wherever possible. In an ideal environment it should be possible, e. g., to switch from structured to unstructured meshes without changing the code for the discretization.

The biggest challenges in the construction of such a software package are:

- Design for change. As new (numerical) algorithms are developed it should be able to incorporate them into the framework. This requires a lot of experience in the design of the interfaces.

- Combination of flexibility and efficiency. A general and flexible code is nice but if it is too slow nobody will use it. These contradictory goals can be achieved by combining a high–level object oriented approach with efficient low–level kernels.

7

Numerical Results

7.1 Introduction

7.1.1 Overview of the Experiments

In this chapter various numerical experiments are performed to illustrate the theoretical considerations concerning the different formulations of the two–phase flow problem and to show the behavior of the numerical algorithms. To that end one or several parameters (mesh size, processor number, other bad parameter) are varied for each experiment. The setup of the experiments is described in detail in order to enable others to verify the results and to provide a basis for comparison with other methods.

The following numerical experiments are performed:

Section 7.2 investigates several variants of a quarter five spot. The reservoir is two–dimensional and horizontal with capillary pressure being neglected in the simulation (hyperbolic case).

Section 7.3 is devoted to two–dimensional vertical DNAPL infiltration. Entry pressure effects in a porous medium with a single low permeable lens and in a medium with geostatistical permeability distribution (and corresponding entry pressure) are of primary importance here.

Section 7.4 covers the simulation of a medium–scale (6.5 by 2.5 meters) experiment performed at the VEGAS facility, see (Kobus 1996). This example is used to show the performance of the parallelization on up to 256 processors.

Section 7.5 treats DNAPL infiltration in three space dimensions. Up to 256 processors are used to do large scale simulations with more than 5 million unknowns.

Section 7.6 shows the application of the simulator to water–gas flow simulating air rising in a heterogeneous, water–saturated porous medium.

Section 7.7 extends the previous experiment to the three–dimensional case.

7.1.2 Parameters and Results

Most simulations are done with the same set of parameters referred to as “standard parameters”. Deviations from these settings are explicitly noted. The standard parameters are given by:

$\theta = 1$	Implicit Euler time–stepping
$\beta = 1$	Fully upwinding of mobility
$\varepsilon_{nl} = 10^{-5}$	Reduction in non–linear solver
$ls = 6$	Maximum number of line–search steps
nested	Nested iteration to obtain initial guess
$\varepsilon_0 = 10^{-4}$	Minimum reduction in linear solver
BiCGSTAB	Krylov subspace solver
MG	Multigrid preconditioner
ILU	Point–block ILU smoother
lexicographic	ordering of degrees of freedom
$\nu_1 = \nu_2 = 2$	Smoothing steps
$\gamma = 1$	V–cycle
$cut = 2$	Truncated restriction parameter

The following quantities are reported in the results for each numerical experiment (not all quantities may be listed for all experiments):

SIZE	Number of elements
S	Number of time steps
EX	Total execution time in seconds
N	Number of Newton iterations for all time steps
MG	Total number of multigrid cycles for all time steps
AVG	Average number of multigrid cycles per Newton iteration
MAX	Maximum number of multigrid cycles per Newton iteration
TN	Computation time per node and time step in milli–seconds
P	Number of processors
TI	Time for one multigrid cycle in seconds

7.1.3 Computer Equipment

Several different computers have been used to obtain the numerical results reported below. Sequential computations have been done on a Power Macintosh G3 with 266 MHz using the Metrowerks CodeWarrior IDE Version 2.1 with all optimizations on. Some sequential computations used a SGI Indigo^2 with 200MHz R4400 processor using the IRIX C compiler with -O2 optimization level.

Parallel computations have been performed on the 512 processor T3E system of HLRS in Stuttgart using Cray Programming Environment Version 3.0 and -O2 optimization level.

7.2 Five Spot Waterflooding

This section shows results for waterflooding of a two–dimensional horizontal oil reservoir. The characteristic feature of this problem is that capillary forces

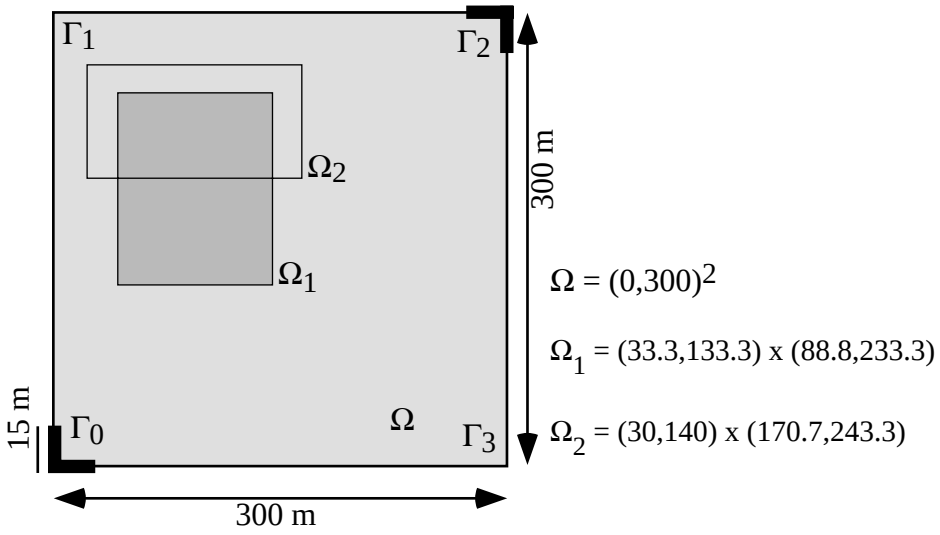


Figure 7.1: Geometry of the quarter five spot.

are neglected, i. e. the saturation equation is hyperbolic. We will investigate the case of a homogeneous permeability field and three cases with a heterogeneous permeability field.

Fig. 7.1 shows the geometry of the five spot problem which is the same in all variants. The reservoir is initially filled with oil (the non–wetting phase). Water is pumped in over Γ_0 and the oil exits the domain over Γ_2, i. e. the wells are implemented as flux–type boundary conditions for simplicity.

7.2.1 Homogeneous Permeability Field

Formulation

(p_n, S_w) with *PPS* method.

Boundary Conditions

$$\begin{aligned} \Gamma_0&: \quad \phi_n = 0, \phi_w = -0.0032\ [kg/(sm^2)] \\ \Gamma_1 \cup \Gamma_3&: \quad \phi_n = \phi_w = 0 \\ \Gamma_2&: \quad p_n = 10^5\ [Pa]\ , S_w = 0 \end{aligned}$$

Note: All boundary conditions are given in three–dimensional form. Computationally the reservoir is assumed to have a thickness of 1 meter, i. e. the inflow of $0.0032\ [kg/(sm^2)]$ over Γ_0 corresponds to an inflow of $8294.4\ kg/day$ in the lower left corner.

Fluid Properties

$$\rho_w = 1000\ [kg/m^3] \qquad \rho_n = 1000\ [kg/m^3]$$
$$\mu_w = 10^{-3}\ [Pa\ s] \qquad \mu_n = 20 \cdot 10^{-3}\ [Pa\ s]$$

Solid Matrix Properties

$\Phi = 0.2$, $K = kI$, $k = 10^{-10}\ [m^2]$.

Table 7.1: Performance statistics for homogeneous five spot simulation on a Power Macintosh G3.

S	SIZE	EX	N	MG	AVG	MAX	TN
50	80^2	694	151	313	2.1	4	2.17
50	160^2	2861	151	323	2.1	4	2.22
50	320^2	12005	151	360	2.4	5	2.34

Constitutive Relations

Brooks–Corey relative permeabilities with $S_{wr} = S_{nr} = 0$ and $\lambda = 2$, no capillary pressure.

Initial Values

$S_w = 0$, $p_n = 10^5$ $[Pa]$.

Mesh & Time Steps

The coarsest mesh (level 0) has 5 by 5 equidistant quadrilateral elements, the finest mesh used is refined six times yielding 320 by 320 elements with 103041 nodes and about 200000 degrees of freedom.

50 time steps of $\Delta t = 15$ $[days]$ are computed (final time 750 $[days]$).

Results

The left column of Fig. 7.3 shows the solution after 750 days of simulated time on the three finest meshes. The solution exhibits a rarefaction wave and a shock as can be expected from the Buckley–Leverett problem.

Table 7.1 shows the results for this simulation for varying spatial mesh size and fixed size of the time step. Standard parameters from Subs. 7.1.2 have been used.

The table shows that overall complexity scales linearly with the number of unknowns. The number of Newton steps on the finest mesh as well as the average and maximum number of multigrid steps per Newton iteration show *h*–independent behavior. The mesh independence of the nonlinear solution algorithm is achieved through the nested iteration technique. The Courant number is about five in the 320 by 320 computation. From the results on the Buckley–Leverett problem in Section 3.8 we expect the solution error to be dominated by temporal error. Nevertheless, the time step is held fixed to show the robustness of the linear and nonlinear scheme.

7.2.2 Geostatistical Permeability Field

The problem setup is the same as in the homogeneous case above except that the (isotropic) permeability field $k(\mathbf{x})$ is now position dependent and provided by geostatistical techniques. Two different permeability fields with 160 by 160 cells have been used with the following properties:

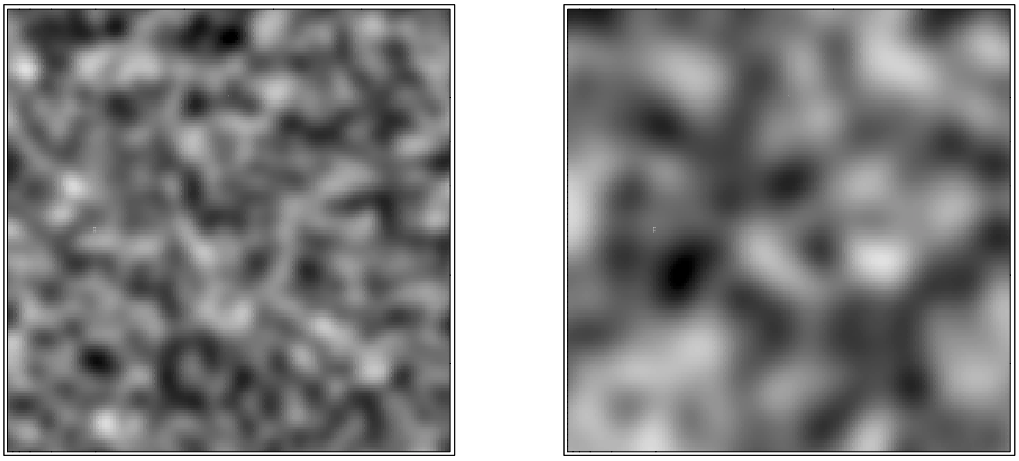


Figure 7.2: Heterogeneous permeability fields for five spot simulations. Mean value is $\bar{k} = 10^{-10}[m^2]$ with 2 orders of magnitude variation up and down. Resolution is 160×160 cells with correlation length 8 cells (left) and 16 cells (right). Darker values indicate lower permeability.

Name	correlation length	$\bar{k}$	k_{min}	k_{max}
C16	16 cells	10^{-10}	$10^{-11.8}$	$10^{-8.31}$
C08	8 cells	10^{-10}	10^{-12}	$10^{-7.98}$

The two permeability fields are visualized in Fig. 7.2. Permeability varies over four orders of magnitude. The corresponding solutions are shown in Fig. 7.3 and 7.4. 40 time steps of $\Delta t = 15\ [days]$ have been computed for field C16 and 45 time steps of the same size for C08. The Courant number is about 6 in the finest calculations. The mesh refinement study indicates that high spatial resolution is definitely needed for this type of problem. The comparison of two different time steps in Fig. 7.4 shows that temporal errors do not play a major rôle.

Solver statistics (standard parameters, see Subs. 7.1.2) for the geostatistical permeability field computations are given in Table 7.2. As in the homogeneous case the overall complexity scales linearly with problem size and the nonlinear solver as well as the linear solver show h–independent behavior. The time per node and time step (TN) indicates an increasing difficulty from the homogeneous case to the case with a correlation length of 8 cells.

7.2.3 Discontinuous Coefficient Case

This example is included to demonstrate the effectiveness of the truncated restriction in the case of discontinuities in the permeability field that are not aligned with coarse grid edges.

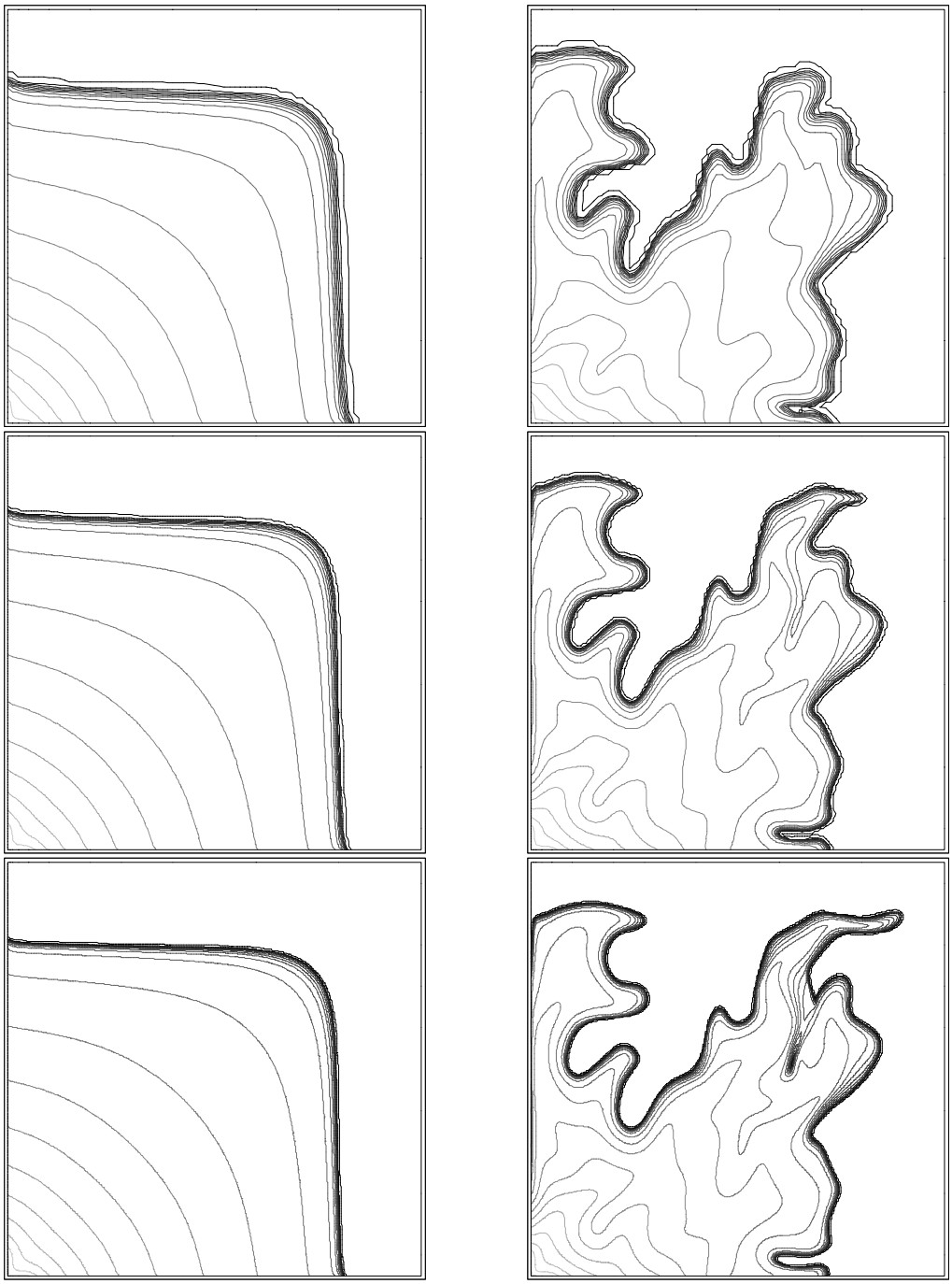


Figure 7.3: Quarter five spot simulation. Homogeneous permeability field with $k = 10^{-10}[m^2]$ left and heterogeneous permeability field with correlation length 16 cells right. Time step was $\Delta t = 15[d]$ and solution is shown after 50 steps in the homogeneous case and after 40 steps in the heterogeneous case. Spatial resolution is 80×80, 160×160 and 320×320 elements (from top). Contour lines are plotted in 0.05 intervals, first contour line is at 0.0001.

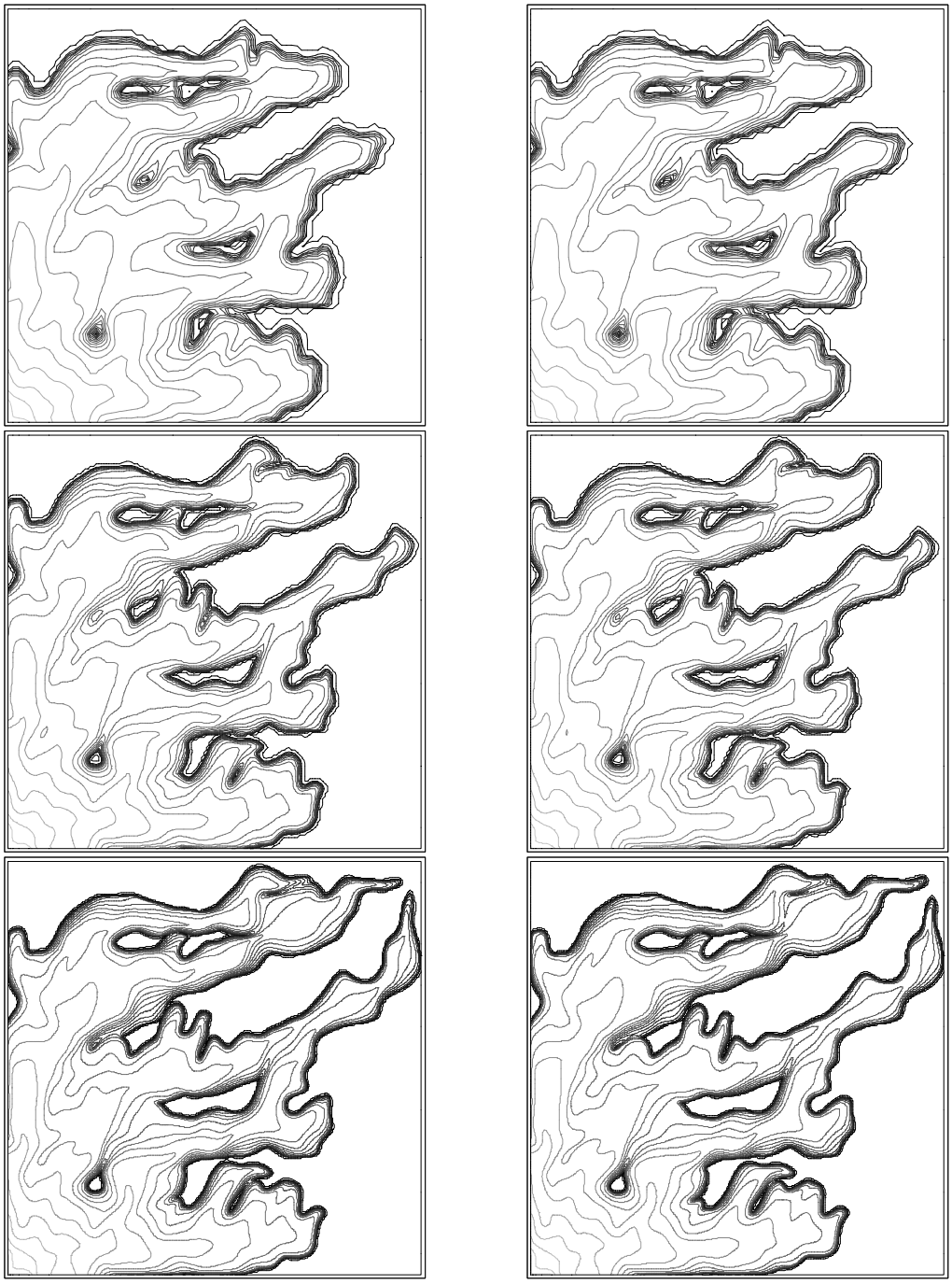


Figure 7.4: Quarter five spot simulation. Heterogeneous permeability field with correlation length 8 cells. Left column with $\Delta t = 15[d]$ after 45 steps and right column with $\Delta t = 7.5[d]$ after 90 steps. Spatial resolution is 80×80, 160×160 and 320×320 elements (from top). Contour lines are plotted in 0.05 intervals, first contour line is at 0.0001.

Table 7.2: Performance statistics for five spot simulation with geostatistical permeability field on a Power Macintosh G3.

Problem	S	SIZE	EX	N	MG	AVG	MAX	TN
C16	40	80^2	948	170	569	3.4	6	3.70
	40	160^2	4070	171	581	3.4	6	3.70
	40	320^2	17866	181	627	3.5	6	3.70
C08	45	80^2	1393	216	899	4.2	7	4.84
	45	160^2	5661	217	835	3.9	5	4.91
	45	320^2	24109	243	849	3.5	6	5.23

The problem setup is the same as in the cases above except for the permeability field which is given by

$$k(\mathbf{x}) = \begin{cases} 10^{-16} & \mathbf{x} \in \Omega_1 \\ 10^{-10} & \text{else} \end{cases}$$

and the initial values of saturation which are given by

$$S_{w0}(\mathbf{x}) = \begin{cases} 0.2 & \mathbf{x} \in \Omega_2 \\ 1 & \text{else} \end{cases} .$$

Subdomains Ω_1 and Ω_2 are defined in Fig. 7.1.

The solution for this problem is shown in Fig. 7.5 and solver statistics are given in Table 7.3. Standard parameters from Subs. 7.1.2 have been employed. Again the solver exhibits linear overall complexity. It should be noted that standard multigrid with discretized coarse grid operator diverges for this problem.

7.3 Vertical 2D DNAPL Infiltration

This section investigates several two–dimensional DNAPL infiltration model problems. The examples include gravitational and capillary pressure effects. In particular we will consider a case where both fluids are present at maximum saturation in the domain, furthermore the flow over a low permeable lens with

Table 7.3: Performance statistics for five spot simulation with discontinuous permeability field on a Power Macintosh G3.

S	SIZE	EX	N	MG	AVG	MAX	TN
25	40^2	119	103	308	3.0	5	2.98
25	80^2	571	118	335	2.8	4	3.57
25	160^2	2787	128	419	3.3	5	4.35
25	320^2	12284	119	469	3.9	5	4.80

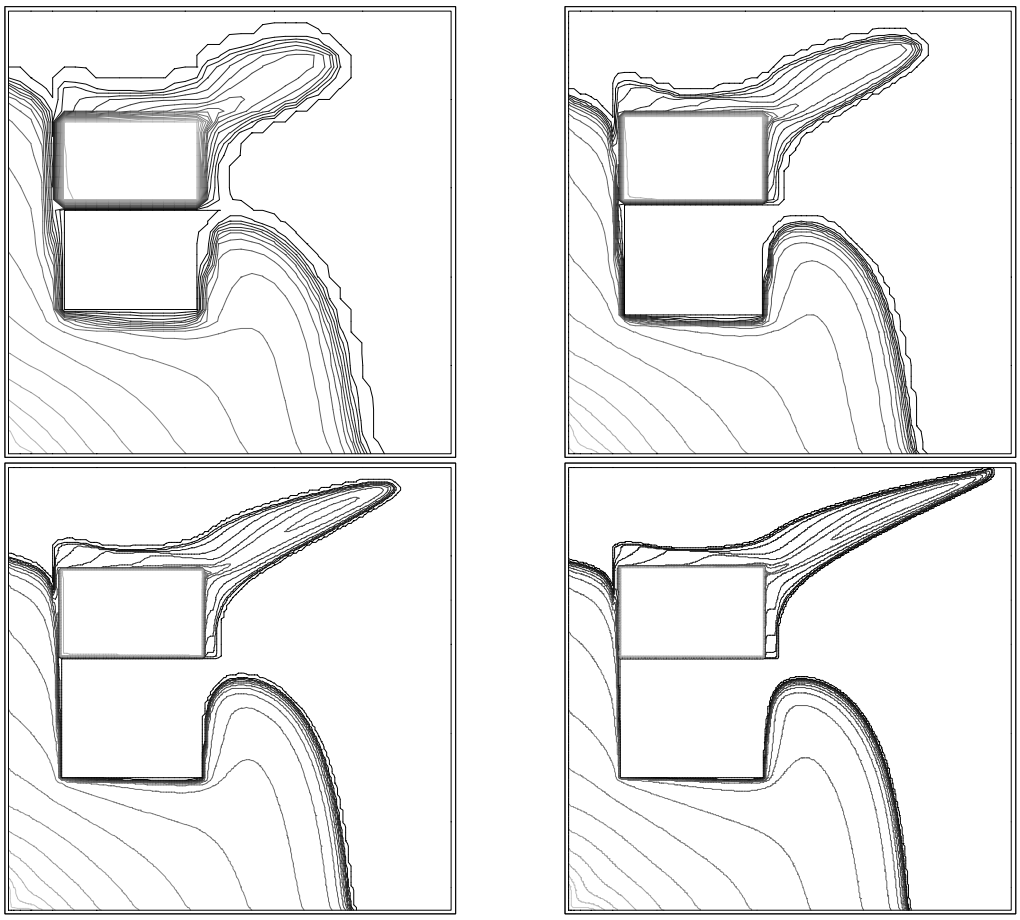


Figure 7.5: Quarter Five Spot simulation with low permeable region not aligned with coarse grid elements. Solution shown on 40^2 up to 320^2 elements after 25 time steps of $\Delta t = 15\ [days]$ (top left to bottom right). Contour lines are plotted in 0.05 intervals, first contour line is at 0.0001.

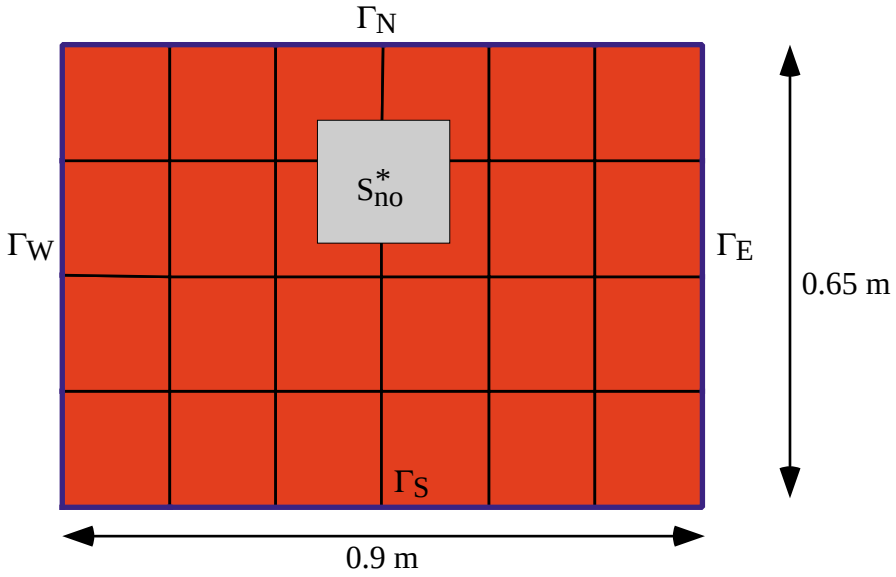


Figure 7.6: Geometry of the 2D DNAPL problem without low permeable lens.

and without infiltration of the lens and finally the flow in a medium with geostatistical permeability field where entry pressure changes from node to node.

7.3.1 Both Fluids at Maximum Saturation

The first example consists of a homogeneous, water–saturated porous medium. A rectangular region within the domain is assumed to be filled with DNAPL initially. Several simulations with increasing initial DNAPL saturation are performed to demonstrate the robustness of the global pressure formulations in contrast to a phase pressure formulation.

Fig. 7.6 shows the geometry of the domain and the coarsest level mesh. The problem parameters are now given in detail.

Formulations Used

(p_w, S_n) with *PPS* method, $(p, \mathbf{u}, S_n)$ with *GPSTV* method and $(p, \mathbf{j}, S_n)$ with *GPSTF* method.

Boundary Conditions

Boundary	*PPS*	*GPSTV*	*GPSTF*
Γ_N	$p_w = 10^5, \phi_n = 0$	$p = 10^5, \phi_n = 0$	$p = 10^5, \phi_n = 0$
Γ_W, Γ_E	$\phi_w = \phi_n = 0$	$\mathbf{u}\cdot\mathbf{n} = 0, \phi_n = 0$	$\mathbf{j}\cdot\mathbf{n} = 0, \phi_n = 0$
Γ_S	$\phi_w = 0, S_n = 0$	$\mathbf{u}\cdot\mathbf{n} = 0, S_n = 0$	$\mathbf{j}\cdot\mathbf{n} = 0, S_n = 0$

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\quad \rho_n = 1460\ [kg/m^3]$

$\mu_w = 10^{-3}\ [Pa\ s]$ $\quad \mu_n = 0.9 \cdot 10^{-3}\ [Pa\ s]$

Solid Matrix Properties
$\Phi = 0.4$, $K = kI$, $k = 6.64 \cdot 10^{-11}\ [m^2]$.

Constitutive Relations
Brooks–Corey relative permeabilities and capillary pressure with $S_{wr} = S_{nr} = 0$, $\lambda = 2$ and $p_d = 755\ [Pa]$.

Initial Values
Hydrostatic water and global pressure conditions are assumed initially (this is only used as an initial guess for the Newton method since both fluids are incompressible):

$$p_{w0}(x,y) = p_0(x,y) = 10^5 + (0.65 - y) \cdot 9810.0$$

and the initial DNAPL saturation is given by

$$S_{n0}(x,y) = \begin{cases} S_{n0}^* & 0.35 \le x \le 0.55 \wedge 0.4 \le y \le 0.55 \\ 0 & \text{else} \end{cases} .$$

Mesh & Time Steps
The coarsest mesh (level 0) has 6 by 4 equidistant quadrilateral elements. After six levels of uniform refinement a mesh with 384 by 256 elements and 98945 nodes is obtained.

50 $[s]$ of simulated time with a maximum time step size of $\Delta t = 10[s]$ are to be computed.

Results
Standard parameters from Subs. 7.1.2 have been used in the simulation with the following modifications: symmetric Gauß–Seidel smoother instead of the ILU smoother and nested iteration has been turned *off* after the first time step, i. e. the converged solution from the preceding time step is used as an initial guess for the next time step on the finest level.

Table 7.4 shows the results for an initial DNAPL saturation of 0.9, 0.99, 0.999 and 0.9999 and varying spatial mesh size. The results clearly indicate that the (p_w, S_n) formulation is not robust in this case as can be expected from the discussion in Subs. 2.1.3. Very small time steps are necessary in the phase–pressure formulation to obtain convergence of the nonlinear solver. In this context it is important to note that the Brooks–Corey capillary pressure curve has been regularized in a differentiable way by a straight line segment if effective water saturation is below $5 \cdot 10^{-5}$, a value not reached in the simulation here (this is to avoid an accidental division by zero).

Both formulations with global pressure show robust behavior at least with respect to the number of nonlinear iterations. The average number of multigrid cycles increases but much slower than for the phase pressure formulation. Total

Table 7.4: Performance statistics for vertical DNAPL infiltration with initial blob on a Power Macintosh G3. Level 3 is a 48×32 mesh and level 6 is a 384×256 mesh.

S^*_{n0}	level	*PPS*			*GPSTV*			*GPSTF*		
		S	N	MG	S	N	MG	S	N	MG
0.9	3	5	19	55	5	17	59	5	16	59
	4	5	25	95	5	26	92	5	19	71
	5	5	35	196	5	24	112	5	21	85
	6	5	56	472	8	48	327	5	26	115
0.99	3	5	52	285	5	19	73	5	17	63
	4	8	129	971	5	24	94	5	19	75
	5	31	409	2139	5	28	118	5	23	127
	6	> 150			6	39	341	5	26	123
0.999	3	9	165	2678	5	20	80	5	18	68
	4	> 75	> 500		5	26	109	5	20	79
	5	–			5	31	144	5	25	156
	6	–			5	35	345	5	32	175
0.9999	3	> 75			5	20	79	5	18	66
	4	> 500			5	26	112	5	20	79
	5	> 1000			5	31	221	5	27	265
	6	–			5	38	552	5	34	541

velocity and total flux formulation give virtually identical results although capillary pressure is not completely eliminated from the pressure equation in the total flux formulation. Fig. 7.7 shows pressure and saturation plots after $50[s]$ of simulated time indicating the strong coupling of pressure p_w and saturation S_n in the phase pressure formulation and the weak coupling in the global pressure formulation.

7.3.2 Flow Over a Low Permeable Lens

The main purpose of this subsection is to compare the *PPS* and *PPSIC* formulations for a porous medium with a discontinuity as described in section 2.3. Two different cases of capillary pressure functions are considered. In the first case the critical saturation is not reached whereas in the second case the critical saturation is reached and infiltration occurs.

Fig. 7.8 shows the geometry of the single lens problem. Numerical computations and experiments for a similar problem are reported in (Helmig 1997). The problem setup is now described in detail.

Formulation

(p_w, S_n) with *PPS* and *PPSIC* methods.

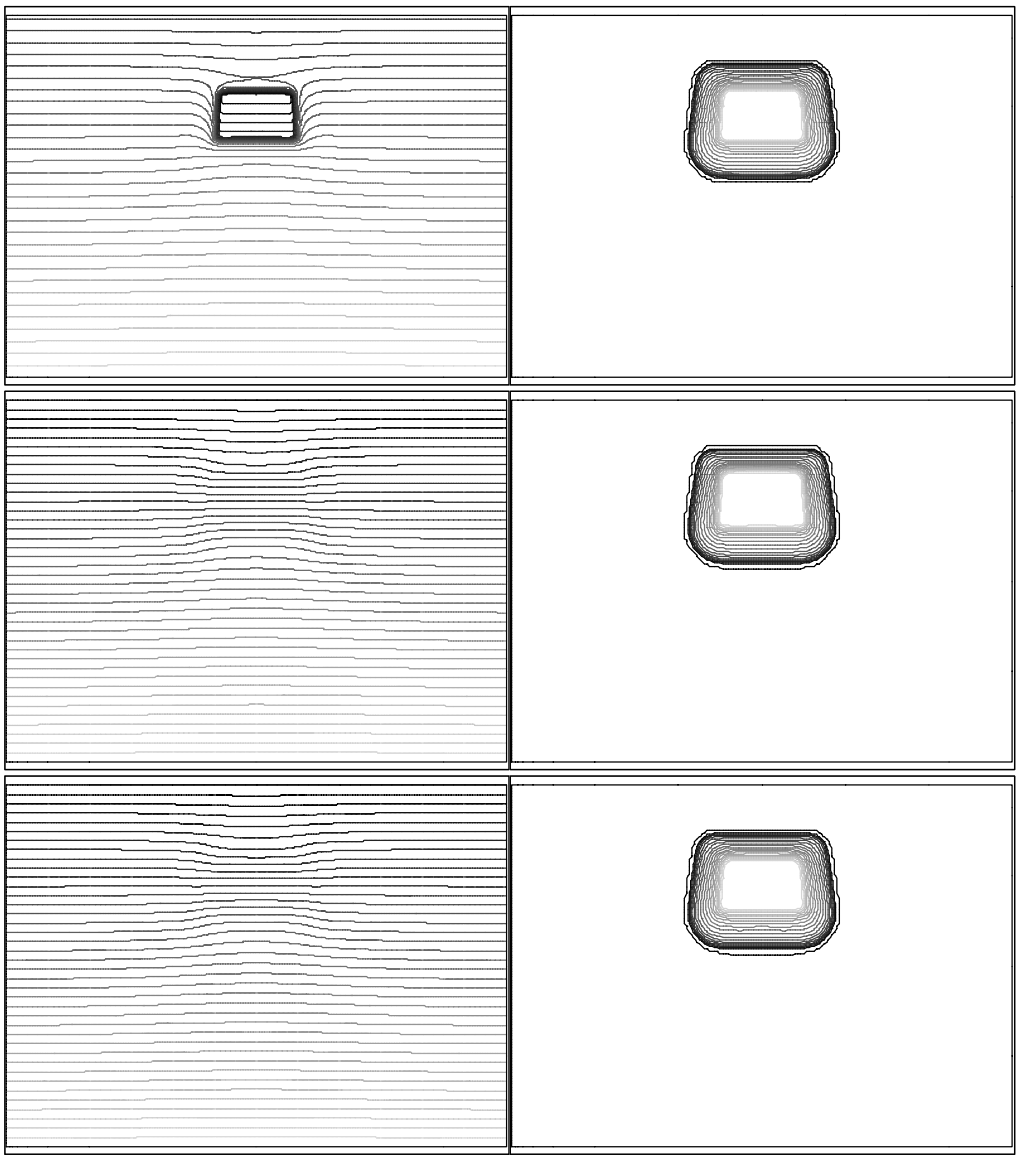


Figure 7.7: Solution of the 2D DNAPL infiltration problem with an initial blob of DNAPL in a rectangular region of the domain after $50[s]$ of simulated time. Initial saturation was 0.99 in this case. Phase or global pressure shown left and DNAPL saturation right. Top plot is from *PPS* method, middle plot is from *GPSTV* method and bottom plot is from *GPSTF* method.

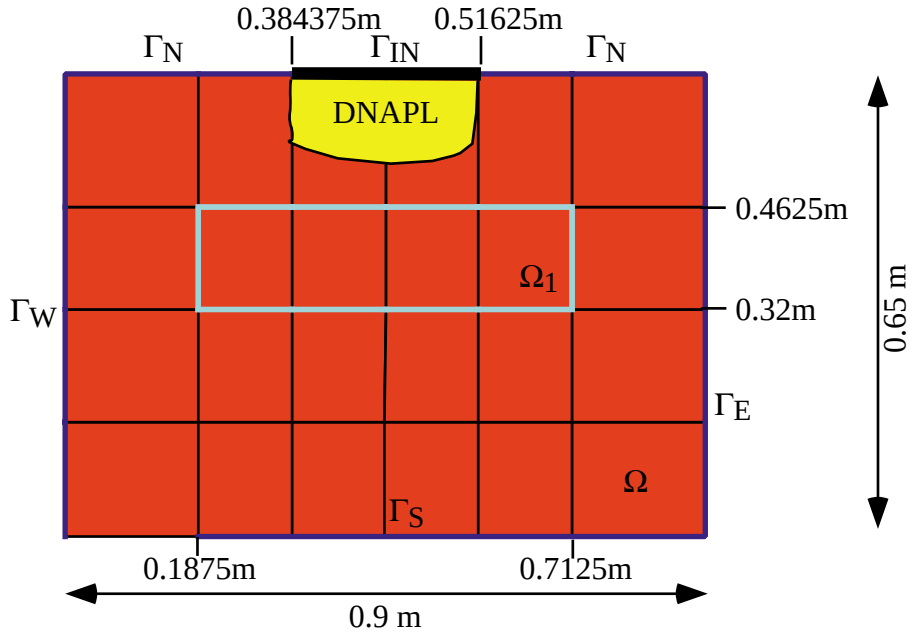


Figure 7.8: Geometry of the 2D DNAPL problem with low permeable lens.

Boundary Conditions

Γ_{IN}	$\phi_n = -0.075[kg/(sm^2)], \phi_w = 0$
Γ_N	$\phi_n = \phi_w = 0$
Γ_E, Γ_W	$p_w = (0.65 - y) \cdot 9810.0\ [Pa]$ (hydrostatic), $S_n = 0$
Γ_S	$\phi_w = 0, S_n = 0$

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\quad \rho_n = 1460\ [kg/m^3]$

$\mu_w = 10^{-3}\ [Pa\ s]$ $\quad \mu_n = 0.9 \cdot 10^{-3}\ [Pa\ s]$

Solid Matrix & Constitutive Relations

Brooks–Corey functions with the following parameters:

Subdomain	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$
Ω_1	0.4	$6.64 \cdot 10^{-11}$	0.1	0.0	2.7	755.0
$\Omega \setminus \Omega_1$	0.39	$3.32 \cdot 10^{-11}$	0.12	0.0	2.0	1163.5/1466.1

Ω_1 is defined in Fig. 7.8. An entry pressure of 1163.5 $[Pa]$ corresponds to a critical saturation of $S_n^* = 0.62$ which is reached in time step 18 (1080 $[s]$). An entry pressure of 1466.1 $[Pa]$ corresponds to a critical saturation of $S_n^* = 0.75$ which is never reached.

Initial Values

$p_{w0}(x,y) = p_0(x,y) = (0.65 - y) \cdot 9810.0, S_n = 0.$

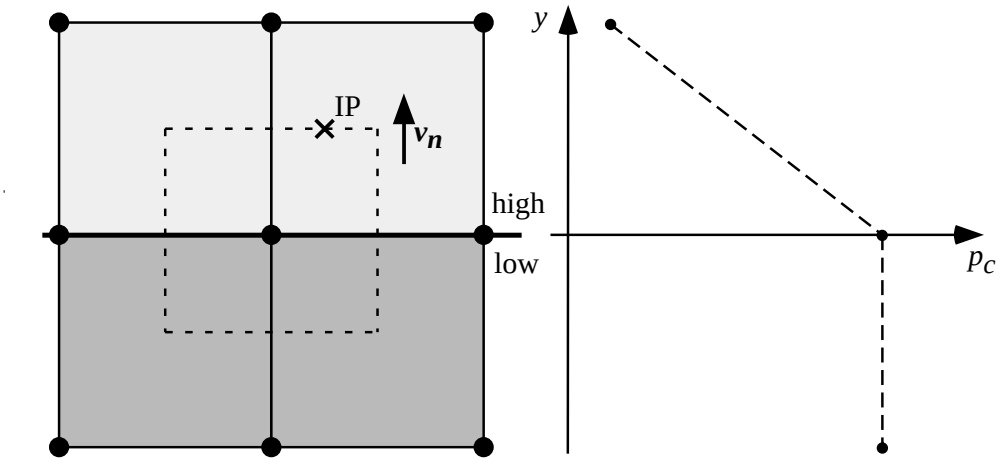


Figure 7.9: Approximation of the entry pressure effect in the *PPS* method with fully upwinding.

Mesh & Time Steps

The coarsest mesh (level 0) has 6 by 4 equidistant quadrilateral elements as shown in Fig. 7.8. After six levels of uniform refinement a mesh with 384 by 256 elements and 98945 nodes is obtained.

75 time steps of $\Delta t = 60\ [s]$ are computed (final time 4500 $[s]$).

Results

A mesh refinement study of the solution after 75 time steps ($T = 4500\ [s]$) is given in Figs. 7.10 and 7.11. Contour lines are spaced in 0.05 intervals, the first (darkest) contour line is at a saturation value of 10^{-6} indicating that the solution has compact support and no spurious oscillations. The free boundary seperating the domains where only water and both phases are present moves about 5 mesh cells per time step in the finest calculations.

In Fig. 7.10 both methods give comparable results with no infiltration of the low permeable lens except for the 48×32 mesh used with the *PPS* method. To explain this behavior consider Fig. 7.9. The figure shows a control volume extending over the interface between high permeability and low permeability (since the *elements* are associated with subdomains). The nodes lying on the interface are assumed to belong to the low permeable region. Consider now a zero DNAPL saturation at all nodes shown in Fig. 7.9, then capillary pressure (which is now the entry pressure) will be larger at the nodes belonging to the low permeable region. Correspondingly, a large gradient of capillary pressure will be computed in the elements directly above the interface as indicated in the right part of Fig. 7.9. If this gradient is large enough the velocity $\mathbf{v}_n = -K(\nabla p_w + \nabla p_c - \rho_n \mathbf{g})$ in the integration point IP will point upward, effectively producing a zero mobility and zero flux of the DNAPL over the sub–control volume face and therefore preventing infiltration of the low permeable lens. If the DNAPL saturation above the lens rises the velocity $\mathbf{v}_n$ will eventually point

downward allowing the DNAPL to infiltrate the lens. Since $-K(\nabla p_w - \rho_n \mathbf{g})$ points downward this will happen *before* the critical saturation defined by the interface condition is reached. The critical saturation is therefore only computed approximately and the accuracy depends on the mesh size. Obviously, the 48×32 in Fig. 7.9 was too coarse to prevent infiltration of the lens. In contrast, the *PPSIC* formulation does not approximate the critical saturation value where infiltration occurs and therefore yields correct results on all grid levels.

It should also be noted that fully upwinding ($\beta = 1$) is required in the *PPS* method to prevent infiltration of the low permeable lens. Otherwise, the mobility at the integration point IP in Fig.7.9 would not be zero and infiltration would occur immediately. Helmig (1997) compares various discretization schemes with respect to a correct representation of the entry pressure effect.

The case with infiltration is shown in Fig. 7.11. Here the approximation of the critical saturation in the *PPS* method (with fully upwinding) allows more DNAPL to penetrate through the lens (since it infiltrates earlier) when compared to the *PPSIC* formulation. Consequently, the fingers extending around the lens are shorter with the *PPS* scheme. The figure also shows the discontinuous representation of the saturation in the *PPSIC* formulation. The discontinuity is resolved within one mesh cell in the *PPS* method.

Table 7.5 lists the solver statistics for this problem. Standard parameters from Subs. 7.1.2 have been used in the solver. Nested iteration has been used to obtain initial guesses. It was important to pay attention to the discontinuous representation of saturation in the *PPSIC* method when interpolating initial guesses from coarse to fine grid. However, standard prolongation is used within the multigrid method!

For both values of the entry pressure the *PPSIC* method performs significantly better than the *PPS* method. The number of Newton steps is nearly independent of the mesh size (fixed time step) with number of Newton steps significantly lower for the *PPSIC* method. The average number of multigrid iterations is (slowly) increasing for the *PPS* scheme while it stays constant for the *PPSIC* method. On the finest mesh *PPSIC* is therefore twice as fast as *PPS*. Also, *PPSIC* behaves the same whether the DNAPL infiltrates or not, while *PPS* performs worse in the case with infiltration (time step reduction was necessary).

We conclude that the *PPSIC* method should be preferred over the *PPS* scheme for discontinuous porous media. The *PPSIC* method gives qualitatively correct results already on coarse meshes, we will show later that the approximation of the critical saturation becomes worse for water–gas flows and problems on larger scales. For tetrahedral elements in three space dimensions the fully upwinding procedure is not able to prevent infiltration of a low permeable lens. Moreover, the number of Newton iterations is lower and the number of multigrid cycles is *h*–independent for the *PPSIC* scheme (for the problem considered here).

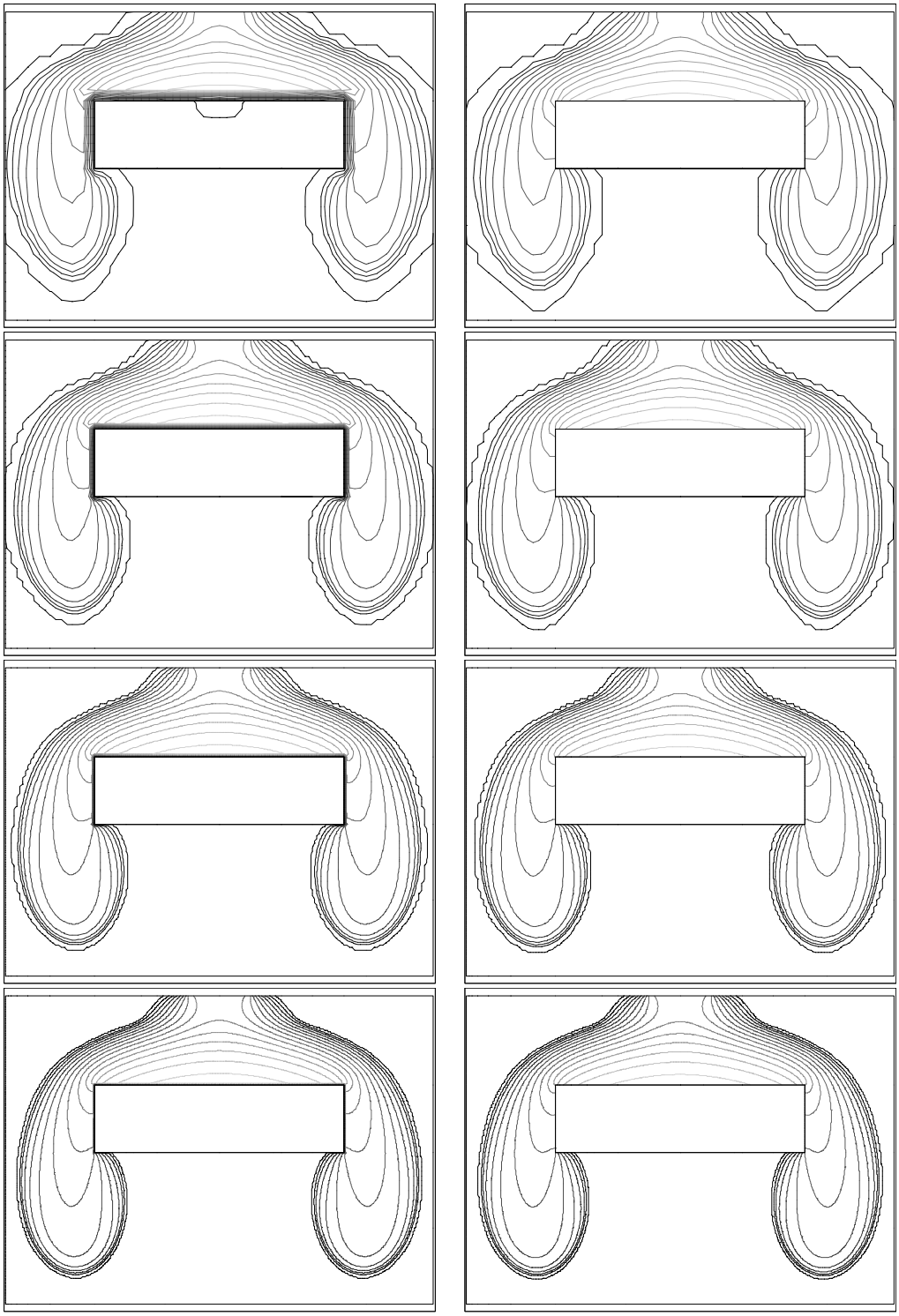


Figure 7.10: Single Lens DNAPL infiltration (high entry pressure). 48×32 to 384×256 meshes. *PPS* left and *PPSIC* right.

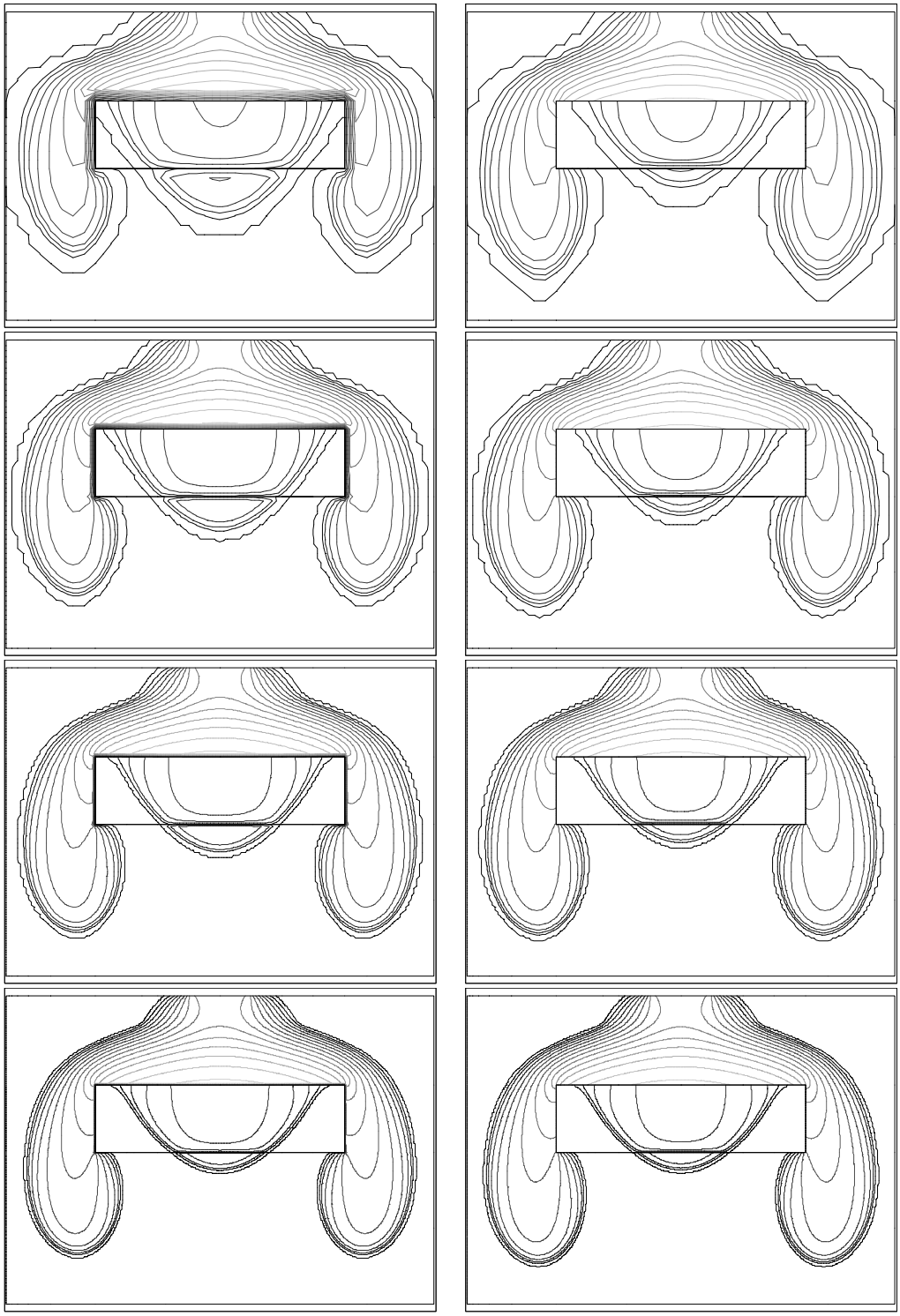


Figure 7.11: Single Lens DNAPL infiltration (low entry pressure). 48×32 to 384×256 meshes. *PPS* left and *PPSIC* right.

Table 7.5: Performance statistics for 2D DNAPL infiltration with low permeable lens on Power Macintosh G3.

Problem	S	SIZE	EX	N	MG	AVG	MAX	TN
PPS	75	48×32	484	405	1271	3.1	5	4.20
high	75	96×64	2264	376	1415	3.8	7	4.91
entry	75	192×128	10711	367	1787	4.9	8	5.81
pressure	75	384×256	54221	370	2409	6.5	15	7.35
PPSIC	75	48×32	398	253	765	3.0	5	3.45
high	75	96×64	1840	248	906	3.7	5	3.99
entry	75	192×128	7601	235	922	3.9	6	4.12
pressure	75	384×256	31369	234	944	4.0	7	4.25
PPS	75	48×32	527	453	1406	3.1	5	4.57
low	75	96×64	2734	449	1749	3.9	6	5.93
entry	79	192×128	13804	425	2247	5.3	8	7.11
pressure	87	384×256	77712	494	3237	6.6	10	9.09
PPSIC	75	48×32	400	254	704	2.8	4	3.47
low	75	96×64	1915	262	925	3.5	5	4.16
entry	75	192×128	7802	245	933	3.8	6	4.23
pressure	75	384×256	32409	254	930	3.7	7	4.40

7.3.3 Geostatistical Permeability Distribution

The problem setup and boundary conditions are taken from Subs. 7.3.2 with the following changes. The permeability field, shown in Fig. 7.12, is geostatistically distributed with a mean value of $\overline{k} = 6.64 \cdot 10^{-11} = 10^{-10.18}[m^2]$, a correlation length of 8 cells and a size of 192 by 128 cells. Its minimum value is $k_{min} = 10^{-11.2}$ and its maximum value is $k_{max} = 10^{-9.24}$, i. e. only one order of magnitude variation around the mean value

Using the correlation of Leverett (1941) between capillary pressure and absolute permeability (porosity is constant $\Phi = 0.4$ in our case) we define a Brooks–Corey type capillary pressure function with entry pressure depending on absolute permeability:

$$p_c = p_d \sqrt{\frac{\overline{k}}{k}} \overline{S}_w^{-1/\lambda}. \tag{7.1}$$

We use $S_{wr} = 0.1$, $S_{nr} = 0$, $p_d = 755[Pa]$ and $\lambda = 2.7$.

60 time steps of $\Delta t = 35[s]$ have been computed. Solution after $2100[s]$ of simulated time is shown in Fig. 7.13. The *PPS* formulation has been used with standard parameters of the solver (see Subs. 7.1.2). The solution shows preferential flow paths due to strong variations in entry pressure.

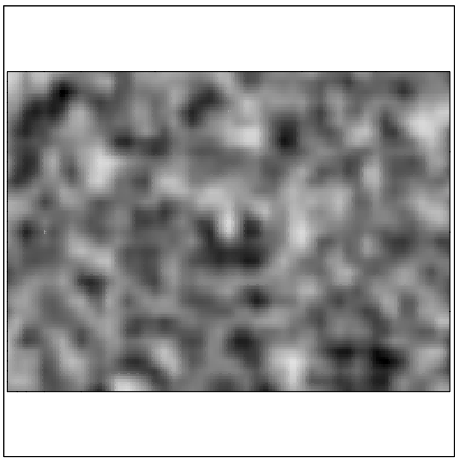


Figure 7.12: Permeability field for vertical DNAPL infiltration.

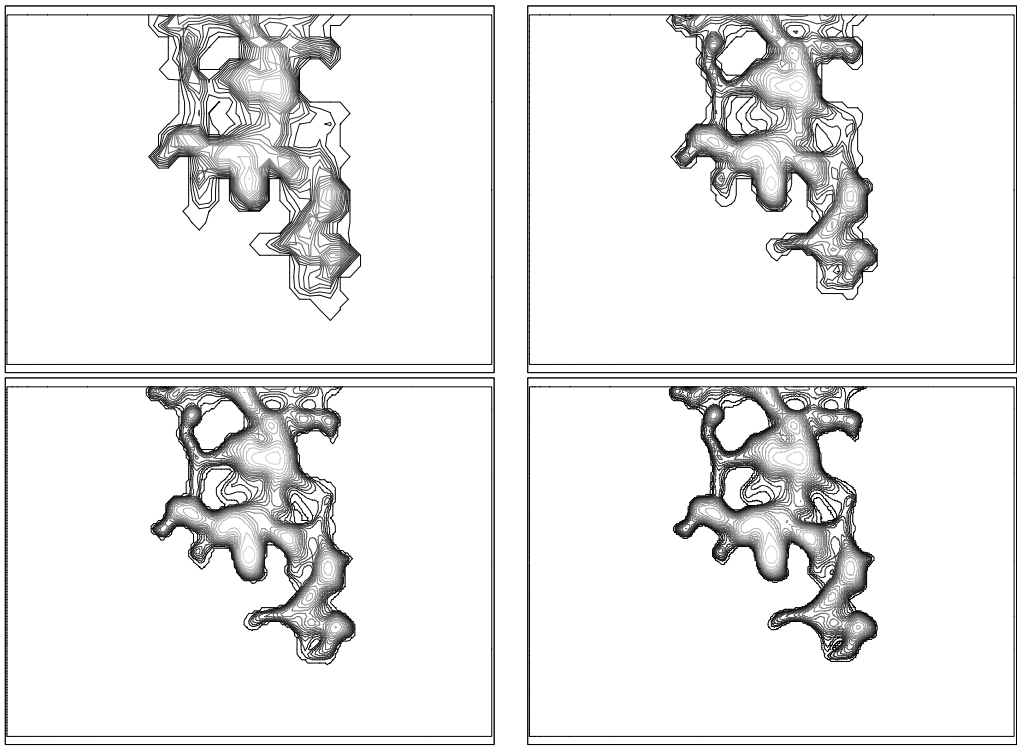


Figure 7.13: DNAPL infiltration in a medium with geostatistical permeability distribution. 48×32 to 384×256 meshes (top left to bottom right). *PPS* method has been used.

Table 7.6: Performance statistics for 2D DNAPL infiltration with geostatistically distributed absolute permeability on a Power Macintosh G3.

S	SIZE	EX	N	MG	AVG	MAX	max S_n
60	48×32	497	364	969	2.7	4	0.850
60	96×64	2689	381	1492	3.9	7	0.866
60	192×128	11502	336	1650	4.9	8	0.869
60	384×256	53168	320	2048	6.4	12	0.872

Solver statistics are shown in Fig. 7.6. Performance is similar to the single lens case with the number of Newton steps being constant and the number of multigrid steps slightly increasing with mesh size.

7.4 VEGAS Experiment

This section is about the numerical simulation of an experiment that has been conducted at the VEGAS facility (in german: "Versuchseinrichtung zur Grundwasser– und Altlastensanierung") in Stuttgart, see (Kobus 1996). Previous results of Sheta in (Helmig et al. 1998) have been used in the design of the pilot experiment.

Fig. 7.14 shows the geometry of the domain which is 6.43 meters long, 2.4 meters high and 0.4 meters thick. The simulation, however, is two–dimensional. DNAPL is released on top and flows downward over the lenses with different slopes. A groundwater flow from left to right and capillary forces enable the DNAPL to migrate upward on the slopes. The U–shaped lens to the right (sand 1) has a relatively low entry pressure and will be invaded if enough DNAPL accumulates.

The parameters of the simulation are given as follows.

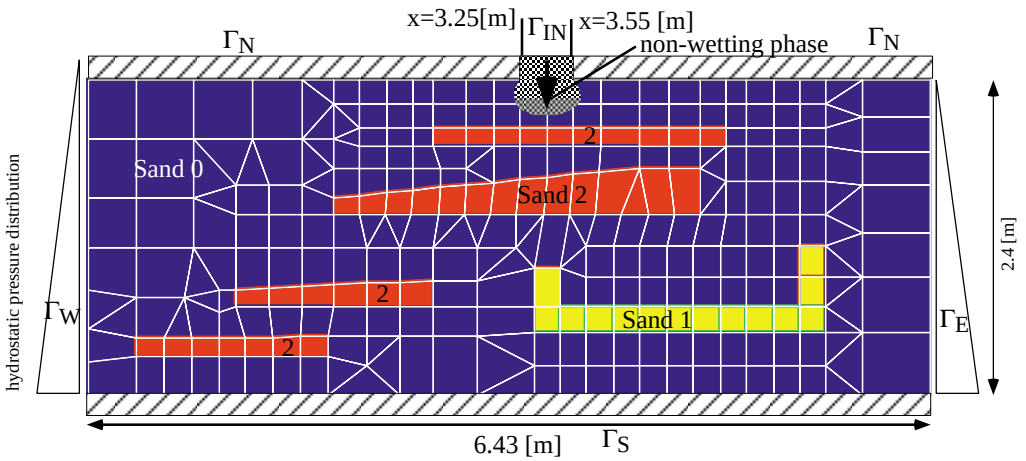


Figure 7.14: Geometry of the two–dimensional VEGAS experiment.

Formulation Used

The *PPSIC* method with (p_w, S_n) as unknowns will be used.

Boundary Conditions

Γ_{IN} $\phi_n = -0.259[kg/(sm^2)], \phi_w = 0$
Γ_N $\phi_n = \phi_w = 0$
Γ_E $p_w = (2.4 - y) \cdot 9810\ [Pa], S_n = 0$
Γ_W $p_w = (2.4 - y) \cdot 9810 + 661.95\ [Pa], S_n = 0$
Γ_S $\phi_w = 0, \phi_n = 0$

For definition of the boundary segments see Fig. 7.14.

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\rho_n = 1400\ [kg/m^3]$
$\mu_w = 10^{-3}\ [Pa\ s]$ $\mu_n = 0.9 \cdot 10^{-3}\ [Pa\ s]$

Solid Matrix & Constitutive Relations

Brooks–Corey functions with the following parameters:

Sand	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$
0	0.4	$4.60 \cdot 10^{-10}$	0.10	0.0	3.0	234.0
1	0.4	$3.10 \cdot 10^{-11}$	0.12	0.0	2.5	755.0
2	0.4	$9.05 \cdot 10^{-12}$	0.15	0.0	2.0	1664.0

The location of the regions with different sands is given in Fig. 7.14.

Initial Values

$p_w(x,y) = (1 - x/6.43) \cdot 661.95 + (2.4 - y) \cdot 9810.0, \qquad S_n = 0\ .$

Mesh & Time Steps

The coarsest mesh consists of 290 quadrilateral and triangular elements as shown in Fig. 7.14

Uniform refinement results in the following meshes:

Level	Elements
0	290
1	1160
2	4640
3	18560
4	74240
5	296960
6	1187840

240 steps of $\Delta t = 30[s]$ are to be computed. The propagation speed of the non–wetting phase infiltration front is more than 6 mesh cells per time step in the finest calculation.

Results

Figs. 7.15 and 7.16 show the results of the numerical computation after $7200[s]$ of simulated time. Comparison with experimental results given in Fig. 7.17 show a qualitatively correct behavior in the sense that the lenses of type 2 are not infiltrated and that the U–shaped lens is infiltrated. The assumption of a homogeneous coarse sand (sand 0), however, is not justified as is shown by the experimental results. Small scale heterogeneities as investigated in Subs. 7.3.3 have a large influence on the flow behavior. Although the porous medium used in the experiment is built up in a controlled laboratory environment the use of natural sand (instead of glass beads) inevitably results in small–scale heterogeneities. Incorporation of these heterogeneities into the simulation with a geostatistical model resulted in solutions with a qualitatively correct representation of the layering effects, see Sheta (1999).

This example is also used to show the effectiveness of the data–parallel implementation. Table 7.7 shows the performance for a scaled computation where the number of elements per processor was about 4600. Standard solver parameters from Subs. 7.1.2 were used with the following modifications: The ILU smoother was replaced by a symmetric Gauß–Seidel smoother with damping factor $\omega = 0.8$ and the truncated restriction was replaced by standard restriction. Level 0 (290 elements) was kept on one processor, levels 1 and higher were mapped to all processors when using up to 64 processors while in the 256 processor case level 1 was mapped to 72 processors and level 2 (4640 elements) used all processors. Recursive spectral bisection with Kernighan–Lin optimization from the CHACO library, (Hendrickson and Leland 1993a), was used as partitioning scheme. Nested iteration was used to obtain good initial guesses for the nonlinear iteration on the finest level. Starting level for the nested iteration was 2 (instead of 0 used in the sequential runs) to save some work on the coarsest grid levels where parallel efficiency is poor. Table 7.7 shows a fourfold increase in total computation time when increasing the problem size *and* the number of processors by a factor of 256. This increase has three reasons: The average number of multigrid iterations per Newton step increased by a factor of two, the number of nonlinear iterations increased by a factor of 1.6 and the work on the coarse meshes during nested iteration does not parallelize well (but this a relatively small amount of work). Nevertheless the overall performance is considered to be quite good. The last column of Table 7.7 labeled TI shows the time for one multigrid cycle on the finest level. The small increase of only 31% shows that load imbalance and communication overhead are small.

Table 7.8 compares multigrid with a single grid iterative scheme as preconditioner in BiCGSTAB. The multigrid V–cycle used a symmetric Gauß–Seidel smoother with two pre–and postsmoothing steps while the single–grid method was one symmetric Gauß–Seidel step (thus the multigrid preconditioner costs about four times as much). Due to time limitations on the CRAY T3E only the first 25 time steps were computed with both methods. Considering total execution time (EX) it is shown that the run using the multigrid preconditioner is faster

Table 7.7: Multigrid solver performance for 2D VEGAS experiment on Cray T3E.

P	S	SIZE	EX	N	MG	AVG	MAX	TI
1	240	4640	9407	827	4546	5.5	10	0.96
4	240	18560	19280	1206	9073	7.5	13	1.06
16	240	74240	23819	1148	9635	8.4	13	1.15
64	240	296960	29624	1219	11477	9.4	15	1.24
256	240	1187840	35669	1297	13407	10.3	15	1.26

by a factor of 21 for the mesh with 1.2 million elements (2.4 million degrees of freedom). The average number of multigrid cycles increases only very slightly while the number of Gauß–Seidel preconditioner steps doubles with each mesh refinement. Table 7.8 clearly indicates that efficient solvers with optimal complexity are a necessity for large scale simulations with parallel computers.

7.5 3D DNAPL Infiltration

This section extends the results of Section 7.3 to the three–dimensional case. The geometry of the domain and the location of the lenses with different properties is shown in Fig. 7.18.

Formulation Used

The *PPSIC* method with (p_w, S_n) as unknowns will be used.

Table 7.8: Comparison of multigrid and single grid preconditioner for 2D VEGAS experiment after 25 time steps on Cray T3E.

Prec.	P	S	SIZE	EX	N	ITER	AVG	MAX
MG–	1	25	4640	887	107	357	3.3	6
SGS(2,2)	4	25	18560	1151	93	396	4.3	8
V–cycle	16	25	74240	1483	104	460	4.4	8
	64	25	296960	1793	105	534	5.1	9
	256	25	1187840	1955	100	560	5.6	9
SGS(1)	1	25	4640	3674	107	8992	84	153
	4	25	18560	4516	93	12780	137	249
	16	25	74240	11244	104	32976	317	450
	64	25	296960	21231	106	57302	541	1149
	256	25	1187840	42040	101	113180	1121	2699

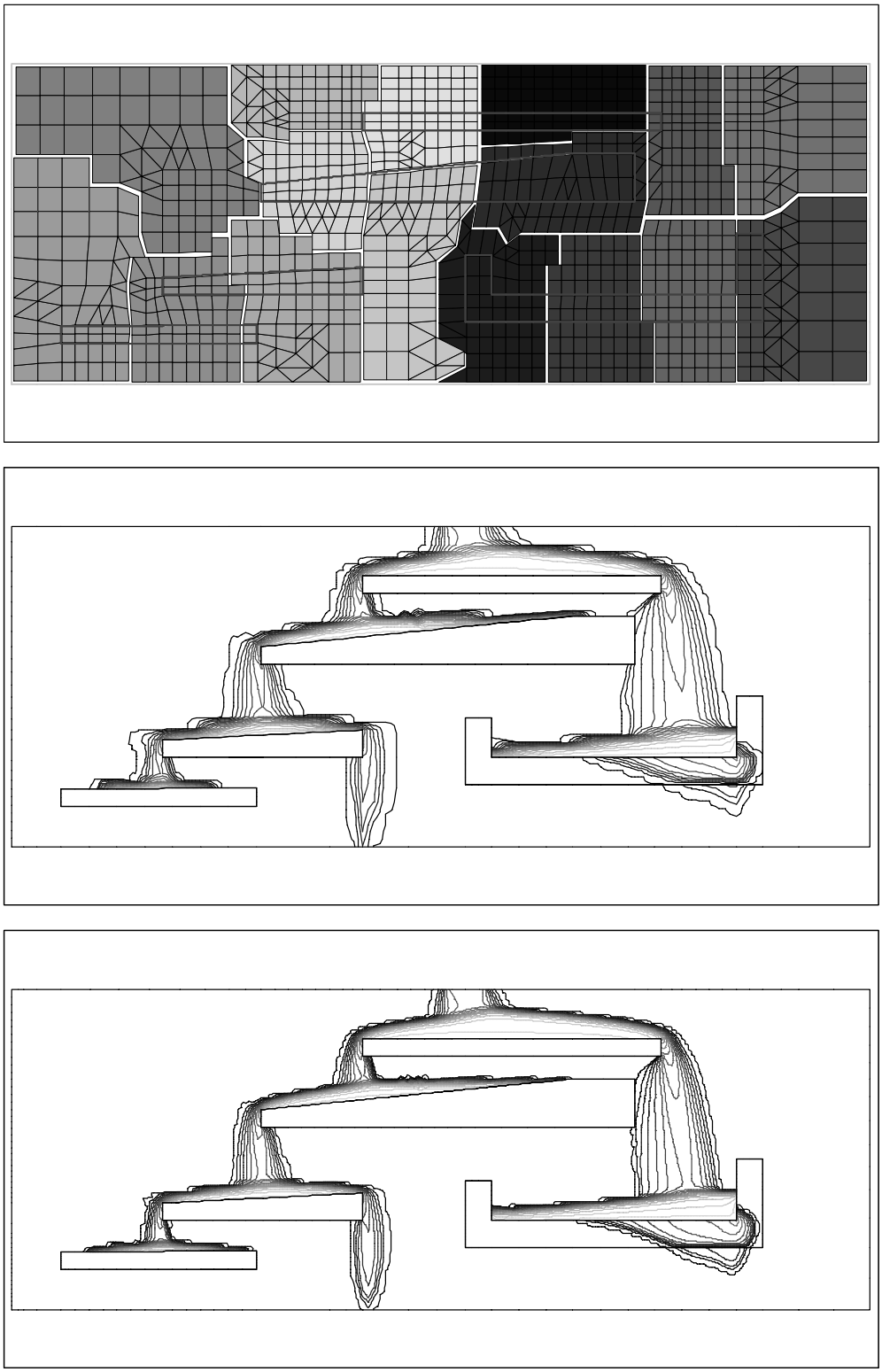


Figure 7.15: Partitioning of the VEGAS mesh (16 processors). DNAPL saturation after $7200[s]$ on levels 2 and 3 (middle and bottom).

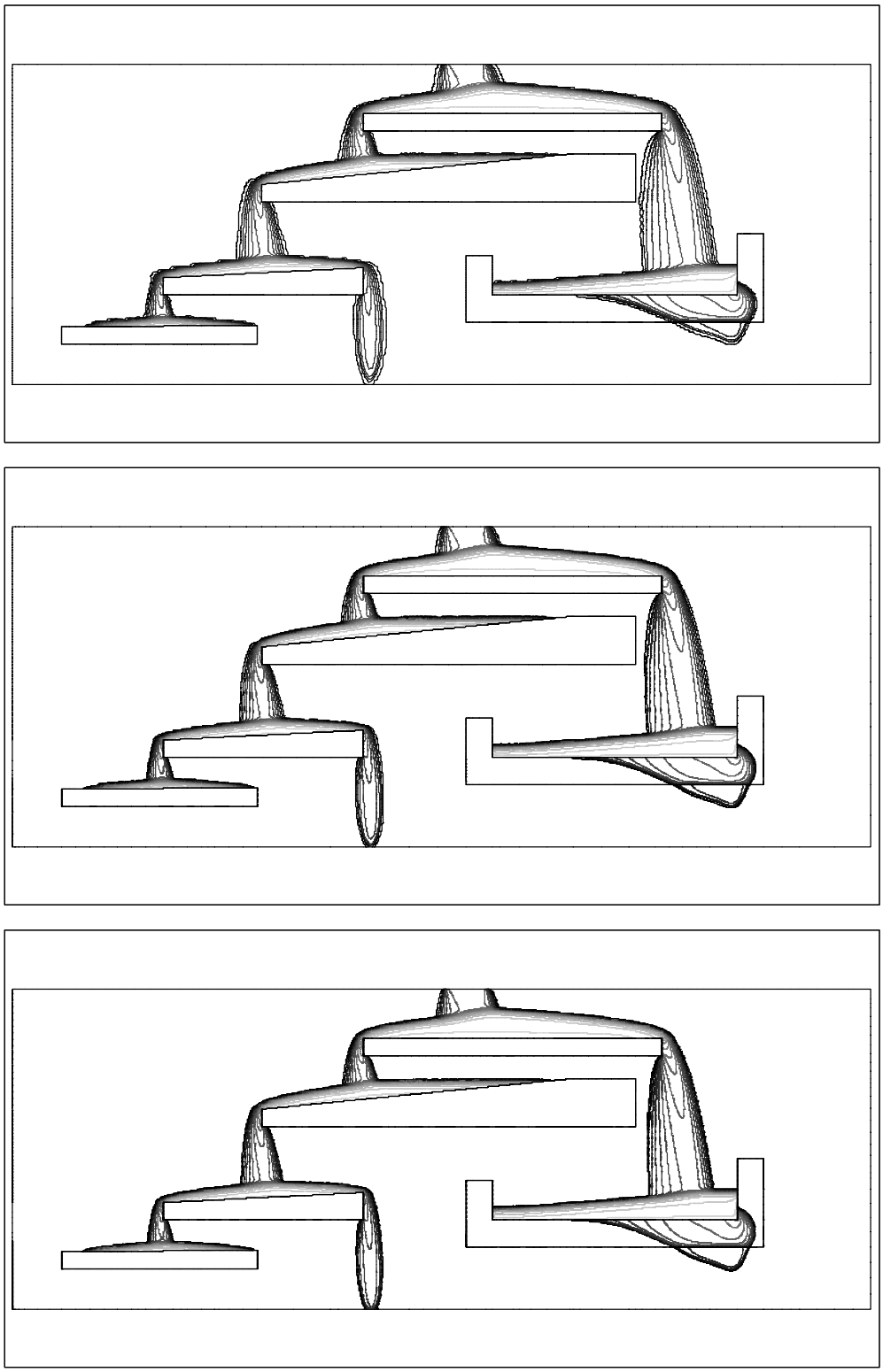


Figure 7.16: Contour plot of DNAPL saturation after 7200$[s]$ on levels 4, 5 and 6 (from top).

Figure 7.17: Experimental result from VEGAS facility.

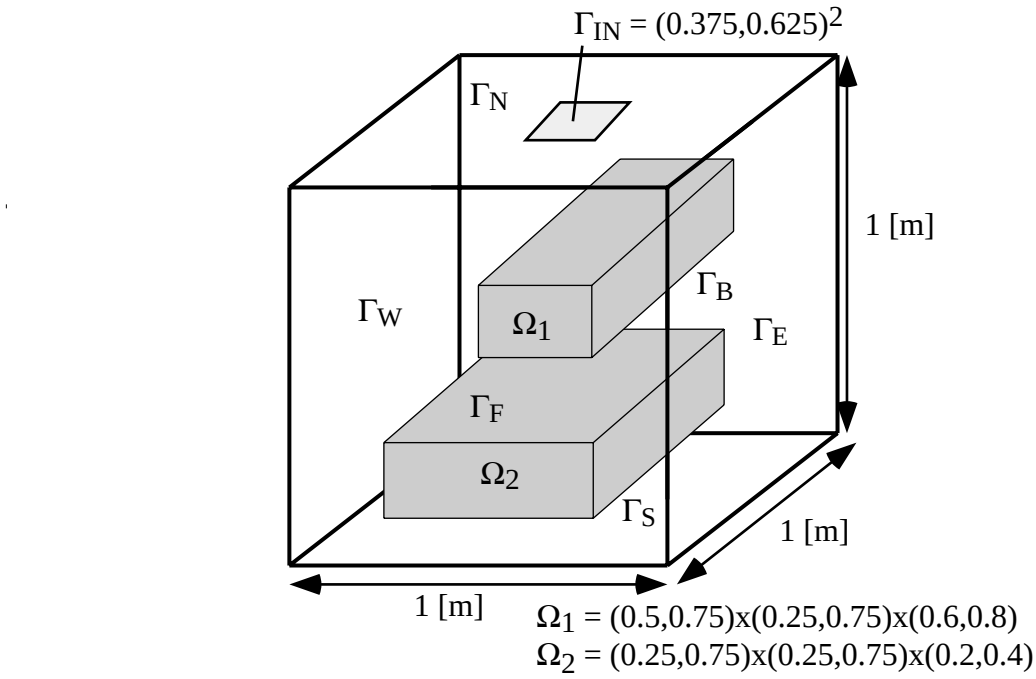


Figure 7.18: Domain for the three–dimensional DNAPL infiltration example.

Boundary Conditions

Γ_{IN} $\quad \phi_n = -0.25[kg/(sm^2)], \phi_w = 0$
Γ_N, Γ_S $\quad \phi_n = \phi_w = 0$
Γ_E $\quad p_w = (1-z)\cdot 9810 + 400\ [Pa], \phi_n = 0$
Γ_W $\quad p_w = (1-z)\cdot 9810\ [Pa], \phi_n = 0$
Γ_F, Γ_B $\quad p_w = (1-z)\cdot 9810 + x\cdot 400\ [Pa], \phi_n = 0$

For definition of the boundary segments see Fig. 7.18.

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\quad \rho_n = 1630\ [kg/m^3]$
$\mu_w = 10^{-3}\ [Pa\ s]$ $\quad \mu_n = 10^{-3}\ [Pa\ s]$

Solid Matrix & Constitutive Relations

Brooks–Corey functions with the following parameters:

Subdomain	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$
Ω_1, Ω_2	0.39	$5.26\cdot 10^{-11}$	0.10	0.0	2.49	2324
Ω	0.4	$5.04\cdot 10^{-10}$	0.08	0.0	3.86	369

The location of the regions with different sands is given in Fig. 7.18.

Initial Values

$p_w(x,y) = x\cdot 400 + (1-z)\cdot 9810.0, \qquad S_n = 0\ .$

Mesh & Time Steps

The coarsest mesh consists of $4\times 4\times 5$ hexahedral elements and resolves the interfaces between low and high permeable regions. Uniform refinement results in the following meshes:

Level	x	y	z	elements
0	4	4	5	80
1	8	8	10	640
2	16	16	20	5120
3	32	32	40	40960
4	64	64	80	327680
5	128	128	160	2621440

50 steps of $\Delta t = 20[s]$ are to be computed. The propagation speed of the infiltration front is between 5 and 6 mesh cells per time step.

Results

This example is intended to show the applicability of the methods in three space dimensions and to show the excellent parallelization properties.

Fig. 7.19 shows a contour plot of DNAPL saturation at $T = 1000[s]$ on two cuts through the domain. The *PPSIC* formulation allows a discontinuous representation of the saturation at the interface. Fig. 7.20 shows isosurfaces of DNAPL concentration at various time steps.

Table 7.9: Performance statistics for 3D DNAPL infiltration with two low permeable lenses on Cray T3E.

P	S	SIZE	EX	N	MG	AVG	MAX	TI
1	50	5120	4187	218	348	1.6	2	2.10
4	50	40960	11589	243	612	2.5	4	4.69
32	50	327680	13214	264	928	3.5	7	4.76
256	50	2621440	14719	255	1098	4.3	9	4.82

The simulation used standard parameters (see Subs. 7.1.2) with the following modifications: The point–block ILU smoother has been damped with $\omega = 0.9$. This is necessary for a block–Jacobi type smoother in the parallel case, the value is not critical for this problem. Nested iteration has been used starting from level 1 (640 elements) instead of level 0. This has been done to improve parallel performance. Note that nested iteration includes more work on coarse meshes where parallelization is less efficient, especially for large processor numbers. On the other hand this effect is less critical in three dimensions than in two dimensions due to the larger growth factor. Load balancing has been done as follows: Level 0 (80 elements) has been kept on one processor in all calculations to enable fast solution with a direct solver. Level 1 (640 elements) has then been distributed to all processors, except in the 256 processor case where only 72 processors have been used on level 1. In the 256 processor run level 2 (5120 elements) has then been distributed to all processors. Load balancing scheme was inertial recursive bisection with Kernighan–Lin optimization, see (Hendrickson and Leland 1993a) for details.

Performance data of the simulation are presented in Table 7.9. Starting with four processors the problem size (in space) is increased by a factor of eight (uniform refinement) while also increasing the number of processors by eight leading to a problem size of about 10000 hexahedral elements per processor. The time step size was the same in all calculations. Results for a single processor having only 5120 elements are included for reference.

The time per multigrid iteration on the finest level (TI) can be used to evaluate the parallel efficiency of the code. In the ideal case it should be constant which it almost is. Note that the time for one processor has to be multiplied by two to be comparable. The average number of multigrid cycles per Newton step indicates that multigrid convergence is almost independent of the mesh size *and* the processor number for this example. Due to the use of nested iteration the number of Newton steps on the finest mesh remains also constant although the time step size is fixed. All components together show excellent scalability of the overall solution process: Total computation time increases by 75% for a 256 fold increase in problem size and processor number!

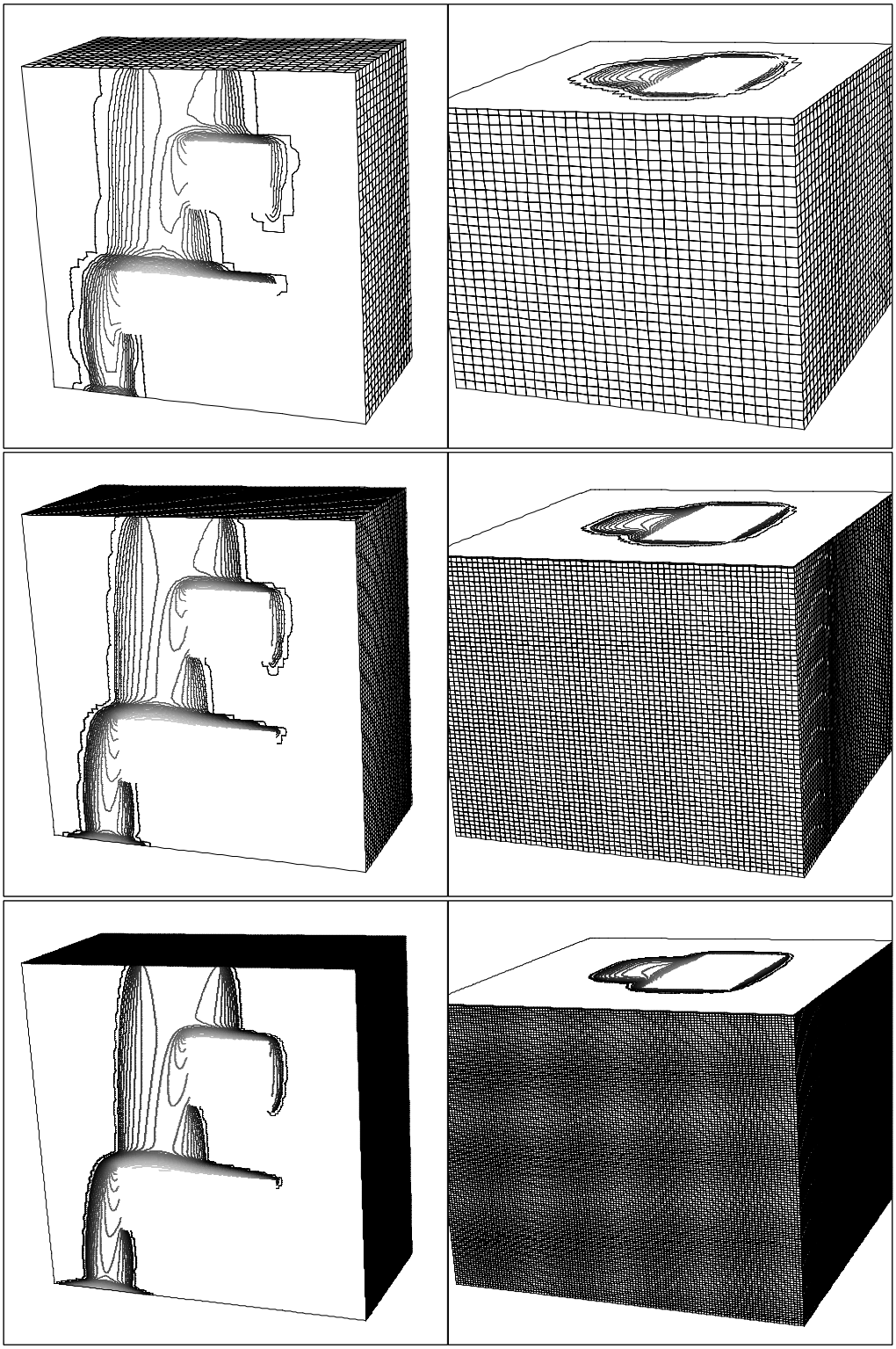


Figure 7.19: Contour plot of DNAPL saturation at $T = 1000[s]$ on levels 3, 4 and 5 (2.6 million elements, 5.2 million unknowns).

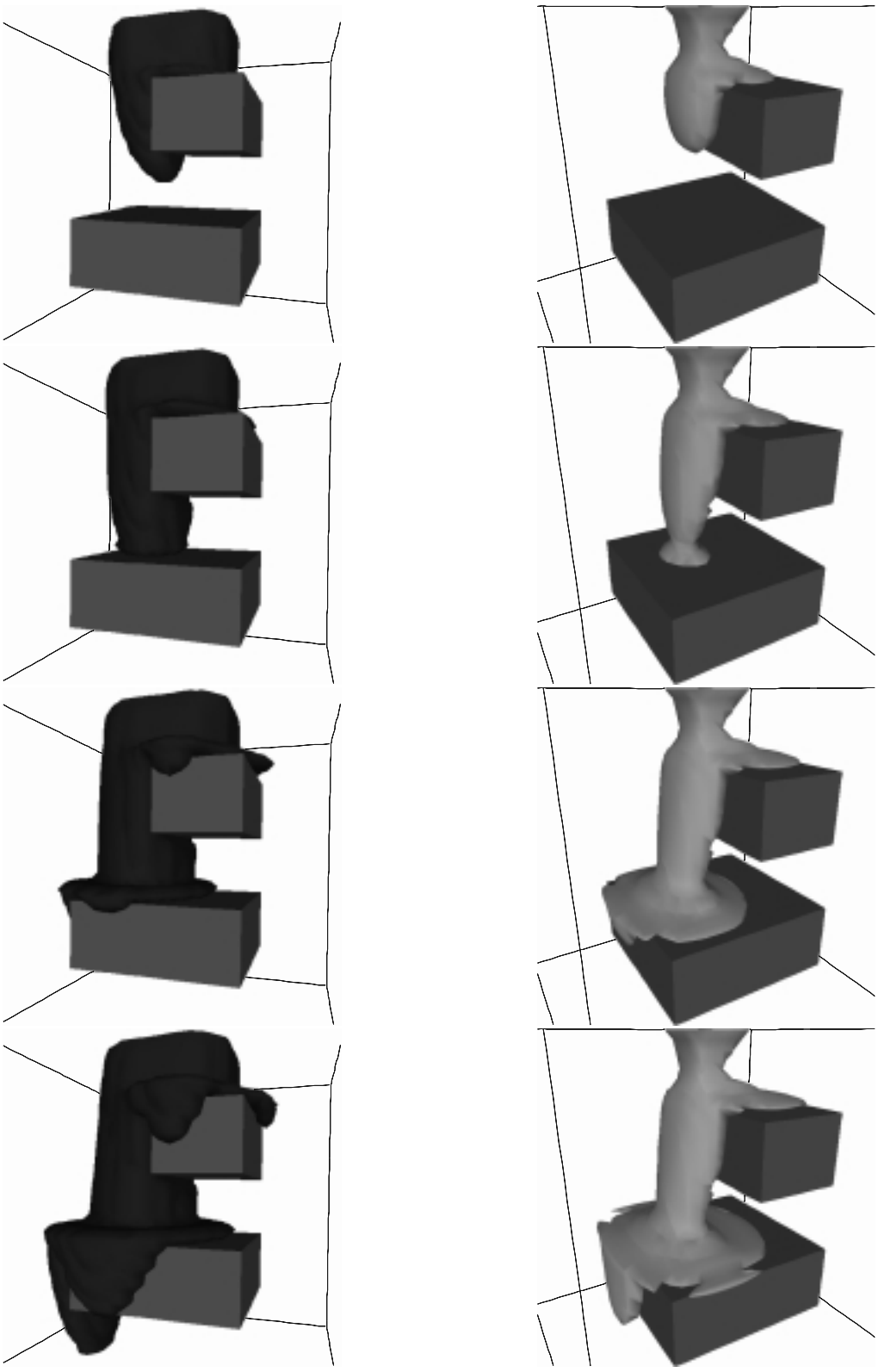


Figure 7.20: Isosurfaces of DNAPL saturation 1% (left) and 30% (right) after 240, 480, 720 and 960 seconds.

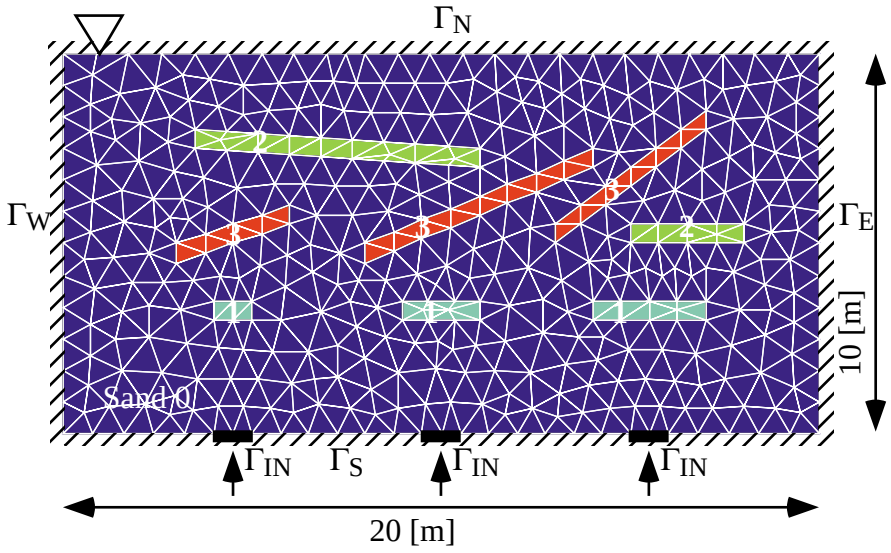


Figure 7.21: Geometry and initial mesh for the two–dimensional air sparging simulation.

7.6 2D Air Sparging

Air sparging refers to a remediation technique where air is injected from below in the saturated zone. The rising air is intended to reach organic liquids trapped there and to enhance microbial degradation and/or volatilization. Experiments revealed that the flow of air is affected strongly by heterogeneities present in the soil, see van Dyke and van der Zee (1998) and the references there.

If we are only interested in the distribution of the injected air in the saturated zone this process can be modeled with a two–phase flow model. Richards equation cannot be used in this case since the air is injected from below and is not in contact with the surface. Compressibility effects will be included via the ideal gas law. Furthermore, we restrict ourselves to the case of a piecewise homogeneous porous medium with the subdomains having different permeability, porosity and constitutive relations. The qualitative behavior of the solutions is the same as for the vertical DNAPL infiltration, only “upside down”. Due to the higher mobility of the air phase the flow of air is much more advection–dominated in the buoyancy–driven regions. The regions just below a low permeable layer where the air accumulates tend to be much thinner if the same constitutive relations are used.

Fig. 7.21 shows the geometry of the domain. Eight low permeable layers with different soil properties and inclinations are distributed over a region of 20 by 10 meters. Air is injected at three different places as indicated. The domain is meshed using triangular elements and an automatic mesh generator to demonstrate the unstructured mesh capabilities of the code.

The following parameters have been used in the simulation.

Formulation Used

The *PPSIC* method with (p_w, S_n) as unknowns will be used.

Boundary Conditions

Γ_N $\quad p_w = 10^5[Pa], \phi_n = 0$

$\Gamma_E, \Gamma_W, \Gamma_S$ $\quad \phi_n = \phi_w = 0$

Γ_{IN} $\quad \phi_n = -7.5 \cdot 10^{-4}[kg/(sm^2)], \phi_w = 0$

For definition of the boundary segments see Fig. 7.21.

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\quad \rho_n = p_n/84149.6\ [kg/m^3]$

$\mu_w = 10^{-3}\ [Pa\ s]$ $\quad \mu_n = 1.65 \cdot 10^{-5}\ [Pa\ s]$

Solid Matrix & Constitutive Relations

Brooks–Corey functions with the following parameters:

Sand	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$	S_n^*
0	0.40	$5.04 \cdot 10^{-10}$	0.10	0.0	2.0	1600.0	–
1	0.39	$2.05 \cdot 10^{-10}$	0.10	0.0	2.0	1959.6	0.30
2	0.39	$5.62 \cdot 10^{-11}$	0.10	0.0	2.0	2565.7	0.55
3	0.41	$8.19 \cdot 10^{-12}$	0.10	0.0	2.0	4800.0	0.80

The location of the regions with different sands is given in Fig. 7.21. The critical saturation refers to an infiltration from sand 0.

Initial Values

$p_w(x,y) = 10^5 + (10 - y) \cdot 9810.0, \qquad S_n = 0\ .$

Mesh & Time Steps

The initial (coarse) mesh had 760 triangular elements and 419 nodes. Uniform refinement resulted in the following mesh hierarchy:

Level	Elements	Nodes
0	760	419
1	3040	1597
2	12160	6233
3	48640	24625
4	194560	97889
5	780288	391361

Final simulation time was $T = 800[s]$, time steps size was $\Delta t = 16[s]$ (50 steps) for levels 0 to 3 and $\Delta t = 8[s]$ for levels 4 and 5. The non–wetting phase front moves about two mesh cells per time step in the finest calculation.

Results

Figs. 7.22 and 7.23 show contour plots of air saturation after $704[s]$ of simulated time for the *PPSIC* method and the *PPS* method respectively. Contour lines are spaced in 0.025 intervals with the first (darkest) contour line at $S_n = 0.0001$.

The contour plots show that with the *PPSIC* method only the three lenses directly above the air inlets are infiltrated. With the *PPS* method the rightmost lens, which is of type 2, is infiltrated on all mesh levels. The refinement study shows, however, that the amount of fluid infiltrating the lens is decreasing with increasing mesh refinement. This is due to the approximation of the interface condition in the *PPS* method. The following argument shows that a very fine mesh spacing is required for the *PPS* scheme to accurately represent the interface condition: We assume that water pressure is hydrostatic, i. e. we have $\nabla p_w = 9810[Pa/m]$. The jump of capillary pressure over the interface from sand 0 to sand 2 is $\approx 950[Pa]$ for zero DNAPL saturation on both sides. For $\nabla p_w + \nabla p_c - \rho_n \mathbf{g}$ to point downward (and produce the correct upwinding) ∇p_c must balance ∇p_w (the gravity term can be neglected since $\rho_n = \rho_w/1000$) which requires a mesh spacing smaller than $0.1[m]$. This is only an upper bound for the mesh spacing. Since the air saturation is increasing below the lens the jump of capillary pressure becomes smaller and the mesh spacing must be even smaller for ∇p_c to balance ∇p_w. This argument shows that *PPS* requires excessively small mesh spacing on the order of $[cm]$ under the lenses which makes the method impractical for field scale models.

The solutions in Figs. 7.22 and 7.23 exhibit significantly more mesh dependence than in the previous examples. This is due to the combination of several effects: Water–gas flow is advection dominated in the buoyancy–driven regions and the unstructured triangular mesh results in a fair amount of numerical diffusion (mostly "crosswind"). Secondly, the layers of air beneath the low permeable lenses are extremely thin (several centimeters) and the better they are resolved the longer is the air path. Because of a large viscosity ratio air saturation is low in the buoyancy–driven regions (about 0.05). Due to these effects the differences in the solution from level 4 to 5 amount only to a small fraction of total mass injected (note that all plots in Figs. 7.22 and 7.23 contain the same total mass).

Computations for this problem have been carried out on the Power Macintosh G3 and performance data is given in Table 7.10. Standard parameters have been employed except that nested iteration was not effective and therefore has *not* been used. We think that this is due the large saturation gradients directly under the lenses which are not infiltrated. As a consequence the number of Newton step increases with mesh fineness or the time step has to be reduced accordingly. The multigrid method however behaves very well as in the previous examples.

7.7 3D Air Sparging

The final example simulates the bubbling of air in a three–dimensional heterogeneous porous medium. The domain is given in Fig. 7.24. It is 5 meters high and about 4 by 5 meters wide. Three lenses with different sand properties are placed within the domain. The remaining parameters are similar to those in the last section:

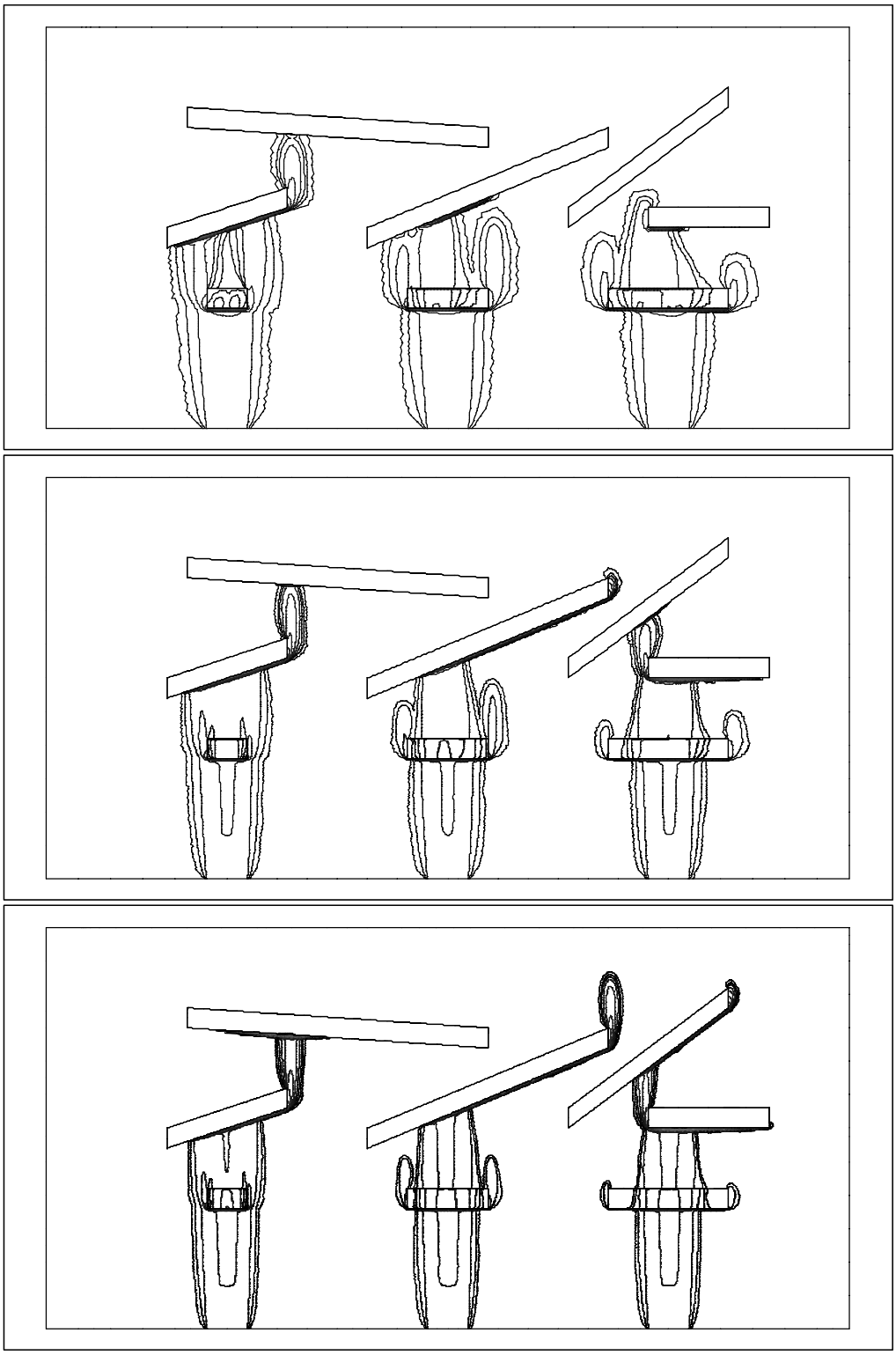


Figure 7.22: Air sparging simulation in 2D on levels 3, 4 and 5 with *PPSIC* method.

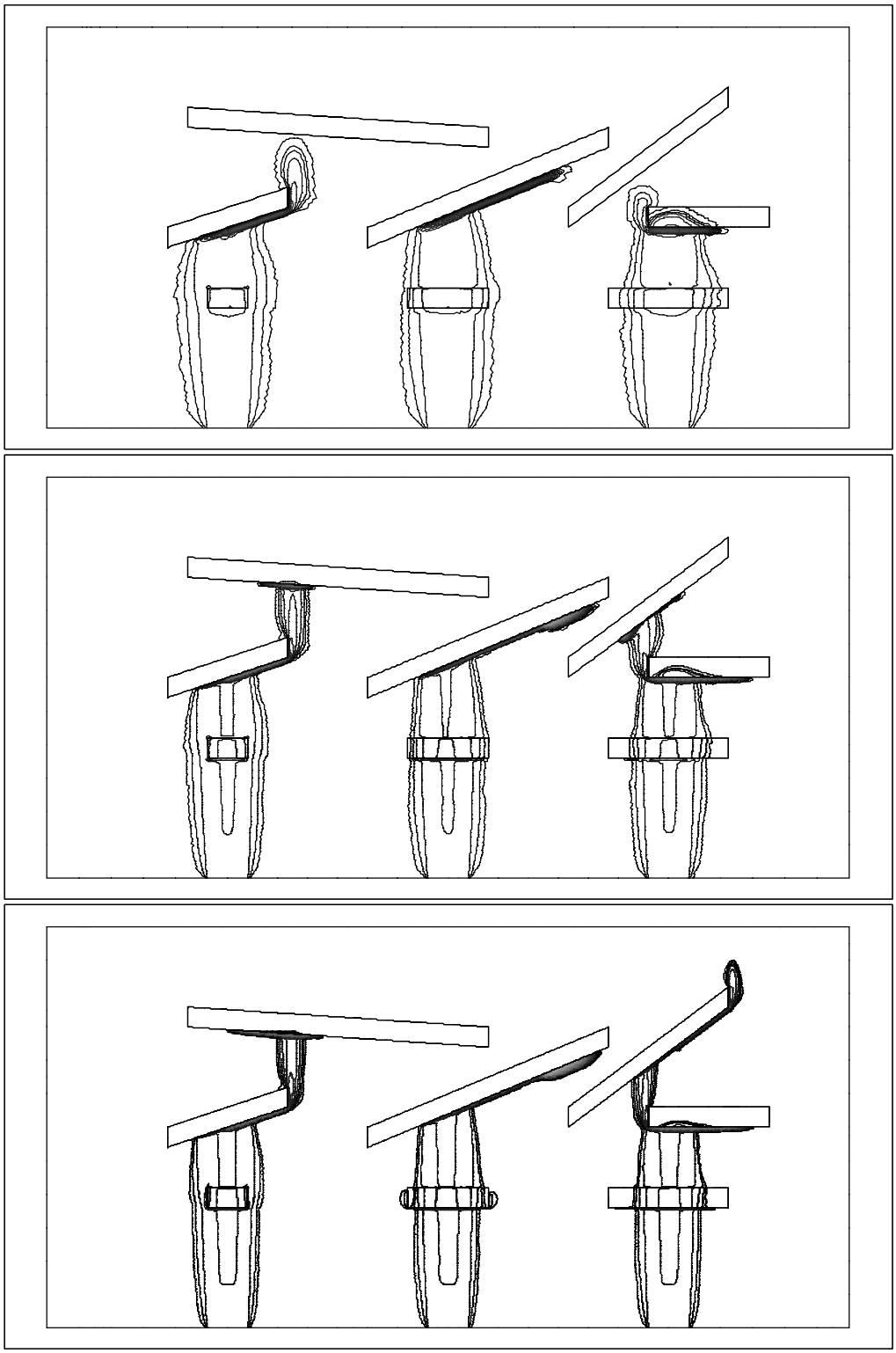


Figure 7.23: Air sparging simulation in 2D on levels 3, 4 and 5 with *PPS* method.

Table 7.10: Performance statistics for 2D air sparging example (sequential calculation on G3).

Method	S	SIZE	EX	N	MG	AVG	MAX
PPS	50	12160	752	198	426	2.2	4
	50	48640	4059	261	610	2.3	5
	100	194560	43785	590	1816	3.1	9
PPSIC	50	12160	1090	210	494	2.4	5
	50	48640	6351	303	749	2.5	5
	100	194560	74546	767	2447	3.2	7

Formulation Used

The *PPSIC* method with (p_w, S_n) as unknowns will be used.

Boundary Conditions

Γ_{TOP} $\quad p_w = 10^5[Pa], \phi_n = 0$

$\Gamma_{SIDE}, \Gamma_{BOT}$ $\quad \phi_n = \phi_w = 0$

Γ_{IN} $\quad \phi_n = -3 \cdot 10^{-3}[kg/(sm^2)], \phi_w = 0$

For definition of the boundary segments see Fig. 7.24.

Fluid Properties

$\rho_w = 1000\ [kg/m^3]$ $\quad \rho_n = p_n/84149.6\ [kg/m^3]$

$\mu_w = 10^{-3}\ [Pa\ s]$ $\quad \mu_n = 1.65 \cdot 10^{-5}\ [Pa\ s]$

Solid Matrix & Constitutive Relations

Brooks–Corey functions with the following parameters:

Sand	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$	S_n^*
0	0.40	$5.04 \cdot 10^{-10}$	0.10	0.0	2.0	1600.0	–
1	0.39	$2.05 \cdot 10^{-10}$	0.10	0.0	2.0	1959.6	0.30
2	0.39	$5.62 \cdot 10^{-11}$	0.10	0.0	2.0	2565.7	0.55
3	0.41	$8.19 \cdot 10^{-12}$	0.10	0.0	2.0	4800.0	0.80

The location of the regions with different sands is given in Fig. 7.24. The critical saturation refers to an infiltration from sand 0.

Initial Values

$p_w(x,y) = 10^5 + (5-y) \cdot 9810.0, \qquad S_n = 0\ .$

Mesh & Time Steps

The coarse mesh is shown in Fig. 7.24. It consists of 1492 tetrahedral elements and all internal boundaries are resolved by faces of the initial mesh. The mesh has been generated with "NETGEN", see Schöberl (1997). Uniform refinement of the tetrahedral coarse mesh resulted in the following multigrid hierarchy:

Table 7.11: Performance statistics for 3D air sparging calculation on CRAY T3E.

P	S	SIZE	EX	N	MG	AVG	MAX	TI
2	80	95488	10771	247	1355	5.5	8	3.44
16	81	763904	15201	320	1909	6.0	9	3.76
128	83	6111232	37297	693	4684	6.8	13	3.99

Level	Elements	Nodes
0	1492	354
1	11936	2124
2	95488	17329
3	763904	132801
4	6111232	1040129

The time step size was $\Delta t = 8[s]$ with final time $T = 640[s]$ (80 steps) unless a time step reduction was enforced by the nonlinear solver.

Results

Fig. 7.25 shows an isosurface of non–wetting phase saturation $S_n = 0.05$ at final time $T = 640[s]$. It shows that the *PPSIC* method also works with three–dimensional unstructured meshes. Visualization has been done with the graphics program GRAPE which is able to visualize large data sets, see (Rumpf et al. 1997).

Computations for this problem have been carried out on the T3E of HLRS, Stuttgart. Table 7.11 contains the performance results on up to a million nodes (2 million unknowns) mapped to 128 processors. Scaling the problem size and the number of processors by a factor 64 results in an almost fourfold increase in total computation time. This is mostly due to the increase in the number of Newton steps on the finest mesh which in turn is due to the fact that nested iteration has *not* been used. Nested iteration was ineffective in this example (as well as in the two–dimensional case). We believe that this is a result of the very thin layers of air under the lenses with a correspondingly large gradient. On the other hand the multigrid method scales very well with respect to the average number of iterations and the time per iteration (parallel efficiency). Standard parameters have been employed with the following modifications: Nested iteration was turned off (see above) and the point–block ILU smoother has been replaced by the point–block Gauß–Seidel smoother.

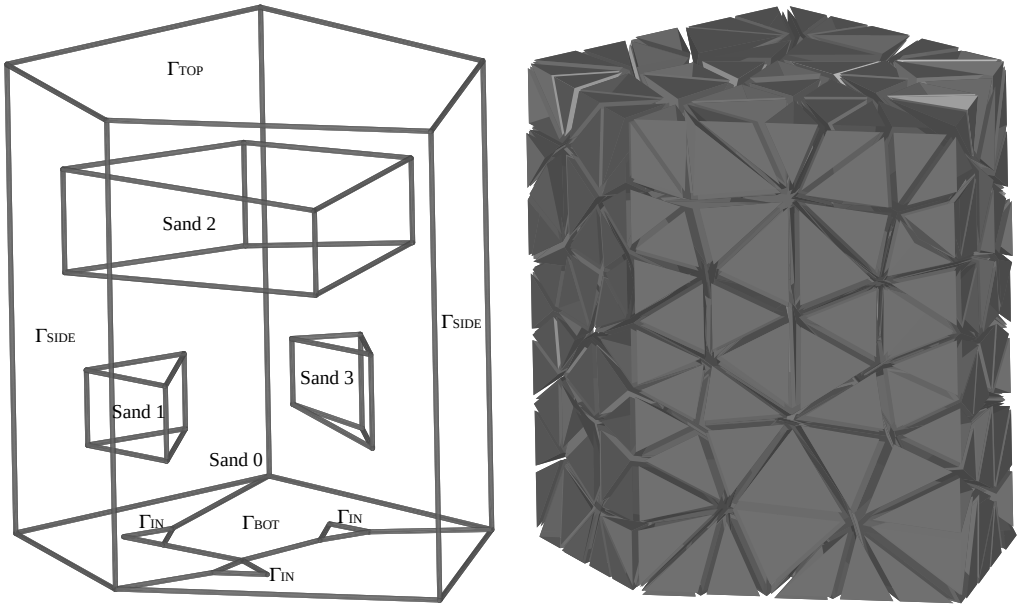


Figure 7.24: Geometry (left) and coarse grid (right) for 3D air sparging problem (Visualization with GRAPE).

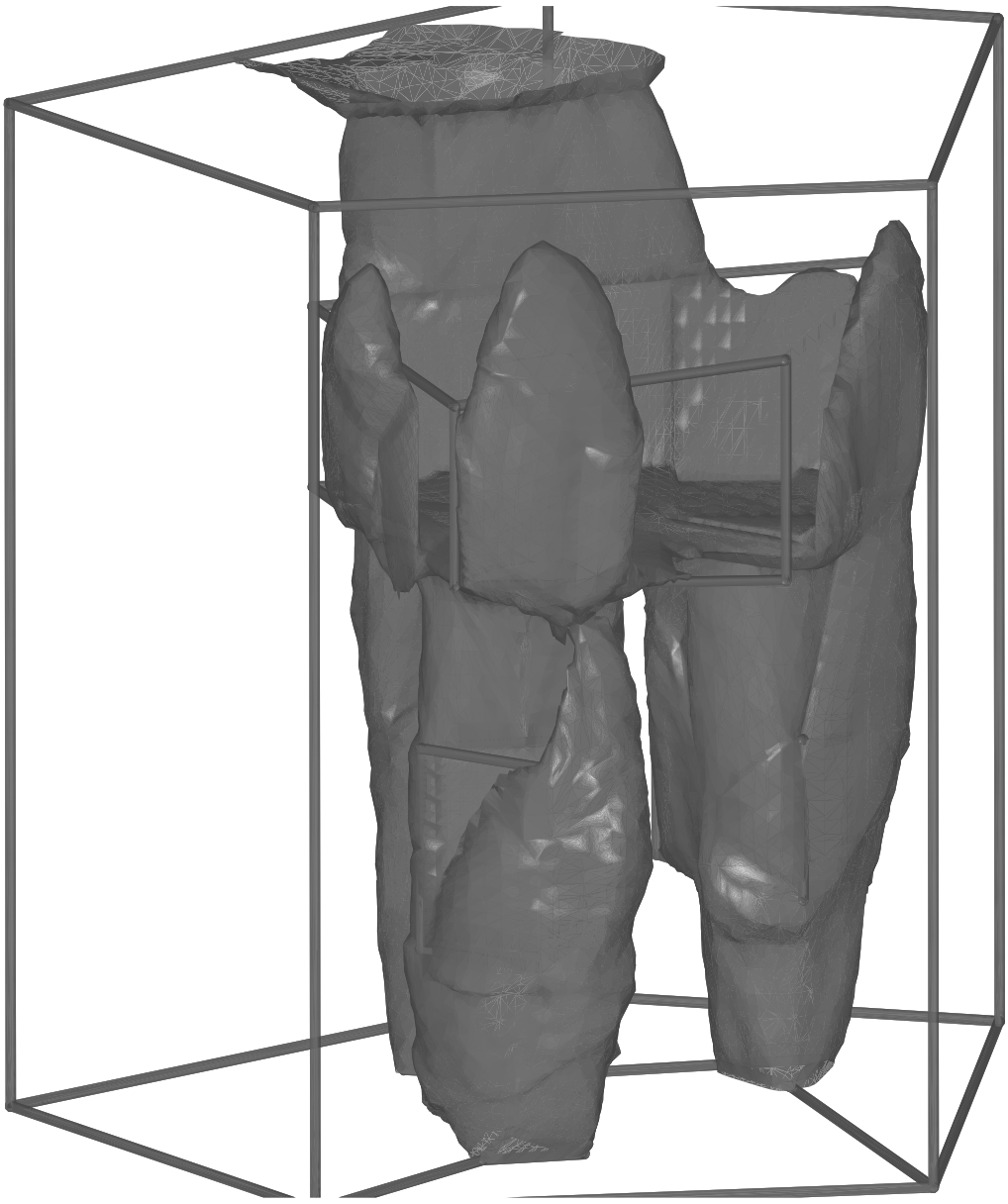


Figure 7.25: Isosurface $S_n = 0.05$ after 640 $[s]$ of simulated time in 3D air sparging problem (Visualization with GRAPE).

Conclusion and Future Work

We have demonstrated the effective use of parallel Newton–multigrid techniques for the fully–coupled solution of the two–phase flow problem in this work. For heterogeneous porous media we compared the fully upwinding method and a formulation with explicit incorporation of the interface conditions. The formulation with interface conditions was found to give qualitatively and quantitatively better results on coarser meshes and lead to linear and nonlinear systems that were easier to solve. A global pressure formulation equipped with interface conditions as described in Subs. 2.3.3 would be the preferred method if the NAPL saturation on the high permeable side becomes large.

The techniques presented as well as the computer implementation based on the PDE software tool–box UG are general enough to allow extensions in various directions. The forthcoming work of Lang (1999) will extend UG with adaptive local mesh refinement and dynamic load balancing capabilities for time–dependent problems. The solutions of multiphase flow problems exhibiting shocks and free boundaries will greatly benefit from the use of adaptive local mesh refinement provided a good error indicator can be found. First results (in sequential mode) are promising.

Another direction of future work will be the incorporation of more complex mathematical models. Three phase/three component models (isothermal and non–isothermal) are currently being developed by R. Helmig and his group on the basis of the simulator developed in this work. First results have been presented in Huber and Helmig (1998). The extension to fractured porous media is also being worked on.

The now existing two–phase simulator is used in the computation of water–gas flows for the purpose of security assessment of underground waste repositories. Its ability to solve large–scale problems makes it also an ideal tool to investigate “numerical upscaling” where one tries to identify effective parameters and/or processes, see Pruess (1996) and Ewing (1997), for coarse grid numerical models that match fine grid computations thus addressing the fundamental problem of porous medium flow modeling.

Bibliography

Alcouffe, R., A. Brandt, J. Dendy, and J. Painter (1981). The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput. 2*, 430–454.

Allen, M. (1985). Numerical modelling of multiphase flow in porous media. *Adv. Water Res. 8*, 162–187.

Allen, M., G. Behie, and J. Trangenstein (1992). *Multiphase Flow in Porous Media*, Volume 34 of *Lecture Notes in Engineering*. Springer–Verlag.

Axelsson, O. and V. Barker (1984). *Finite Element Solution of Boundary Value Problems*. Academic Press.

Aziz, K. and A. Settari (1979). *Petroleum Reservoir Simulation*. Elsevier.

Balay, S., W. Gropp, L. McInnes, and B. Smith (1997). Efficient management of parallelism in object–oriented numerical software libraries. In E. Arge, A. Bruaset, and H. Langtangen (Eds.), *Modern Software Tools for Scientific Computing*. Birkhäuser. `http://www.mcs.anl.gov/petsc/petsc.html`.

Bank, R., A. Sherman, and A. Weiser (1983). Refinement algorithms and data structures for regular local mesh refinement. In *Scientific Computing*. IMACS, North–Holland.

Bank, R. and C. Wagner (1998). Multilevel ILU decomposition. to appear in Numerische Mathematik.

Barrett, R., M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM.

Bastian, P. (1993). Parallel adaptive multigrid methods. Technical Report 93–60, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen.

Bastian, P. (1996). *Parallele adaptive Mehrgitterverfahren*. Teubner–Verlag.

Bastian, P. (1998). Load balancing for adaptive multigrid methods. *SIAM J. Sci. Stat. Comput. 19*(4), 1303–1321.

Bastian, P., K. Birken, S. Lang, K. Johannsen, N. Neuß, H. Rentz-Reichert, and C. Wieners (1997). UG: A flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science 1*, 27–40.

Bear, J. (1972). *Dynamics of Fluids in Porous Media*. Dover Publications.

Bear, J. (1979). *Hydraulics of Groundwater*. McGraw–Hill.

Bear, J. and Y. Bachmat (1991). *Introduction to Modeling of Transport Phenomena in Porous Media*. Kluwer Academic Publishers.

Bey, J. (1995). Tetrahedral grid refinement. *Computing 55*, 355–378.

Bey, J. (1997). *Finite–Volumen– und Mehrgitterverfahren für elliptische Randwertprobleme*. Ph. D. thesis, Universität Tübingen.

Bey, J. and G. Wittum (1997). Downwind numbering: Robust multigrid for convection–diffusion problems. *Appl. Numer. Math. 23*, 177–192.

Binning, P. and M. Celia (1994). Eulerian–Lagrangian localized adjoint methods for contaminant transport simulations. In A. P. et al. (Ed.), *Computational Methods in Water Resources X*. Kluwer Academic Publishers.

Birken, K. (1998). *Ein Modell zur effizienten Parallelisierung von Algorithmen auf komplexen, dynamischen Datenstrukturen*. Ph. D. thesis, Universität Stuttgart.

Birken, K. and P. Bastian (1994). Distributed Dynamic Data (DDD) in a parallel programming environment - Specification and functionality. Technical Report RUS–22, Rechenzentrum der Universität Stuttgart.

Bokhari, S. (1981). On the mapping problem. *IEEE Transactions on Computers 30*(3), 207–214.

Braess, D. (1992). *Finite Elemente*. Springer–Verlag.

Braess, D. (1995). Towards algebraic multigrid for elliptic problems of second order. *Computing 55*, 379–393.

Brakhagen, F. and T. Fogwell (1990). Multigrid for the fully implicit formulation of the equations for multiphase flow in porous media. In *Multigrid Methods: Special topics and applications*, Volume II, pp. 31–42.

Bramble, J. (1993). *Multigrid Methods*. Pitman Research Notes in Mathematics Series. Longman Scientific & Technical.

Bramble, J., J. Pasciak, J. Wang, and J. Xu (1991). Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comput. 57*, 1–22.

Brenner, S. and R. Scott (1994). *The mathematical theory of finite element methods*. Springer.

Briggs, W. (1987). *A Multigrid Tutorial*. SIAM.

Brooks, A. and T. Hughes (1982). Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equation. *Computer Methods in Applied Mechanics and Engineering 32*, 199–259.

Brooks, R. and A. Corey (1964). *Hydraulic Properties of Porous Media*, Volume 3 of *Colorado State University Hydrology Paper*. Colorado State University.

Bruaset, A. and H. Langtangen (1997). A comprehensive set of tools for solving partial differential equations; Diffpack. In M. Dæhlen and A. Tveito

(Eds.), *Numerical Methods and Software Toolsin Industrial Mathematics*. Birkhäuser.

Cai, X. (1998). Domain decomposition in high–level parallelization of PDE codes. http://www.ifi.uio.no/∼xingca.

Celia, M. (1994). Two–dimensional Eulerian–Lagrangian localized adjoint method for the solution of the contaminant transport equation in the saturated and unsaturated zones. In A. P. et al. (Ed.), *Computational Methods in Water Resources X*. Kluwer Academic Publishers.

Celia, M., T. Russel, I. Herrera, and R. Ewing (1990). An Eulerian–Lagrangian localized adjoint method for the advection–diffusion equation. *Adv. Water Resources 13(4)*, 187–206.

Chavent, G. and J. Jaffré (1978). *Mathematical Models and Finite Elements for Reservoir Simulation*. North–Holland.

Chen, Z., R. Ewing, and M. Espedal (1994). Multiphase flow simulation with various boundary conditions. In *Proceedings of the International Conference on Computational Methods in Water Resources X*, pp. 925–932.

Chung, T. (1996). *Applied Continuum Mechanics*. Cambridge University Press.

Corey, A. (1994). *Mechanics of Immiscible Fluids in Porous Media* (3rd ed.). Water Resources Publications.

Cybenko, G. (1989). Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing 7*, 279–301.

Dawson, C. (1991). Godunov–mixed methods for advective flow problems in one space dimension. *SIAM J. Numer. Anal. 28(5)*, 1282–1309.

Dawson, C., H. Klíe, M. Wheeler, and C. Woodward (1997). A parallel, implicit, cell centered method for two–phase flow with a preconditioned Newton–Krylov solver. Technical Report UCRL–JC–127724, Lawrence Livermoore National Laboratory.

de Keyser, J. and D. Roose (1991). Adaptive irregular multiple grids on a distributed memory multiprocessor. In A. Bode (Ed.), *Proc. of the 2nd European Distributed Memory Computing Conference*, pp. 153–162.

de Keyser, J. and D. Roose (1992). Partitioning and mapping adaptive multigrid hierarchies on dirstributed memory computers. Technical Report TW 166, Dept. of Computer Science, K. U. Leuven.

de Neef, M. and J. Molenaar (1997). Analysis of DNAPL infiltration in a medium with a low permeable lense. *Computational Geosciences 1*, 191–214.

Dendy Jr., J. (1987). Two multigrid methods for three–dimensional problems with discontinuous and anisotropic coefficients. *SIAM J. Sci. Stat. Comput. 8*, 673–685.

Diffpack (1998). Diffpack Home Page. http://www.noobjects.com/products/diffpack.

Donea, J. (1984). A Taylor–Galerkin method for convective transport problems. *Int. J. for Numerical Methods in Engineering 20*, 101–119.

Douglas Jr., J., F. Furtado, and F. Pereira (1997). On the numerical simulation of waterflooding of heterogeneous petroleum reservoirs. *Computational Geosciences 1*, 155–190.

Douglas Jr., J., D. Peaceman, and H. Rachford Jr. (1959). A method for calculating multi–dimensional displacement. *Trans. AIME 216*, 297–308.

Douglas Jr., J. and T. Russel (1982). Numerical methods for convection dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures. *SIAM J. Numer. Anal. 19(5)*, 871–885.

Dryja, M., M. Sarkin, and O. Widlund (1996). Multilevel Schwartz methods for elliptic problems with discontinuous coefficients in three space dimensions. *Numer. Math. 72*, 313–348.

Dryja, M. and O. Widlund (1990). Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. Chan, R. Glowinski, J. Périaux, and O. Widlund (Eds.), *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pp. 3–21. SIAM.

Durlofsky, L. (1993). A triangle based mixed finite element—finite volume technique for modeling two–phase flow through porous media. *Journal of Computational Physics 105*, 252–266.

Durlofsky, L. (1994). Accuracy of mixed and control volume finite element approximations to Darcy velocity and related quantities. *Water Resources Research 30(4)*, 965–973.

Eisenstat, S. and H. Walker (1996). Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Stat. Comput. 17*(1), 16–32.

Emmert, M. (1997). *Numerische Modellierung nichtisothermer Gas–Wasser Systeme in porösen Medien*. Ph. D. thesis, Universität Stuttgart.

Eriksson, K., D. Estep, P. Hansbo, and C. Johnson (1995). Introduction to adaptive methods for differential equations. Acta Numerica.

Espedal, M. and R. Ewing (1987). Characteristic Petrov–Galerkin subdomain methods for two–phase immiscible flow. *Computer Methods in Applied Mechanics and Engineering 64*, 113–135.

Ewing, R. (1983). Problems arising in the modeling of processes for hydrocarbon recovery. In R. Ewing (Ed.), *Research Frontiers in Applied Mathematics*, Volume 1, pp. 3–34. SIAM.

Ewing, R. (1991). Operator splitting and Eulerian–Lagrangian localized adjoint methods for multiphase flow. In *The Mathematics of Finite Elements and Applications VII*.

Ewing, R. (1997). Aspects of upscaling in simulation of flow in porous media. *Adv. Water Resources 20*(5-6), 349–358.

Ewing, R., R. Lazarov, J. Pasciak, and A. Vassilev (1995). Mathematical modeling, numerical techniques, and computer simulation of flows and transport in porous media. In *Computational Techniques and Applications*, pp. 1–17.

Ewing, R., H. Wang, and R. Sharpley (1994). Eulerian–Lagrangian localized adjoint methods for transport of nuclear waste contamination in porous media. In A. P. et al. (Ed.), *Computational Methods in Water Resources X*. Kluwer Academic Publishers.

Ewing, R. and M. Wheeler (1980). Galerkin methods for miscible displacement problems in porous media. *SIAM J. Numer. Anal. 17*, 351–365.

Falta, R. (1992). *Multiphase Transport of Organic Chemical Contaminants in the Subsurface*. Ph. D. thesis, Department of Material Sciences and Mineral Engineering, University of California, Berkeley.

Fein, E. (Ed.) (1998). *d^3f – Ein Programmpaket zur Modellierung von Dichteströmungen*.

Fiduccia, C. and R. Mattheyses (1982). A linear time heuristic for improving network partitions. In *Proceedings of the 19th IEEE Design Automation Conference*, pp. 175–181.

Forsyth, P. (1991). A control volume finite element approach to NAPL groundwater contamination. *SIAM J. Sci. Stat. Comput. 12(5)*, 1029–1057.

Forsyth, P. and B. Shao (1991). Numerical simulation of gas venting for NAPL site remediation. *Adv. Water Resources 14(6)*, 354–367.

Fox, G. (1986, November). A graphical approach to load balancing and sparse matrix vector multiplication on the hypercube. Presented at Minnesota Institute for Mathematics and its Applications Workshop.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1995). *Design Patterns*. Addison–Wesley.

Garey, M., D. Johnson, and L. Stockmeyer (1976). Some simplified NP–complete graph problems. *Theoretical Computer Science 1*, 237–267.

Glimm, J., E. Isaacson, B. Lindquist, O. McBryan, and S. Yaniv (1983). Statistical fluid dynamics: The influence of geometry on surface instabilites. In R. Ewing (Ed.), *Research Frontiers in Applied Mathematics*, Volume 1, pp. 137–160. SIAM.

Glimm, J., B. Lindquist, O. McBryan, and L. Padmanabhan (1983). A front tracking reservoir simulator, five–spot validation studies and the water coning problem. In R. Ewing (Ed.), *Research Frontiers in Applied Mathematics*, Volume 1, pp. 101–136. SIAM.

Glimm, J., D. Marchesin, and O. McBryan (1981). Unstable fingers in two phase flow. *Comm. Pure Appl. Math. 34*, 53–75.

Golub, G. and C. Van Loan (1989). *Matrix Computations*. John Hopkins University Press.

Griebel, M. and G. Zumbusch (1998). Hash–storage techniques for adaptive multilevel solvers and their domain decomposition parallelization. *Contemporary Mathematics 218*, 279–286.

Gundersen, E. and H. Langtangen (1997). Finite element methods for two–phase flow in heterogeneous porous media. In *Numerical Methods and Software Tools in Industrial Mathematics*. Birkhäuser.

Hackbusch, W. (1985). *Multi–Grid Methods and Applications*. Springer–Verlag.

Hackbusch, W. (1994). *Iterative Solution of Large Sparse Systems of Linear Equations*. Springer.

Hackbusch, W. (1997). On the feedback vertex set problem for a planar graph. *Computing 58*(2), 129–155.

Hackbusch, W. and T. Probst (1997). Downwind Gauß–Seidel smoothing for convection dominated problems. *Numerical Linear Algebra With Applications 4*(2), 85–102.

Hairer, E. and G. Wanner (1991). *Solving ordinary differential equations II*. Springer, Berlin.

Hassanizadeh, M. and W. Gray (1979a). General conservation equations for multiphase systems: 1. averaging procedure. *Adv. Water Res. 2*, 1–14.

Hassanizadeh, M. and W. Gray (1979b). General conservation equations for multiphase systems: 2. mass momentum energy and entropy conditions. *Adv. Water Res. 2*, 191–203.

Hassanizadeh, M. and W. Gray (1980). General conservation equations for multiphase systems: 3. constitutive theory for porous media flow. *Adv. Water Res. 3*, 30–44.

Heinrich, J., P. Huyakorn, O. Zienkiewicz, and A. Mitchell (1977). An upwind finite element scheme for two–dimensional convective transport equation. *Int. J. for Numerical Methods in Engineering 11*, 131–143.

Heise, B. and M. Jung (1995). Comparison of parallel solvers for nonlinear elliptic problems based on domain decomposition ideas. Technical report, Johannes Kepler Universität Linz, Institut für Mathematik. Institutsbericht Nr. 494.

Helmig, R. (1997). *Multiphase Flow and Transport Processes in the Subsurface – A Contribution to the Modeling of Hydrosystems*. Springer–Verlag.

Helmig, R., H. Class, R. Huber, H. Sheta, J. Ewing, R. Hinkelmann, H. Jakobs, and P. Bastian (1998). Architecture of the modular program system MUFTE–UG for simulating multiphase flow and transport processes in heterogeneous porous media. to appear.

Helmig, R. and R. Huber (1996). Multiphase flow in heterogeneous porous media: A classical finite element method versus an IMPES–based mixed FE/FV approach. Technical Report 19, Sonderforschungsbereich 404, Universität Stuttgart. to appear in Int. J. Numer. Meth. in Fluids.

Helmig, R. and R. Huber (1998). Comparison of Galerkin–type discretization techniques for two–phase flow in heterogenous porous media. *Adv. Water Resources 21*(8), 697–711.

Hendrickson, B. and R. Leland (1992). An improved spectral graph partitioning method for mapping parallel computations. Technical Report SAND92–1460, Sandia National Laboratory.

Hendrickson, B. and R. Leland (1993a). The CHACO user's guide 1.0. Technical Report SAND93–2339, Sandia National Laboratories.

Hendrickson, B. and R. Leland (1993b). A multilevel algorithm for partitioning graphs. Technical Report SAND93–1301, Sandia National Laboratory.

Hornung, U. (1997). *Homogenization and Porous Media*. Springer–Verlag.

Huber, R. and R. Helmig (1998). Simulation of multiphase and compositional flow in porous media. In *Proceedings of XII International Conference on Computational Methods in Water Resources*. Crete.

Hvistendahl Karlsen, K., K. Lie, N. Risebro, and J. Frøyen (1997). A front tracking approach to a two–phase fluid flow model with capillary forces. Technical report, University of Bergen.

Jahresbericht der Wasserwirtschaft (1993). Gemeinsamer Bericht der mit der Wasserwirtschaft befassten Bundesministerien – Haushaltsjahr 1992. *Wasser und Boden 45*(5), 504–516.

Jones, M. and P. Plassmann (1997). Parallel algorithms for adaptive mesh refinement. *SIAM J. on Scientific Computing 18*, 686–708.

Karypis, G. and V. Kumar (1995). Multilevel *k*-way partitioning scheme for irregular graphs. Technical Report 95–064, University of Minnesota, Department of Computer Science.

Karypis, G. and V. Kumar (1996). Parallel multilevel *k*-way partitioning scheme for irregular graphs. Technical Report 96–036, University of Minnesota, Department of Computer Science.

Kernighan, B. and S. Lin (1970). An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal 49*, 291–307.

Kettler, R. (1982). Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In *Multi–grid methods*. Springer. Lecture Notes in Math 960.

Kinzelbach, W. and W. Schäfer (1992). Stochastic modeling of in–situ bioremediation in heterogeneous aquifers. *Journal of Contaminant Hydrology 10*, 47–73.

Klaas, O., R. Niekamp, and E. Stein (1994). Parallel adaptive finite element computations with hierarchical preconditioning. Technical Report IBNM–Bericht 94/4, IBNM, Uni Hannover.

Kobus, H. (1996). The role of large–scale experiments in groundwater and subsurface remediation research: The VEGAS concept and approach. In H. Kobus, B. Barczewski, and H. Koschitzky (Eds.), *Groundwater and Subsurface Remediation*, pp. 1–18. Springer–Verlag.

Kroener, D. and S. Luckhaus (1984). Flow of oil and water in a porous medium. *Journal of Differential Equations 55*, 276–288.

Kueper, B. and E. Frind (1988). An overview of immiscible fingering in porous media. *Journal of Contaminant Hydrology 2*, 95–110.

Lampe, M. (1997). Parallelisierung eines Grafiksubsystems in einem Paket zur numerischen Lösung partieller Differentialgleichungen. Master's thesis, Uni Stuttgart.

Lang, S. (1999). *Parallele adaptive Mehrgitterverfahren für dreidimensionale instationäre Berechnungen*. Ph. D. thesis, Universität Heidelberg. to appear, tentative title.

LeVeque, R. (1992). *Numerical Methods for Conservation Laws*. Birkhäuser.

Leverett, M. (1941). Capillary behavior in porous solids. *Trans. AIME 142*, 152–169.

McWhorter, D. and D. Sunada (1990). Exact integral solutions for two–phase flow. *Water Resources Research 26(3)*, 399–413.

Michev, I. (1996). *Finite volume and finite volume element methods for nonsymmetric problems*. Ph. D. thesis, Texas A&M University.

Mitchell, W. (1998). FUDOP Home Page. `http://math.nist.gov/Staff/WMitchell`.

Molenaar, J. (1994). A simple multigrid method for 3D interface problems. Technical report, TU Delft. Technical Report 94–44.

Molenaar, J. (1995). Multigrid methods for fully implicit oil reservoir simulation. In *Proceedings Copper Mountain Conference on Multigrid Methods*.

Mulder, W. and R. G. Meyling (1993). Numerical simulation of two–phase flow using locally refined grids in three space dimensions. *SPE Advanced Technology Series 1(1)*, 36–41.

Muskat, M., R. Wyckoff, H. Botset, and M. Meres (1937). Flow of gas–liquid mixtures through sands. *Pet. Trans. AIME 123*, 69–82.

Neuß, N. (1999). A new sparse matrix storage method for adaptive solving of large systems of reaction-diffusion-transport equations. Technical Report 1999–04, IWR, Uni Heidelberg.

PadFEM (1998). PadFEM Home Page. `http://www.uni-paderborn/fachbereich/AG/monien/SOFTWARE/PADFEM`.

Parker, J., R. Lenhard, and T. Kuppusami (1987). A parametric model for constitutive properties governing multiphase flow in porous media. *Water Resources Research 23(4)*, 618–624.

Peaceman, D. W. (1977). *Fundamentals of Numerical Reservoir Simulation*. Elsevier.

POET (1998). POET Home Page. `http://glass-slipper.ca.sandia.gov/poet`.

POOMA (1998). POOMA Home Page. `http://www.acl.lanl.gov/pooma`.

Pothen, A., H. Simon, and K. Liou (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl. 11*(3), 430–452.

Pruess, K. (1991). TOUGH2–A general purpose numerical simulator for multiphase fluid and heat flow. Technical Report LBL–29400, Lawrence Berkeley Laboratory.

Pruess, K. (1996). Effective parameters, effective processes: From porous flow physics to in–situ remediation technology. In H. Kobus, B. Barczewski, and H. Koschitzky (Eds.), *Groundwater and Subsurface Remediation*, pp. 183–193. Springer–Verlag.

Rannacher, R. (1988). Numerical analysis of nonstationary fluid flow. Technical Report 492, SFB 123, Universität Heidelberg.

Rannacher, R. (1994). Accurate time discretization schemes for computing nonstationary incompressible fluid flow. In *Proceedings of the International Conference on Computational Methods in Water Resources X*, pp. 1239–1246.

Raw, M. (1996). Robustness of coupled algebraic multigrid for the Navier–Stokes equations. Technical Report 96–0297, AIAA.

Renardy, M. and R. Rogers (1993). *An Introduction to Partial Differential Equations*. Springer–Verlag.

Rentz–Reichert, H. (1996). *Robuste Mehrgitterverfahren zur Lösung der inkompressiblen Navier–Stokes Gleichung: Ein Vergleich*. Ph. D. thesis, Universität Stuttgart.

Reusken, A. (1995a). Fourier analysis of a robust multigrid method for convection–diffusion equations. *Numer. Math. 71*, 365–397.

Reusken, A. (1995b). A multigrid method based on incomplete Gaussian elimination. Technical report, Eindhoven University of Technology. Department of Mathematics and Computing Science, Report RANA 95–13.

Reusken, A. (1996). On a robust multigrid solver. *Computing 56*, 303–322.

Richards, L. (1931). Capillary conduction of liquids in porous media. *Physics 1*, 318–333.

Risebro, N. and A. Tveito (1991). Front tracking applied to a nonstrictly hyperbolic system of conservation laws. *SIAM J. Sci. Stat. Comput. 12*, 1401–1419.

Ruge, J. and K. Stüben (1987). Algebraic multigrid. In S. F. McCormick (Ed.), *Multigrid Methods*. SIAM.

Rumpf, M., R. Neubauer, M. Ohlberger, and R. Schwörer (1997). Efficient visualization of large–scale data on hierarchical meshes. In W. Lefer and M. Grave (Eds.), *Visualization in Scientific Computing '97*. Springer.

Russel, T. (1985). Time stepping along characteristics with incomplete iteration for a Galerkin approximation of miscible displacement in porous media. *SIAM J. Numer. Anal. 22(5)*, 970–1013.

Sadayappan, P. and F. Ercal (1987). Nearest–neighbor mapping of finite element graphs onto processor meshes. *IEEE Transactions on Computers C-36*(12), 1408–1424.

Scheidegger, A. (1961). General theory of dispersion in porous media. *Journal of Geophysical Research 66*, 3273–3278.

Scheidegger, A. (1974). *The Physics of Flow Through Porous Media*. University of Toronto Press.

Schloegel, K., G. Karypis, and V. Kumar (1997). Multilevel diffusion schemes for repartitioning of adaptive meshes. Technical Report 97–013, University of Minnesota, Department of Computer Science.

Schöberl, J. (1997). A rule–based tetrahedral mesh generator. *Computing and Visualization in Science 1*, 1–26.

Schroll, H. and A. Tveito (1997). Local existence and stability for a hyperbolic–elliptic system modeling two–phase reservoir flow. Technical Report 136, Institut für Geometrie und Praktische Mathematik, RWTH Aachen.

SCOREC (1998). SCOREC Home Page. `http://www.scorec.rpi.edu`.

Scott, T. (1985). Multi–grid methods for oil reservoir simulation in two and three dimensions. *J. Comput. Phys. 59*, 290–307.

Sheta, H. (1999). *Einfluss der Hysterese bei Infiltrations– und Ausbreitungsvorgängen in der gesättigten und ungesättigten Bodenzone*. Ph. D. thesis, Universität Stuttgart, Institut für Wasserbau.

Smith, B., P. Bjørstad, and W. Gropp (1996). *Domain Decomposition*. Cambridge University Press.

Stone, H. (1973). Estimation of three–phase relative permeability and residual oil data. *Journal Can. Petro. Technol. 12(4)*, 53–61.

Sumaa3d (1998). Sumaa3d Home Page. `http://www.mcs.anl.gov/sumaa3d`.

Van de Velde, E. (1993). *Concurrent Scientific Computing*. Springer Verlag.

Van der Vorst, H. (1992). BiCGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of non–symmetric linear systems. *SIAM J. Sci. Stat. Comput. 13*, 631–644.

Van Driesche, R. and D. Roose (1995). An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing 21*, 29–48.

van Duijn, C., J. Molenaar, and M. de Neef (1995). Effects of capillary forces on immiscible two–phase flow in heterogeneous porous media. *Transport in Porous Media 21*, 71–93.

van Dyke, M. and S. van der Zee (1998). Modeling of air sparging in a layered soil: Numerical and analytical approximations. *Journal of Geophysical Research 34*, 341–353.

Van Genuchten, M. (1980). A closed form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. J. 44*, 892–898.

Vaněk, P., J. Mandel, and M. Brezina (1996). Algebraic multi–grid by smoothed aggregation for second and forth order elliptic problems. *Computing 56*, 179–196.

Varga, R. (1962). *Matrix Iterative Analysis*. Prentice Hall.

Verfürth, R. (1988). Multi–level algorithms for mixed problems II. Treatment of the Mini–Element. *SIAM J. Numer. Anal. 25*, 285–293.

Wagner, C., W. Kinzelbach, and G. Wittum (1997). A robust multigrid method for groundwater flow. *Numer. Math. 75*, 523–545.

Walshaw, C. and M. Berzins (1993). Enhanced dynamic load–balancing of adaptive unstructured meshes. In *Proc. of the 6th Conf. on Parallel Processing*, pp. 971–979.

Walshaw, C. and M. Cross (1998). Mesh partitioning: A multilevel balancing and refinement algorithm. Technical Report 98/IM/35, University of Greenwich, Centre for Numerical Modelling and Process Analysis.

Walshaw, C., M. Cross, and M. Everett (1997). Dynamic load–balancing for parallel adaptive unstructured meshes. In *Proc. of the 8th Conf. on Parallel Processing*.

Watson, A., J. Wade, and R. Ewing (1994). Parameter and system identification for fluid flow in underground reservoirs. In H. Engl and J. McLaughlin (Eds.), *Proceedings of the Conference on "Inverse Problems and Optimal Design in Industry"*, Volume 10, pp. 81–108. B. G. Teubner. Europ. Cons. for Meth. Ind.

Wesseling, P. (1992). *An Introduction to Multigrid Methods*. John Wiley.

Whitaker, S. (1986a). Flow in porous media I: A theoretical derivation of Darcy's law. *Transport in Porous Media 1*, 3–25.

Whitaker, S. (1986b). Flow in porous media II: The governing equations for immiscible two–phase flow. *Transport in Porous Media 1*, 105–125.

Wieners, C. (1997). The implementation of parallel multigrid methods for finite elements. available in electronic form: `ftp://ftp.ica3.uni-stuttgart.de/pub/text/wieners/wieners13.ps.gz`.

Williams, R. (1990). Performance of dynamic load balancing algorithms for unstructured mesh calculations. Technical Report C3P 913, California Institute of Technology.

Wittum, G. (1989). On the robustness of ILU smoothing. *SIAM J. Sci. Stat. Comput. 10*, 699–717.

Wittum, G. (1990). On the convergence of multigrid methods with transforming smoothers. *Numerische Mathematik 57*, 15–38.

Xu, J. (1992). Iterative methods by space decomposition and subspace correction. *SIAM Review 34*, 581–613.

Yotov, I. (1997). A mixed finite element discretization on non–matching multiblock grids for a degenerate parabolic equation arising in porous media flow. *East–West J. Numer. Math. 5*, 211–230.

Young, D. (1971). *Iterative Solution of Large Linear Systems*. Academic Press.

Index