Efficient Fully–Coupled Solution Techniques for Two–Phase Flow in Porous Media

Parallel multigrid solution and large scale computations

Peter Bastian
Institut für Computeranwendungen III,
Universität Stuttgart
Pfaffenwaldring 27, 70569 Stuttgart,
email: peter@ica3.uni-stuttgart.de

Rainer Helmig
Institut für Angewandte Mechanik und Bauinformatik
FG für Numerische Methoden und Informationsverarbeitung
Technische Universität Braunschweig,
Pockelsstr. 3, D–38106 Braunschweig
email: r.helmig@tu-bs.de

Abstract

This paper is concerned with the fast resolution of nonlinear and linear algebraic equations arising from a fully implicit finite volume discretization of two–phase flow in porous media. We employ a Newton–multigrid algorithm on unstructured meshes in two and three space dimensions. The discretized operator is used for the coarse grid systems in the multigrid method. Problems with discontinuous coefficients are avoided with a novel truncated restriction operator and use of an outer Krylov–space method.

We show experimentally an optimal order of convergence for a wide range of two–phase flow problems including heterogeneous media and vanishing capillary pressure. Furthermore, we also present a data parallel implementation of the algorithm with speedup results.

1 Introduction

This paper is concerned with the fast resolution of nonlinear algebraic equations arising from a fully implicit finite volume discretization of two–phase flow in porous media and the presentation of large scale computations on sequential and parallel computers.

Multigrid methods, [10, 21], are among the fastest methods to solve large sparse systems of linear equations arising from the discretization of partial differential equations. The most prominent feature of these methods is their optimal order of convergence when applied to elliptic model problems, i. e. the time required to solve a system of linear equations up to a certain accuracy is proportional to the number of unknowns.

Within a decoupled approach using the fractional flow formulation, [8], the application of multigrid to two–phase flow problems is rather straightforward. It can be applied independently to the (scalar, linearized) pressure equation and the saturation equation. Often multigrid is only applied to the elliptic pressure equation, for this approach see e. g. [16].

The fully coupled solution method with a pressure–saturation formulation leads to a set of nonlinear algebraic equations to be solved per time step. The resolution of nonlinear equations by multigrid is possible with two different approaches. In the first approach, a global linearization (e. g. Newton's method) is performed and the resulting linear equations are solved with multigrid. The second approach is nonlinear multigrid where the linearization is only done within the smoothing iteration and a nonlinear coarse grid problem is set up. Molenaar, [15], has compared the two approaches and found that Newton–multigrid is more efficient in terms of computer time. In this paper we also follow the global linearization approach since the reduction of computer time is of primary importance and also the robust multigrid techniques are more developed within the linear framework.

The linear multigrid method offers several choices for its three basic components: smoother, grid transfer and coarse grid operator. The suitable selection of components depends strongly on the problem to be solved. Therefore, we take a closer look at the linearized two–phase equations whose discretization is the Jacobian system. We will show that the linearized operator displays solution–dependent, strong variations in the coefficients that are not aligned with coarse grid element boundaries. Moreover, in the case of vanishing capillary pressure, it contains a dominant convective term. All

these effects are complicated by the coupling of the system. In addition, anisotropy may be introduced through the absolute permeability tensor or the grid construction (however we will not concentrate on anisotropy effects here).

Because of the varying coefficients, care must be taken in the construction of the coarse grid correction. The two options are either a Galerkin approach or using the discretized operator on the coarser level. Due to the character of the two–phase system, a standard Galerkin method may lead to instability but an aggregation type approach has been used successfully by Molenaar, [15]. See also [17, 6] for this kind of coarse grid operator. In this paper, we will use the discretized (linearized) operator on the coarse grid. The successful use of this approach is made possible by a novel truncated restriction operator.

With the methods proposed in this paper, the computer time needed for the resolution of the linear systems is (for a large class of problems) comparable to the time needed for setting up the linear systems and independent of the mesh size. However, a time–dependent, three–dimensional calculation still requires a substantial amount of computer time. Therefore, we consider the parallelization of the multigrid algorithm via data partitioning as described in [4]. The respective algorithms and constitutive relationships are incorporated in the numerical simulator MUFTE–UG which is based on the software toolbox *UG* [5].

The rest of this paper is organized as follows. In the next section we shortly review the numerical model and the discretization scheme. Then we describe the damped, inexact Newton–method that is used to solve the nonlinear systems. The following section then describes the multigrid method and its components. After describing the parallelization strategy, the numerical results are presented.

2 Numerical Model

2.1 Pressure–Saturation Formulation

Let $\Omega \in \mathbb{R}^d$, $d = 2, 3$ be an open domain and $I = (0, T)$ the time interval of interest. The equations for the flow of two immiscible fluid phases w (wetting) and n (nonwetting) in a porous medium are given by the conservation

of mass ($\alpha = w, n$)

$$\frac{\partial(\Phi \varrho_\alpha S_\alpha)}{\partial t} + \nabla \cdot (\varrho_\alpha v_\alpha) = \varrho_\alpha q_\alpha, \quad \text{in } \Omega \times I, \tag{1}$$

and the generalized *Darcy Law*

$$v_\alpha = -\frac{k_{r\alpha}(x, S_\alpha)}{\mu_\alpha} \boldsymbol{K}(\nabla p_\alpha - \varrho_\alpha g) \quad , \tag{2}$$

where Φ is the porosity of the porous medium, ϱ_α is the density of phase α, S_α is the unknown saturation of phase α, v_α is the volumetric flux vector, q_α is the source/sink term, $\boldsymbol{K}$ is the absolute permeability tensor, $k_{r\alpha}(x, S_\alpha)$ is the relative permeability, μ_α is the dynamic viscosity of the fluid α, p_α is the unknown pressure of phase α and g is the vector of gravitational forces. The model (and its implementation) also handles full compressibility of both fluid phases [12]. In addition to these differential equations, we have the algebraic relations

$$S_w(x,t) + S_n(x,t) = 1 \tag{3}$$

$$p_n(x,t) - p_w(x,t) = p_c(x, S_w(x,t)) \quad . \tag{4}$$

Inserting (2) into (1) for both phases and using the relations $S_w = 1 - S_n$ and $p_n = p_w + p_c(1 - S_n)$, we obtain the pressure–saturation formulation with p_w and S_n as unknowns:

$$\begin{aligned}
&\mathcal{L}_w(p_w, S_n) = \frac{\partial(\Phi \varrho_w (1 - S_n))}{\partial t} + \\
&\nabla \cdot \left\{ -\varrho_w \frac{k_{rw}(1-S_n)}{\mu_w} \boldsymbol{K}(\nabla p_w - \varrho_w g) \right\} \\
&- \varrho_w q_w = 0 \\
&\mathcal{L}_n(p_w, S_n) = \frac{\partial(\Phi \varrho_n S_n)}{\partial t} + \\
&\nabla \cdot \left\{ -\varrho_n \frac{k_{rn}(S_n)}{\mu_n} \boldsymbol{K}(\nabla p_w + \nabla p_c(1 - S_n) - \varrho_n g) \right\} \\
&- \varrho_n q_n = 0
\end{aligned} \tag{5}$$

The system of two coupled nonlinear partial differential equations is supplemented by the following boundary conditions

$$\begin{aligned}
&p_w = g_{wd} \text{ on } \Gamma_{wd}, \quad && \varrho_w v_w \cdot \nu = g_{wn} \text{ on } \Gamma_{wn}, \\
&S_n = g_{nd} \text{ on } \Gamma_{nd}, \quad && \varrho_n v_n \cdot \nu = g_{nn} \text{ on } \Gamma_{nn},
\end{aligned} \tag{6}$$

and initial conditions

$$p_w(x,0) = g_{w0}(x), \quad S_n(x,0) = g_{n0}(x) \quad . \tag{7}$$

The Dirichlet boundary Γ_{wd} must have a positive measure and ν denotes the outward unit normal.

2.2 Discrete Formulation

Eq. (5) is discretized on an unstructured mesh T_h with mesh width h consisting of triangles and quadrilaterals if $d = 2$ or tetrahedra, pyramids, prisms and hexahedra if $d = 3$. Associated with T_h, we have the space of lowest order conforming finite element functions V_h and the space of test functions W_h which are the characteristic functions of vertex centered finite volumes. For $p_{wh}, S_{nh} \in V_h$, an implicit Euler discretization in time and the abbreviation $(u, v) = \int_\Omega uvdx$, we get the discrete form of the equations

$$\begin{aligned}
&\tfrac{1}{\Delta t}\left[\left(\Phi\varrho_w\left(1-S_{nh}^{k+1}\right),w_h\right)-\left(\Phi\varrho_w\left(1-S_{nh}^{k}\right),w_h\right)\right]\\
&-\left(\nabla\cdot\left\{\varrho_w\lambda_{wh}^{k+1}\boldsymbol{K}\left(\nabla p_{wh}^{k+1}-\varrho_w g\right)\right\},w_h\right)\\
&-(\varrho_w q_w,w_h)=0\\
&\tfrac{1}{\Delta t}\left[\left(\Phi\varrho_n S_{nh}^{k+1},w_h\right)-\left(\Phi\varrho_n S_{nh}^{k},w_h\right)\right]\\
&-\left(\nabla\cdot\left\{\varrho_n\lambda_{nh}^{k+1}\boldsymbol{K}\left(\nabla p_{wh}^{k+1}+\nabla p_{ch}^{k+1}-\varrho_n g\right)\right\},w_h\right)\\
&-(\varrho_n q_n,w_h)=0
\end{aligned} \tag{8}$$

$\forall w_h \in W_h$. Superscripts k and $k+1$ denote time levels t^k, t^{k+1} and $\Delta t = t^{k+1} - t^k$. In addition, we used the mobilities defined as

$$\lambda_w = \frac{k_{rw}(1-S_n)}{\mu_w}, \quad \lambda_n = \frac{k_{rn}(S_n)}{\mu_n} \quad . \tag{9}$$

Instead of the implicit Euler scheme we can also use the Crank–Nicholson or BDF(2) methods.

After inserting a basis function representation

$$p_{wh}^k(x) = \sum_{i=1}^{n_h} \hat{p}_{wh,i}^k N_i(x), \quad S_{nh}^k(x) = \sum_{i=1}^{n_h} \hat{S}_{nh,i}^k N_i(x) \tag{10}$$

of the unknown finite element functions (n_h denotes the number of nodes in mesh T_h) and evaluation of all the integrals, we obtain a system of nonlinear algebraic equations for the coefficients $\hat{p}_{wh,i}$ and $\hat{S}_{nh,i}$:

$$F(x^{k+1}) = 0 \quad . \tag{11}$$

The nonlinear function F depends on the time levels t^k and t^{k+1} as well as x^k. The vector $x^{k+1} \in \mathbb{R}^{2n_h}$ contains all unknowns in the following order

$$x^{k+1} = (\hat{p}_{wh,1}^{k+1}, \ldots, \hat{p}_{wh,n_h}^{k+1}, \hat{S}_{nh,1}^{k+1}, \ldots, \hat{S}_{nh,n_h}^{k+1})^T \quad . \tag{12}$$

3 Nonlinear Solution Method

3.1 Newton's Method

The nonlinear system (11) is solved by a damped inexact Newton method given in the following algorithm:

Choose $x^{k+1,0}$; *set* $m = 0$;
while ($\|F(x^{k+1,m})\|_2 / \|F(x^{k+1,0})\|_2 > \varepsilon_{nl}$)
{
  Solve $K(x^{k+1,m})u = -F(x^{k+1,m})$
    with accuracy ε_{lin};
  $x^{k+1,m+1} = x^{k+1,m} + \eta u$;
  $m = m + 1$;
}

The double superscript $k+1, m$ denotes time step $k+1$ and Newton iteration m and $\|.\|_2$ is the euclidean norm of a vector. The damping factor $\eta = (1/2)^q$ is chosen in each step such that

$$\|F(x^{k+1,m+1})\|_2 \leq \left[1 - \frac{1}{4}\left(\frac{1}{2}\right)^q\right] \|F(x^{k+1,m})\|_2 \tag{13}$$

holds for the smallest possible $q \in \{0, 1, \ldots, n_{ls}\}$. The number of line search steps n_{ls} is between 4 and 6. If no such q can be found, the size of the time step is reduced.

The Jacobian matrix K evaluated at $x^{k+1,m}$ is defined as

$$K_{ij}(x^{k+1,m}) = \left.\frac{\partial F_i}{\partial x_j}\right|_{x^{k+1,m}}, \quad 1 \leq i, j \leq 2n_h \quad . \tag{14}$$

As indicated in the algorithm, the Jacobian system need not be solved exactly. We choose the linear reduction ε_{lin} as

$$\varepsilon_{lin} = \min\left(\varepsilon_{min}, \left(\frac{\|F(x^{k+1,m})\|_2}{\|F(x^{k+1,m-1})\|_2}\right)^2\right) \quad . \tag{15}$$

This choice allows quadratic convergence of the Newton method in the final steps. The minimal reduction ε_{min} should not be too large since the convergence of iterative methods may not be monotonic in the sense that saturation is maintained between 0 and 1 (we typically chose $\varepsilon_{min} = 10^{-4}$ in the examples below).

3.2 Linearized Operator

By applying the linearization step already in Eq. (8) we can interpret the Jacobian as the discretization of a linear differential operator. To that end, we write the finite element functions corresponding to the Newton iterates as follows:

$$\begin{aligned} p_{wh}^{k+1,m+1} &= p_{wh}^{k+1,m} + \delta p_{wh} \\ S_{nh}^{k+1,m+1} &= S_{nh}^{k+1,m} + \delta S_{nh} \end{aligned} \quad . \tag{16}$$

Linearising k_{rw}, k_{rn} and p_c at $S_{nh}^{k+1,m}$ and dropping all higher order correction terms yields the following linear equations for the corrections δp_{wh} and δS_{nh}:

$$\begin{aligned} & L_w'\left(p_{wh}^{k+1,m}, S_{nh}^{k+1,m}; \delta p_{wh}, \delta S_{nh}\right) = \\ & -\left(\nabla\cdot\left\{\varrho_w \lambda_{wh}^{k+1,m} \boldsymbol{K}\nabla\delta p_{wh}\right\}, w_h\right) \\ & -\tfrac{1}{\Delta t}\left(\Phi\varrho_w\delta S_{nh}, w_h\right) - \left(\nabla\cdot\left\{\vec{\beta}_w\delta S_{nh}\right\}, w_h\right) \\ & = -\tfrac{1}{\Delta t}\left(\Phi\varrho_w\left(S_{nh}^{k} - S_{nh}^{k+1,m}\right), w_h\right) \\ & -\left(\nabla\cdot\left\{-\varrho_w\lambda_{wh}^{k+1,m}\boldsymbol{K}\left(\nabla p_{wh}^{k+1,m} - \varrho_w g\right)\right\}, w_h\right) \\ & +\left(\varrho_w q_w, w_h\right) \end{aligned} \tag{17}$$

$$\begin{aligned} & L_n'\left(p_{wh}^{k+1,m}, S_{nh}^{k+1,m}; \delta p_{wh}, \delta S_{nh}\right) = \\ & -\left(\nabla\cdot\left\{\varrho_n \lambda_{nh}^{k+1,m} \boldsymbol{K}\nabla\delta p_{wh}\right\}, w_h\right) \\ & +\tfrac{1}{\Delta t}\left(\Phi\varrho_n\delta S_{nh}, w_h\right) + \left(\nabla\cdot\left\{\vec{\beta}_n\delta S_{nh}\right\}, w_h\right) \\ & +\left(\nabla\cdot\left\{\varrho_n\lambda_{nh}^{k+1,m}\ {p_{ch}'}^{k+1,m}\ \boldsymbol{K}\nabla\delta S_{nh}\right\}, w_h\right) \\ & = -\tfrac{1}{\Delta t}\left[\left(\Phi\varrho_n S_{nh}^{k+1,m}, w_h\right) - \left(\Phi\varrho_n S_{nh}^{k}, w_h\right)\right] \\ & -\left(\nabla\cdot\left\{-\varrho_n\lambda_{nh}^{k+1,m}\boldsymbol{K}\left(\nabla p_{nh}^{k+1,m} - \varrho_n g\right)\right\}, w_h\right) \\ & +\left(\varrho_n q_n, w_h\right) \end{aligned} \tag{18}$$

$\forall w_h \in W_h$ and the vectors $\vec{\beta}_\alpha$ being defined as

$$\vec{\beta}_w = -\varrho_w\ {\lambda_w'}^{k+1,m}\ \boldsymbol{K}\left(\nabla p_{wh}^{k+1,m} - \varrho_w g\right) \tag{19}$$

and

$$\begin{aligned} \vec{\beta}_n &= \varrho_n\lambda_n^{k+1,m}\boldsymbol{K}\nabla\left(p_{ch}'\right)^{k+1,m} \\ &\quad -\varrho_n\ {\lambda_n'}^{k+1,m}\ \boldsymbol{K}\left(\nabla p_{nh}^{k+1,m} - \varrho_n g\right) \end{aligned} \tag{20}$$

(remember $p_n = p_w + p_c$). Note that on the right hand side of Eqs. (17, 18) we have $-F(x^{k+1,m})$ as desired.

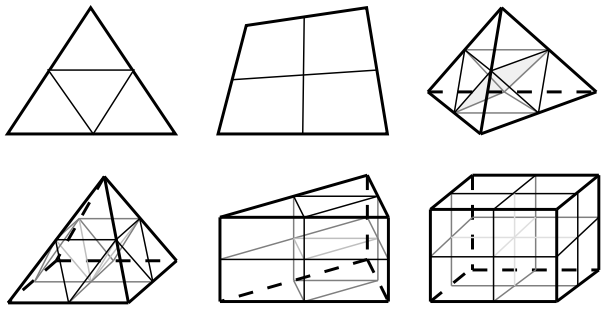

Figure 1: Regular refinement rules.

We will use the linearized operator defined in this section solely for the purpose of getting some insight into the qualitative character of the Jacobian matrix K. A discretization of Eqs. (17) and (18) is equivalent to the Jacobian only up to discretization error. Therefore, the numerical computations will always be done with the true Jacobian given by Eq. (14).

4 Multigrid Method

4.1 Basic Algorithm

For an introduction to multigrid methods, we refer the reader to [10, 21, 7]. The standard multigrid method uses a sequence of $J+1$ nested meshes with increasing fineness

$$T_0 \subset T_1 \subset \ldots \subset T_J = T_h \tag{21}$$

and corresponding finite element spaces

$$V_0 \subset V_1 \subset \ldots \subset V_J = V_h \quad . \tag{22}$$

T_0 is an intentionally coarse mesh and T_l, $l > 0$ is obtained by regular subdivision of each element of T_{l-1}. See Fig. 1 for the refinement rules. Local grid refinement is possible with our code but is not considered in this paper.

The mesh hierarchy induces a Jacobian system

$$K_l u_l = f_l, \quad l = 0, \ldots, J, \tag{23}$$

on each grid level l. K_J is obtained by (14), the construction of the coarse grid matrices K_l, $l < J$, is discussed below.

Furthermore, we need the linear mappings

$$\begin{array}{lcll} R_l & : & \mathbb{R}^{2n_l} \to \mathbb{R}^{2n_{l-1}} & \text{(Restriction)}, \\ P_l & : & \mathbb{R}^{2n_{l-1}} \to \mathbb{R}^{2n_l} & \text{(Prolongation)}, \end{array} \tag{24}$$

where n_l denotes the number of nodes in mesh T_l.

The multigrid iteration for the iterative improvement of a given vector u_l then reads as follows:

mgc (l, u_l, f_l)
{
  if ($l == 0$) $u_0 = K_0^{-1} f_0$;
  else {
    Apply ν_1 *smoothing iterations to* $K_l u_l = f_l$;
    $d_{l-1} = R_l(f_l - K_l u_l)$;
    $v_{l-1} = 0$;
    for $(g = 1, \ldots, \gamma)$ mgc $(l-1, v_{l-1}, d_{l-1})$;
    $u_l = u_l + P_l v_{l-1}$;
    Apply ν_2 *smoothing iterations to* $K_l u_l = f_l$;
  }
}

The parameter γ determines the cycle form. Typical values are $\gamma = 1$ (V-cycle) and $\gamma = 2$ (W-cycle).

4.2 Smoothing Iteration

In order to decide on the smoothing iteration we need some knowledge about the structure of the Jacobian system. For that purpose the Jacobian is viewed as a discretization of the linear equations (17) and (18) as described above.

The components of the vectors u and f are assumed to be in *equation–wise ordering*, i. e. all pressure corrections are numbered before all saturation corrections as in (12). This induces a 2 by 2 block structure on the Jacobian matrix K (we omit the level index l):

$$K = \begin{pmatrix} K^{ww} & K^{wn} \\ K^{nw} & K^{nn} \end{pmatrix} \tag{25}$$

where each block is of dimension $n_h \times n_h$. A comparison with Eqs. (17,18) shows that K_{ww} comes from the discretization of an elliptic term, and K_{nn} comes from the discretization of a parabolic/hyperbolic term depending on the magnitude of the derivative of the capillary pressure–saturation function. This corresponds directly to the characterization of the two–phase flow equations from the fractional flow formulation.

The following observation shows that simple point–wise smoothing is not applicable to the fully–coupled system. Assume a (p_w, S_n) formulation, then if $S_w = 0$ at a node we will also have that $\lambda_w(S_w) = 0$ and the whole row of K^{ww} corresponding to this node will have zero entries. If $S_w = 0$ initially in the whole domain we find that $K^{ww} = 0$ and K is indefinite. Fortunately there is a simple remedy for this problem. One can see that the following matrix

$$D = \begin{pmatrix} \mathrm{diag}(K^{ww}) & \mathrm{diag}(K^{wn}) \\ \mathrm{diag}(K^{nw}) & \mathrm{diag}(K^{nn}) \end{pmatrix} \tag{26}$$

is always invertible (except if a Dirichlet boundary condition $S_n = 1$ and a flux (Neumann) boundary condition for the w–phase is prescribed at a node, obviously a contradiction) and that the transformed system $D^{-1}Ku = D^{-1}f$ is therefore amenable to point–wise smoothing.

The matrix D defined above becomes block–diagonal if the unknowns are reordered in the *point–block ordering* where pressure and saturation at each node are numbered consecutively. Equation– and point–wise orderings are connected through a permutation of the indices, i. e.

$$\bar{u} = P^T u \tag{27}$$

with a permutation matrix P. Matrix D in point–block form is given by

$$\bar{D} = P^T D P. \tag{28}$$

The permuted Jacobian system is written as $\bar{K}\bar{u} = \bar{f}$ with $\bar{K} = P^T K P$ and $\bar{f} = P^T f$. $\bar{K}$ has $n_h \times n_h$ blocks of size 2×2:

$$\bar{K} = \begin{pmatrix} \bar{K}_{11} & \dots & \bar{K}_{1n_h} \\ \vdots & & \vdots \\ \bar{K}_{n_h 1} & \dots & \bar{K}_{n_h n_h} \end{pmatrix}. \tag{29}$$

Application of standard iterative schemes like Jacobi, Gauß–Seidel or incomplete decompositions (ILU) to the little 2 by 2 blocks instead of scalars

leads to the so–called *point–block smoothers*. Point–block smoothers provide more coupling than the corresponding equation–wise variant and are therefore preferred. Moreover, point–block ILU is robust for certain cases of anisotropic permeabilities in two space dimensions. As in the scalar case discussed in [10] matrix $\bar{K}$ becomes point–block tridiagonal provided the mesh is structured, grid points are ordered lexicographically and anisotropy is in the x or y direction only. Point–block tridiagonal systems are solved exactly by the point–block ILU iteration. Therefore the smoother becomes an exact solver in the limit case (large anisotropy). For the scalar case this is discussed in detail in [22].

For these reasons, and because they are supported efficiently by our sparse matrix data structure, we will use point–block ILU or Gauß–Seidel smoothers in the examples below. As noted above, point–wise iterative schemes may only be applied to the transformed system $D^{-1}K$, e. g. in the context of a left–transforming iteration [11, 8.1].

4.3 Coarse Grid Correction

This subsection is concerned with the construction of the grid transfer operators P_l and R_l as well as the coarse grid matrices K_l, $l < J$. Basically, there are two options for doing this. For given P_l and R_l, one can compute K_{l-1} recursively via the Galerkin approach $K_{l-1} = R_l K_l P_l$. Alternatively, the coarse grid matrices can be set up using (14) on each grid level. We will use the latter approach in this paper. In order to motivate this decision, we will first discuss some properties of the Galerkin approach.

In the Galerkin approach, we need to specify only P_l and R_l. The canonical choice for P_l on the grid hierarchy given by (21) is the standard finite element interpolation (note that $V_{l-1} \subset V_l$). For finite element or finite volume discretizations, we then can set $R_l = P_l^T$. Such an approach is very well suited for a scalar diffusion equation with a smooth permeability coefficient. If the diffusion coefficient varies strongly or dominating convection is present, P_l has to be chosen matrix–dependent, see [10, 10.3] or [1]. Very good results for various scalar problems are reported in [20]. Typically, this approach is used with structured meshes; it is not clear how to define a matrix–dependent P_l on an unstructured mesh such that stability (e. g. M–matrix property) is preserved in the case of dominating convection and upwind discretization. Application of the Galerkin approach to the fully

coupled system is also not straightforward. Consider the matrix K given in (25). Presumably prolongation matrix P_l for the system would have the following form:

$$P_l = \begin{pmatrix} P_l^w & 0 \\ 0 & P_l^n \end{pmatrix}, \tag{30}$$

i. e. pressures are only interpolated from pressures and saturations from saturations. In the case of matrix–dependent prolongations P_l^w would be chosen according to K_l^{ww} and P_l^n according to K_l^{nn}. Note that in our case here K_l^{ww} is the discretization of a second order elliptic term and K_l^{nn} that of a possibly hyperbolic term. Assuming $R_l = P_l^T$, the resulting Galerkin coarse grid matrix then reads:

$$K_{l-1} = \begin{pmatrix} R_l^w K_l^{ww} P_l^w & R_l^w K_l^{wn} P_l^n \\ R_l^n K_l^{nw} P_l^w & R_l^n K_l^{nn} P_l^n \end{pmatrix}. \tag{31}$$

The wn and nw blocks in K_{l-1} are multiplied with different prolongations and restrictions from left and right. It is not clear wether the recursive application of this process leads to reasonable coarse grid matrices. Some tests of this procedure indicated that it is not the case.

The stability problem for the coarse grid operator in the case of fully–coupled systems can be circumvented by an aggregation multigrid approach [17, 6, 18, 15], i. e. piecewise constant prolongation. Very good results are reported for a fully–coupled solution of the Navier–Stokes equations in [17]. Multigrid theory [10, Note 6.3.37] predicts that piecewise constant prolongation and restriction is not optimal for second order problems. Braess, [6], therefore uses an overrelaxation of the coarse grid correction which will fail, however, if dominating convection is present in parts of the domain. Molenaar suggests in [15] to treat the convective and diffusive parts separately. This is impossible, however, when the Jacobian is computed with numerical differentiation. In addition, it introduces an unwanted dependence of the multigrid solver on the discretization scheme.

For these reasons, we compute the Jacobian on each grid level and use it as coarse grid matrix. The current solution $x^{k+1,m}$ needed on the coarse grid for this computation is defined by injection, i. e. for all coarse grid nodes we take the value in the corresponding fine grid node.

The use of the discretized operator on the coarse grid together with canonical prolongation and restriction is known to fail for problems with

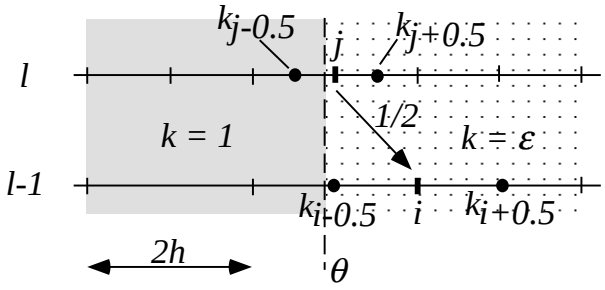

Figure 2: One–dimensional model situation for jumping coefficients.

jumps in the diffusion coefficient that are not aligned with coarse grid element boundaries. To understand this (and remedy the situation), we consider a simple one–dimensional model problem:

$$-\frac{d}{dx}\left(k(x)\frac{du}{dx}\right) = q \quad \text{in } (0,1), \tag{32}$$

$$k(x) = \begin{cases} 1 & x < \theta \\ \varepsilon \ll 1 & \text{else,} \end{cases} \tag{33}$$

and boundary conditions $u(0) = 0, u(1) = 1$. A finite volume discretization of (32) yields

$$-\frac{k_{i-\frac{1}{2}}}{h}u_{i-1} + \left(\frac{k_{i-\frac{1}{2}} + k_{i+\frac{1}{2}}}{h}\right)u_i - \frac{k_{i+\frac{1}{2}}}{h}u_{i+1} = hq_i \tag{34}$$

at grid point i using a mesh width h. $k_{i-\frac{1}{2}}$ and $k_{i+\frac{1}{2}}$ denote point–wise evaluation of (33) half way between the grid points.

Let us now consider a two–grid situation as shown in Fig. 2. The value θ happens to be such that $k_{j-\frac{1}{2}} = 1$, $k_{j+\frac{1}{2}} = \varepsilon$ for fine grid node j and $k_{i-\frac{1}{2}} = k_{i+\frac{1}{2}} = \varepsilon$ for coarse grid node i. The defect computed at fine grid node j is of size $O(1)$ and is restricted to the right hand side of equation j on the coarse grid. All coefficients in row j on the coarse grid are of order ε, therefore a correction of size $O(1/\varepsilon)$ is computed which leads to divergence of the multigrid method if ε is small enough. The problem would disappear if the equations on all grid levels would be scaled to unit diagonal elements. However, scaling the equations in multigrid must be done with care since the discretization also involves powers of h depending on the order of the operator and the space dimension. In the following, we show how to use this observation in the construction of the coarse grid correction.

Application of 34 on all grid levels leads to the linear equations $K_l u_l = f_l$, $l = 0, \ldots, J$. The two–grid correction is given by

$$u_l^{new} = u_l^{old} + P_l K_{l-1}^{-1} R_l \left(f_l - K_l u_l^{old} \right) \quad . \tag{35}$$

A left transormation with D_l^{-1}, $D_l = \mathrm{diag}(K_l)$ yields the transformed equations $\tilde{K}_l u_l = \tilde{f}_l$ with $\tilde{K}_l = D_l^{-1} K_l$ and $\tilde{f}_l = D_l^{-1} f_l$. An *equivalent* form of the coarse grid correction (35) using the transformed systems would read

$$u_l^{new} = u_l^{old} + P_l \tilde{K}_{l-1}^{-1} D_{l-1}^{-1} R_l D_l \left(\tilde{f}_l - \tilde{K}_l u_l^{old} \right) \quad . \tag{36}$$

The ε–dependence has now been moved to the "new" restriction operator $\tilde{R}_l = D_{l-1}^{-1} R_l D_l$ as the situation of Fig. 2 leads to $\tilde{R}_{l,ij} = (1+\varepsilon)/2\varepsilon = O(1/\varepsilon)$. The idea is now to replace $\tilde{R}_l$ by the truncated version r_l:

$$r_{l,ij} = R_{l,ij} \cdot \min \left(cut, \frac{K_{l,jj}}{K_{l-1,ii}} \right) \quad . \tag{37}$$

Note that, for constant k, $r_{l,ij}$ contains the correct scaling when the order of the differential operator is greater or equal than the space dimension. A typical value for *cut* is 2.0 since $K_{l,jj}/K_{l-1,ii} = 2$ near Neumann boundaries in the case of a constant diffusion coefficient.

The definition of the truncated restriction can be extended to the two–phase system by considering the point–block ordering (29). The equations on all grid levels are scaled by the block diagonal matrix $\bar{D}$ from 28. The 2×2 block structure naturally carries over to the restriction matrices giving the block–truncated version $\bar{r}_{l,ij}$,

$$\begin{aligned} &(\bar{r}_{l,ij})_{\alpha\beta} = \\ &\quad (\bar{R}_{l,ij})_{\alpha\beta} \cdot \max \left(0, \min \left(cut, \left(\bar{K}_{l-1,ii}^{-1} \bar{K}_{l,jj} \right)_{\alpha\beta} \right) \right) \end{aligned} \tag{38}$$

where we have $1 \le i \le n_{l-1}$, $1 \le j \le n_l$ and $\alpha, \beta \in \{1, 2\}$.

In Table 1 we compare three different multigrid methods for a two–dimensional variant of (32). The diffusion coefficient is chosen as depicted in Fig. 3. TRUNC denotes the multigrid method with discretized coarse grid operator and truncated restriction given by (37), MDGAL denotes a multigrid method with matrix–dependen prolongation similar to that in [10, 10.3.10a-d] and SC denotes the Schur–complement multigrid method described in [19]. All experiments used an ILU smoother, $\nu_1 = \nu_2 = 1$ and $\gamma = 1$. Table 1 shows the number of iterations needed for a 10^{-8} reduction

Table 1: Iteration numbers for a two–dimensional interface problem. First table is for quadrilateral elements, second table is for triangular elements.

mesh type	h	TRUNC	MDGAL	SC
quadri-	1/8	6	6	5
laterals	1/16	7	7	6
	1/32	7	7	6
	1/64	7	7	5
triangles	1/8	9	11	8
	1/16	11	12	12
	1/32	13	14	15
	1/64	14	15	17

of the residual norm for the different methods on a quadrilateral mesh and a triangular mesh. Both meshes had a coarse mesh width of $h_0 = 1/2$ and the initial guess was always zero (no nested iteration). The table shows that the methods behave very similar. The cost per iteration is approximately equal for TRUNC and MDGAL and one iteration of SC is about 25% more expensive than the other two methods.

5 Parallel Implementation

Despite the good convergence properties of the multigrid method, computer time requirements for three–dimensional nonlinear time–dependent problems are still enormous. The UG–toolbox, [5], on which our two–phase simulator is based, offers a portable parallel computing environment. Grid refinement, discretization and solution process have been parallelized. The multigrid solver itself is readily parallelizable. See [14] for a review and [4] for a description of parallel adaptive multigrid methods on unstructured meshes.

5.1 Data Decomposition

The parallel multigrid algorithm exploits the inherent data parallelism of the method by mapping the unstructured mesh data structure to the set of processors P. In our approach, the individual elements $t \in T_l$ of all mesh levels l are assigned uniquely the processors resulting in a minimal overlap

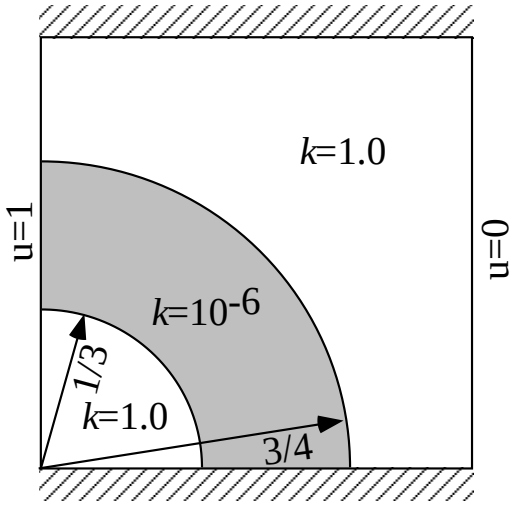

Figure 3: Setup for a two–dimensional interface problem.

as shown in Fig. 4.

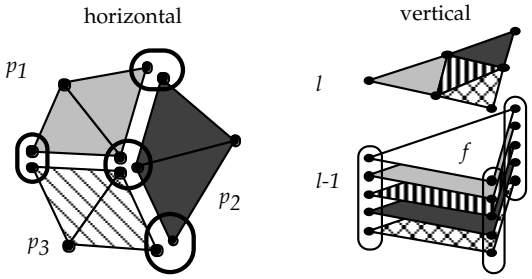

Figure 4: Data decomposition.

The mesh nodes are duplicated at processor boundaries resulting in a horizontal overlap (Fig. 4 left). In a similar way, a level–$l-1$–element f is duplicated whenever a child element is not mapped to the same processor as f. This is called vertical overlap and is shown in Fig. 4 right. Besides mapping an equal number of elements to each processor, the load balancing scheme tries to create a small horizontal and vertical overlap; for details refer to [4, 3].

The decomposition of the mesh data structure implies a decomposition of vectors and matrices used in the numerical algorithm. Let P be the set of processors and T_l^p the set of level–l–elements that have been mapped to processor $p \in P$. Further, let I_l be the set of node indices on level l

and $I_l^p \subset I_l$ the set of indices that are mapped to processor p. Note that $I_l^p \cap I_l^q \neq \emptyset$ is possible for $p \neq q$ due to horizontal overlap. The set

$$P_l(i) = \{p \in P | i \in I_l^p\} \tag{39}$$

gives all processors storing the i-th node on level l. Since the degrees of freedom are associated with the nodes of the mesh, the components of a vector x_l are also duplicated at processor boundaries. Let $x_{l,i}^p$ be the copy of the i-th component $x_{l,i}$ stored in processor $p \in P_l(i)$. Then we can define the following three states of a parallel vector:

$$x_{l,i}^p = x_{l,i}^{seq} \quad \forall p \in P_l(i) \qquad \text{(consistent)}$$

$$\sum_{p \in P_l(i)} x_{l,i}^p = x_{l,i}^{seq} \qquad \text{(inconsistent)}$$

$$x_{l,i}^p = \begin{cases} x_{l,i}^{seq} & p = p_l^*(i) \in P_l(i) \\ 0 & \text{else} \end{cases} \qquad \text{(unique)}$$

where $x_{l,i}^{seq}$ denotes the component $x_{l,i}$ computed by a sequential version of the code. In consistent mode, each processor knows the same value as the sequential version. In inconsistent mode, the sum of all copies of a component gives the sequential value and in unique mode the sequential value is stored in exactly one copy $x_{l,i}^{p_l^*(i)}$. The function $p_l^*(i) : I_l \to P$ associates each index with a unique processor. Note that a unique vector is also inconsistent. The transformation of an inconsistent vector to a consistent or unique one requires local communication. Lists of interface nodes are maintained for efficient implementation of this communication.

5.2 Assembling the Jacobian System

In finite element and finite volume discretizations, the stiffness matrix is assembled element by element. Therefore, processor p will be able to generate all local stiffness matrices for elements $t \in T_l^p$ without any communication, provided that the current solution in the Newton iteration is stored consistently. Accumulation of local stiffness matrices in each processor produces an inconsistent global stiffness matrix since we have for all i, j:

$$\sum_{p \in P_l(i) \cap P_l(j)} K_{l,ij}^p = K_{l,ij}^{seq} \quad . \tag{40}$$

The same arguments apply to the right hand side f_l which is also assembled inconsistently without communication. If load balance is perfect, we can expect a perfect speedup for the assembly process.

5.3 Parallel Iterative Solver

In this subsection we provide the parallel algorithm for an iterative solver using a single grid scheme as basic iteration (we omit the level subscript for clarity):

psolve (K, u, f, ε_{lin})
{
given u *consistent*, K, f *inconsistent*;

(UM) $$Q_{ij}^p = \begin{cases} \sum\limits_{q \in P(i) \cap P(j)} K_{ij}^q & p = p^*(i) = p^*(j) \\ 1 & i = j \wedge p \neq p^*(i) \\ 0 & \text{else} \end{cases} ;$$

for $(m = 0, 1, \ldots)$ {

(D) $$d_i^p = f_i^p - \sum_{j \in I^p} K_{ij}^p u_j^p,\ p \in P, i \in I^p;$$

(UD) $$d_i^p = \begin{cases} \sum\limits_{q \in P(i)} d_i^q & p = p^*(i) \\ 0 & \text{else} \end{cases} \quad p \in P, i \in I^p;$$

(N) $$r_m = \sqrt{\left(\sum_{p \in P} \sum_{i \in I^p} (d_i^p)^2 \right)};$$

(E) **if** $(r_m < r_0 \varepsilon_{lin})$ **break** ;

(BJ) $v^p = (M^p)^{-1} d^p$, $Q^p = M^p - N^p$, $\forall p \in P$;

(C) $$v_i^p = \sum_{q \in P(i)} v_i^q,\ p \in P, i \in I^p;$$

(U) $$u_i^p = u_i^p + \omega v_i^p,\ p \in P, i \in I^p;$$

}
}

The algorithm is called with a consistent initial guess u and matrix K and vector f in inconsistent form. The preparation step (UM) will be explained below. In the iteration loop step (D), computes the defect d in *inconsistent* form without communication. Step (UD) transforms the defect d from inconsistent to unique form which requires an interface communication. This is necessary to compute the euclidean norm of d in (N) correctly.

The norm computation requires a global communication. Step (E) checks the convergence criterion.

In order to compute the new iterate in the following steps, a splitting of the form $K = M - N$ is required. However, not every splitting leads to a parallelizable algorithm. In order to allow independent computation in each processor, we require

$$M_{ij} = 0 \Leftrightarrow p^*(i) \neq p^*(j) \quad . \tag{41}$$

In other words, a processor may only (approximately) invert the matrix Q^p computed in the preparation step (UM) of algorithm psolve. Note that the unique defect d^p (part of d stored in p) is compatible with the consistent Q^p. An exact inversion of Q^p ($M^p = Q^p$ in (BJ)) would lead to a block–Jacobi type iteration with M given by $M_{ij} = K_{ij} \Leftrightarrow p^*(i) = p^*(j)$ and (41). We intend to use the iteration as a smoother in multigrid. Therefore, one step of an ILU decomposition or Gauss–Seidel applied to Q^p is sufficient in step (BJ). Since the correction v is in unique mode, it must be transformed to consistent mode in (C) before it can be added to the current iterate in (U).

It is clear that the convergence behaviour of this method depends on the number of processors and the individual mesh partitioning. In addition to the losses due to load imbalance and communication overhead in the parallel solver, we also have to consider carefully the number of iterations needed in comparison to the sequential version.

The parallel solution algorithm can be extended to multigrid by replacing steps (BJ), (C), and (U) with one multigrid iteration. One smoothing step in multigrid is identical to the body of the **for**–loop in psolve without steps (N) and (E), i.e. it requires two interface communications per smoothing iteration and level. Discussion of the grid transfer operators and the coarse grid solver remain.

It can be shown that the restriction of an inconsistent defect and the prolongation of a consistent correction can be computed without communication when only the coarsest mesh T_0 is partitioned to the processors and refined elements are mapped to the same processor as their father element. However, such a mapping is only efficient if $|T_0| \gg |P|$ (Note that, on the other hand, $|T_0|$ should not be too large in multigrid). If the number of elements in T_0 is comparable to the number of processors or less, the coarsest

grid should be mapped to a subset of the processors P. If the number of processors is very large, it may be necessary to map $T_0, \ldots, T_{b-1}$ for some $b > 0$ to fewer than all processors. In that case, grid transfers up to level b require local communication. It should be noted that, in our approach, the size of the coarsest grid T_0 is not related to the number of processors in any way as in some domain decomposition approaches.

6 Numerical Results

In this section we present some numerical results using the methods discussed in this paper. We are especially interested in the robustness of the multigrid procedure, i. e. the number of iterations should not depend on the mesh size.

Sequential Results have been obtained on an Apple Power Macintosh G3 Computer (266 MHz) with the CodeWarrior IDE 2. 1 and full optimization. Parallel Computations have been carried out on the 512 Processor Cray T3E system of HLRS Stuttgart using Cray Programming Environment Version 3. 0 and -O2 optimization level.

Unless otherwise noted, the multigrid parameters were: Two pre– and post–smoothing steps ($\nu_1 = \nu_2 = 2$) with point–block–ILU and V–cycle ($\gamma = 1$). The multigrid cycle is always used as a preconditioner in BiCGSTAB (see [9]). The nonlinear reduction required in the Newton procedure within each time step was set to $\varepsilon_{nl} = 10^{-5}$ and the linear reduction per Newton step was computed by (15) using $\varepsilon_{min} = 10^{-4}$. At most $q = 6$ line search steps were allowed in the Newton algorithm. No time step reduction was necessary in all the examples presented here. BDF(1) (i. e. implicit Euler) was used as a time–stepping scheme in all examples.

A nested iteration procedure is used in the very first time step to obtain better starting values on the finer grids. For the second and later time steps, the value of the preceding time step is used as initial guess on the finest level.

Table 2 explains the column labels used in the presentation of the numerical results.

6.1 Five Spot

This example tests the multigrid method in the case of pure displacement, i.e. no capillary pressure. In this case saturation is governed by a hyperbolic

Table 2: Notation used in the result presentation.

P	Number of processors used in parallel computation
STEPS	Number of time–steps computed
MESH	Indicates the number of elements in the finest mesh
EXECT	Reports total computation time in seconds
NLIT	Number of Newton iterations for the total computation
AVG	Average number of multigrid iterations per Newton step
MAX	Maximum number of multigrid iterations in any Newton step
TI	Time per multigrid iteration

equation. The problem is treated for different discretisation techniques in [12].

The setup for the five–spot problem is given by Fig. 5. The reservoir is initially filled completely by the non–wetting phase fluid and is displaced by the wetting phase fluid from the lower left corner. At the inflow boundary Γ_{IN} a flux of $Q_w = 0.0032[\mathrm{kg}/(\mathrm{sm}^2)]$ is prescribed while the flux of the non–wetting phase is zero here. At the outflow boundary Γ_{OUT} the pressure p_n is fixed to 10^5[Pa] and the wetting phase saturation is assumed to be zero: $S_w = 0$. On all other boundaries, zero flux conditions are supplied. The initial conditions are $S_n = 1$ and $p_w = 10^5$[Pa] throughout the reservoir.

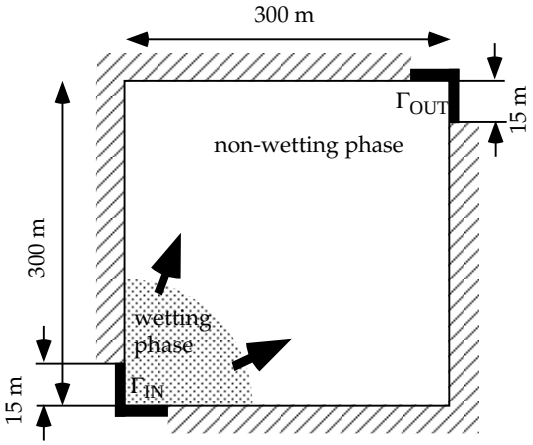

Figure 5: Domain for the five–spot problem.

The porosity was set to $\Phi = 0.2$ and the fluid parameters were $\varrho_w = \varrho_n = 1000[\mathrm{kg/m}^3]$ and $\mu_w = 0.001$[s Pa], $\mu_n = 20\mu_w$. For the absolute

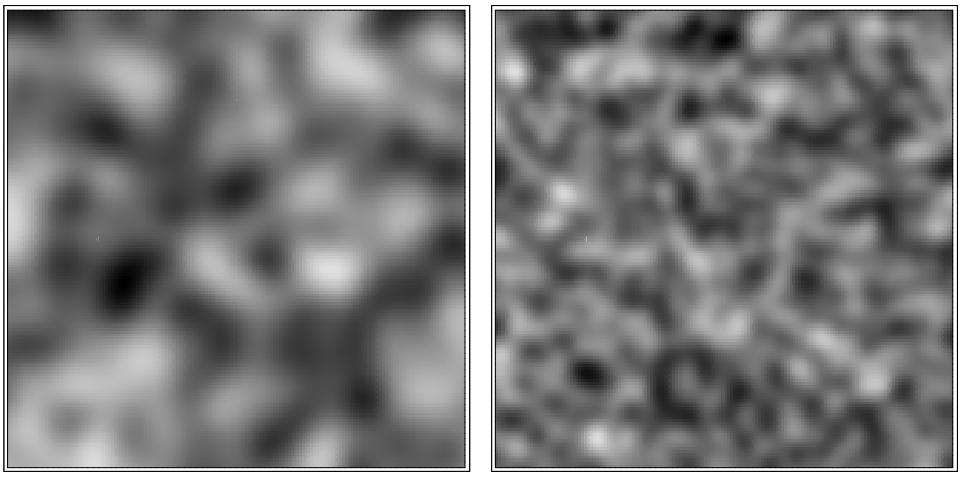

Figure 6: Permeability fields for the five spot problem. Case (B) with correlation lenth 16 shown left and case (C) with 8 cells right.

permeability, we consider three different cases: In case (A), we use a homogeneous field with $k = 10^{-10}[\mathrm{m}^2]$. Cases (B) and (C) use a geostatistically generated permeability field with 160^2 cells, a mean value of $k = 10^{-10}[\mathrm{m}^2]$ and a variation of four orders of magnitude (i. e. permeability is in the range of $10^{-8} \ldots 10^{-12}$). Case (B) uses a correlation length of 16 cells and Case (C) one of 8 cells. The permeability fields are shown in Fig. 6.

In all cases the relative permeability is given by the relations of Brooks and Corey with $\lambda = 2$:

$$k_{rw}(S_w) = S_w^4, \quad k_{rn}(S_n) = S_n^2 \left(1 - (1 - S_n)^2\right) \quad . \tag{42}$$

Fig. 7 shows saturation plots for cases (B) and (C) after 600 and 675 days of simulated time. Table 3 gives the results for the five spot problem variants (A) to (C). Standard parameters were used as described above. The initial guess for the Newton iteration has been obtained recursively by solving the time–step on the coarse mesh and interpolating the result. The coarsest mesh T_0 had 5 by 5 quadrilateral elements in all computations, the finest mesh used 7 levels corresponding to 320 by 320 elements.

For each case the size of the finest mesh is varied while using a *fixed* time step. This has been done in order to test the robustness of the multigrid procedure and the Newton iteration with respect to iteration numbers. The Courant number is between 5 and 6 in the finest computations.

Table 3 shows a *linear* increase in total computation time with growing

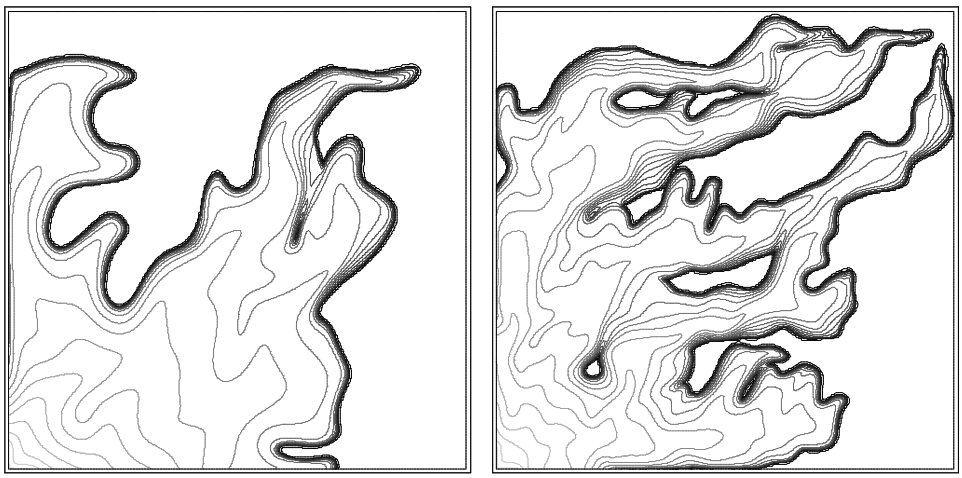

Figure 7: Saturation plot for case (B) after 40 steps of 15 days on the left and for case (C) after 45 steps of 15 days to the right.

Table 3: Results for five spot problem obtained on an Apple Power Macintosh G3 computer. The time step was $\Delta t = 15$[d] in all cases.

Case	STEPS	MESH	EXECT	NLIT	AVG	MAX
(A)	50	80^2	694	151	2.1	4
	50	160^2	2861	151	2.1	4
	50	320^2	12005	151	2.4	5
(B)	40	80^2	948	170	3.4	6
	40	160^2	4070	171	3.4	6
	40	320^2	17866	181	3.5	6
(C)	45	80^2	1393	216	4.2	7
	45	160^2	5661	217	3.9	5
	45	320^2	24109	243	3.5	6

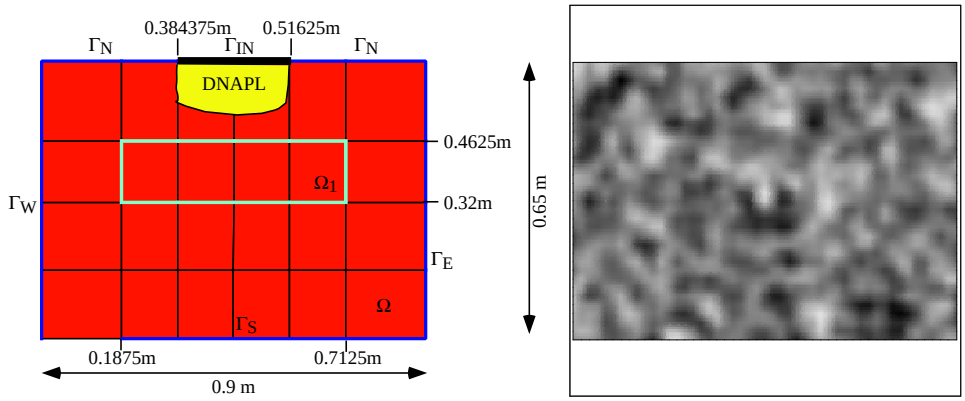

Figure 8: Domain and permeability field for the DNAPL infiltration problem.

mesh size. This is due to the fact that the number of Newton iterations *and* the number of multigrid iterations do not depend on the mesh size. One can also see that the number of Newton and multigrid iterations depends on the form of the permeability field. The computation on the highly varying permeability field from case (C) is twice as expensive as that on the homogeneous field from case (A). For the highly varying permeability fields spatial resolution is more important than temporal resolution and the use of large time steps is reasonable even with a first order scheme. As far as robustness is concerned the time step size can be increased further (say Courant number $15 \ldots 20$) without changing the number of Newton or multigrid iterations.

6.2 2D DNAPL Infiltration

The second test case simulates the vertical infiltration of a DNAPL into a fully water–saturated reservoir. This problem is treated in detail in [12] where experimental results are also reported. The problem setup is given in Fig. 8. The following three variants are investigated:

(A) A low permeable lens is placed into the interior of the reservoir as shown in Fig. 8 on the left. Capillary pressure relations after Brooks and Corey with different entry pressures are assumed in the lens and the surrounding matrix. Entry pressure of the lens is high enough that *no* infiltration of the lens is possible in case (A) (see below for specific values).

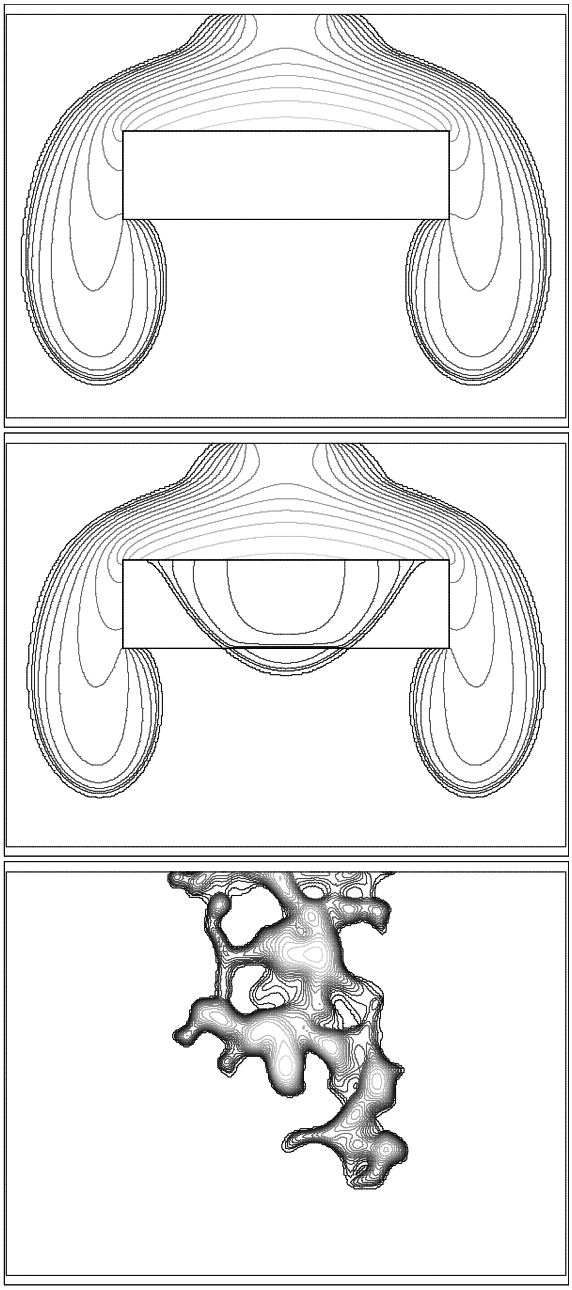

Figure 9: Solution plots for the 2D DNAPL infiltration after 75 steps of $60[s]$ for cases (A) and (B) and 60 steps of $35[s]$ for case (C) (from top). Mesh size was 384 times 256 for all cases.

Table 4: Sand properties for the 2D DNAPL infiltration problem.

Sand	Φ	$k[\mathrm{m}^2]$	S_{wr}	λ	p_d[Pa]
matrix	0.4	$6.64 \cdot 10^{-11}$	0.10	2.7	755
lens	0.39	$3.32 \cdot 10^{-12}$	0.12	2.0	1163.5/1466.1

(B) Same as case (A) but entry pressure is lower. Infiltration of the lens is possible.

(C) Randomly generated porous medium with permeability field shown in Fig. 8 on the right. The permeability has the same mean value as the homogeneous case, is defined on a 192 by 128 mesh and has a correlation length of 8 cells. Permeability varies by two orders of magnitude. The entry pressure in the Brooks–Corey relationship is now different for each fine grid element (see below).

The geometry of the reservoir is given in Fig. 8 (left). The bottom of the reservoir is impermeable for both phases. Hydrostatic conditions for pressure p_w and homogeneous Dirichlet conditions for S_n are prescribed at the left and right boundaries. At the inlet on the top boundary, a flux $Q_n = 0.075[\mathrm{kg}/(\mathrm{sm}^2)]$ for the non–wetting phase is prescribed. Initial conditions were $S_n = 0$ and a hydrostatic pressure distribution. The fluid parameters were $\varrho_w = 1000[\mathrm{kg}/\mathrm{m}^3]$, $\varrho_n = 1460[\mathrm{kg}/\mathrm{m}^3]$, $\mu_w = 0.001$[s Pa] and $\mu_n = 0.0009$[s Pa].

Relative permeabilities and capillary pressure are defined after the model of Brooks and Corey:

$$\begin{aligned} k_{rw}(S_e) &= S_e^{\frac{2+3\lambda}{\lambda}}, \\ k_{rn}(S_e) &= (1-S_e)^2(1-S_e^{\frac{2+\lambda}{\lambda}}), \\ p_c(S_e) &= p_d S_e^{-1/\lambda} \quad . \end{aligned} \tag{43}$$

where the effective saturation is $S_e = (1 - S_n - S_{wr})/(1 - S_{wr})$. The parameters p_d, λ and S_{wr} and other properties are given in Table 4. The value of the entry pressure for the low permeable lens is 1466.1 Pascal for case (A) and 1163.5 Pascal for case (B).

In the random case (C), the capillary pressure curve depends on the local value of absolute permeability:

$$p_c(x, S_e) = p_d \sqrt{\frac{\bar{k}}{k(x)}} S_e^{-1/\lambda} \tag{44}$$

Table 5: Multigrid performance for 2D DNAPL infiltration problem on an Apple Power Macintosh G3 computer.

Case	STEPS	MESH	EXECT	NLIT	AVG	MAX
(A)	75	48×32	398	253	3.0	5
	75	96×64	1840	248	3.7	5
	75	192×128	7601	235	3.9	6
	75	384×256	31369	234	4.0	7
(B)	75	48×32	400	254	2.8	4
	75	96×64	1915	262	3.5	5
	75	192×128	7802	245	3.8	6
	75	384×256	32409	254	3.7	7
(C)	60	48×32	497	364	2.7	4
	60	96×64	2689	381	3.9	7
	60	192×128	11502	336	4.9	8
	60	384×256	53168	320	6.4	12

where $\bar{k} = 6.64 \cdot 10^{-11}$ is the mean value of the permeability field and the parameters p_d and λ are taken from the matrix.

Figure 9 shows solution plots for all three cases on a mesh with 384 by 256 elements. In case (A), it can be seen that no DNAPL infiltrates the fine sand lens, saturation is discontinuous over the lens boundary. In case (C), the flow is restricted to channels of highest permeability and DNAPL pools are formed above zones of low permeability and high entry pressure.

Table 5 shows the solver performance for all three cases. Standard parameters as listed above have been used. The initial guess for Newtons' method has been obtained through nested iteration. The coarse mesh had 6 by 4 elements, the finest mesh hierarchy had 7 levels. Again the time step size has been fixed (60 [s] for cases (A) and (B), 35 [s] for case (C)) while the spatial mesh size has been increased. Total execution time increases linearly for cases (A) and (B) and slightly more than linear in case (C) indicating that both the number of Newton iterations and the number of multigrid cycles per Newton step remain (almost) constant. The propagation speed of the free boundary (seperating the domains where only water and both phases are present) is about 5 mesh cells per time step on the finest meshes in cases (A) and (B).

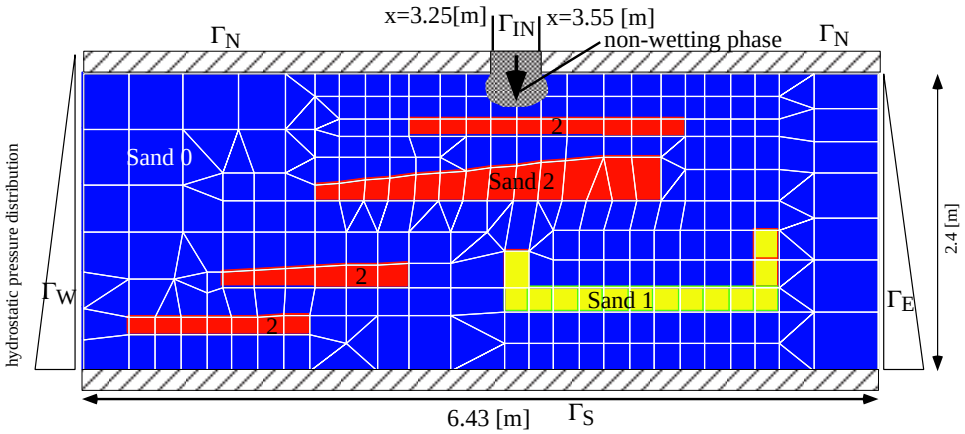

Figure 10: Setup of the two–dimensional VEGAS experiment.

Table 6: Parameters for the different types of sand in the VEGAS infiltration problem.

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

6.3 VEGAS Infiltration Problem

The setup shown in Fig. 10 has been used in an experiment at the VEGAS facility in Stuttgart (cf. [13]). It consists of several fine sand lenses with different inclinations within a coarse sand. It is very similar to the previous example but with a more complicated geometry. We use this example to show the parallel performance of the simulator and the superiority of the multigrid method compared to a single grid scheme.

Table 6 shows the parameters for the different types of sand in the VEGAS problem as indicated in Fig. 10. Brooks–Corey constitutive relations are used for relative permeability and capillary pressure. Boundary conditions at the left and right boundaries of the domain are hydrostatic for p_w and $S_n = 0$. All other boundaries are impermeable except for the inflow of 0.259 [kg/(sm^2)] of DNAPL as indicated in Fig. 10.

Fig. 11 shows the saturation obtained after 2 hours of simulated time. Note that the U–shaped lens (sand 1) to the right has been infiltrated by the DNAPL.

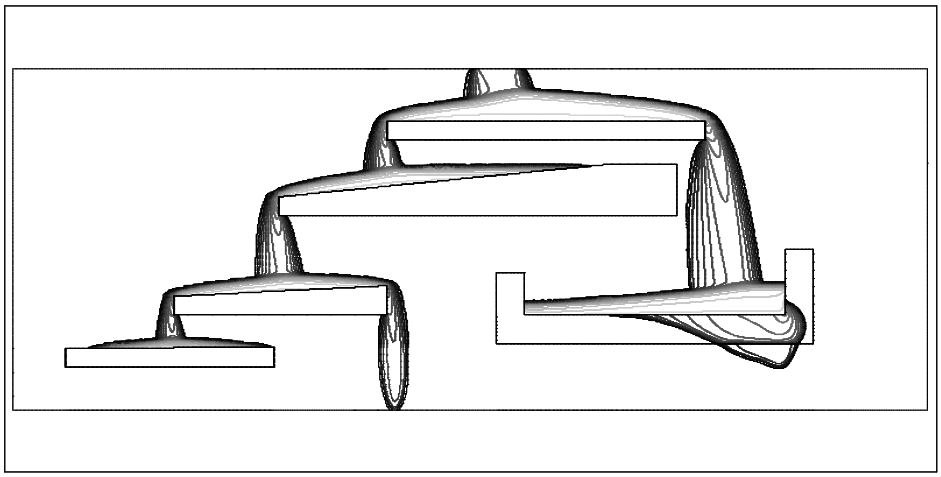

Figure 11: Saturation plot for the VEGAS Experiment after 240 steps of 30 [s].

Table 7: Multigrid solver performance for 2D VEGAS experiment on Cray T3E.

P	STEPS	MESH	EXECT	NLIT	AVG	MAX	TI
1	240	4640	9407	827	5.5	10	0.96
4	240	18560	19280	1206	7.5	13	1.06
16	240	74240	23819	1148	8.4	13	1.15
64	240	296960	29624	1219	9.4	15	1.24
256	240	1187840	35669	1297	10.3	15	1.26

Table 7 gives the solver performance on up to 256 processors of the CRAY T3E. The coarsest mesh was that shown in Fig. 10 with 290 elements. Six levels of uniform refinement resulted in a finest mesh with about 1.2 million elements or 2.4 million unknowns. Multigrid parameters were as listed above execept that a symmetric point–block Gauß–Seidel smoother has been used instead of the point–block ILU smoother. The Gauß–Seidel smoother showed slightly lower iteration numbers in the parallel case. The initial guess for the Newton iteration was obtained through nested iteration. The number of elements *per processor* was fixed at 4640, i. e. we consider a scaled computation where the problem size increases linearily with the number of processors. The time step size was fixed at 30 [s]. Therefore execution time should remain constant if all components of the algorithm scale in the optimal way. This is not the case. Table 7 shows a fourfold increase in execution time from one processor to 256 processors. This increase can be explained by the increase in the the number of Newton iterations NLIT (56 %), multigrid iterations AVG (87%) and the time per multigrid iteration TI (31%). The increase in the number of multigrid iterations is due to the fact that information is lagged by one iteration at processor boundaries in the smoother and not due to the decreasing spatial mesh size. Nevertheless the scalability of the whole algorithm is quite acceptable. The propagation speed of the non–wetting phase infiltration front in this example is more than 6 mesh cells per time step in the finest calculation.

Table 8 compares the multigrid preconditioner with symmetric point–block Gauß–Seidel smoother to a stand–alone symmetric point–block Gauß–Seidel preconditioner. Ordering of the grid points was lexicographic within each processor. The comparison is made after 25 time steps of the calculation (the time limit on the CRAY is 12 hours for a single job). The table shows that the number of multigrid cycles remains nearly constant while the number of Gauß–Seidel preconditioning iterations doubles with each mesh refinement (as one would expect for an elliptic model problem, but this is a fully coupled system here). Similar behavior has been observed for a point–block ILU(0) preconditioner (within each processor). While it may be possible to improve iteration numbers for single–grid preconditioners through various techniques (diagonal scaling, ordering of unknowns, more fill–in in ILU) the dependence on the mesh size essentially remains the same for all these variants. “Modification” of the ILU scheme (see [2]) is able to

Table 8: Comparison of multigrid and single grid preconditioner for 2D VEGAS experiment after 25 time steps on Cray T3E.

Prec.	P	STEPS	MESH	EXECT	NLIT	AVG	MAX
MG–	1	25	4640	887	107	3.3	6
SGS(2,2)	4	25	18560	1151	93	4.3	8
V–cycle	16	25	74240	1483	104	4.4	8
	64	25	296960	1793	105	5.1	9
	256	25	1187840	1955	100	5.6	9
SGS(1)	1	25	4640	3674	107	84	153
	4	25	18560	4516	93	137	249
	16	25	74240	11244	104	317	450
	64	25	296960	21231	106	541	1149
	256	25	1187840	42040	101	1121	2699

Table 9: Sand properties for the 3D DNAPL infiltration problem.

Sand	Φ	$k\ [m^2]$	S_{wr}	S_{nr}	λ	$p_d\ [Pa]$
coarse	0.4	$5.04 \cdot 10^{-10}$	0.08	0.0	3.86	369
fine	0.39	$5.26 \cdot 10^{-11}$	0.10	0.0	2.49	2324

achieve an order reduction (doubling of iteration numbers only with every *two* mesh refinements) but this has only been shown for certain scalar elliptic problems and not for the fully–coupled systems considered here.

The use of the multigrid preconditioner resulted in a 21–fold improvement in total execution time in the example presented above. While it may very well be that multigrid is not necessary for problem sizes that can be treated on a workstation it is a necessity for large scale parallel computations.

6.4 3D DNAPL Infiltration

A three–dimensional DNAPL infiltration problem with two fine sand lenses has been setup. The porosity, permeability and Brooks–Corey parametersare given in Table 9.

The position of the lenses and isosurfaces for DNAPL saturation are shown in Fig. 12. A water flow from right to left (in the top picture) has been prescribed. The solution shows a counter current flow of the DNAPL

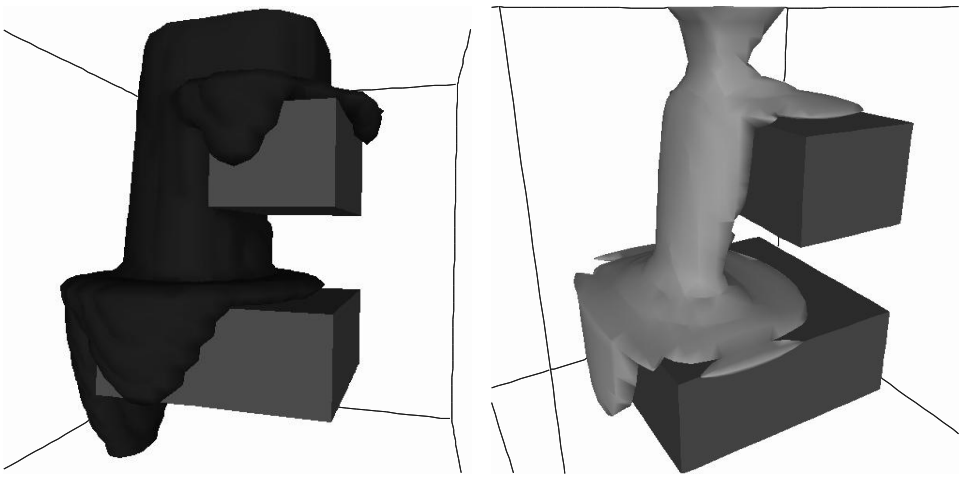

Figure 12: Isosurfaces for $S_n = 0.01$ and $S_n = 0.30$ after 960 [s] in the 3D DNAPL infiltration problem.

and no infiltration of the fine sand lenses.

Table 10 gives results for a scaled parallel computation on the CRAY T3E system. Standard parameters have been employed as descrived above. The size of the coarsest mesh was 4 by 4 by 5 hexahedral elements. Five levels of uniform refinement yield a fine mesh of 128 by 128 by 160 elements corresponding to 5.2 million unknowns. 50 time steps of 20 [s] have been computed. Fig. 12 shows isosurfaces of non–wetting phase saturation for two different values of saturation after 960 [s] of simulated time. The propagation speed of the non–wetting phase front is between five and six mesh cells per time step in the finest calculation. The number of elements per processor was about 10000 for 4, 32 and 256 processors. The calculation with one processor had only 5120 elements, since mesh size grows by a factor of 8 from level to level in 3d. Therefore the execution time for the one processor calculation has to be doubled to be comparable to the time needed by 4, 32 and 256 processors. The scalability in this example is better than in the two–dimensional VEGAS example: From 4 to 256 processors total execution time increases only by 27%.

Conclusions

In this paper we presented a fully–coupled solution method for two–phase flow in porous media. The large non–linear systems were solved per time

Table 10: Performance statistics for 3D DNAPL infiltration with two low permeable lenses on Cray T3E.

P	STEPS	MESH	EXECT	NLIT	AVG	MAX	TI
1	50	5120	4187	218	1.6	2	2.10
4	50	40960	11589	243	2.5	4	4.69
32	50	327680	13214	264	3.5	7	4.76
256	50	2621440	14719	255	4.3	9	4.82

step by a Newton–Multigrid algorithm. The different terms in the Jacobian and their influence on the multigrid method were discussed in detail. A truncated restriction operator enabled us to use the discretized operator on the coarse meshes. In order to achieve a further reduction in computation time a data parallel implementation has been developed.

Experimental results have been presented for various two–phase flow situations including vanishing capillary pressure and heterogeneous media with entry pressure effects as well as two– and three–dimensional examples. Very satisfactory multigrid performance was observed in all examples and a comparison with an ILU preconditioner showed the superiority of multigrid. In two–dimensional computations (not reported here) with 2.4 million unknowns on 112 processors an improvement of a factor eleven in wall clock time for a complete calculation could be observed in comparison to a BiCGSTAB-ILU solver.

Since a convergence proof of multigrid is not available for the problems discussed in this paper, more practical experience is necessary. The code is also able to handle compressible fluids and fully unstructured meshes although neither aspect has been tested extensively up to now.

The fully–coupled solution method is also the basis for multiphase– multicomponent simulators. The development of effective multigrid preconditioners for these models will be a challenge for the future.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft in Sonderforschungsbereich 404, project A3. We thank all members of the UG group at ICA 3, Stuttgart, especially Klaus Birken, Stefan Lang and Chris-

tian Wieners that we could use their work on the parallel version and Klaus Johannsen for his implementation of BiCGSTAB.

References

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[2] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems*. Academic Press, 1984.

[3] P. Bastian. Load balancing for adaptive multigrid methods. *SIAM J. Sci. Stat. Comput.*, page to appear.

[4] P. Bastian. *Parallele adaptive Mehrgitterverfahren*. Teubner–Verlag, 1996.

[5] P. Bastian, K. Birken, S. Lang, K. Johannsen, N. Neuß, H. Rentz-Reichert, and C. Wieners. Ug: A flexible software toolbox for solving partial differential equations. *Computing and Visualization in Science*, 1, 1997.

[6] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55:379–393, 1995.

[7] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*. Springer, 1994.

[8] G. Chavent and J. Jaffré. *Mathematical Models and Finite Elements for Reservoir Simulation*. North–Holland, 1978.

[9] H. A. Van der Vorst. 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, 1992.

[10] W. Hackbusch. *Multi–Grid Methods and Applications*. Springer–Verlag, 1985.

[11] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Linear Equations*. Springer, 1994.

[12] R. Helmig. *Multiphase Flow and Transport Processes in the Subsurface — A Contribution to the Modeling of Hydrosystems.* Springer Verlag, 1997.

[13] H. Kobus. The role of large–scale experiments in groundwater and subsurface remediation research: The vegas concept and approach. In *Groundwater and Subsurface Remediation.* Springer–Verlag, 1996.

[14] O. A. McBryan, P. O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C. A. Thole, and U. Trottenberg. Multigrid methods on parallel computers — a survey of recent developments. *Impact of Computing in Science and Engineering*, 3:1–75, 1991.

[15] J. Molenar. Multigrid methods for fully implicit oil reservoir simulation. Technical report, TWI, Delft University of Technology, 1995.

[16] W. A. Mulder and R. H. J. Gmelig Meyling. Numerical simulation of two–phase flow using locally refined grids in three space dimensions. *SPE Advanced Technology Series*, 1(1):36–41, 1993.

[17] M. Raw. Robustness of coupled algebraic multigrid for the Navier–stokes equations. Technical Report 96–0297, AIAA, 1996.

[18] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multi–grid by smoothed aggregation for second and forth order elliptic problems. *Computing*, to appear.

[19] C. Wagner and G. Wittum. A robust multigrid method for groundwater flow. *Numer. Math.*, to appear, 1997.

[20] T. Washio and C. W. Oosterlee. Experiences with robust parallel multilevel preconditioners for BiCGSTAB. Technical Report 949, Gesellschaft für Mathematik und Datenverarbeitung, 1995.

[21] P. Wesseling. *An Introduction to Multigrid Methods.* John Wiley, 1992.

[22] G. Wittum. On the robustness of ILU smoothing. *SIAM J. Sci. Stat. Comput.*, 10:699–717, 1989.