A method for simulating sharp fluid interfaces in groundwater flow

Erik J. Holm a, Hans Petter Langtangen b, *

a Institute for Energy Technology (IFE), 2007 Kjeller, Norway
b Mechanics Division, Department of Mathematics, University of Oslo, P.O. Box 1053 Blindern, N-0316 Oslo, Norway

Received 24 April 1998; received in revised form 23 November 1998; accepted 24 November 1998

Abstract

We present a numerical model for two-phase porous media flow, where the phases are separated by a sharp interface. The model is based on a unified pressure equation, and an advection equation for tracking a pseudo-concentration function. The zero-level set of this function defines the interface between the fluids. The finite element method is used for spatial discretization, with local grid refinements in the vicinity of the interface. Examples on applications involving moving interface and steady-state seepage problems are investigated. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Groundwater flow; Finite element methods; Front tracking; Two-fluid model; Free boundary problem

1. Introduction

Many groundwater flow problems involve the computation of macroscopic fluid interfaces. Saltwater intrusion in aquifers and the water–air phreatic surface are two common examples. Due to heterogeneous features of the medium and microscopic hydrodynamic dispersion, the fluid interfaces are seldom sharp at the microscopic scale. However, the assumption of a sharp interface is often appropriate from a modeling point of view and consistent with the lack of detailed physical insight into the microscopic flow processes in the interface region.

Seepage problems with free boundaries are classical model problems in numerics and mathematics. Considering seepage through a dam, the large difference in density between water and air makes it natural to neglect the movement of the air phase. The flow domain then covers water only, but the extent of the domain is unknown. The handling of moving, sharp interfaces is normally a challenging problem from a numerical point of view [27]. Numerical solution approaches can be classified into two main categories, using either a moving or a fixed grid. Moving grid here means that only the water saturated part of the medium is gridded, and that the extent of this grid must be computed as part of the solution method. We refer to Chung and Kikuchi [6] for an application of this approach to steady-state seepage through a dam. Only fixed grid methods are considered in the present paper. In this class of methods, a pressure field is extended from the water zone into the air zone by various strategies. The classical Baiocchi transformation [1] is often used to convert free boundary problems into formulations involving variational inequalities. More information about relevant methods based on variational inequalities and their applications to groundwater flow is found in Bruch [5]. Unfortunately, it seems difficult to create efficient methods based on the Baiocchi transformation and variational inequalities for handling general irregular domains. Another popular class of methods for free boundary problems in groundwater flow is the residual flow procedure introduced by Desai [8]. Westbrook [26] presents a comparison of residual flow procedures and variational inequalities. The approach followed in the present work is different, and more general, than the residual flow procedures or variational inequalities, although there are certain similarities with the residual flow formulation. It should also be mentioned that boundary integral methods constitute an efficient way of solving sharp interface problems in groundwater flow [17], especially in 2D problems. However, this class of methods are not readily applied to heterogeneous media.

The purpose of this paper is to present a unified framework for simulating flow with sharp interfaces in heterogeneous porous media. We will focus mainly on...
the two-fluid situation, and particularly at the computation of the phreatic surface in the numerical examples. The cited contributions so far deal primarily with steady-state seepage problems. This is also a central application in the present paper, but our method has a true time-dependent nature, and can therefore be applied to transient two-phase flow (provided that the saturation of each phase is either 0 or 1) as well as to steady-state free boundary problems. In contrast to standard approaches to the seepage problem, our formulation works with the movement of both fluid phases through a fixed domain. The advantage of this generalized formulation is first of all easier grid handling and no explicit movement of the interface surface. Moreover, the extension to more complicated free boundary problems is conceptually straightforward. Just as in the residual flow procedures, we work with a unified pressure field over the complete domain.

The fluid interface is defined as the zero level set of a function $F$ that is advected with the velocity field. This is principally the same approach as used in volume-of-fluids methods, but without concern of fill factors or couplings between nodes and control volumes. Our front tracking can also be classified as a simplified version of the original level set methods [21]. The simplification consists in not letting the set function $F$ measure the distance to the zero level set. An extra transient nonlinear PDE [12,22], to be integrated to a stationary state at each time level, is hence avoided. The paper demonstrates that a simplified approach can work satisfactorily in many applications. If the densities or the viscosities of the fluids are significantly different, the coefficients in the governing partial differential equation (PDE) exhibit large jumps. To increase the accuracy when such jumps appear in the interior of the elements, we propose to perform local mesh refinements in the vicinity of the interface. Other authors have also addressed sharp interface problems in groundwater flow using grid refinement methods, see e.g. [6,20]. Contrary to these works, we focus on refinements along the interface, inspired by the techniques from, for example, Refs. [7,24,25]. An obvious extension of our sharp interface model is to allow for an arbitrarily variable density. Kolditz et al. [18] have recently published a comprehensive study of numerical models for variable density flows. Hou et al. [12] have developed a finite difference method for Hele-Shaw flow, based on a full level set method for tracking the water/air surface and an immersed interface method for constructing accurate finite difference approximation in the cells containing the surface. Combination of adaptivity and the level set method when solving the Navier–Stokes equations is reported by Sussman et al. [23].

The equation that is used for tracking the interface is closely related to equations for specie transport. We will therefore show how models for contaminant transport, when hydrodynamic dispersion effects are neglected, can be trivially included in a simulator that implements our numerical framework, especially when the implementation makes use of object-oriented programming techniques.

Section 2 presents the mathematical model, and section 3 is devoted to numerical methods. A one-dimensional two-fluid problem is used for partial verification of the model in Section 4, whereas numerical examples of more general flows constitute the topic of Section 5.

2. The mathematical model

Consider the flow of two incompressible, homogeneous fluids in a porous medium. The fluids will be labeled by a superscript $\ell$, where $\ell = 1,2$. We assume that the saturation of a fluid is either zero or one. That is, there is no macroscopic mixing of the two fluids, and the interface is sharp. Fluid 1 occupies the domain $\Omega(1)(t)$, while fluid 2 occupies $\Omega(2)(t)$, where $t$ denotes the time. The union $\Omega = \Omega(1) \cup \Omega(2)$ is constant in time. Assuming that Darcy’s law holds for each fluid, the mass and momentum equations read

$$\nabla \cdot \mathbf{v}^1 = \sum_w Q_w \delta(x_w - x), \quad \text{in } \Omega(1),$$  

$$\nabla \cdot \mathbf{v}^2 = \sum_w Q_w \delta(x_w - x), \quad \text{in } \Omega(2),$$  

$$\mathbf{v}^{(1)} = -\frac{K}{\mu^{(1)}} (\nabla p^{(1)} - \rho^{(1)} \mathbf{g}), \quad \text{in } \Omega(1),$$  

$$\mathbf{v}^{(2)} = -\frac{K}{\mu^{(2)}} (\nabla p^{(2)} - \rho^{(2)} \mathbf{g}), \quad \text{in } \Omega(2).$$  

Here, $\mathbf{v}^{(\ell)}$ is the velocity in fluid $\ell$, $Q_w$ is the strength of a well $w$ located at the spatial point $x_w \in \Omega$, $x$ is a spatial point in $\Omega$, $K$ is the permeability tensor, $\mu^{(\ell)}$ is the dynamic coefficient of viscosity, $\rho^{(\ell)}$ is the pressure, $\rho^{(\ell)}$ is the density, and $\mathbf{g}$ is the acceleration due to gravity. Wells are modeled by Dirac delta functions $\delta$ in the continuity equations. The velocities can trivially be eliminated to yield two variable-coefficient Poisson equations for the pressures.

At the interface $\Gamma$ between the two fluids we have a kinematic and a dynamic condition

$$\mathbf{v}^{(1)} \cdot \mathbf{n} = \mathbf{v}^{(2)} \cdot \mathbf{n},$$  

$$p^{(1)} = p^{(2)},$$  

where $\mathbf{n}$ is a unit normal vector to $\Gamma$. The interface is unknown and a part of the solution of the flow problem. Let $\Gamma$ be given by $\Gamma = \{ x | F(x,t) = 0 \}$.

Since fluid particles on the interface will always stay there, the material derivative of $F$ must vanish:

$$\frac{\partial F}{\partial t} + \mathbf{V} \cdot \nabla F = 0,$$  

where $\mathbf{V}$ is the velocity field.

84  

E. J. Holm, H. P. Langtangen / Advances in Water Resources 23 (1999) 83–95
where \( \mathbf{V} \) is the velocity of particles belonging to the interface. We have the relation \( \mathbf{v} = n_1 \mathbf{V} \), with \( n_1(x) \) being the effective porosity with respect to the flow \([2]\). The value of \( F \) in fluid 1 and 2 is greater than and less than zero, respectively.

We may now formulate the initial-boundary value problem as follows:

\[
- \nabla \cdot \left[ \frac{K}{\mu^{(1)}} (\nabla p^{(1)} - \varrho^{(1)} \mathbf{v}) \right] = \sum_w Q_w \delta(x_w - x), \quad \text{in } \Omega^{(1)},
\]

\[
- \nabla \cdot \left[ \frac{K}{\mu^{(2)}} (\nabla p^{(2)} - \varrho^{(2)} \mathbf{v}) \right] = \sum_w Q_w \delta(x_w - x), \quad \text{in } \Omega^{(2)},
\]

\[
\frac{K}{\mu^{(1)}} (\nabla p^{(1)} - \varrho^{(1)} \mathbf{v}) \cdot \mathbf{n} = \frac{K}{\mu^{(2)}} (\nabla p^{(2)} - \varrho^{(2)} \mathbf{v}) \cdot \mathbf{n},
\]

on \( F = 0 \),

\[
p^{(1)} = p^{(2)}, \quad \text{on } F = 0.
\]

In addition, we must require zero flux \( \mathbf{v}^{(1)} \cdot \mathbf{n} \) at impermeable boundaries, \( \ell = 1, 2 \). The rest of the boundary of \( \Omega \) must have \( p^{(1)} \) or \( p^{(2)} \) prescribed. Finally, the \( F \) function must have initial values at time \( t = 0 \), taken to be 1 in fluid 1 and \(-1\) in fluid 2. From a mathematical point of view, this discontinuous initial \( F \) field must be smoothed such that \( F \) for \( t > 0 \) is continuous and the zero-level set is well defined.

The mathematical problem can be significantly simplified by introducing a weak formulation. Let \( W \) be a test function in a suitable function space. A weak formulation of Eqs. (8)-(11) might then read

\[
\int_{\Omega^{(1)}} \nabla W \cdot \frac{K}{\mu^{(1)}} (\nabla p^{(1)} - \varrho^{(1)} \mathbf{v}) \, dx
\]

\[
+ \int_{\Omega^{(2)}} \nabla W \cdot \frac{K}{\mu^{(2)}} (\nabla p^{(2)} - \varrho^{(2)} \mathbf{v}) \, dx
\]

\[
+ \int_{\Gamma} W \frac{K}{\mu^{(1)}} (\nabla p^{(1)} - \varrho^{(1)} \mathbf{v}) \cdot \mathbf{n} \, ds
\]

\[
- \int_{\Gamma} W \frac{K}{\mu^{(2)}} (\nabla p^{(2)} - \varrho^{(2)} \mathbf{v}) \cdot \mathbf{n} \, ds
\]

\[
= \sum_w Q_w W(x_w).
\]

The two integrals along \( \Gamma \) cancel due to the kinematic boundary condition (10). Moreover, we can combine the integrals over \( \Omega^{(1)} \) and \( \Omega^{(2)} \) by defining a variable fluid property, like \( \varrho \), according to

\[
\varrho(F) = \begin{cases} 
\varrho^{(1)}, & F \geq 0, \\
\varrho^{(2)}, & F < 0.
\end{cases}
\]

with a similar definition of \( \mu(F) \). Since \( p^{(1)} = p^{(2)} \) on \( \Gamma \), we can introduce a unified pressure field \( p \) in \( \Omega \). This leads to the weak form

\[
\int_a \nabla W \cdot \frac{K}{\mu(F)} (\nabla p - \varrho(F) \mathbf{v}) \, dx = \sum_w Q_w W(x_w).
\]

The value of \( F \) is governed by Eq. (7), which can be alternatively written

\[
\frac{\partial F}{\partial t} - \frac{K}{n_1 \mu(F)} (\nabla p - \varrho(F) \mathbf{v}) \cdot \nabla F = 0.
\]

Eqs. (13) and (14), together with the boundary conditions for \( p \) and the initial condition for \( F \), govern the motion of the two fluids. Notice that the internal boundary is not explicitly a part of the formulation. Eq. (13) is most naturally solved by finite element techniques, whereas Eq. (14) can be solved by any numerical scheme appropriate for hyperbolic problems. Notice that the smoothing of an initially discontinuous \( F \) field is normally automatically accomplished through some built-in interpolation in the numerical formulation. This is particularly the case in the finite element method we will apply for Eq. (14). The smoothing has nothing to do with artificinal diffusion; it is just a consequence of interpolating a discontinuous function by a continuous finite element field. Our convention of letting \( F \) be initially constant in each fluid is chosen for mathematical convenience – any smooth function that has the same zero level set can in fact also be used.

We remark that Eq. (13) is consistent with a unified continuity equation of the form \( \nabla \cdot \mathbf{v}(F) = 0 \). The validity of this equation arise from the underlying assumption that, although the density varies throughout the domain, the density remains constant for a fluid particle.

3. Numerical formulation

With today’s emerging geological information systems, researchers and engineers will get access to mathematical representations of the complicated geometry often encountered in groundwater reservoirs. The gridding of such geometric models and the accompanying simulation are most easily carried out in a finite element framework. We have therefore based our numerical approach on finite element discretization.

The weak formulation of the time dependent boundary-value problem for the unified pressure field, Eq. (13), can straightforwardly be transformed to a discrete form by standard finite element techniques. The discrete, unified pressure field \( \hat{p} \) is written as

\[
\hat{p} = \sum_{j=1}^m N_j(x) p_j(t),
\]

where \( N_j \) denotes finite element basis functions, and \( p_j(t) \) are functions to be computed at discrete time levels, i.e., we seek \( p_j(k \Delta t), \ k = 0, 1, 2, \ldots \) Using a Galerkin approach, Eq. (13) is then supposed to hold for all \( W = N_j \),
i = 1, . . . , m. This gives, as usual, a linear system that must be solved at each time level. The coefficient matrix of this linear system is positive definite, and can be solved by, for instance, preconditioned conjugate gradient methods. In the numerical examples herein we have used a standard modified incomplete LU factorization (MILU) preconditioner [3].

Normally, Eq. (14) makes special demands on the numerical methods to avoid nonphysical oscillations near sharp fronts. In the present problem, the accuracy of the $F(x, t)$ function itself is not of major interest, since we only make use of the sign of $F$. The central point is to keep the position of the zero-level set of $F$ accurate enough for the pressure computation. This can be accomplished by, for example, a streamline-upwind/Petrov-Galerkin (SUPG) finite element method in space [14] and a two-level finite difference approximation (θ-rule) in time. Using the same finite element basis functions for $F$ as for $p$, but using SUPG-type weighting functions $L_i$, we have the following discrete form of Eq. (14):

$$\sum_{j=1}^{m} \left( \int_{\Omega} \left( L_i N_j - L_i \theta \Delta t \frac{K}{n_e \mu(F)} \left( \nabla \hat{p}^{k+1} - q(F^k) g \right) \cdot \nabla N_j \right) d\Omega \right) F_j^{k+1}$$

$$= \int_{\Omega} \left( L_i \hat{F}^k + L_i (1 - \theta) \Delta t \frac{K}{n_e \mu(F)} \left( \nabla \hat{p}^k - q(F^k) g \right) \cdot \nabla \hat{F}^k \right) d\Omega,$$

(15)

where $i = 1, . . . , m$, $\hat{F}^k$ is the discrete counterpart to $F$, $\hat{F}^k = \sum_{j=1}^{m} N_j(x) F_j^k$, with $F_j^k$ being the value of $F$ at node $j$ at time level $k$. Furthermore, $\Delta t$ is the current time step length, which will normally vary throughout the simulation, and $L_i$ has typically the form $L_i = N_i + \tau \nabla N_i$, where $\tau$ is a parameter that depends on the mesh size and $V$ [14].

The discrete pressure and front tracking equations are coupled in a nonlinear manner. To simplify the solution process, we employ the common strategy of splitting the compound differential operator in our PDE system. At time level $k + 1$, we first solve the discrete pressure equation for $\hat{p}^{k+1}$, and then we solve Eq. (15) with respect to $\hat{F}^{k+1}$. An iteration between the two equations can be introduced at each time level, but our experiments do not indicate that such an approach is necessary.

When the jump in density or viscosity is significant, which is the case for water–air systems, the suggested method can be inaccurate in the vicinity of the front. Moving the mesh such that the front coincides with the element boundaries, will normally ensure a satisfactory treatment of the jump conditions in a finite element setting. Such moving mesh algorithms are, however, complicated and computationally expensive. The present paper suggests to use a stationary grid, where the fluid interface intersects the elements. To improve the accuracy near the front, we apply local mesh refinements [11]. The criterion for refining elements is based on the elements’ distance to the interface, defined by $F = 0$. A posteriori error estimation might be complicated in the present problem, since the error should be based on $p$, rather than $F$, while the refinement criterion arising from the error estimator must definitely focus on refinements where $F$ changes rapidly. Error control in level set-like methods therefore seems to be a challenging task. Nevertheless, the geometric properties of the $F$ function can easily be analyzed and used in mesh refinement criteria. This is the refinement approach followed in the present work.

Although the refinement method is described in Ref. [11], some of the technical terms need to be explained here since we make use of these later when describing the numerical examples. The elements are restricted to be tensor products of 1D Lagrange elements. Each box-shaped element is subdivided into $(n + 1) \times (n + 1)$ elements, i.e., $n$ measures the subdivision of each element in local mesh refinements. If the neighbor elements are not refined, this subdivision leads to irregular nodes, also called slave, constrained, hanging, or improper nodes. The function values at the irregular nodes are constrained by the function values of the surrounding regular nodes, using standard finite element interpolation. It is common to refer to such a grid as an $n$-irregular grid, where the value $n = 1$ is dominating in the literature. As pointed out in Ref. [11], $n > 1$ can be advantageous in some problems. The subdivision can be repeated $M$ times, yielding a mesh with $M$ refinement levels.

The numerical scheme is implemented in a C++ code based on the Diffpack [9,15] libraries. The implementation makes heavy use of object-oriented programming techniques. We create two solvers, one for the discrete pressure equation and one for the discrete front tracking Eq. (15). These solvers are realized as classes in C++. The implementation of one equation is completely independent of the implementation of the other equation. This makes it trivial to reuse the code in other problems or to replace our Poisson equation solver by a more efficient one based on, e.g., adaptive multigrid methods. The present front tracking solver is, in fact, reused in a manager class that establishes communication between the solver classes and administers the time stepping algorithm. Notice that this implementational approach relies on the operator splitting technique, such that the PDEs are solved in sequence at each time level. The general details of the software design are documented in Refs. [4,16]. From a
practical point of view, building solvers for systems of PDEs by assembling solvers for each scalar PDE proves to be a very efficient and reliable way of developing simulation codes for a wide range of scientific and engineering problems.

As a by-product of the software design, it was a trivial task to add a second front tracking solver to the manager class. This solver also deals with Eq. (14), but now \( F \) can be given the interpretation as the concentration of a pollutant. We refer to this concentration as \( c \) in the following. Although we use identical code for the former since all values of \( c \in [0, 1] \) have physical significance. The proposed SUPG/\( \theta \)-rule approach seems, however, to be acceptable. We remark that hydrodynamic dispersion is an important effect in contamination transport and that a proper diffusion term should be included in the transport equation for \( c \) to model this phenomenon. In a more accurate groundwater model one can therefore not simply reuse the front tracking solver for contaminant transport.

Using the Diffpack libraries, it is straightforward to implement a general solver where the number of space dimensions is just a parameter that can be set at run time. Therefore, our simulator is capable of dealing with 1D, 2D, and 3D problems.

4. Analysis of a one-dimensional problem

For partial evaluation of the numerical method and for verification of the computer implementation, we have analyzed a one-dimensional two-phase flow problem with a sharp interface. The domain of interest is taken as \([0, 1]\). There are no wells (\( Q_o = 0 \)). Initially, the domain is filled with fluid 2, which means that \( F = -1 \) at \( t = 0 \). Fluid 1 is then injected at \( x = 0 \), implying the boundary condition \( F = 1 \) at \( x = 0 \) and \( t > 0 \). The pressure is kept constant at unity for \( x = 0 \) and vanishes for \( x = 1 \). Gravity effects are neglected, and \( K \) is treated as a constant.

The analytical solution can be found from the formulation Eqs. (8)–(11), with Eq. (7) for the movement of the front. It is clear that in one dimension, the velocity must be constant in space for each fluid. From the interface condition it follows that the velocity is a pure function of time, which is denoted by \( v(t) \). We can then solve Eq. (7) for the front movement:

\[
F(x, t) = 1 - 2H(x - \Phi(t)),
\]

where \( H(\xi) \) is the Heaviside function and

\[
\Phi(t) = \int_0^t v(\tau) \, d\tau.
\]

Integrating the pressure in each fluid domain and applying the interface conditions, leads to

\[
p^{(1)}(x, t) = C_1 \mu^{(1)} x + 1,
\]

\[
p^{(2)}(x, t) = C_1 \mu^{(2)} (x - 1),
\]

with

\[
C_1 = \frac{1}{\mu^{(1)}} \left[ a(t) \left( 1 - \frac{\mu^{(2)}}{\mu^{(1)}} \right) + \frac{\mu^{(2)}}{\mu^{(1)}} \right]^{-1}.
\]

This implies that the velocity in both phases equals \(-KC_1\). The zero-level set of \( F \) determines the front position \( x = a(t) \). This leads to

\[
a(t) = \Phi(t), \quad \frac{d\Phi}{dt} = v(t).
\]

The latter of these gives a differential equation for \( a(t) \), with initial condition \( a(0) = 0 \). If \( \mu^{(1)} = \mu^{(2)} \), the solution is \( a(t) \sim t \), which is expected if the medium is completely saturated with a single fluid.

The complete solution to the initial-boundary value problem now becomes

\[
p(x, t) = \begin{cases} 
C_1 \mu^{(1)} x + 1, & x < a(t), \\
C_1 \mu^{(2)} (x - 1), & x \geq a(t), 
\end{cases}
\]

\[
C_1 = \frac{1}{\mu^{(1)}} \left[ a(t) \left( 1 - \frac{\mu^{(2)}}{\mu^{(1)}} \right) + \frac{\mu^{(2)}}{\mu^{(1)}} \right]^{-1},
\]

\[
a(t) = \frac{\left( \frac{2(1 - \mu^{(2)} / \mu^{(1)})}{\mu^{(1)}} t + \left( \frac{\mu^{(2)}}{\mu^{(1)}} \right)^2 \right)^{1/2} - \mu^{(2)} / \mu^{(1)}}{1 - \mu^{(2)} / \mu^{(1)}}.
\]

Consider the limit \( \mu^{(2)} / \mu^{(1)} \to 0 \). We then achieve the reasonable result

\[
p(x, t) = \begin{cases} 
1 - x / a(t), & x < a(t), \\
0, & x \geq a(t)
\end{cases}
\]

with

\[
a(t) = \sqrt{2Kt / \mu^{(1)}}.
\]

The numerical method described in the previous section can be evaluated in the present one-dimensional problem. For the particular numerical example to be reported here, the values of \( \mu^{(1)}, \mu^{(2)}, \) and \( K \) equal 1, \( 1.8 \cdot 10^{-2} \), and 1, respectively. Linear finite elements are used for \( p^{(1)} \) and \( \Phi^{(1)} \), with \( \theta = 1/2 \) in the time integration scheme for Eq. (15). The SUPG method reproduces a standard, two-point upwind scheme for the front tracking equation in 1D. During the simulation we monitor the Courant number and adaptively choose \( \Delta t \) such that the maximum Courant number is close to unity. At higher Courant numbers the solution method is numerically stable, but the front can then move several element lengths during one time step and be located outside the refined front region. Limiting the Courant number to be of order unity avoids the need for iterating on the refinements and the front position. We have seen from the numerical experiments that limiting the Courant number to unity and employing explicit time
integration of the advection equation, gives a generally less robust overall solution method. Since the convective term in the front tracking equation needs to be re-assembled at every time level and the iterative solution of the linear system in Eq. (15) is very fast (using BiCGStab or Orthomin, combined with ILU preconditioning), the additional work implied by $\theta > 0$ is almost negligible.

Both uniform and locally refined meshes have been tested. The uniform meshes have 26, 51, 101, 251, 751 and 1001 nodes respectively, corresponding to element lengths varying from 0.04 down to 0.001. The refinement is determined dynamically from the location of the front in the previous time step. The elements containing the front, as well as the neighbor element on both sides, are subdivided. This refinement is performed on the coarse background mesh and then repeated recursively down to the desired level. We have performed seven such computations, using from one up to seven levels of refinement. Moreover, $n = 1$, i.e. each selected element is divided into two new elements. The meshes then become 1-irregular. The coarse background mesh contains 10 elements. The element sizes closest to the fluid front hence vary from 0.05 down to 0.00078 in the respective computations with refined meshes. The number of nodes in the refined computations varies slightly with time and equals, on average, 16, 22, 34, 53, 85, 154 and 282, respectively.

We have also performed computations using a higher subdivision inside the selected elements, i.e. $n > 1$. This leads to larger jumps in the element size and is often avoided by finite element practitioners. Nevertheless, such large jumps might lead to an overall more efficient computational algorithm in some problems [11]. The computations with $n > 1$ use one level of refinement from a background mesh with 10 elements, and a sub-partition of 4, 8, and 16 elements, respectively. In the computation with sub-partition 16, the element size hence jumps directly from $\Delta x = 0.1$ to $\Delta x = 0.00625$. This results in meshes of approximately 23, 39 and 71 nodes, respectively.

Fig. 1(a) depicts the absolute error of the front position at $t = 0.4$ as the mesh is refined, either locally or uniformly. The convergence is linear, i.e., the error in front position is of order $\Delta x^l$. Notice that this represents an accumulated error in the velocity, cf. Eq. (18). The convergence of the $n$-irregular computations is also shown in the figure, being almost as good as the 1-irregular meshes, but the efficiency of the adaptive grid construction is better than in the case $n = 1$. We have also computed the accumulated $L^2$ norm of the pressure error in the time interval $[0, 0.5]$. This quantity is displayed in Fig. 1(b) as a function of the number of nodes in uniform and refined grids.

The experience from this case study shows that we can use a coarse background mesh and employ a significant reduction in the element size around the fluid front and thereby improve the accuracy of the method. The 1-irregular mesh shows the best performance in terms of convergence rates, but the $n$-irregular meshes
are cheaper to compute and may therefore be the most CPU-time effective solution strategy. An important message from these experiments is that significant jumps in the element size does not destroy the accuracy when solving the present PDE system consisting of an elliptic Poisson equation and a hyperbolic advection equation.

5. Numerical examples

In this section we present an evaluation of the proposed numerical framework in three case studies. The first study aims at demonstrating a potential numerical problem with the proposed methodology. The next example evaluates the mass conservation properties of a discrete two-fluid system, whereas the final example shows a more engineering-oriented application of the method.

5.1. Possible numerical instability of a water–air system

We consider a simple problem with a horizontal water–air interface at $z = 0.5$, as depicted in Fig. 2. The computational domain is a two-dimensional box: $[0, 1] \times [0, 1]$, but the box walls are completely permeable, with a prescribed hydrostatic pressure distribution in the $z$ direction. The ratio of the densities is 1000, while the ratio of the viscosities is 55. The permeability is constant. The coarse background mesh has only $8 \times 8$ bilinear elements. In the computation, the interface $z = 0.5$ coincides with the element boundaries. Starting with the exact solution for $F$ results in stable computations, that is, the same steady-state solution is reproduced from time step to time step. However, if we raise the interface slightly, such that it intersects the interior of the elements, and then simulate the motion towards the expected stationary state, the method might face substantial problems; numerical instabilities start to develop, and the solution process becomes unstable. Fig. 3 shows a slightly wavy interface shape already after a short simulation time. Using bilinear or linear elements, the pressure gradient and the velocity are naturally treated as constant in each element, with the constant value being computed from finite element data at the centroid of the element. The computed pressure gradient in a transition element depends on the values of the material properties at the integration points used for evaluating the finite element data. These values can be computed in several ways. The most straightforward approach is to apply either air or water properties, according to the sign of $F$, as in Eq. (12). The velocity is then strongly dependent on whether the centroid is in the air or water phase. With a slightly wavy front, it is easy to imagine that the velocities of two neighboring elements can differ highly and lead to amplified instabilities, as shown in Fig. 3.

Alternative velocity computations consist of interpolating $\mu$ and $\varrho$ at the centroid or using more sophisticated averaging techniques, like harmonic averaging. The velocity computation based on interpolation will correctly use a mean value of the material properties applied in the pressure computation. The numerical experiments show that the interpolation procedure produces a stable computation of the phreatic surface towards the stationary state, when homogeneous Neumann conditions are applied on the walls. Using material properties based on $F$, or based on a harmonic mean, provides unstable computations. Furthermore, the hydrostatic boundary condition implies a constant density in a boundary element and will strongly affect the pressure gradient in the transition elements at the boundary, as seen in Fig. 2(a). This is a second effect for creating unstable velocity conditions.

Using material properties for air as constant properties for the transition elements, in the finite element integrals as well as in the velocity computation, and making the element boundaries coincide with the prescribed water level at the boundaries, we achieve the necessary consistency between the boundary conditions and the material properties used in the integration points. In the presence of the Dirichlet-type conditions, i.e. hydrostatic pressure, using air in the transition elements produces stable computations. On the other hand, the interpolation procedure, the straightforward $F$-based method, and the harmonic mean, all provide unstable computations in the Dirichlet case. Our experimentation with different interpolation strategies indicates that using a signed distance function $F$, which is standard in the level set method, only yields other perturbations of the interpolation procedure, and will most likely not be able to cure the instability problem.
5.2. Convergence towards equilibrium interfaces

The purpose of this test problem is to investigate the ability of the method to converge to a trivial solution, starting with an arbitrary shape of the interface between two fluids, this time confined in a box with impermeable walls. We will also study the loss of mass of each phase during the simulation. The pressure equation we use conserves the total mass, because \( N_i \) is a function in the finite element space, but the front computation may be suspected to move mass from one phase to the other. We start with an interface, between water and air, with the shape of a Gaussian bell function, as shown in Fig. 4(a), and let gravity move the interface towards the horizontal equilibrium position. Air, being the lighter fluid, is on top of water such that the interface can move towards the equilibrium position (Fig. 4(c)) in a stable manner.

We use the same physical parameters as in the previous test case, but the background mesh, before any refinements, now consists of 10 \( \times \) 10 bilinear elements. The elements adjacent to the front are refined with a prescribed number of refinement levels, using a sub-partition of two elements in each direction, and then we smooth the refinement to produce a 1-irregular mesh. We have performed 6 computations, using from 0 to 5 refinement levels. The minimum element size will then vary from 0.1 down to 0.003125. The time step is adjusted during the simulation to produce a maximum local Courant number close to unity. The absence of the hydrostatic boundary conditions implies that the strategy from the previous section, of using air values for the material properties in the transition elements, is not necessary in this case. The material properties are instead taken according to the continuous finite element interpolation.

In Fig. 4, we show a snapshot of the mesh from the beginning of the computation, using four refinement levels. Table 1 shows the convergence results, that is, the deviation between the expected and the computed steady water level. From these results we can conclude that the front computation method shows a satisfactory mass conservation property of the individual phases, which can be controlled by local mesh refinement.

The convergence to an equilibrium state in a closed box is far more challenging when we initially have heavy fluid over light fluid. In a closed box, as above, we start with the unstable equilibrium position, where salt water is on top of fresh water. The ratio of the viscosities equals unity, and 1.05 is the ratio of the densities. The

![Fig. 3. Results from the simulation in Section 5.1. (a) The computed pressure field. Notice that the scale is focused on the air pressure distribution. (b) The resulting velocity field. (c) The water/air interface.](image)

![Fig. 4. From the computation in Section 5.2, using four levels of refinement. (a) The initial Gauss bell shaped interface. (b) The corresponding mesh. (c) The resulting interface.](image)
permeability is constant, and the porosity was set to 0.2. A basic assumption is, of course, that salt and fresh water do not mix. That is, their saturations must be 0 or 1, such that the mathematical model of the present paper is applicable. A somewhat related example, the so-called Elder problem, which is free convection problem with smoothly varying density, is treated by Kolditz et al. [18].

If the initial interface is perfectly flat, the method preserves this unstable equilibrium position. To initiate movement of the fluids, we impose a perturbation of the interface, with the shape of a small semi-disk on top of the horizontal surface, as shown in Fig. 5(a). The simulations make use of a uniform 80 × 80 mesh. Because the shape of the interface becomes very complicated, local mesh refinements will easily cover the whole domain, and it is more efficient from a computational point of view to use a fixed, uniform mesh. Fig. 5(b) through (d) show three snapshots from the simulated process. The length scales resolved in the simulation depend on the element size, and the results from an 80 × 80 grid should only be regarded as a qualitative indication of the formations that appear in this instability problem. A more thorough investigation of instabilities of the present kind requires a more comprehensive set of simulations on very fine grids and is considered as beyond the scope of the present paper. Our main intention with this example was to see if the method is capable of reproducing the main qualitative features of the instability and if there could be a significant mass transfer from the salt water \((F < 0)\) to the fresh water fluid phase \((F > 0)\) when the interface undergoes very complicated deformations. The resulting loss of mass on an 80 × 80 grid is about 10%, which must be considered as satisfactory on background of the chosen (quite coarse) discretization and the complexity of the problem.

### 5.3. A seepage problem

We now apply our numerical model to a common porous media flow problem, namely seepage through a dam or a hill. Fig. 6 shows the porous medium between two lakes. The two-dimensional computational domain is bounded by the thick solid and dashed lines. A prescribed hydrostatic pressure distribution serves as boundary conditions at the sides. First, we solve the seepage problem and establish the stationary fluid flow through the medium. Thereafter, we trace the transport of a concentrated pollutant.

#### 5.3.1. The seepage problem

In the dynamic simulation for obtaining the phreatic surface, the interesting result is usually the final steady state of the flow and the interface. This simulation can therefore be performed by successive computations, using increasing refinement near the front. The initial free surface is taken as a straight line between the two fixed water/air interface points at the boundary. The first computation is performed with the coarse background mesh shown in Fig. 7(b). The ratio of the densities and the viscosities are 1000 and 55, respectively. The porosity is 0.2, and \(K\) is scaled to unity. As concluded in Section 5.1, material properties for air are used in the transition elements, due to the hydrostatic pressure distribution (Dirichlet-type conditions) at the boundaries. When an approximate steady surface is obtained, the computation is stopped.

To make the present flow case more challenging, we have introduced impermeable zones (seen as black areas in Fig. 7(a), (c) and (e). Instead of removing these zones from the grid, we assign a very small value of the permeability to the impermeable areas. This approach is essentially the same as the method of domain imbedding.

---

**Table 1**

Error in the mass of the water phase as a function of different refinement levels

<table>
<thead>
<tr>
<th>Ref. levels</th>
<th>Min. element size</th>
<th>Mass error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1</td>
<td>0.125</td>
</tr>
<tr>
<td>1</td>
<td>0.05</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>0.025</td>
<td>0.0005</td>
</tr>
<tr>
<td>3</td>
<td>0.0125</td>
<td>0.0062</td>
</tr>
<tr>
<td>4</td>
<td>0.00625</td>
<td>0.0031</td>
</tr>
<tr>
<td>5</td>
<td>0.003125</td>
<td>0.0016</td>
</tr>
</tbody>
</table>

---

![Fig. 5. Plots of the interface between fresh and salt water during in a problem that is initially unstable. The fresh water has the darkest grey tone and is the lighter fluid. (a) The initial distribution of fresh and salt water. (b) and (c) Snapshots of the fresh and salt water distribution during the flow. (d) The final, stable equilibrium state with fresh water on top of salt water.](image-url)
[19], but has a clear physical relevance in the present problem. Since the boundary of the impermeable zones do not coincide with the element boundaries, the large jumps in the permeability will give rise to inaccuracies of the same type as those due to jumps in the viscosity and density. Again, local mesh refinements around the low permeable zones can be used to increase the accuracy of the numerical results. The time step is adaptively controlled in order to keep the maximum Courant number close to unity, of previously mentioned reasons.

The steady-state surface, seen in Fig. 7(a), is used as initial condition in the next computation, where the mesh is locally refined one level near the front. The subpartition in this case has four elements in each space

Fig. 6. Seepage problem with two lakes and a dam or a hill in between.

Fig. 7. The steady result for the phreatic surface in Section 5.3.1, using different levels of refinement around the water–air interface: (a) uniform mesh; (c) one level; (e) two levels of refinement. Figures (b), (d) and (f) show the corresponding meshes. The black areas in figures (a), (c), and (e) represent impermeable regions.
direction, producing 3-irregular meshes. The improved phreatic surface and the computational mesh can be seen in Fig. 7(c) and (d). The procedure is repeated for the third computation, where the mesh is refined another level with the same subpartition. The \( F \) field and the mesh are shown in Fig. 7(e) and (f).

Several computations have been performed, using different subpartition and different number of refinement levels. The conclusion seems to be as for the one-dimensional problem: the 1-irregular meshes show the best performance concerning accuracy versus the number of nodes. But again, meshes with larger jumps in the element size, as in the presented computation, can perform quite close to the 1-irregular results and can be significantly computationally faster to establish, due to fewer iterations on the mesh construction.

5.3.2. Contamination transport

Having the stationary seepage flow field, one can study the transport of contaminants. The motion of a contaminant is governed by the same advection equation as the one used for tracking the fluid front, i.e. Eq. (7), if we neglect the effects of hydrodynamic dispersion. Using the object-oriented design of the compound simulator, as presented in Section 3, it is very easy to include the computation of contamination transport. The component simulator for contamination transport is essentially an object of the same class as the front tracker. The additional work is just to slightly adjust the managing simulator, which now must administer three component simulators (of course, the computation of the interface can be turned off if the flow field is considered as steady).

We place a contaminant with a Gaussian distribution in the upstream end of the domain, see Fig. 8(a). A 2-irregular mesh was used in the presented computation. This is an example where it is computationally more effective to use larger jumps in the element size than what is implied by the 1-irregular meshes that dominate in the literature. Fig. 8(b) shows the contaminant field and how it is affected by the impermeable zone. In addition to the refinement around the front, we also refine the entire region covered by the contaminant. Some of the contaminant will leak into the low-permeable zones, because the boundary of the zones intersects the interior of the elements. There will also be some artificial diffusion due to the numerical method. All these effects can be reduced by the mesh refinement. A snapshot of the mesh can be seen in Fig. 8(c). Although we have neglected hydrodynamical dispersion here for simplicity of the implementation, it is of course straightforward to extend the advection equation for the contaminant concentration with a standard model for hydrodynamical dispersion. This can be done, e.g., through a simple variable-coefficient Laplace operator term with a velocity-dependent dispersion parameter. Such a second-order derivative term will normally stabilize the computation of the concentration field.

6. Concluding remarks

We have presented a numerical framework for simulating sharp fluid interfaces in porous media flow. The method employs a unified pressure equation, covering both fluids and an advection equation that implicitly tracks the interface between the fluids. The grid is not following the front, but local mesh refinements are used dynamically around the front to improve the accuracy of the method. Using object-oriented programming techniques, the simulator is quickly realized as a combination of a general finite element Poisson equation solver and a standard finite element advection equation solver.

The verification of the proposed simulation model and its computer implementation is based on three ingredients: (1) the code is carefully examined in a 1D problem for which we can derive a complete analytical
solution, as treated in Section 4, (2) the number of space dimensions is parameterized in the computer code, and (3) the model is tested on multi-dimensional problems for which some qualitative and quantitative features are known. The one-dimensional analysis, together with the space-dimension parameterized code, gives credibility to the implementation, whereas the more challenging test examples reveal some experience with the method and indicate potential difficulties in practical applications. A fundamental assumption is, of course, that the discrete model converges to the solution of the mathematical problem defined in Section 2.

The two-dimensional numerical examples considered interfaces with large jumps in fluid density and viscosity, typically the jumps that appear in water-air systems. When such jumps are located in the interior of finite elements, they make strong demands on the numerics. Satisfactory robustness was obtained by defining an element as being air if at least one integration point is in the air phase. The need for this crude, but stable, approximation is dependent on the type of boundary conditions. Smaller jumps in the coefficients, e.g., the jumps encountered in saltwater intrusion problems, are easily handled by using standard finite element interpolation of material properties inside the elements. In one flow case, large interior jumps in the permeability were also present. It is clear that a more detailed analysis of the effects of letting the interface intersect the elements, and thereby introducing large jumps in the coefficients of the governing partial differential equation in the interior of elements, needs consideration. The scope of the present paper has been to outline a numerical approach and its performance in some porous media flow applications and to point out the advantages and deficiencies of the methodology.

The advantage of the present numerical method, compared with previous approaches to the seepage problem in the literature, is the simple, implicit handling of the interface between two fluids, as well as the straightforward implementation of the method, at least if a suitable finite element library is available. Contrary to most other methods for the seepage problem, which neglect the motion of air, the present method requires computation of the motion of both fluid phases. Hence, some computational efficiency is sacrificed for increased flexibility, but our method is of course much more efficient than a complete two-phase model with variable interface [12]). Furthermore, stable time integration of this auxiliary PDE is a nontrivial task, but promising steps have recently been taken in this direction [13].

The presented numerical model is limited to flow problems with sharp interfaces. Capillary pressure effects cannot be modeled, as the fundamental continuity assumption on the pressure across the interface is then no longer valid. This limits the application areas. For example, oscillatory movement of the phreatic surface due to pumping can be influenced by capillary effects.

Since our implementation of the present method works in 1D, 2D and 3D, we have also run three-dimensional experiments. Besides significantly longer CPU times, the main experience from the 3D cases confirm the general conclusions about the method from the 2D cases shown in detail herein. The method can easily be generalized to more complicated mathematical models. For example, we have applied a similar method to injection molding problems, where the momentum equation is a highly nonlinear Poisson equation [10].

Acknowledgements

The work has been financed by the Research Council of Norway (NFR) under grant 100640/410.

References


[16] Langtangen HP, Munthe O. Simulation of generalized Newtonian pipelie flow with thermal effects using object-oriented techniques, to be submitted.


