Ecient fully-coupled solution techniques for two-phase flow in porous media
Parallel multigrid solution and large scale computations

Peter Bastian a,1, Rainer Helmig b,*

a Institut f"ur Computeranwendungen III, Universit"at Stuttgart, Pfaffenwaldring 27, 70569 Stuttgart, Germany
b Institut f"ur ComputerAnwendungen im Bauingenieurwesen, Technische Universit"at Braunschweig, Pockelsstr. 3, D-38106 Braunschweig, Germany

Received 31 March 1998; received in revised form 1 February 1999; accepted 31 March 1999

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 by using a newly truncated restriction operator and an outer Krylov-space method. We show an optimal order of convergence for a wide range of two-phase flow problems including heterogeneous media and vanishing capillary pressure in an experimental way. Furthermore, we present a data parallel implementation of the algorithm with speedup results. © 1999 Elsevier Science Ltd. All rights reserved.

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, [11,23], 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, [9], 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. [18].

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, [17], 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 the discretization of which 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

---

* Corresponding author. Tel.: +49-5313917588; fax: +49-5313917599; e-mail: r.helmig@tu-bs.de
1 E-mail: peter@ica3.uni-stuttgart.de

0309-1708/99/$ - see front matter © 1999 Elsevier Science Ltd. All rights reserved.
PII: S 0 3 0 9 - 1 7 0 8 ( 9 9 ) 0 0 0 1 4 - 7
boundaries. Moreover, in the case of vanishing capillary pressure, it contains a dominant convective term. All these effects become 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, the construction of the coarse grid correction has to be carried out carefully. The two options are either a Galerkin approach or the usage of 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, [17]. See also [19,6] 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 newly 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, 3D calculation still requires a considerable amount of computer time. Therefore, we consider the parallelization of the multigrid algorithm via data partitioning as described in [3,8]. The respective algorithms and constitutive relationships are incorporated in the numerical simulator MUFTE-UG [14] which is based on the software toolbox UG [5].

The rest of this paper is organized as follows. In Section 2 we shortly review the numerical model and the discretization scheme. Then we describe the damped, inexact Newton-method that is used to solve the non-linear systems in Section 3. Section 4 then describes the multigrid method and its components. After having described the parallelization strategy in Section 5, the numerical results are presented in Section 6.

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 (\( x = w, n \))

\[
\frac{\partial (\Phi \rho_x S_x)}{\partial t} + \nabla \cdot (\rho_x v_x) = \rho_q q_x \quad \text{in} \ \Omega \times I
\]

and the generalized Darcy Law

\[
v_x = -\frac{k_x(x,S_x)}{\mu_x} K(\nabla p_x - q_x g),
\]

where \( \Phi \) is the porosity of the porous medium, \( \rho \) the density of phase \( x \), \( S_x \) the unknown saturation of phase \( x \), \( v_x \) the volumetric flux vector, \( q_x \) the source/sink term, \( K \) the absolute permeability tensor, \( k_x(x,S_x) \) the relative permeability, \( \mu_x \) the dynamic viscosity of the fluid \( x \), \( p_x \) the unknown pressure of phase \( x \) and \( g \) the vector of gravitational forces. The model (and its implementation) also handles full compressibility of both fluid phases [13]. In addition to these differential equations, we have the algebraic relations:

\[
S_w(x,t) + S_n(x,t) = 1,
\]

\[
p_w(x,t) - p_n(x,t) = p_c(x,S_w(x,t)).
\]

Inserting Eq. (2) into Eq. (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 \( S_w \) as unknowns:

\[
L_w(p_w, S_w) = \frac{\partial (\Phi \rho_w (1 - S_w))}{\partial t} + \nabla \cdot \left\{ -\rho_w k_w(1 - S_w) \frac{K(\nabla p_w - \rho_w g)}{\mu_w} \right\} - \rho_w q_w
\]

\[
= 0 L_w(p_n, S_n)
\]

\[
\frac{\partial (\Phi \rho_n S_n)}{\partial t} + \nabla \cdot \left\{ - \frac{\rho_n k_n(S_n)}{\mu_n} K(\nabla p_n) \right. \right.

\left. \left. + \nabla p_c(1 - S_n) - \rho_n g \right\} - \rho_n q_n
\]

\[= 0. \] (5)

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

\[p_w = g_{wd} \quad \text{on} \ \Gamma_{wd}, \quad \rho_n v_w \cdot n = g_{wn} \quad \text{on} \ \Gamma_{wn}, \]

\[S_n = g_{nd} \quad \text{on} \ \Gamma_{nd}, \quad \rho_n v_n \cdot n = g_{mn} \quad \text{on} \ \Gamma_{mn}, \] (6)

and the initial conditions

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

Vector \( v \) 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 shown in Fig. 1 (\( b_1 \) is the control volume corresponding to node \( v_1 \)). For \( p_{w,h}, S_{h} \in V_h \), an implicit Euler discretization in time, and the abbreviation \( (u,v) = \int_T u v \ dx \), we get the discrete form of the equations:
equations for the coefficients in the following order: number of nodes in mesh density on pressure in Eq. (8), but the code can treat the storage term. Mobilities in the flux terms are evaluated using the midpoint rule. This results in a lumped mass method for both phases as compressible. Instead of the implicit Euler scheme it is also possible to use the Crank–Nicholson or BDF(2) methods (however, this implies a time-step restriction if the saturation equation is convection-dominated). After inserting a basis function representation of the unknown finite element functions ( denotes the number of nodes in mesh ) and evaluating all the integrals, we obtain a system of nonlinear algebraic equations for the coefficients and : The nonlinear function depends on the time levels and as well as . The vector contains all unknowns in the following order:

\[ x^{k+1} = \left( \hat{\mathbf{p}}^{k+1}_{wh,1}, \ldots, \hat{\mathbf{p}}^{k+1}_{wh,n_h}, \hat{S}^{k+1}_{nh,1}, \ldots, \hat{S}^{k+1}_{nh,n_l} \right)^T. \] (12)

3. Nonlinear solution method

3.1. Newton’s method

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

Choose \( x^{k+1,0} \); set \( m = 0 \);

while \( \|F(x^{k+1,m})\|_2 / \|F(x^{k+1,0})\|_2 > \varepsilon_{nl} \) do

\[ \begin{align*}
&\text{Solve } K(x^{k+1,m})u = -F(x^{k+1,m}) \\
&\text{with accuracy } \varepsilon_{lin}; \\
&x^{k+1,m+1} = x^{k+1,m} + \eta u; \\
&m = m + 1;
\end{align*} \]

end

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

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

holds for the smallest possible \( q \in \{0, 1, \ldots, n_h\} \). The number of line search steps \( n_h \) 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

\[ K_{ij}(x^{k+1,m}) = \left. \frac{\partial F_j}{\partial x_i} \right|_{x^{k+1,m}}, \quad 1 \leq i, j \leq 2n_h. \] (14)

As indicated in the algorithm, the Jacobian system does not need to be solved exactly. We chose the linear reduction \( \varepsilon_{lin} \) as

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

This choice allows quadratic convergence of the Newton method in the final steps. The minimal reduction \( \varepsilon_{min} \) 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 already having applied the linearization step in Eq. (8) we can interpret the Jacobian system as the discretization of a linear differential operator. To that end, we write the finite element functions corresponding to the Newton iterates as follows:
\[
p_{i+1,m+1}^{k+1} = p_{i+1,m}^k + \delta p_{i+1,m}, \quad S_{i+1,m+1}^{k+1} = s_{i+1,m}^k + \delta s_{i+1,m}.
\]

Linearising \( k_{nu} \), \( k_{na} \) and \( p_c \) at \( S_{i+1,m}^{k+1} \) and dropping all higher order correction terms yields the following linear equations for the corrections \( \delta p_{i+1,m} \) and \( \delta s_{i+1,m} \):

\[
L'_{nu}(p_{i+1,m}^{k+1}, S_{i+1,m}^{k+1}; \delta p_{i+1,m}, \delta S_{i+1,m}) = -\left( \nabla \cdot \{ q_{i+1,m}^{k+1} K \nabla p_{i+1,m} \} , w_h \right) - \frac{1}{\Delta t} (\Phi q_{i+1,m}^k F_{i+1,m}, w_h)
\]

\[
- \left( \nabla \cdot \{ \tilde{p}_{i+1,m} \delta S_{i+1,m} \}, w_h \right) = - \frac{1}{\Delta t} (\Phi q_{i+1,m}^k (S_{i+1,m}^k - S_{i+1,m}^{k+1}), w_h)
\]

\[
- \left( \nabla \cdot \{ - q_{i+1,m}^{k+1} K (\nabla p_{i+1,m}^k - q_w g) \}, w_h \right) + (\Phi q_{i+1,m}^k, w_h),
\]

\[
L'_{nu}(p_{i+1,m}^{k+1}, S_{i+1,m}^{k+1}; \delta p_{i+1,m}, \delta S_{i+1,m})
\]

\[
= -\left( \nabla \cdot \{ q_{i+1,m}^{k+1} K \nabla p_{i+1,m} \} , w_h \right) + \frac{1}{\Delta t} (\Phi q_{i+1,m}^k \delta S_{i+1,m}, w_h)
\]

\[
+ \left( \nabla \cdot \{ \tilde{p}_{i+1,m} \delta S_{i+1,m} \}, w_h \right) + \left( \nabla \cdot \{ q_{i+1,m}^{k+1} mK \nabla S_{i+1,m} \}, w_h \right)
\]

\[
= - \frac{1}{\Delta t} [(\Phi q_{i+1,m}^k S_{i+1,m}^k - (\Phi q_{i+1,m}^k S_{i+1,m}^{k+1})]
\]

\[
- \left( \nabla \cdot \{ - q_{i+1,m}^{k+1} K (\nabla p_{i+1,m}^k - q_w g) \}, w_h \right) + (\Phi q_{i+1,m}^k, w_h),
\]

\[
\forall w_h \in W_h \text{ and the vectors } \tilde{p}_{i+1,m} \text{ being defined as}
\]

\[
\tilde{p}_{i+1,m} = -q_{i+1,m}^{k+1} K (\nabla p_{i+1,m}^k - q_w g)
\]

and

\[
\tilde{p}_{i+1,m} = -q_{i+1,m}^{k+1} K (\nabla p_{i+1,m}^k - q_w g)
\]

\[
- q_{i+1,m}^{k+1} mK (\nabla p_{i+1,m}^k - q_w g)
\]

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

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). This is also important if the nonlinear equations shall be solved very accurately which in turn is required to reduce material balance errors to neglectable values. (Discrete conservation of mass in the finite volume scheme is only guaranteed if the nonlinear systems per time-step are solved exactly.) Finally, we remark that the linearized operator has been derived on the assumption of incompressibility for both phases. The code, however, takes compressibility into account.

### 4. Multigrid method

#### 4.1. Basic algorithm

For an introduction to multigrid methods, we recommend to read Refs. [11,23,8]. The standard multigrid method uses a sequence of \( J + 1 \) nested meshes with increasing fineness

\[
T_0 \subset T_1 \subset \cdots \subset T_J = T_h
\]

and corresponding finite element spaces

\[
V_0 \subset V_1 \subset \cdots \subset V_J = V_h.
\]

\( T_0 \) is an intentionally coarse mesh and \( T_1, l > 0 \) is obtained by regular subdivision of each element of \( T_{l-1} \). See Fig. 2 for the refinement rules. Local grid refinement is possible with our code but it is not considered in this paper.

The mesh hierarchy induces a Jacobian system

\[
K_l u_l = f_l, \quad l = 0, \ldots, J,
\]

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

Furthermore, we need the linear mappings

\[
R_l : \mathbb{R}^{2n_l} \to \mathbb{R}^{2n_l-1} \quad \text{(restriction)},
\]

\[
P_l : \mathbb{R}^{2n_{l-1}} \to \mathbb{R}^{2n_l} \quad \text{(prolongation)},
\]

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:

\[
\text{mgc} (l, u_l, f_l)
\]

\[
\text{if } (l == 0) u_0 = K_0^{-1} f_0;
\]

\[
\text{else } \{
\]

\[
\text{Apply } v_1 \text{ smoothing iterations to } K_l u_l = f_l;
\]

\[
d_{l-1} = R_l (f_l - K_l u_l);
\]

\[
v_{l-1} = 0;
\]

\[
\text{for } (g = 1, \ldots, \gamma) \text{ mgc } l-1, v_{l-1}, d_{l-1};
\]

\[
u_l = u_l + P_l v_{l-1};
\]

\[
\text{Apply } v_2 \text{ smoothing iterations to } K_l u_l = f_l;
\]

\[
\}\}

Fig. 2. Regular refinement rules.
The parameter \( \gamma \) 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 Eqs. (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 Eq. (12). This induces a \( 2 \times 2 \) block structure on the Jacobian matrix \( K \) (we omit the level index \( l \)):

\[
K = \begin{pmatrix} K_{ww} & K_{wn} \\ K_{wn} & K_{nn} \end{pmatrix},
\]

where each block has the dimension \( n_h \times n_h \). A comparison with Eqs. (17) and (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 pointwise smoothing is not applicable to the fully coupled system. Assume a \( (p_w, S_w) \) formulation, then if \( S_w = 0 \) at a node we will also have that \( \hat{\lambda}_w(S_w) = 0 \) 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. It is obvious that the following matrix

\[
D = \begin{pmatrix} \text{diag}(K_{ww}) & \text{diag}(K_{wn}) \\ \text{diag}(K_{wn}) & \text{diag}(K_{nn}) \end{pmatrix}
\]

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

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

\[
\tilde{u} = P^T u
\]

with a permutation matrix \( P \). Matrix \( D \) in point-block form is given by

\[
\tilde{D} = P^T DP.
\]

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

\[
\tilde{K} = \begin{pmatrix} \tilde{K}_{11} & \cdots & \tilde{K}_{1n_h} \\ \vdots & \ddots & \vdots \\ \tilde{K}_{n_h1} & \cdots & \tilde{K}_{n_hn_h} \end{pmatrix}.
\]

The application of standard iterative schemes like Jacobi, Gauß–Seidel, or incomplete decompositions (ILU) to the little \( 2 \times 2 \) 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(0) (i.e., no additional fill-in) is robust for certain cases of anisotropic permeabilities in two space dimensions. As in the scalar case discussed in [11] matrix \( \tilde{K} \) 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(0) iteration. Therefore, the smoother becomes an exact solver in the limited case (large anisotropy). For the scalar case this is discussed in detail in [24].

For these reasons, which are supported efficiently by our sparse matrix data structure, we will use point-block ILU(0) 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 [12, 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 proceeding in this way. For given \( P_l \) and \( R_l \), one can compute \( K_{i-1} \) recursively via the Galerkin approach \( K_{i-1} = R_l K_l P_l \). Alternatively, the coarse grid matrices can be set up using Eq. (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 Eq. (21) is the standard finite element interpolation (note that \( V_{l-1} \subset V_l \)). For finite element or finite volume discretizations, we can then 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 if dominating convection is present, \( P_l \) has to be chosen matrix-dependent, see [11, 10.3] or [1]. Very good results for various scalar problems are reported in [22]. Typically, this approach is used with structured meshes; it is not
clear how to define a matrix-dependent $P_i$ on an unstructured mesh in such a way that stability (e.g., $M$-matrix property) is preserved in the case of dominating convection and upwind discretization. The application of the Galerkin approach to the fully coupled system is neither straightforward. Consider the matrix $K$ given in Eq. (25). Presumably, prolongation matrix $P_i$ for the system would have the following form:

$$P_i = \begin{pmatrix} P^w_i & 0 \\ 0 & P^o_i \end{pmatrix},$$

i.e., pressures are only interpolated from pressures and saturations from saturations. In the case of matrix-dependent prolongations $P^w_i$ would be chosen according to $K^{wn}$ and $P^o_i$ according to $K^{on}$. Note that in our case $K^{wn}$ is the discretization of a second order elliptic term and $K^{on}$ that of a possibly hyperbolic term. Assuming $R_i = P^T_i$, the resulting Galerkin coarse grid matrix then reads:

$$K_{i-1} = \begin{pmatrix} R^w_i K^{wn} P^w_i & R^w_i K^{wn} P^o_i \\ R^o_i K^{on} P^o_i & R^o_i K^{on} P^w_i \end{pmatrix}.$$

The $wn$ and $nw$ blocks in $K_{i-1}$ are multiplied with different prolongations and restrictions from the left and the right. It is not clear whether 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 [19,6,21,17], i.e., piece-wise constant prolongation. Very good results are reported for a fully coupled solution of the Navier–Stokes equations in [19]. Multigrid theory [11, Note 6.3.37] predicts that piece-wise 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 [17] 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 assemble the Jacobian on each grid level directly and use it as coarse grid matrix. The current solution $u^{k+1,n}$ needed on the coarse grid for this computation is defined by injection. Then coarse grid matrices are reasonable (e.g., the discretization of the convective part in $K^{wn}$ is always an $M$-matrix). However, the use of the discretized operator on the coarse grid together with canonical prolongation and restriction is known to fail for problems with 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 1D model problem:

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

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

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

$$-\frac{k_{i-1/2} u_{i-1} + (k_{i-1/2} + k_{i+1/2})}{h} u_i - \frac{k_{i+1/2}}{h} u_{i+1} = h q_i$$

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

Let us now consider a two-grid situation as shown in Fig. 3. The value $\theta$ happens to be such that $k_{j-1/2} = 1$, $k_{j+1/2} = \varepsilon$ for fine grid node $j$ and $k_{j-1/2} = k_{j+1/2} = \varepsilon$ 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 $\varepsilon$, therefore a correction of size $O(1/\varepsilon)$ is computed there, which leads to a divergence of the multigrid method if $\varepsilon$ is small enough.

The problem would disappear if the equations on all grid levels were scaled to unit diagonal elements. However, scaling the equations in multigrid must be done carefully 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 $K_i u_i = f_i$, $i = 0, \ldots, J$. The two-grid correction is given by

$$u_i^{\text{new}} = u_i^{\text{old}} + P_i K_{i-1}^{-1} f_i - K_i u_i^{\text{old}}.$$  

A left transformation with $D_i^{-1}$, $D_i = \text{diag}(K_i)$ yields the transformed equations $K_i u_i = f_i$ with $K_i = D_i^{-1} K_i$ and $f_i = D_i^{-1} f_i$. An equivalent form of the coarse grid correction (35) using the transformed systems would read

$$\begin{pmatrix} \hat{K}_{-1} & \hat{K}_0 & \hat{K}_1 \\ \hat{K}_0 & \hat{K}_1 & \hat{K}_2 \\ \hat{K}_1 & \hat{K}_2 & \hat{K}_3 \end{pmatrix} \begin{pmatrix} \hat{u}_{-1} \\ \hat{u}_0 \\ \hat{u}_1 \end{pmatrix} = \begin{pmatrix} \hat{f}_{-1} \\ \hat{f}_0 \\ \hat{f}_1 \end{pmatrix}$$

Fig. 3. One-dimensional model situation for jumping coefficients.
The ε-dependence has now been moved to the “new” restriction operator \( \tilde{R}_l = D_l^{-1} R_l D_l \) as the situation of Fig. 3 leads to \( \tilde{R}_{l,j} = (1 + \varepsilon)/\varepsilon = O(1/\varepsilon) \). The idea is now to replace \( \tilde{R}_l \) by the truncated version \( r_l \):

\[
r_{l,j} = R_{l,j} \cdot \min \left( \text{cut}, \frac{K_{l,j}}{K_{l-1,j}} \right),
\]

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

Coefficient jumps that are not aligned with coarse grid element boundaries are just one difficulty in (standard) multigrid. Another is the problem of a large isolated coefficient as depicted in Fig. 4. For small \( \varepsilon \) the interior boundary acts like a homogeneous Neumann boundary condition and a constant error on the subdomain with a large coefficient is not removed effectively by the coarse grid correction. The situation is remedied to a considerable extent, however, by using the multigrid cycle as a preconditioner in a Krylov method such as the conjugate gradient method with respect to discontinuities in the diffusion coefficient. This is illustrated below with several diffusion model problems. All problems \( i = 1, \ldots, 4 \) solve the equation

\[
\nabla \cdot (k(x,y) \nabla u) = 0 \quad \text{in} \ \Omega = (0, 1)^2
\]

with Dirichlet boundary conditions on the east and west boundaries and Neumann conditions on the north and south boundaries. Problem 1 uses the diffusion coefficient \( k_1(x,y) = 1 \) just for comparison. Problem 2 uses

\[
k_2(x,y) = \begin{cases} 
10^{-6} & 1/3 \leq x^2 + y^2 \leq 3/4, \\
1 & \text{else.} 
\end{cases}
\]

Problem 3 is taken (with modifications) from [1] and has an isolated big coefficient:

\[
k_3(x,y) = \begin{cases} 
10^{-6} & 1/3 \leq x,y \leq 2/3, \\
1 & \text{else.} 
\end{cases}
\]

Problem 4 is the hardest one and uses a checkerboard-like distribution of the diffusion coefficient. The distribution is such that the solution has internal singularities and several isolated big coefficients:

\[
k_4(x,y) = \begin{cases} 
10^4 & \text{mod 2 = 0,} \\
10^{-3} & \text{mod 2 = 1,} \\
10^2 & \text{mod 2 = 1,} \\
10^{-1} & \text{mod 2 = 2.} 
\end{cases}
\]

Eq. (39) is discretized with a vertex centered finite volume scheme on a sequence of structured quadrilateral meshes with \( h = 1/2^{i+1} \), i.e., the coarsest mesh has four quadrilateral elements unless otherwise stated. Note that the discontinuities in the diffusion coefficients \( k_2, \ldots, k_4 \) are not aligned with coarse grid element boundaries.

For the solution of the discretized equations we use the multigrid method (\( \gamma = 1 \)) with truncated restriction (TR) as a pre-conditioner for the BiCG-STAB method (since the pre-conditioner is not symmetric the conjugate gradient method cannot be used; it is, however, possible to define a symmetric variant). Two steps of symmetric Gauss-Seidel (lexicographic ordering) are used for pre- and post-smoothing (\( v_1 = v_2 = 2 \)). The initial iterate is set to 0 on all grid levels (no nested iteration). For comparison we also provide iteration numbers for an algebraic multigrid pre-conditioner (AMG) as it has been presented in [6]. This pre-conditioner was used inside a conjugate gradient iteration and used the same smoother.

Table 1 reports the number of multigrid cycles needed for a \( 10^{-10} \) reduction in residual norm for both pre-conditioners and the four problems stated above. Note that a standard multigrid pre-conditioner (discretized coarse grid operator, standard grid transfers) does not converge for any of the problems 2, \ldots, 4. The
pre-conditioner with truncated restriction converges quite well for problems 2 and 3 (for problem 2 it converges with similar rates, also without Krylov acceleration). In problem 4 the coarsest grid size has been set to \( h = 1/16 \). Therefore, iteration numbers are only reported beginning with \( h = 1/32 \). Problem 4 also shows the limit of the pre-conditioned truncated restriction method: it is not robust with respect to a growing number of isolated subdomains with large coefficients. Interestingly, the algebraic multigrid shows a very robust behavior. While subdomains with large coefficients. Interestingly, the algebraic multigrid shows a very robust behavior. While

### 5. Parallel implementation

Despite the good convergence properties of the multigrid method, computer time requirements for 3D 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 [16] for a review and [3] 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 as shown in Fig. 5.

The mesh nodes are duplicated at processor boundaries resulting in a horizontal overlap (Fig. 5, 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. 5, 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 [3,4].

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 \( I^l_i \subset I_l \) the set of indices that are mapped to processor \( p \). Note that \( I^l_i \cap I^l_p \neq \emptyset \) is possible for \( p \neq q \) due to horizontal overlap. The set

\[
P^l_i := \{ p \in P : i \in I^l_i \}
\]

(43) 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 \( x^{\text{seq}}_{ij} \) be the copy of the \( j \)th component \( x_{ij} \) and stored in processor \( p \in P^l_i \).

Then we can define the following three states of a parallel vector:

\[
x^{\text{seq}}_{ij} = x^{\text{seq}}_{ij} \quad \forall p \in P^l_i \quad \text{(consistent)},
\]

\[
\sum_{p \in P^l_i} x^{\text{seq}}_{ij} = x^{\text{seq}}_{ij} \quad \text{(inconsistent)},
\]

\[
x^{\text{seq}}_{ij} = \begin{cases} x^{\text{seq}}_{ij} & p = p^l_i(i) \in P^l_i \in P^l_i(i) \\ 0 & \text{else} \end{cases} \quad \text{(unique),}
\]

where \( x^{\text{seq}}_{ij} \) denotes the component \( x_{ij} \) computed by a sequential version of the code. In the consistent mode, each processor knows the same value as the sequential version. In the 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 \( x^{\text{seq}}_{ij} \). The function \( p^l_i(i) : I_l \rightarrow P \) 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 the efficient implementation of this communication.

### Table 1

Number of pre-conditioner evaluations for several diffusion problems using the multigrid pre-conditioner with truncated restriction or an algebraic multigrid pre-conditioner

<table>
<thead>
<tr>
<th>( h^{-1} )</th>
<th>Problem 1</th>
<th>Problem 2</th>
<th>Problem 3</th>
<th>Problem 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TR AMG</td>
<td>TR AMG</td>
<td>TR AMG</td>
<td>TR AMG</td>
</tr>
<tr>
<td>16</td>
<td>5 8</td>
<td>7 8</td>
<td>12 9</td>
<td>– –</td>
</tr>
<tr>
<td>32</td>
<td>5 11</td>
<td>8 10</td>
<td>14 11</td>
<td>20 9</td>
</tr>
<tr>
<td>64</td>
<td>5 14</td>
<td>9 12</td>
<td>14 14</td>
<td>44 12</td>
</tr>
<tr>
<td>128</td>
<td>5 17</td>
<td>11 16</td>
<td>15 17</td>
<td>47 17</td>
</tr>
<tr>
<td>256</td>
<td>5 20</td>
<td>9 20</td>
<td>16 20</td>
<td>58 21</td>
</tr>
</tbody>
</table>

![Fig. 5. Data decomposition.](image-url)
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 $i \in T_p^l$ 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 $ij$:

$$\sum_{p \in P_l(i) \cap P_l(j)} K_{ij}^p = K_{ij}^{\text{consist}}. \quad (44)$$

The same arguments apply to the right-hand side of $f_i$ which is also assembled inconsistently without communication. If the 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):

```c
psolve (K, u, f, e\text{linit})
{
  given u consistent, K, f inconsistent;
  (UM) Q_p = \begin{cases}
    \sum_{q \in P_l(p) \cap P_l(q)} K_{pq}^q & p = p^*(i) = p^*(j) \\
    1 & i = j \land p \neq p^*(i) \\
    0 & \text{else}
  \end{cases}
  \quad (m = 0, 1, \ldots)

  (D) d_p = f_p - \sum_{p \in P_l} K_{ij}^p u_j^p, \quad p \in P, \ i \in P_p;

  (UD) d_i^p = \begin{cases}
    \sum_{q \in P_l(p)} d_q^i & p = p^*(i) \\
    0 & \text{else}
  \end{cases} \quad p \in P, \ i \in P_i;

  (N) r_m = \sqrt{\sum_{p \in P_l} \sum_{i \in P} (d_i^p)^2};

  (E) if (r_m < r_{\text{tolinit}}) break;

  (BJ) v_p = (M_p)^{-1} d_p, \quad Q_p = M_p - N_p \quad \forall p \in P;

  (C) v_i^p = \sum_{q \in P_l(p)} v_i^q, \quad p \in P, \ i \in P_i;

  (U) u_i^p = u_i^p + ov_i^p, \quad p \in P, \ i \in P_i;
}
```

The algorithm with a consistent initial guess is called $u$ 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 an inconsistent form without communication. Step (UD) transforms the defect $d$ from the inconsistent to the 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

$$M_{ij} = 0 \iff p^*(i) \neq p^*(j). \quad (45)$$

In other words, a processor may only (approximately) invert the matrix $Q_p$ computed in the preparation step (UM) of algorithm solve. Note that the unique defect $d_p^*(\text{part of } d \text{ stored in } p)$ 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 $M_{ij} = K_{ij}, \iff p^*(i) = p^*(j)$ and Eq. (45). 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 the unique mode, it must be transformed to the 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 solve without steps (N) and (E), i.e., it requires two interface communications per smoothing iteration and level. The discussion of the grid transfer operators and the coarse grid solver remains.

It can be shown that the restriction of an inconsistent defect and the prolongation of a consistent correction can be computed without communication if only the coarsest mesh $T_0$ is partitioned to the processors and if 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 $T_0, \ldots, T_{n-1}$ 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 the Cray Programming Environment Version 3.0 and -O2 optimization level.

Unless otherwise noted, the multigrid parameters were: two pre- and post-smoothing steps \((v_1 = v_2 = 2)\) with point-block-ILU\((0)\) and V-cycle \((\gamma = 1)\). The multigrid cycle was always used as a pre-conditioner in BiCGSTAB (see [10]). The nonlinear reduction required in the Newton procedure within each time-step was set to \(\epsilon_{\text{nl}} = 10^{-5}\) and the linear reduction per Newton step was computed by (15) using \(\epsilon_{\text{min}} = 10^{-4}\). At most \(q = 6\) line search steps were allowed in the Newton algorithm. Time-step reduction was not necessary in any of the examples presented here. BDF \((1)\) (i.e., implicit Euler) was used as a time-stepping scheme in all examples.

A nested iteration procedure was 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 was used as an 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., without capillary pressure. In this case saturation is governed by a hyperbolic equation. The problem is treated for different discretisation techniques in [13].

The setup for the five-spot problem is given in Fig. 6. The reservoir is initially filled completely by the nonwetting phase fluid and is displaced by the wetting phase fluid from the lower left corner. At the inflow boundary \(\Gamma_{\text{IN}}\) a flux of \(Q_w = 0.0032\) kg/s/m\(^2\) is prescribed while the flux of the nonwetting phase is zero here. At the outflow boundary \(\Gamma_{\text{OUT}}\) the pressure \(p_n\) is fixed to \(10^5\) Pa and the wetting phase saturation is assumed to be zero: \(S_w = 0\). At all other boundaries, zero flux conditions are supplied. The initial conditions are \(S_n = 1\) and \(p_n = 10^5\) Pa throughout the reservoir.

The porosity is set to \(\Phi = 0.2\) and the fluid parameters are \(\rho_w = \rho_n = 1000\) kg/m\(^3\) and \(\mu_w = 0.001\) s Pa, \(\mu_n = 20\mu_w\). For the absolute permeability, we consider three different cases: in case \((A)\), we use a homogeneous field with \(k = 10^{-10}\) m\(^2\). Cases \((B)\) and \((C)\) use a geostatistically generated permeability field with \(160^2\) cells, a mean value of \(k = 10^{-10}\) 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. 7.

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

\[
\begin{align*}
{k_w(S_n)} &= S_n^\lambda, \\
{k_m(S_n)} &= S_n^\lambda \left(1 - (1 - S_n)^2\right).
\end{align*}
\]
Fig. 8 shows saturation plots for the cases (B) and (C) after 600 and 675 days of simulated time. Table 3 gives the results for the five-spot problem variants (A)–(C). Standard parameters are used as described above. The initial guess for the Newton iteration is obtained recursively by solving the time-step on the coarse mesh and interpolating the result. The coarsest mesh $T_0$ has $5 \times 5$ quadrilateral elements in all computations, the finest mesh uses 7 levels corresponding to $320 \times 320$ elements.

For each case the size of the finest mesh is varied while using a fixed time-step. This is 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 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 of case (C) is twice as expensive as that on the homogeneous field of 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, . . . , 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 [13], where experimental results are also reported. The problem setup is given in Fig. 9. The following three variants are investigated:

(A) A low permeable lens is placed into the interior of the reservoir as shown in Fig. 9 on the left.
Capillary pressure relations after Brooks and Corey with different entry pressures are assumed in the lens and the surrounding matrix. The entry pressure of the lens is high enough to avoid infiltration of the lens in case (A) (see below for specific values).

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

(C) Randomly generated porous medium with the permeability field shown in Fig. 9 on the right. The permeability which has the same mean value as the homogeneous case, is defined on a 192 \times 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. 9 (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 \( Q_n \) \( \hat{\mathbf{e}}_0 \) of \( 0.75 \text{ kg/(sm}^2) \) for the nonwetting phase is prescribed. Initial conditions were \( S_n \) \( \hat{\mathbf{e}}_0 \) and a hydrostatic pressure distribution. The fluid parameters were \( \rho_w = 1000 \text{ kg/m}^3, \rho_n = 1460 \text{ kg/m}^3, \mu_w = 0.001 \text{ s Pa} \) and \( \mu_n = 0.0009 \text{ s Pa} \).

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

\[
\begin{align*}
  k_w(S_e) &= S_e^{2+3\lambda/2}, \quad k_m(S_e) \\
  &= (1 - S_e)^2(1 - S_e^{2+3\lambda/2}), \quad p_c(S_e) = p_d S_e^{-1/\lambda},
\end{align*}
\]

where \( k = 6.64 \times 10^{-11} \) is the mean value of the permeability field and the parameters \( p_d \) and \( \lambda \) are taken from the matrix.

Fig. 10 shows solution plots for all three cases on a mesh with 384 \times 256 elements. In case (A), it can be observed that no DNAPL infiltrates the fine sand lens, while 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 are used. The initial guess for Newton’s method is obtained through nested iteration. The coarse mesh has \( 6 \times 4 \) elements, the finest mesh hierarchy has 7 levels. Again the time-step size is fixed (60 s for cases (A) and (B), 35 s for case (C)) while the spatial mesh size is 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 (separating 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).

### 6.3. VEGAS infiltration problem

The setup shown in Fig. 11 has been used in an experiment at the VEGAS facility in Stuttgart (cf. [15]). It consists of several fine sand lenses with different inclinations within coarse sand. It is very similar to the previous example but with a more complicated

<table>
<thead>
<tr>
<th>Sand</th>
<th>( \Phi )</th>
<th>( k ) (m(^2))</th>
<th>( S_{wr} )</th>
<th>( \lambda )</th>
<th>( p_d ) (Pa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td>0.4</td>
<td>( 6.64 \times 10^{-11} )</td>
<td>0.10</td>
<td>2.7</td>
<td>755</td>
</tr>
<tr>
<td>Lens</td>
<td>0.39</td>
<td>( 3.32 \times 10^{-12} )</td>
<td>0.12</td>
<td>2.0</td>
<td>1163.5</td>
</tr>
</tbody>
</table>

Fig. 9. Domain and permeability field for the DNAPL infiltration problem.
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. 11. Brooks–Corey’s 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. 11.

![Fig. 10. 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 x 256 for all the cases.](image)

Table 5

<table>
<thead>
<tr>
<th>Case</th>
<th>STEPS</th>
<th>MESH</th>
<th>EXECT</th>
<th>NLIT</th>
<th>AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>75</td>
<td>48 x 32</td>
<td>398</td>
<td>253</td>
<td>3.0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>96 x 64</td>
<td>1840</td>
<td>298</td>
<td>3.2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>192 x 128</td>
<td>7601</td>
<td>235</td>
<td>3.9</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>384 x 256</td>
<td>31 369</td>
<td>234</td>
<td>4.0</td>
<td>7</td>
</tr>
<tr>
<td>(B)</td>
<td>75</td>
<td>48 x 32</td>
<td>400</td>
<td>254</td>
<td>2.8</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>96 x 64</td>
<td>1915</td>
<td>262</td>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>192 x 128</td>
<td>7802</td>
<td>245</td>
<td>3.8</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>384 x 256</td>
<td>32 409</td>
<td>254</td>
<td>3.7</td>
<td>7</td>
</tr>
<tr>
<td>(C)</td>
<td>60</td>
<td>48 x 32</td>
<td>497</td>
<td>364</td>
<td>2.7</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>96 x 64</td>
<td>2689</td>
<td>381</td>
<td>3.9</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>192 x 128</td>
<td>11502</td>
<td>336</td>
<td>4.9</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>384 x 256</td>
<td>53 168</td>
<td>320</td>
<td>6.4</td>
<td>12</td>
</tr>
</tbody>
</table>

Fig. 12 shows the saturation obtained after 2 h of simulated time. Note that the U-shaped lens (sand 1) to the right has been infiltrated by the DNAPL.

Table 7 gives the solver performance on up to 256 processors of the CRAY T3E. The coarsest mesh is that shown in Fig. 11 with 290 elements. Six levels of uniform refinement result in the finest mesh with about 1.2 million elements or 2.4 million unknowns. Multigrid parameters are used as listed above except for a symmetric point-block Gauß–Seidel smoother which is used instead of the point-block ILU(0) smoother. The Gauß–Seidel smoother shows slightly lower iteration numbers in the parallel case. The initial guess for the Newton iteration is obtained through nested iteration. The number of elements per processor is fixed at 4640, i.e., we consider a scaled computation where the problem size increases linearly with the number of processors. The time-step size is 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 four-fold increase in execution time from one processor to 256 processors. This increase can be explained by the increase in 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 to the decreasing spatial mesh size. Nevertheless the scalability of the whole algorithm is quite acceptable. The propagation speed of the nonwetting phase infiltration front in this example is more than 6 mesh cells per time-step in the finest calculation.

Table 8 compares the multigrid pre-conditioner with the symmetric point-block Gauß–Seidel smoother to a stand-alone symmetric point-block Gauß–Seidel pre-conditioner. The 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 h 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 the system in question is a fully coupled one). 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. A “Modification” of the ILU scheme (see [2]) makes an order reduction possible (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 be true 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 3D DNAPL infiltration problem with two fine sand lenses has been set up. The porosity, permeability, and Brooks–Corey parameters are given in Table 9.

---

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

<table>
<thead>
<tr>
<th>Sand</th>
<th>$\Phi$</th>
<th>$k \text{ (m}^2\text{)}$</th>
<th>$S_w$</th>
<th>$S_r$</th>
<th>$\lambda$</th>
<th>$\mu_0$ (Pa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4</td>
<td>$4.60 \times 10^{-10}$</td>
<td>0.10</td>
<td>0.0</td>
<td>3.0</td>
<td>234.0</td>
</tr>
<tr>
<td>1</td>
<td>0.4</td>
<td>$3.10 \times 10^{-11}$</td>
<td>0.12</td>
<td>0.0</td>
<td>2.5</td>
<td>755.0</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>$9.05 \times 10^{-12}$</td>
<td>0.15</td>
<td>0.0</td>
<td>2.0</td>
<td>1664.0</td>
</tr>
</tbody>
</table>

Fig. 11. Setup of the two-dimensional VEGAS experiment.

Fig. 12. Saturation plot for the VEGAS Experiment after 240 steps of 30 s.
The position of the lenses and isosurfaces for DNAPL saturation are shown in Fig. 13. A water flow from the right to the left (in the top picture) has been prescribed. The solution shows a counter current flow of the DNAPL and no infiltration of the fine sand lenses.

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

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

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

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

Table 9
Sand properties for the 3D DNAPL infiltration problem

<table>
<thead>
<tr>
<th>Sand</th>
<th>$\phi$</th>
<th>$k$ (m²)</th>
<th>$S_w$</th>
<th>$S_n$</th>
<th>$\lambda$</th>
<th>$p_d$ (Pa)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse</td>
<td>0.40</td>
<td>$5.0460 \times 10^{-10}$</td>
<td>0.08</td>
<td>0.0</td>
<td>3.86</td>
<td>369</td>
</tr>
<tr>
<td>Fine</td>
<td>0.39</td>
<td>$5.26 \times 10^{-11}$</td>
<td>0.10</td>
<td>0.0</td>
<td>2.49</td>
<td>2324</td>
</tr>
</tbody>
</table>

The position of the lenses and isosurfaces for DNAPL saturation are shown in Fig. 13. A water flow from the right to the left (in the top picture) has been prescribed. The solution shows a counter current flow of the DNAPL and no infiltration of the fine sand lenses.

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

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

6.5. 3D Air sparging simulation

The final example simulates the bubbling of air in a 3D heterogeneous porous medium. The domain is given in Fig. 14. It is 5 m high and about $4 \times 5$ m wide. Three lenses with different sand properties are placed within
the domain as shown in the figure and the air is injected
from below at three inlets into the initially water-satu-
rated porous medium.

The fluid properties are \( \rho_w = 1000 \, \text{kg/m}^3 \), \( \mu_w = \mu_n / 84,149.6 \, \text{kg/m}^3 \), \( \mu_n = 10^{-3} \, \text{Pa} \, \text{s} \), and \( \mu_n = 1.65 \times 10^{-5} \, \text{Pa} \, \text{s} \). The properties of the sands are given in
Table 11.

The coarse mesh is shown in Fig. 14. It consists of
1492 tetrahedral elements and all internal boundaries
are resolved by faces of the initial mesh. The mesh was
generated with “NETGEN”, see [20]. Four levels of
uniform refinement resulted in a fine mesh with six
million tetrahedral elements and about two million un-
knowns. The time-step size was \( \Delta t = 8 \, \text{s} \) and 80 time-
steps were computed. The nonwetting phase front
moved about 2 mesh cells per time-step in the finest
calculation. Fig. 15 shows an isosurface of the satu-
rati on \( S_w = 0.05 \) at final time that was obtained with the
visualization software GRAPE.

The performance of the solver is shown in Table 12.
The problem size is scaled by a factor of 64 with a
corresponding increase in the number of processors.
The increase in total computation time is almost en-
tirely due to the increase in Newton iterations. Newton
iterations increased in this example because interpola-
tion of initial guesses from a coarse grid solution was
not effective. Using the solution of the preceeding time-
step as initial guess for the new time-step lead to fewer
iterations but was not robust with respect to Courant
number. The multigrid pre-conditioner using a
symmetric Gauß–Seidel smoother parallelized well (TI)
and the number of iterations remained almost

<table>
<thead>
<tr>
<th>Sand</th>
<th>( \Phi )</th>
<th>( k ) (m²)</th>
<th>( S_w )</th>
<th>( S_r )</th>
<th>( \lambda )</th>
<th>( p_d ) (Pa)</th>
<th>( S_w^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.40</td>
<td>( 5.04 \times 10^{-10} )</td>
<td>0.10</td>
<td>0.0</td>
<td>2.0</td>
<td>1600.0</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>0.39</td>
<td>( 2.05 \times 10^{-10} )</td>
<td>0.10</td>
<td>0.0</td>
<td>2.0</td>
<td>1959.6</td>
<td>0.30</td>
</tr>
<tr>
<td>2</td>
<td>0.39</td>
<td>( 5.62 \times 10^{-11} )</td>
<td>0.10</td>
<td>0.0</td>
<td>2.0</td>
<td>2565.7</td>
<td>0.55</td>
</tr>
<tr>
<td>3</td>
<td>0.41</td>
<td>( 8.19 \times 10^{-12} )</td>
<td>0.10</td>
<td>0.0</td>
<td>2.0</td>
<td>4800.0</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Fig. 14. Geometry (left) and coarse grid (right) for the 3D air sparging problem (visualization with GRAPE).

Fig. 15. Isosurface \( S_w = 0.05 \) after 640 s of simulated time in 3D air sparging problem (visualization with GRAPE).
constant when \( P \) was increased and the mesh size was reduced.

7. Conclusions

In this paper we have presented a fully-coupled solution method for two-phase flow in porous media. The large nonlinear systems per time have been solved stepwise by a Newton-multigrid algorithm. The different terms in the Jacobian and their influence on the multigrid method have been discussed in detail. A truncated restriction operator has 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 2D and 3D examples. Very satisfactory multigrid performance has been observed in all examples and a comparison with a pre-conditioned single grid method has shown the superiority of multigrid. Up to 5.2 million unknowns could be treated in three space dimensions.

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

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

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft in the Sonderforschungsbereich 404, project A3. We wish to thank all members of the UG group at ICA III, Stuttgart, especially Klaus Birken, Stefan Lang, and Christian Wiener for being allowed to use their work on the parallel version and Klaus Johannsen for his implementation of BiCGSTAB.

The use of the visualization software GRAPE is greatly acknowledged.

References

