Front Tracking for Two-Phase Flow in Reservoir Simulation by Adaptive Mesh

Mazen Saad, Huilong Zhang
Mathématiques Appliquées de Bordeaux
Université de Bordeaux I
351 cours de la Libération
33405 Talence Cedex, France

Received May 10, 1996; accepted December 13, 1996

In this article, an algorithm for the numerical approximation of two-phase flow in porous media by adaptive mesh is presented. A convergent and conservative finite volume scheme for an elliptic equation is proposed, together with the finite difference schemes, upwind and MUSCL, for a hyperbolic equation on grids with local refinement. Hence, an IMPES method is applied in an adaptive composite grid to track the front of a moving solution. An object-oriented programmation technique is used. The computational results for different examples illustrate the efficiency of the proposed algorithm. © 1997 John Wiley & Sons, Inc. Numer Methods Partial Differential Eq **13:** 673–697, 1997

Keywords: porous media; adaptive mesh; MUSCL scheme; OOP

I. INTRODUCTION

Many of the physical phenomena that govern enhanced recovery processes have extremely important local properties. Thus, the models used in simulators of these problems must be able to solve these critical local features. Also, in order to be useful in large-scale dynamic simulators, these models must be self-adaptive and extremely efficient. The development of adaptive grid refinement techniques must take into account the rapid development of advanced computer architectures and the use of performance language programming.

In this article, the method of secondary recovery (waterflooding) is modelized, it is used in many enhanced recovery processes: fluids are injected into some wells in a reservoir while the resident hydrocarbons are produced from other wells, designed production wells. The two-phase incompressible flow, without considering capillary effects, is governed by a system coupling an elliptic equation (pressure equation) and an hyperbolic equation (saturation equation).

As one fluid displaces the others, the physical processes that occur along the moving interface between the fluids govern the effectiveness of the process.

 CCC 0749-159X/97/060673-25

In order to model these important extremely localized phenomena, a grid spacing that is more nearly of the order of the process must be used.

For transport partial differential equations, sharp fronts move along characteristic or near characteristic directions. Therefore, the computed velocity, obtained by resolution of the pressure equation, determines the location of the regions where local refinement will be needed at the next time step.

The interaction between the fluids leads to a sharp front of arbitrary shape, which generally cannot be predicted in advance. This makes useless the well-known patch refinement techniques ([1, 2]); the concept does not require as complex a data structure but involves ideas of passing information from one uniform grid to another.

For an evolutive local refinement, we propose an appropriate data structure permitting us to generate a general composite grid. The choice of this composite grid and the different type of cells allow the refinement of a small area, which tracks accurately the moving front, since efficiency is crucial in large-scale reservoir simulation. An object-oriented programming language, C++, is selected for the implementation [3]. It provides dynamic memory management, and a convenient data structure for evolutive problems.

In Section II we give a brief presentation of the mathematical model. The numerical schemes used are presented in Section III. First, a finite volume method to discretize the elliptic equation on a composite grid (the flux between two neighboring cells is discretized by finite differences) is proposed. This approximation is stable, conservative, and convergent. It enables us to check flux continuity at the interface. Next, finite difference schemes are presented to approach the hyperbolic equation. We show that the upwind scheme on a composite grid ensures the positivity of the saturation, and we give the algorithm to implement a MUSCL scheme. In Section IV, we give an algorithm for solving two-phase flow by an adaptive mesh. The data structure is presented in Section V, and the numerical results are presented in the last section.

II. FORMULATION

Fluid flow in a homogeneous porous medium is governed by a system of nonlinear partial differential equations. The basic equations used for two-phase flow consist of conservation equations of oil and water. The phase velocity is given by Darcy's law [4].

Let $\Omega =]0, L[\times]0, l[$ denote the domain of the study, with boundary $\Gamma = \partial\Omega$, which is divided as follows:

$$\Gamma = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{imp}$$

and $\Gamma_a \cap \Gamma_b = \emptyset$ if $a \neq b$.

Γ_{in} : water injection boundary,
Γ_{out} : fluid production boundary,
Γ_{imp} : impervious boundary.

The following notations are used in the sequel:

w : water phase
o : oil phase
η : η phase, $\eta = w, o$
$S_\eta(x, y, t)$: saturation of the η phase, $(0 \leq S_\eta \leq 1)$
$\phi(x, y)$: porosity
$\mathbf{K}(x, y)$: permeability tensor of the porous medium
$kr_\eta(S_\eta)$: relative permeability of the η phase
$P(x, y, t)$: pressure in the medium (no capillary pressure)
μ_η : viscosity of the η phase
$M_\eta(S_\eta)$: mobility of the η phase $(M_\eta = k_\eta/\mu_\eta \geq 0)$
$M(S_\eta)$: total mobility $(M = M_w + M_o > 0)$
$f_\eta(S_\eta)$: fractional flow of the η phase $(f_\eta(S_\eta) = M_\eta(S_\eta)/M(S_\eta), 0 \leq f_\eta \leq 1)$
$\mathbf{V}_\eta(x, y, t)$: velocity of the η phase
$\mathbf{V}(x, y, t)$: total velocity $(\mathbf{V} = \mathbf{V}_w + \mathbf{V}_o), \mathbf{V} = (u, v)^T$.

For an incompressible flow, where capillary and gravitational effects are negligible, the equations for two-phase flow are given by:

Mass conservation of oil:

$$\phi(x, y)\partial_t S_o(x, y, t) + div(\mathbf{V}_o(x, y, t)) = 0. \tag{1}$$

Mass conservation of water:

$$\phi(x, y)\partial_t S_w(x, y, t) + div(\mathbf{V}_w(x, y, t)) = 0. \tag{2}$$

By definition of the saturations, one gets

$$S_w(x, y, t) + S_o(x, y, t) = 1. \tag{3}$$

The *Darcy velocity* is defined as follows:

$$\mathbf{V}_o = -\mathbf{K}\frac{kr_o(S_o)}{\mu_o}\,\nabla P, \mathbf{V}_w = -\mathbf{K}\frac{kr_w(S_w)}{\mu_w}\,\nabla P. \tag{4}$$

The porous media is considered to be homogeneous, then without loss of generality, we take

$$\phi(x, y) = \phi, \text{ and } \mathbf{K}(x, y) = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix},$$

where ϕ, k_x, and k_y are positive.

By these formulations, we can see that the unknown functions are the saturation of one phase (e.g., water saturation) and pressure. In order to obtain a nondegenerate pressure equation, we express the total velocity of the flow depending on P; Eqs. (1) and (2) are added together, and, following by (3), we have

$$div(\mathbf{V}_o + \mathbf{V}_w) = div(\mathbf{V}) = div(-\mathbf{K}M\nabla P) = 0, \tag{5}$$

which is an elliptic equation.

Using the notation of total velocity, the phase velocity can be expressed as follows:

$$\mathbf{V}_\eta(x, y, t) = f_\eta(S_\eta)\mathbf{V}(x, y, t), \text{ where } f_\eta(S_\eta) = \frac{M_\eta(S_\eta)}{M(S_\eta)}. \tag{6}$$

To simplify our notation, we shall write S for S_w and $f(S)$ for $f_w(S_w)$. Then, the problem (1)–(4) is equivalent to finding (P, S) a solution of the following nonlinear partial differential

system:

$$\begin{cases} div(-\mathbf{K}M(S)\nabla P) = 0 \\ \phi\partial_t S + div(\mathbf{V}f(S)) = 0 \\ \mathbf{V} = -\mathbf{K}M(S)\nabla P \\ 0 \le S \le 1. \end{cases} \tag{7}$$

In petroleum engineering, the function $S \to f(S)$ is increasing and satisfies $0 \le f(S) \le 1$ for every $S \in [0,1]$, and the total mobility $M(S)$ is positive. For example, the Corey model ([4, 5]) with square relative permeabilities ($k_w = S^2, k_o = (1-S)^2$) gives

$$f(S) = \frac{\frac{S^2}{\mu_w}}{\frac{S^2}{\mu_w} + \frac{(1-S)^2}{\mu_o}}, M(S) = \frac{S^2}{\mu_w} + \frac{(1-S)^2}{\mu_o}.$$

In order to deal separately with the two different mathematical equations of this system, a splitting method is used, which leads to solving an elliptic and hyperbolic equations. It is based on an IMPES method ([4–6]) on a composite grid. This algorithm is described in Section IV.

For more clarity, we are interested in discretizing two prototypical problems:

- **Elliptic problem.** For $M(x,y)$ a continuous and positive function in Ω finds P, which is a solution of

$$\begin{cases} -div(M(x,y)\nabla P) = 0 & \text{in } \Omega \\ \nabla P \cdot \mathbf{n} = 0 & \text{on } \Gamma_{imp} \\ P = P_{in} & \text{on } \Gamma_{in} \\ P = P_{out} & \text{on } \Gamma_{out}. \end{cases} \tag{8}$$

  Here $\mathbf{n}$ is the outward unit normal vector defined on Γ.
- **Hyperpolic problem.** For $\mathbf{V}(x,y)$ a given vector in Ω that satisfies $div\mathbf{V} = 0$ finds S, which is a solution of

$$\begin{cases} \partial_t S + div(\mathbf{V}f(S)) = 0 & \text{in } \Omega \\ S(x,y,t) = S_{in} & \text{on } \Gamma_{in} \\ S(x,y,0) = S_0(x,y) & \text{on } \Omega, \end{cases} \tag{9}$$

  where f has $C^1[0,1]$ regularity and is an increasing function.

III. DISCRETIZATION OF THE PROBLEM

In this section we will discuss a discretization scheme of Eqs. (8) and (9) on a composite mesh. The construction of a cell-related composite mesh is presented. We apply the finite volume method to get a convergent and conservative scheme for the elliptic Eq. (8); we also present a monotone and a consistent finite difference scheme to approach the hyperbolic Eq. (9). Numerical tests are presented, in Section VI, for simple problems to validate proposed schemes.

A. Composite Mesh

For the adaptive mesh refinement technique, the cells are managed by means of algorithms such as refinement and clustering. These procedures, which manipulate the mesh on the basis of clearly defined criteria, directly influence the number of grid blocks and, hence, the neighborhood of certain cells. It appears clear that if the number of neighbors for a cell is too large, the writing of

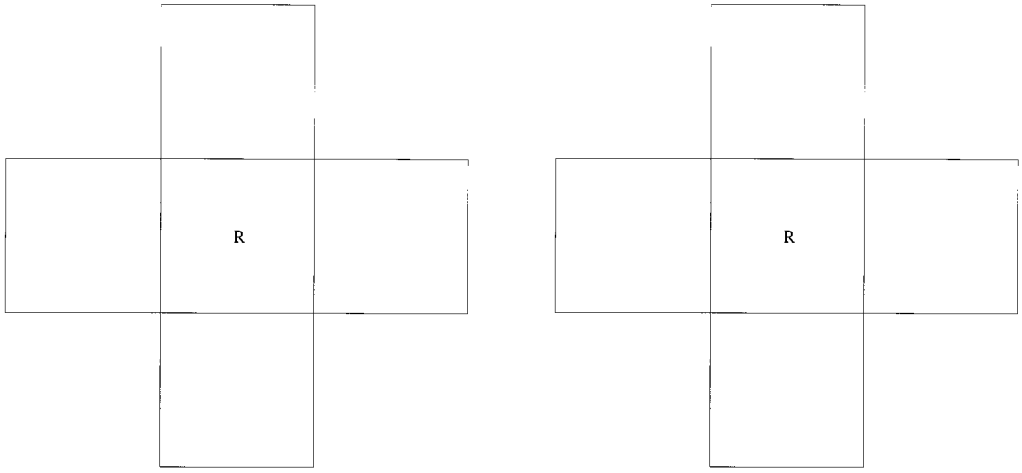

FIG. 1. Regular **R,** coarse fine **CF,** fine coarse **FC,** middle **MD,** and corner **CN.**

the scheme will be difficult. A great rate of refinement between two neighbors will deteriorate discretization precision. To avoid the occurrence of such cases, a certain refinement rule is established at the end of this subsection.

The cell-related discretization scheme detailed in the next subsection is based on a finite volume method. Refinement is performed by dividing a grid cell into four cells having the same volume, in other words, the size of the cell in each direction is divided by two. The geometric configuration of a cell in the composite mesh can sometimes be very sophisticated. (Figure 2 is an example of a 3-level composite mesh.)

The level of a cell is the level of refinement to which it belongs; coarse grid cells are allocated to level 1. A cell is identified by its geometric position (i.e., the coordinate of its barycenter), its four edges, its size, and its level. It is surrounded by several cells, called neighbors. Precisely, two cells are said to be neighbors if they have an edge in common. Sometimes, one or two of these neighbors can reduce to a boundary of the discretization domain.

Let's proceed to a classification for every cell in a composite mesh. Given a cell of level l, five classes are distinguished as follows:

- **Regular (R).** The most simple cell. The neighbors in 4 directions (some of these directions can reduce to a boundary) are in the same level as it.
- **Coarse fine (CF).** A bigger cell in the interface of two levels, i.e., at least one neighbor is of level $l + 1$, the others of level l.
- **Fine coarse (FC).** A finer cell in the interface of two levels, i.e., at least one neighbor is of level $l - 1$, the others of level l.
- **Middle (MD).** An interface cell among 3 levels: $l-1, l$, and $l+1$, i.e., at least one neighbor is of level $l + 1$, and one neighbor is of $l - 1$.
- **Corner (CN).** Two neighbors are of **MD** type cells and the other neighbors are of **FC.**

These classes are shown in Fig. 1.
The cells of classes **CF, FC, MD,** and **CN** find themselves only in the interfaces of different refinement regions. Particularly, **MD** and **CN** appear only when the level of composite grids is superior or equal to 3. For a given composite mesh of level l, the refinement rule for the construction to a composite mesh of level $l + 1$ is that only the regular cell of level l can be

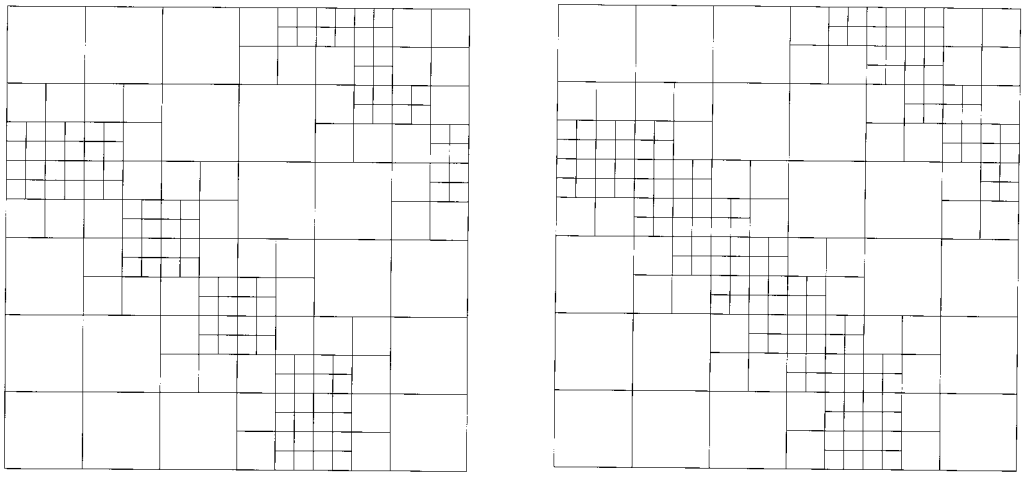

FIG. 2. The roll of **MD** and **CN** type cells. The left composite mesh is obtained without these types of cells, while the right one uses them.

refined. An artificial rule can be imposed, so that the last two types disappear. This will make the implementation more simple, but will restrict considerably the geometrical capacity to represent complex refinement regions, especially when the first level is very coarse (see Fig. 2).

B. Discretization of the Elliptic Equation on a Composite Grid

We use a cell-centered grid to discretize the computational domain and a finite volume method to approach the elliptic problem (8). The reasons for this choice include its ability to be faithful to the physical situation in general and conservation in particular.

Now, let's recall the approximation of Eq. (8). Suppose that C is a given regular cell called *control volume* in Ω with boundary Γ_C as showed in Fig. 3. Using the divergence theorem

$$\int_C div(M\nabla P)d\omega = \int_{\Gamma_C} M\nabla P \cdot \mathbf{n} d\gamma = fl(C) = 0, \tag{10}$$

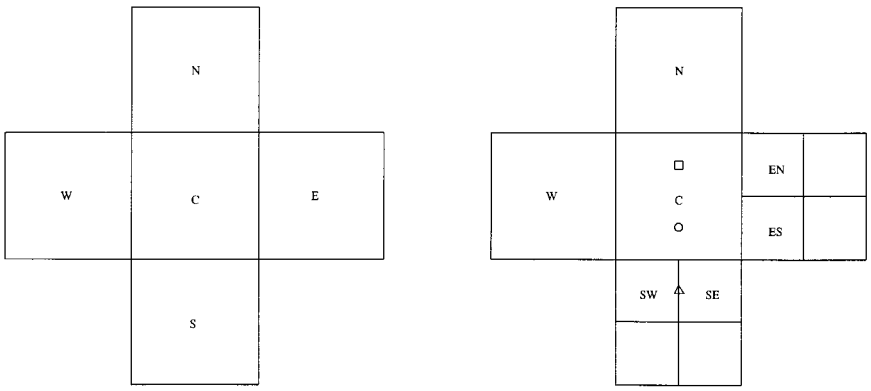

FIG. 3. Regular and coarse fine cells.

where $fl(C)$ is a sum of four flux corresponding to four edges of the cell.

Equation (10) can thus be interpreted as a conservation law for volume C, which states that the net flow rate of the fluid across the surface Γ_C balances with the net flow rate from the interior source, which is zero.

Let's denote by capital letters S, W, E, and N (resp. small letters s, w, e, and n) the four neighbors (resp. the four edges) of the regular cell C. They also designate the center of the cell (resp. the middle of the edge), e.g., for a given function Φ,

$$\begin{aligned} \Phi_N &: \text{ Value of } \Phi \text{ at the point } N, \\ \Phi_n &: \text{ Value of } \Phi \text{ at the middle of the edge } n. \end{aligned}$$

The total flux $fl(C)$ is then written:

$$fl(C) = fl_e(C) + fl_w(C) + fl_n(C) + fl_s(C), \tag{11}$$

where $fl_i(C)$ refers to the flux across the edge i of C $(i = s, w, e, n)$. In other words,

$$fl_i(C) = \int_{\Gamma_i(C)} M \nabla P \cdot \mathbf{n} d\gamma_i.$$

As above, $M_i, i = s, w, e, n$ represents the value of the mobility at the middle of the common edge between cell C and its neighbors. These flux can be approached by

$$fl_i(C) = M_i \int_{\Gamma_i(C)} \nabla P \cdot \mathbf{n} d\gamma_i.$$

The gradient ∇P will be discretized by finite difference. Written in stencil form, a regular interior cell has a 5 point scheme:

$$\begin{pmatrix} & -M_n & \\ -M_w & -\Sigma & -M_e \\ & -M_s & \end{pmatrix} P_C = 0. \tag{12}$$

We denote by $\mathbf{V}_i = (u_i, v_i)^T$ for $i = s, e, n, w$, the average velocity at the common face between the cell C and its neighbors. The scheme (12) is equivalent to the discretization of $div\mathbf{V} = 0$ with $\mathbf{V} = -M\nabla P$, then we have the relation

$$\frac{u_e - u_w}{h} + \frac{v_n - v_s}{h} = 0, \tag{13}$$

where

$$u_e = -M_e \frac{P_E - P_C}{h}, u_w = -M_w \frac{P_C - P_W}{h}, v_n = -M_n \frac{P_N - P_C}{h}, v_s = -M_s \frac{P_C - P_S}{h}.$$

The discretization for the other types of cell is more laborious. We will describe our scheme, without loss of generality, for the case of a coarse fine cell depicted in Fig. 3.

In the same way, $M_i, i = w, n, sw, se, en$, and es denote the value of the mobility at the interface between the cell C and its neighbors. The cells in the north and west directions being nonrefined, the expression of flux across the north and west edges is exactly the same as in the case of regular cells, that is,

$$fl_n(C) = M_n(P_N - P_C) \text{ and } fl_w(C) = -M_w(P_C - P_W).$$

To obtain a convenient expression for the east flux, we assume that the pressure is a linear function between the cell C and its neighbors. Moreover, to ensure the total flux conservation at the east face of the cell C, we impose

$$fl_e(C) = -fl_w(EN) - fl_w(ES),$$

where, $fl_w(EN)$ [resp. $fl_w(ES)$] designates the flux across the west edge of the cell *EN* (resp. *ES*).

Some slave nodes obtained by linear interpolation are then introduced to compute the last two flux. Precisely, to calculate the flux $fl_w(EN)$, a slave node named $P_\square$ is defined by

$$P_\square = \frac{3}{4}P_C + \frac{1}{4}P_N$$

so that

$$fl_w(EN) \simeq -M_{en}\frac{P_\square - P_{EN}}{\frac{3h}{4}}\ \frac{h}{2} = -M_{en}\left(\frac{1}{2}P_C + \frac{1}{6}P_N - \frac{2}{3}P_{EN}\right).$$

To approach the flux $fl_w(ES)$, two slave nodes $P_\triangle$ and $P_\circ$ are considered:

$$P_\triangle = \frac{1}{2}P_{SW} + \frac{1}{2}P_{SE}, \text{ and } P_\circ = \frac{2}{3}P_C + \frac{1}{3}P_\triangle = \frac{2}{3}P_C + \frac{1}{6}P_{SW} + \frac{1}{6}P_{SE};$$

therefore,

$$fl_w(ES) \simeq -M_{es}\frac{P_\circ - P_{ES}}{\frac{3h}{4}}\ \frac{h}{2} = -M_{es}\left(\frac{4}{9}P_C + \frac{1}{9}P_{SW} + \frac{1}{9}P_{SE} - \frac{2}{3}P_{ES}\right).$$

Similarly, with some other slave nodes, we have

$$\begin{aligned} fl_s(C) &= -fl_n(SW) - fl_n(SE) \\ &\simeq -M_{sw}\left(\frac{1}{2}P_C + \frac{1}{6}P_W - \frac{2}{3}P_{SW}\right) - M_{se}\left(\frac{4}{9}P_C + \frac{1}{9}P_{EN} + \frac{1}{9}P_{ES} - \frac{2}{3}P_{SE}\right). \end{aligned}$$

Finally, the stencil for this coarse fine type cell can be written as

$$\begin{pmatrix} & -M_n + \frac{1}{6}M_{en} & \\ -M_w + \frac{1}{6}M_{sw} & -\Sigma & -\frac{2}{3}M_{en} + \frac{1}{9}M_{se} \\ & \frac{1}{9}M_{es} - \frac{2}{3}M_{sw} \quad \frac{1}{9}M_{es} - \frac{2}{3}M_{se} & -\frac{2}{3}M_{es} + \frac{1}{9}M_{se} \end{pmatrix} P_C = 0. \quad (14)$$

In same way to obtain (13), we denote by $\mathbf{V}_i = (u_i, v_i)^T$, the average velocity at the common face between the cell C and its neighbors, for $i = n, w, en, es, sw, se$. The discretization of $div\ \mathbf{V} = 0$ leads to (14), if we set

$$\frac{\frac{u_{en}+u_{es}}{2} - u_w}{h} + \frac{v_n - \frac{v_{sw}+v_{se}}{2}}{h} = 0 \quad (15)$$

with

$$u_{en} = -\frac{M_{en}}{h}\left(\frac{4}{3}P_{EN} - P_C - \frac{1}{3}P_N\right), u_{es} = -\frac{M_{es}}{h}\left(\frac{4}{3}P_{ES} - \frac{8}{9}P_C - \frac{2}{9}P_{SW} - \frac{2}{9}P_{SE}\right),$$

$$v_{sw} = -\frac{M_{sw}}{h}\left(P_C + \frac{1}{3}P_W - \frac{4}{3}P_{SW}\right), v_{se} = -\frac{M_{se}}{h}\left(\frac{8}{9}P_C + \frac{2}{9}P_{EN} + \frac{2}{9}P_{ES} - \frac{4}{3}P_{SE}\right),$$

$$u_w = -M_w\frac{P_C - P_W}{h}, v_n = -M_n\frac{P_N - P_C}{h}.$$

The surface of the boundary control volumes coincide with the boundary of the computational domain, so the boundary condition for problem (8) is easily incorporated into their equations, and conservation for the entire domain is, therefore, assumed. For example, if the west side of the cell C coincides with the boundary domain, we simply let $f_w(C) = 0$ to impose a homogeneous Neumann condition or $f_w(C) = \frac{1}{2}M_w(P_C - P_{imposed})$ to impose a Dirichlet condition.

The proposed scheme is conservative by its construction. Several authors have studied the properties of finite volume schemes in the case of composite mesh ([1, 7]). It is obvious that the scheme is not consistent. Nevertheless, one can prove the convergence of this scheme by using the same method in ([7, 8]). In particular, one gets that the order of convergence is locally 1 (resp. 2) for a interface cell (resp. regular cell).

C. Discretization of the Hyperbolic Equation on a Composite Grid

In this section, finite difference schemes are used to discretize the hyperbolic Eq. (9) on a composite grid. We suppose that the velocity field is given by the solution of (8). The saturation of water in an oil reservoir may have jump discontinuities of several orders of magnitude on the front of contact between these two fluids; this discontinuity is caused by relative permeabilities and the viscosity rate.

The scheme widely used for this kind of hyperbolic equation in petroleum engineering is an upwind scheme; it is based on decentering of mobility of each phase according to its own velocity (see Aziz et al. [4], Forsyth [5], Liu et al. [6], Hermitte [2]). This scheme, which is of first order, presents a principal undesirable behavior: numerical diffusion. This phenomenon increases near the shock fonts. A possible improvement can be obtained by using a second-order scheme such as slope-limiter and flux-limiter schemes (Yee [9], Roe's second-order scheme) or an anti-diffusion scheme (Harten [10]).

It is known, in general, that a higher order scheme becomes first order on a composite mesh. Hence, we propose an upwind scheme based not on mobility but on fractional flux. It is assumed that the fluid speed propagation is the total velocity, and this leads to a simple hyperbolic equation. We will prove that the proposed scheme is stable under an appropriate CFL condition and is of first order. In order to diminish the numerical diffusion introduced by the upwind schemes, we propose then a MUSCL method (see Van Leer ([11, 12]), Randall et al. [13]). As we will see in Sections IV and VI, the front tracking method allows a local refinement near the shocks by a certain criterion, which reduces considerably the numerical diffusion.

Consider the hyperbolic equation

$$\partial_t S + \mathbf{V}\cdot\nabla f(S) = \partial_t S + u\partial_x f(S) + v\partial_y f(S) = 0, \tag{16}$$

where $\mathbf{V}(x,y) = (u(x,y), v(x,y))^T$ is a given function satisfying

$$div\,\mathbf{V} = 0,$$

and f is a monotone increasing function.

Let Ω_h^k be a composite grid and S_h^k a saturation vector on this grid. Our goal is to establish an explicit finite difference scheme to obtain S_h^{k+1} in the same grid. The construction of the composite grid Ω_h^k at the time t^k is described in the Section IV. Here, we suppose that Ω_h^k and

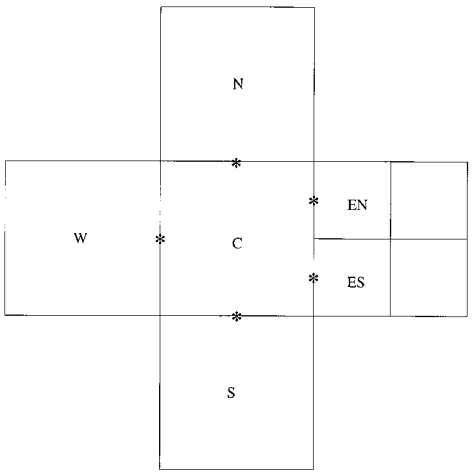

FIG. 4. General composite grid for two-level refinement. (*) represent the points where piecewise linear solution do not create a local extremum for the MUSCL scheme.

S_h^k are given. Denote S_C^k a component of S_h^k representing an approximation of the exact solution associated to the cell C at time t^k.

The Euler time scheme implies

$$\frac{S^{k+1} - S^k}{\Delta t} + \mathbf{V} \cdot \nabla f(S^k) = 0. \tag{17}$$

In order to present the discretization of Eq. (17), we consider, without loss of generality, a coarse-fine cell depicted in Fig. 4.

The average of velocity $\mathbf{V}_i = (u_i, v_i)^T$ $(i = s, w, es, en, n)$ is calculated in the same way as in (15).

• **Upwind scheme.** The approximate solution is considered to be piecewise constant. The second term of Eq. (17) can be considered as a sum of four quantities corresponding to contribution of each side. We approach the flux gradient across the south, west, and north edges by the upwind scheme usually used in the literature ([10, 14, 15]). Hence,

$$\begin{aligned}\frac{S_C^{k+1} - S_C^k}{\Delta t} &+ F_s(C) + F_n(C) + F_w(C) + F_e(C) \\ &= \frac{S_C^{k+1} - S_C^k}{\Delta t} + v_s^+ \frac{f_C^k - f_S^k}{h} + v_n^- \frac{f_N^k - f_C^k}{h} + u_w^+ \frac{f_C^k - f_W^k}{h} + F_e(C) = 0, \end{aligned} \tag{18}$$

where

$$\alpha^+ = \frac{\alpha + |\alpha|}{2}, \alpha^- = \frac{\alpha - |\alpha|}{2} \text{ for } \alpha \in \mathbf{R}.$$

There are two contributions corresponding to the east-north (EN) and the east-south (ES) cells so that the east flux $F_e(C)$ is expressed as follows:

$$F_e(C) = \frac{1}{2}\left(u_{en}^- \frac{f_{EN}^k - f_C^k}{\frac{3}{4}h} + u_{es}^- \frac{f_{ES}^k - f_C^k}{\frac{3}{4}h} \right). \tag{19}$$

Remark 3.1. In the case of regular cell, the scheme (18)–(19) is reduced to

$$\frac{S_C^{k+1}-S_C^k}{\Delta t}+v_s^+\frac{f_C^k-f_S^k}{h}+v_n^-\frac{f_N^k-f_C^k}{h}+u_w^+\frac{f_C^k-f_W^k}{h}+u_e^-\frac{f_E^k-f_C^k}{h}=0.$$

By definition of $v^\pm = v \mp v^\mp$ and $u^\pm = u \mp u^\mp$, the above equation can be rewritten as

$$\frac{S_C^{k+1}-S_C^k}{\Delta t}+\frac{v_n^+f_C^k+v_n^-f_N^k-(v_s^+f_S^k+v_s^-f_C^k)}{h} + \frac{u_e^+f_C^k+u_e^-f_E^k-(u_w^+f_W^k+u_w^-f_C^k)}{h}+f_C^k\left(\frac{v_n-v_s}{h}+\frac{u_e-u_w}{h}\right)=0. \quad (20)$$

From (13), the discretization of $(div\ \mathbf{V}=0)$ on a regular cell implies that the last term in (20) is zero. So, this scheme is identical to the scheme proposed in [15], which guarantees the conservation and consistency. Moreover, when $u = v = constant$, the scheme coincides with Murmann–Roe's scheme [14].

Now, we discuss some properties of scheme (18)–(19), corresponding to the general cell C in Fig. 4.

The corresponding scheme is

$$\frac{S_C^{k+1}-S_C^k}{\Delta t}+v_s^+\frac{f_C^k-f_S^k}{h}+v_n^-\frac{f_N^k-f_C^k}{h}+u_w^+\frac{f_C^k-f_W^k}{h} + \frac{2}{3}u_{en}^-\frac{f_{EN}^k-f_C^k}{h}+\frac{2}{3}u_{es}^-\frac{f_{ES}^k-f_C^k}{h}=0. \quad (21)$$

Let's define

$$\alpha_i=\begin{cases}1 & \text{if } i=s,w,n\\ \frac{2}{3} & \text{if } i=es,en\end{cases}$$

and

$$D_i^\pm=\begin{cases}\mp\alpha_i v_i^\pm\frac{\Delta t}{h}\frac{f_C^k-f_I^k}{S_C^k-S_I^k} & \text{if } S_C^k-S_I^k\neq 0\\ \mp\alpha_i v_i^\pm\frac{\Delta t}{h}f'(S_C) & \text{otherwise}\end{cases}$$

in this definition, when $i = s, w, n, es, en$ then $I = S, W, N, ES, EN$.

One observes that $D_i^\pm$ is always positive, because the flux f is an increasing function.

With the above notations, the following equation results:

$$S_C^{k+1}=S_C^k-D_s^+(S_C^k-S_S^k)-D_n^-(S_C^k-S_N^k)-D_w^+(S_C^k-S_W^k) - D_{en}^-(S_C^k-S_{EN}^k)-D_{es}^-(S_C^k-S_{ES}^k). \quad (22)$$

Proposition 3.2. *Under the following CFL condition*,

$$D_s^+ + D_n^- + D_w^+ + D_{en}^- + D_{es}^- \le 1, \quad (23)$$

the scheme (22) *is* L^∞*-stable*.

Moreover, *the scheme is of first order*.

Proof. Equation (22) can be written equivalently as

$$S_C^{k+1}=(1-D_s^+-D_n^--D_w^+-D_{en}^--D_{es}^-)S_C^k + D_s^+S_S^k+D_n^-S_N^k+D_w^+S_W^k+D_{en}^-S_{EN}^k+D_{es}^-S_{ES}^k.$$

Since $D_i^\pm \ge 0$, and, by taking into account the CFL condition (23), we have

$$
\begin{aligned}
|S_C^{k+1}| \;\le\; & (1 - D_s^+ - D_n^- - D_w^+ - D_{en}^- - D_{es}^-)|S_C^k| \\
& + D_s^+|S_S^k| + D_n^-|S_N^k| + D_w^+|S_W^k| + D_{en}^-|S_{EN}^k| + D_{es}^-|S_{ES}^k| \\
\le\; & (1 - D_s^+ - D_n^- - D_w^+ - D_{en}^- - D_{es}^-) \max_{j\in J} |S_J^k| \\
& + (D_s^+ + D_n^- + D_w^+ + D_{en}^- + D_{es}^-) \max_{i\in J} |S_j^k|,
\end{aligned}
$$

where J represents the set of cells in the composite grid. ∎

The property of the L^∞ stability is a consequence of

$$|S_C^{k+1}| \le \max_{j\in J} |S_j^k|, \forall C \in J.$$

To prove the consistency of the scheme, take $V = (-1, 1)$ to simplify and denote by $S(x, y, t) = S$ the exact solution at cell C. Then, the finite difference operator associated to the scheme (21) is defined by

$$
\begin{aligned}
L_{h,\Delta t} S(x, y, t) \;=\; & \frac{S(x, y, t + \Delta t) - S(x, y, t)}{\Delta t} + \frac{f(S(x, y, t)) - f(S(x, y - h, t))}{h} \\
& - \frac{1}{2} \frac{f(S(x + \frac{3}{4}h, y + \frac{1}{4}h, t)) - f(S(x, y, t))}{\frac{3}{4}h} \\
& - \frac{1}{2} \frac{f(S(x + \frac{3}{4}h, y - \frac{1}{4}h, t)) - f(S(x, y, t))}{\frac{3}{4}h}.
\end{aligned}
$$

By a simple Taylor expansion, we have

$$\frac{S(x, y, t + \Delta t) - S(x, y, t)}{\Delta t} = \partial_t S + O(\Delta t)$$

$$\frac{f(S(x, y, t)) - f(S(x, y - h, t))}{h} = \partial_y f(S) + O(h)$$

and

$$\frac{f(S(x + \frac{3}{4}h, y + \frac{1}{4}h, t)) - f(S(x, y, t))}{\frac{3}{4}h} = \partial_x f(S) + \frac{1}{3}\partial_y f(S) + O(h),$$

$$\frac{f(S(x + \frac{3}{4}h, y - \frac{1}{4}h, t)) - f(S(x, y, t))}{\frac{3}{4}h} = \partial_x f(S) - \frac{1}{3}\partial_y f(S) + O(h).$$

Replacing these expressions by their values in $L_{h,\Delta t}$, one gets

$$L_{h,\Delta t} S(x, y, t) = \partial_t S - \partial_x f(S) + \partial_y f(S) + O(\Delta t) + O(h),$$

which shows the consistency of scheme (21).

- **MUSCL scheme.** The basic idea is to replace piecewise constant representation of the solution by some accurate representation, say piecewise linear.

Methods of this type were first introduced by Van Leer in a series of articles ([12, 11], ...), where he develops the MUSCL scheme (standing for Monotonic Upstream-centered Scheme for Conservations Laws). The slope-limiter method is well known for a nonlinear scalar equations in one-dimensional space, which leads to TVD (Total Variation Diminishing) and stable schemes. These methods are generalized to nonlinear hyperbolic systems ([11, 13]).

In two-dimensional space and on an irregular grid, the MUSCL methods are more difficult to put in place. We now give an algorithm of the MUSCL schemes employed on our composite grid. The proposed scheme is based on a gradient-limiter method.

Give data $\{S_C^k\}_C$ a vector on the composite grid Ω_h^k. We replace the piecewise constant solution by a more accurate reconstruction, taking the piecewise linear function

$$\tilde{S}_X^k = S_C^k + Q_C^k \cdot \vec{CX},$$

where Q_C^k is an approximation of the gradient on the cell C, which is based on the data $\{S_C^k\}_C$.

The most interesting question is, how do we choose the gradients? A classical way is to limit the gradients Q_C^k by a coefficient γ_C^k with $0 \leq \gamma_C^k \leq 1$. This coefficient is chosen such that the saturations on the interfaces, precisely where the flux will be evaluated, do not create a local extremum with respect to the saturations of neighbor cells.

Denote by S_{max} (resp. S_{min}) the highest (resp. lowest) admissible value of the saturation on the interface, and by $\tilde{S}_X^k$ the saturation of X on an interface of cell C: $\tilde{S}_X^k = S_C^k + \gamma_C^k Q_C^k \cdot \vec{CX}$. The algorithm to compute γ_C^k is as follows: Choose $\gamma_C^k = 1$. In the case where $\tilde{S}_X^k > S_{max}$, the gradient of the cell C is then limited by $\gamma_C^k = \frac{S_{max} - S_C}{\tilde{S}_X^k - S_C}$, and if $\tilde{S}_X^k < S_{min}$ then $\gamma_C^k = \frac{S_{min} - S_C}{\tilde{S}_X^k - S_C}$.

We now give the MUSCL scheme for the situation in Fig. 4. A natural approximation of gradient of cell C is $Q_C^k = (A_C^k, B_C^k)$ with

$$A_C^k = \frac{\frac{S_{EN}^k + S_{ES}^k}{2} - S_W^k}{\frac{7}{4}h}, B_C^k = \frac{S_N^k - S_S^k}{2h}.$$

S_{max} and S_{min} are considered to be

$$S_{max} = \max_I S_I^k, S_{min} = \min_I S_I^k, I = \{CN, W, S, EN, ES\}.$$

The coefficient γ_C^k is limited such that the saturations on the middle of the common faces between the cell C and its neighbors, denoted by * in Fig. 4, remain in the interval $[S_{min}, S_{max}]$.

Finally, to compute the cell averages S_C^{k+1}, the scheme (21) is replaced by

$$\begin{aligned}
\frac{S_C^{k+1} - S_C^k}{\Delta t} &+ v_s^+ \frac{f\left(S_C^k + \frac{h}{2}\gamma_C^k B_C^k\right) - f\left(S_S^k + \frac{h}{2}\gamma_S^k B_S^k\right)}{h} \\
&+ v_n^- \frac{f\left(S_N^k - \frac{h}{2}\gamma_N^k B_N^k\right) - f\left(S_C^k - \frac{h}{2}\gamma_C^k B_C^k\right)}{h} \\
&+ u_w^+ \frac{f\left(S_C^k + \frac{h}{2}\gamma_C^k A_C^k\right) - f\left(S_W^k + \frac{h}{2}\gamma_W^k A_W^k\right)}{h} \\
&+ \frac{1}{2}u_{en}^- \frac{f\left(S_{EN}^k - \frac{h}{4}\gamma_{EN}^k A_{EN}^k\right) - f\left(S_C^k - \frac{h}{4}\gamma_C^k A_C^k + \frac{h}{4}\gamma_C^k B_C^k\right)}{\frac{3}{4}h} \\
&+ \frac{1}{2}u_{es}^- \frac{f\left(S_{ES}^k - \frac{h}{4}\gamma_{ES}^k A_{ES}^k\right) - f\left(S_C^k - \frac{h}{4}\gamma_C^k A_C^k - \frac{h}{4}\gamma_C^k B_C^k\right)}{\frac{3}{4}h} = 0. \quad (24)
\end{aligned}$$

Note that taking $Q_C^k = 0$ for every cell C, the MUSCL scheme reduces to the upwind scheme (21). Even if the scheme (24) remains of first order on the composite grid, the numerical results presented in Section VI show that the MUSCL scheme considerably diminishes the numerical diffusion introduced by the upwind schemes.

IV. ALGORITHM FOR SOLVING TWO-PHASE FLOW BY ADAPTIVE MESH

The whole idea of the adaptive method is to produce a stable, physically reasonable solution while saving computational work. The flexibility to dynamically change the number of grid points and, thus, the number of unknowns can create difficulties in the linearization and linear solution algorithms. These algorithms are extremely difficult to vectorize effectively. Also, the adaptivity of the local refinement methods must be driven both by rapid changes in solution properties and by a simple and inexpensive method of mesh construction techniques.

Now, we are going to describe the algorithm for the resolution of system (7) by adaptive techniques. An IMPES method (implicit in pressure, explicit in saturation) is used ([4–6]).

Denote by Ω_H a fixed coarse grid, $\Omega_{h_l}^k$ a composite grid at time t^k where $(h_l = H/2^{l-1})$ is the space step of discretization of top level l, and $(P_{h_l}^k, S_{h_l}^k)$ is a given grid vector in this composite grid.

The goal of the algorithm is to construct a new composite grid $\Omega_{h_l}^{k+1}$ at time t^{k+1} and the associated solution $(P_{h_l}^{k+1}, S_{h_l}^{k+1})$. The triplet $(\Omega_{h_l}^{k+1}, P_{h_l}^{k+1})$ is to be carried by l step.

The difficulty in the case of self-adaptive methods is to pass information from one composite grid to another. For this, an interpolation operator $I_{\Omega_h}^{\Omega'_{h'}}$ is introduced. This operator designates prolongation or restriction of a grid vector from Ω_h to $\Omega'_{h'}$.

In the first step, we start by a regular coarse grid $\Omega_{h_1}^{k+1} = \Omega_H$, independent of $\Omega_{h_l}^k$. Then, the IMPES scheme is applied to the coarse grid; precisely, with the help of the interpolation operator $I_{\Omega_{h_l}^k}^{\Omega_{h_1}^{k+1}}$, we have

$$\begin{cases} \hat{S}_{h_1}^k = I_{\Omega_{h_l}^k}^{\Omega_{h_1}^{k+1}} S_{h_l}^k \\ -div(M(\hat{S}_{h_1}^k)\nabla P_{h_1}^{k+1}) = 0 \\ \mathbf{V}_{h_1}^{k+1} = -M(\hat{S}_{h_1}^k)\nabla P_{h_1}^{k+1} \\ \frac{S_{h_1}^{k+1} - \hat{S}_{h_1}^k}{\Delta t} + \mathbf{V}_{h_1}^{k+1} \cdot \nabla f(\hat{S}_{h_1}^k) = 0. \end{cases} \tag{25}$$

The resolutions of the pressure and saturation equations have been described, respectively, in Sections III.B and III.C.

In the second step, a criterion based on truncation error permits us to find refinement regions; therefore, a composite grid $\Omega_{h_2}^{k+1}$ is constructed. In the same way, the IMPES scheme is used on the new composite grid [see system (26)]. Hence, composite grids $\Omega_{h_j}^{k+1}$ for $k = 3, \ldots, l$ are successively constructed. We note that $\Omega_{h_j}^{k+1}$ is constructed by refining only the regular type cells, which find themselves in the interior of refined regions of $\Omega_{h_{j-1}}^{k+1}$. So, for an arbitrary $\Omega_{h_j}^{k+1}$, the IMPES scheme consists of the following algorithm:

$$\begin{cases} \hat{S}_{h_j}^k = I_{\Omega_{h_l}^k}^{\Omega_{h_j}^{k+1}} S_{h_l}^k \\ -div(M(\hat{S}_{h_j}^k)\nabla P_{h_j}^{k+1}) = 0 \\ \mathbf{V}_{h_j}^{k+1} = -M(\hat{S}_{h_j}^k)\nabla P_{h_j}^{k+1} \\ \frac{S_{h_j}^{k+1} - \hat{S}_{h_j}^k}{\Delta t} + \mathbf{V}_{h_j}^{k+1} \cdot \nabla f(\hat{S}_{h_j}^k) = 0. \end{cases} \tag{26}$$

In much literature [2, 16] one can find refinements and constructions of composite grids just after the first step. In other words, the truncation error reaches a threshold so that the level of

refinement is determined according to the magnitude of error tolerated. Other methods are based on refinement and derefinement regions [16]. The advantage of our proposed algorithm is to enable us to treat a complex problem, for example, where the solution changes radically from step to step (appearance picks, turbulence, . . .). The successive mesh construction induces a more accurate localization of the regions where refinement will be needed; therefore, a better profile of the front can be captured.

The solution of the pressure equation, which represents the most time-consuming part of the algorithm, is efficiently and accurately obtained by a bi-gradient like method. The choice of the initial solution for this iteration process is crucial to accelerate the convergence speed. A natural way is to take the interpolation of $P_{h_{j-1}}^{k+1}$ (defined on $\Omega_{h_{j-1}}^{k+1}$) to $\Omega_{h_j}^{k+1}$.

Recently, locally computed *a posteriori* error estimators have been developed by Babuska and Rheinbolds [17], Bieterman [18, 5] and Bank [19]. These *a posteriori* error estimators are extremely important for problems involving elliptic partial differential equations discretized by the finite element method. Here, we propose a widely used heuristic *a posteriori* criterion based on the magnitude of the discretization truncation error. It is a local computation, which takes little time. Using a simple threshold decision rule, we decide whether this point is going to be refined or not. Notice that the nested iteration of our algorithm offers a numerically cheap way to estimate the local truncation error. In practice this simple criterion appears to be quite efficient and reliable. The following method is used to switch cells:

$$\max_I |S_C^{k+1} - S_I^{k+1}| < \textit{ tol,} \text{ for } I \in \text{ neighbors } C.$$

V. DATA STRUCTURE

The adaptive solution of partial differential equation by mesh refinement is well known to be difficult. The data structure plays an important role in simulation [1]. Modification of the mesh by creating and deleting some grid blocks over time requires that the data structure used be dynamic. Present static data structure, in which the mesh is fixed permanently when the simulation is initialized, lends poorly to such a technique. Since the purpose of adaptive mesh refinement is to solve evolution problems more accurately, but at reasonable cost, a data structure must be worked out that is appropriate to the problem posed.

The programming languages FORTRAN 90, Ada, and C are becoming widely used in scientific computation. They provide so-call Abstract Data Types (ADT). Particularly, by means of `struct`, `pointer`, and `list`, one can construct some specific ADT data structure, adapted to the discretization. Furthermore, the allocation of memory in these languages can be dynamic, i.e., variables of any type can be allocated during execution, when the need arises, under the control of the program, by means of special instructions. See, for example, Cai, Le Gland, and Zhang [20] for an example of a finite difference scheme. The language C++ invented at Bell labs by B. Stroustrup [3] is an extension of the C programming language that provides an efficient implementation of *object-oriented programming* techniques. What is truly novel about C++ is its aggregate ADT type `class`. A class is an extension of the idea of `struct` in traditional C. It provides the means for implementing a user-defined data type and associated functions and operators. An interesting discussion about application of this language in multigrid method can be found in Rüede [21].

A closer analysis of the classification of composite mesh cells explained in Section III.A shows that different cells perform different tasks. A regular cell is the most simple case to treat. But in one or several directions, a complex cell can have the same geometric configuration as a regular

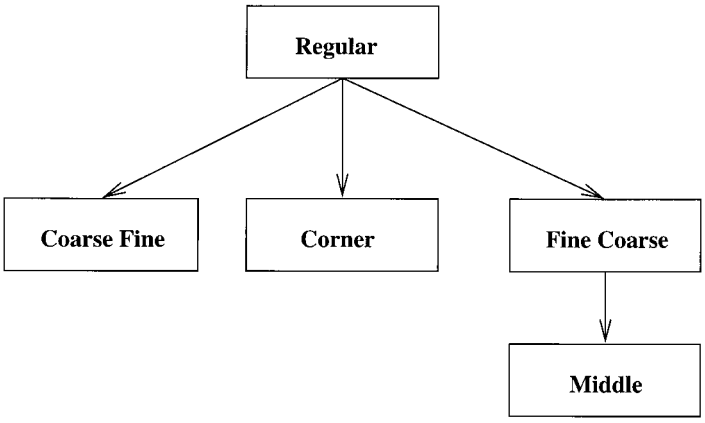

FIG. 5. Hierarchy.

cell. For instance, in Fig. 3, a **CF** cell can be considered as a regular cell when we regard the north or west direction. It is tedious to produce the same code for each ADT type. Based on this observation, we construct a hierarchy ADT classes of cell by using `derive class`. The derive class mechanism is call *inheritance* and is an important tool for object-oriented modularization and program development.

As showed in Fig. 5, a regular cell is defined as a base class. It can then be altered by adding members, overloading an existing member function, to create the other type of derived class cell. The advantage of this mechanism is that the derived class cell can share the information and code of its parent and ancestor classes. Code fragments for the definitions of the regular cell and coarse fine cell are shown in Tables I and II, respectively.

TABLE I. Data structure of a regular cell.

```
class R {
 public:
  int x, y;                    /* coordinate of barycenter */
  int size;                    /* cell size */
  double pressure, saturation; /* physique valeur */
  R      *N,                   /* pointers of 4 neighbors */
   *W,      *E,
         *S;
double    n,                  /* matrix coefficients */
       w, c, e,
          s;
R *next;
R();
void ErrorCriteria();
void Refinement();
...
virtual void Matrix()
virtual void Saturation();
...
};
```

TABLE II. Data structure of a coarse fine cell.

```
class CF : public R {
 public:
 R        *NW,   *NE,        /* pointers of its neighbors */
        *WN,         *EN,
        *WS,         *ES,
          *SW,    *SE;
 double   nw,   ne,          /* matrix coefficients */
       wn,          en,
       ws,          es,
         sw,     se;
 CF();
 void Matrix();
 void Saturation();
 ...
};
```

For any regular cell $\mathbf{x} = (x, y)$, in the constructed composite grid, the pointers S, W, E, and N contain the addresses of their neighbors, each of which is represented itself by same type of structure. If one or several of these neighbors do not exist (e.g., at the boundary), then the corresponding pointers will be allocated to NULL. The pointer `next` is reserved to the construction of chained lists. This is used mainly as a counter for iteration. The definition of coarse fine cell shows that a **CF** cell is a derive class of **R** cell, the eight pointers SW, SE, WS, WN, ES, EN, NW, and NE stock the addresses of their neighbors whose levels are inferior.

As an extension of the idea of `struct` in C, a class can have members that are functions. They allow the ADT to have particular functions that act on its private representation. Furthermore, it supports virtual member functions, which are functions declared in the base class and overloaded in a derived class. By accessing a virtual function through pointers, C++ selects the appropriate overloaded function at run-time. For instance, to calculate the matrix coefficients, the functions (both named `Matrix`() in the two structures) have different codes in **R** and **CF,** respectively, and the second one can partially share the codes of the first. The distinction is made dynamically; each class knows which one will be loaded. The function declarations R() and CF() are *constructor,* essentially containing the dynamic storage allocation of a single cell. The ADTs for the other types of cells are omitted. Note that it is convenient to declare the **MD** cell as a derived class of **FC.** We conclude this section by pointing out that the C++ language can be used to develop an efficient and modular implementation of the ADT for scientific computation, in particular for the adaptive mesh problem. Maximal flexibility and extensibility can be achieved. A good construction of the ADT hierarchy plays an important role in getting an efficient implementation.

VI. COMPUTATIONAL RESULTS

Several examples are presented. The first example validates the scheme described in Section III.B. In the second test, a comparison is given between the upwind scheme and the MUSCL scheme presented in Section III.C. The other examples treat models of waterflooding in porous media.

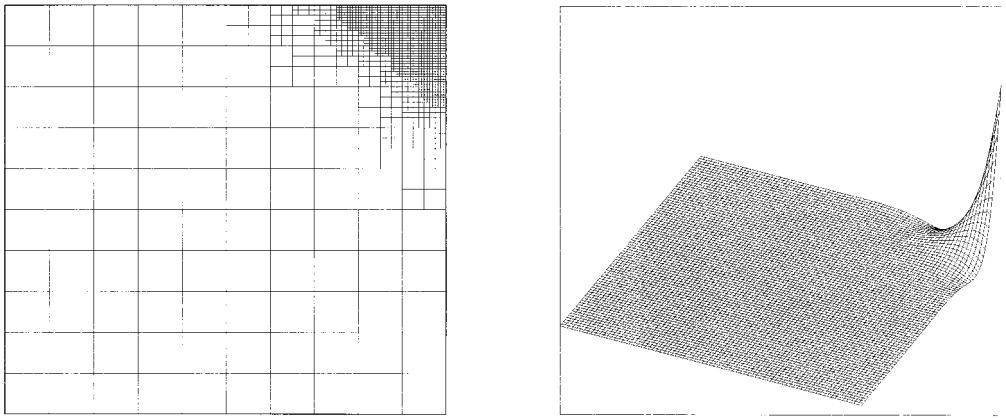

FIG. 6. Poisson equation, composite mesh of 5 levels and solution (number of cells = 1264).

A. Poisson Equation

The first example demonstrates the efficiency of the finite volume scheme on a composite grid. To test this scheme we numerically solve a classical equation of Poisson on a composite mesh:

$$\begin{cases} -\Delta u = 0 & \text{sur } \Omega = (0,1) \times (0,1) \\ u_\Gamma = g(x,y) & \text{sur } \Gamma = \partial\Omega, \end{cases}$$

where

$$g(x,y) = \cos(4\pi(x-y))\frac{\sin h(4\pi(x+y+2))}{\sin h(16\pi)}.$$

The analytic solution is $u(x,y) = g(x,y)$ in Ω. The solution is localized around the point (1, 1). The switching refinement criterion and the algorithm to obtain the composite grid are described in Section IV.

Figure 6 depicts the composite mesh of 5 levels, the coarse grid is 10×10 grids. A linear interpolation is used to draw the composite solution on a 160×160 mesh. Note that the refined composite grid localizes and closes to the solution.

Table III gives the run statistics to compare the adaptive method on composite grids and the classic method on regular finer grids. The maximum error between the composite (resp. the finer) and the analytic solutions are also tabulated. It is clear that the gain in CPU time is more and more important when the levels of refinement increase, moreover the quality of obtained composite solution favors the adaptive method.

TABLE III. Run statistics for Poisson equation.

Levels	Number of cells		CPU time		Max-error	
L	Composite	Finer	Composite	Finer	Composite	Finer
3	253	40×40	0.33	2.11	5.25×10^{-3}	5.24×10^{-3}
4	544	80×80	0.96	40.64	1.64×10^{-3}	1.64×10^{-3}
5	1264	160×160	4.09	400.18	0.45×10^{-3}	0.45×10^{-3}

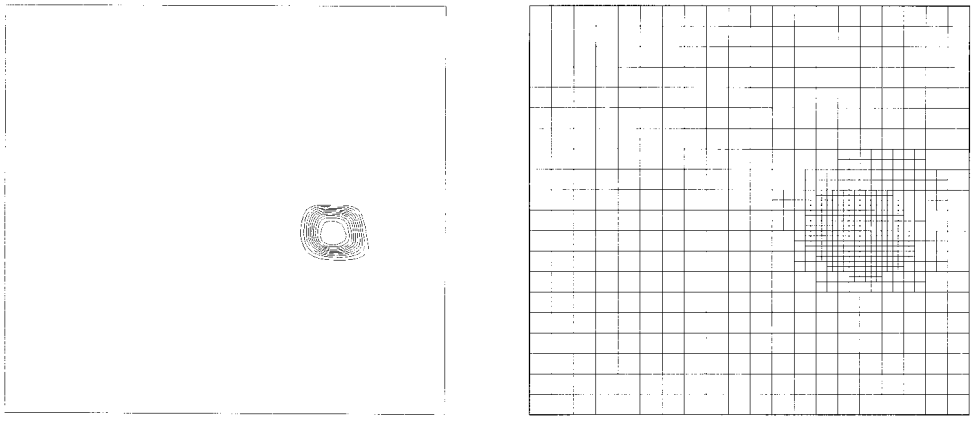

FIG. 7. Water saturation contours obtained by the MUSCL scheme and composite grid for the circular example, time = 1.0 day, number of cells = 754.

B. Circular Example

To validate the finite differences scheme presented in Section III.C, we are interested in a typical problem where the effects of convection are the strongest. The test consists of simulating transport of a saturation field around the center of a domain $\Omega = (0,1) \times (0,1)$ by a circular velocity field $\mathbf{V} = (u, v)^T$, defined as follows:

$$u = -\frac{\pi}{2}\left(y - \frac{1}{2}\right); v = \frac{\pi}{2}\left(x - \frac{1}{2}\right).$$

Then, the hyperbolic equation

$$\partial_t S + \mathbf{V} \cdot \nabla S = 0 \text{ in } \Omega$$

is solved.

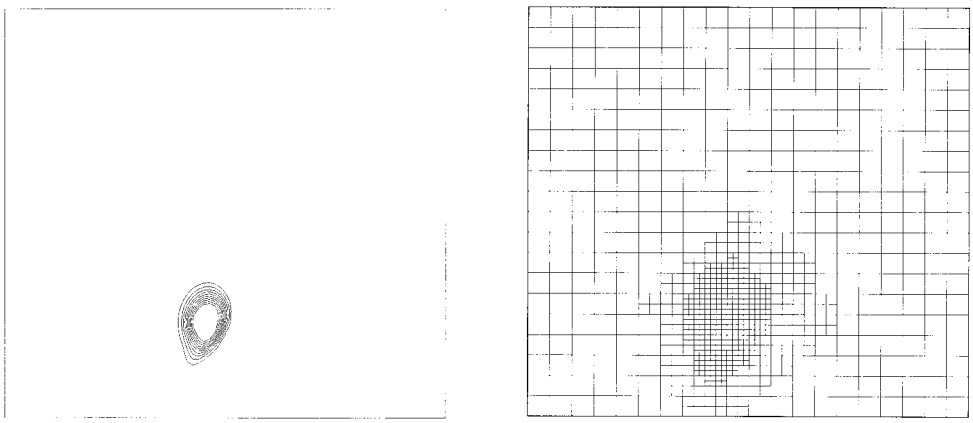

FIG. 8. Water saturation contours obtained by the MUSCL scheme and composite grid for the circular example, time = 4.0 day, number of cells = 793.

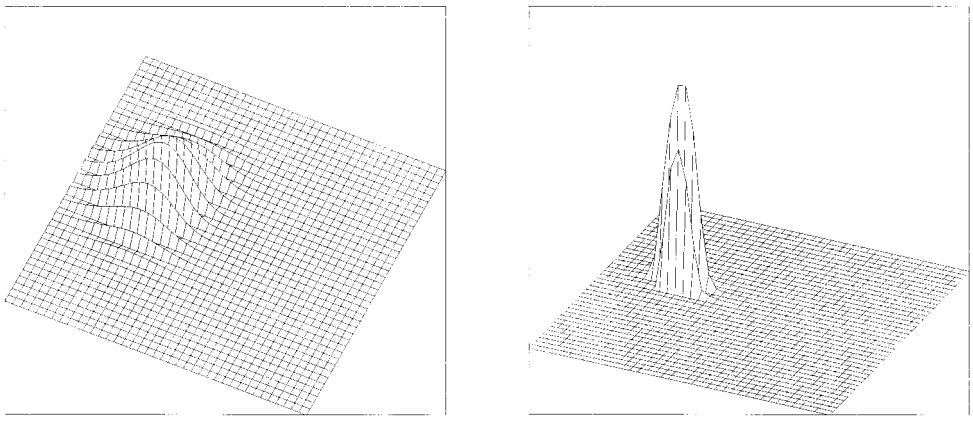

FIG. 9. Comparison between upwind (left) and MUSCL (right) schemes on composite grid for the circular example, time = 3.0 days. The maximums of saturation are 0.2 and 0.87, respectively.

The initial condition is taken as

$$S(x,y) = \begin{cases} 1, & 0.4 \le x \le 0.5; 0.2 \le y \le 0.3 \\ 0, & \text{otherwise,} \end{cases}$$

and all boundaries have no flow.

Figures 7 and 8 show the saturation contours obtained by the MUSCL scheme and the composite grids at various times. The coarse grid is a 20 $\times$ 20 mesh, and 3 levels are required.

The reduction in CPU time of the adaptive method with respect to that of the regular finest 80 $\times$ 80 grids is not significant. This is due to the fact that the used scheme is explicit. Note that the location of the refined area tracks the shock front quite closely during the time. One can remark also that the number of cells in the composite grids is far less then the number of cells in the finest grid. These numbers seem to be sufficient to obtain a good approximation for the solution.

Figure 9 shows a comparison between the solutions obtained by the upwind scheme and the MUSCL scheme on a composite grid. Note that the numerical diffusion introduced by the upwind scheme is considerably reduced.

C. Waterflood Examples

Two examples, treating models of waterflooding simulation, are considered for a two-phase flow in porous media. They differ by their geometry and boundary conditions. Both of them are computed by the MUSCL scheme. In the second test a comparison of upwind and MUSCL scheme is carried out.

• **Corner example.** A one-quarter five-spot [4], two-phase, incompressible water-injection problem is shown in Fig. 10. The data is summarized in Table IV. Water is injected at a constant pressure in the top northwest corner, and a constant pressure is specified in the lower southeast corner. All other boundaries have no-flow. The initial water saturation is the critical saturation (i.e., it is considered to be zero).

The coarse mesh is 10 $\times$ 10 grids, and four levels of refinement are used. Figures 11 and 12 show the water saturation contours and the composite grid at time 34 and 93 days. The number of cells on composite grids are compared to 80 $\times$ 80 cells. Note that the location of the refined area tracks the shock front quite closely during the time. One can remark also that the number of

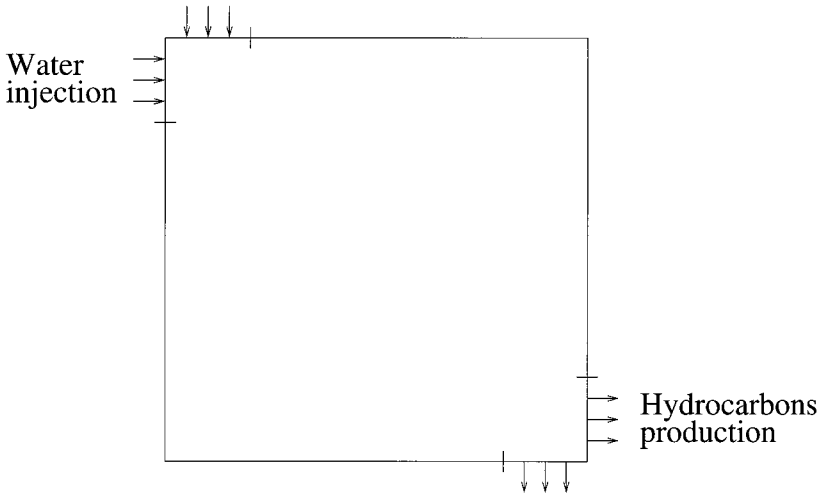

FIG. 10. Boundary conditions for the corner example.

TABLE IV. Data of corner example.

Absolute permeability	$k_x = 10^{-13}m^2$, $k_y = 10^{-13}m^2$
Porosity	$\phi = 0.206$
Viscosities	$\mu_w = 1.0 \times 10^{-3}pa \cdot s$; $\mu_o = 4.0 \times 10^{-3}pa \cdot s$
Relative permeability	$k_w = S^2, k_o = (1 - S)^2$;
Pressure of water injection	1500 kpa
Pressure of production	1000 kpa

cells in the composite grids is far less then the number of cells in the finest grid. These numbers seem to be sufficient to obtain a good approximation for the solution.

The run statistics illustrated in Table V reveal the gain of adaptive method in computer memory and in CPU time.

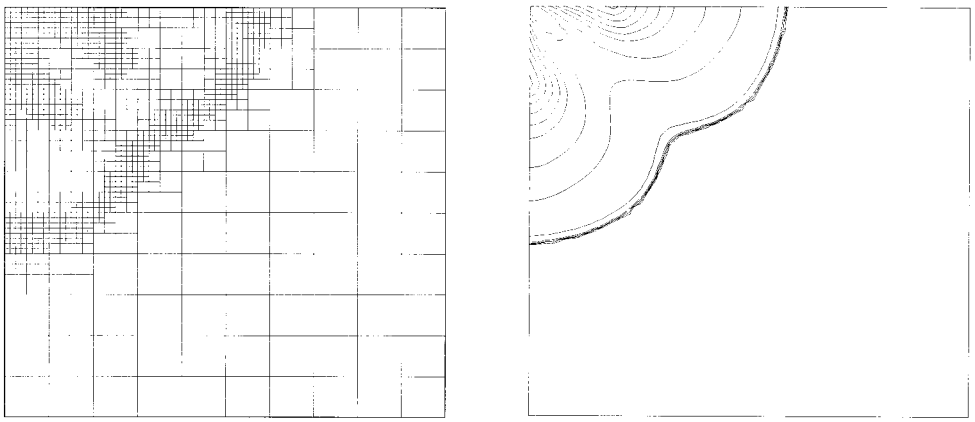

FIG. 11. Water saturation contours and composite grid for the corner example, time = 34.0 days, number of cells = 1051.

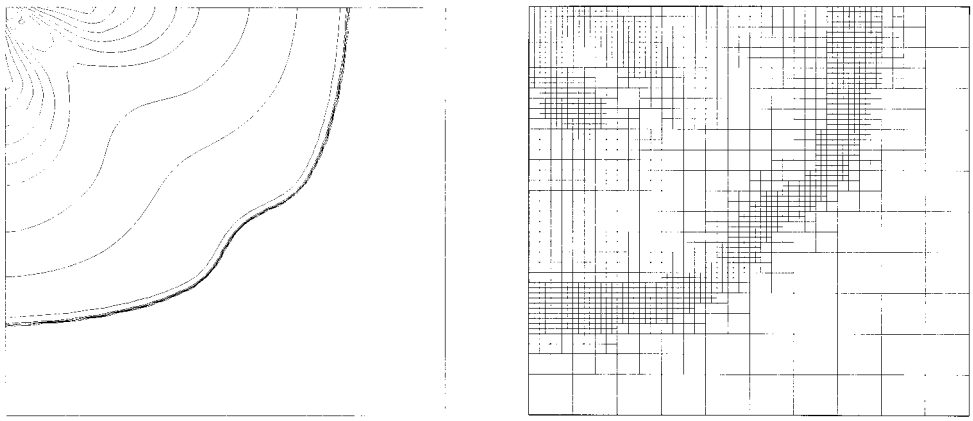

FIG. 12. Water saturation contours and composite grid for the corner example, time = 93.0 days, number of cells = 1735.

TABLE V. Run statistics for corner example.

Times (t)	Number of cells		Normalized CPU time	
	Composite	Finer	Composite	Finer
10.0	586	80×80	1.58	88.60
34.0	1038	80×80	5.24	98.65
43.0	1051	80×80	6.60	99.26
71.0	1243	80×80	9.70	100.28
93.0	1735	80×80	15.5	100.48

• **Rectangular example.** Figure 13 shows a two-dimensional rectangular reservoir. The formulation of the problem has the same form as the corner example, except that the area of the study domain is $\Omega = (0, 10) \times (0, 5)$, and water is injected in the top of the west side and a constant pressure is imposed on the bottom of the east side.

The coarse mesh is 10×5 grids and the level of refinement is four. Figures 14–16 depict the water saturation contours and the composite grids at times 30, 50, and 100 days. As in the corner example, the composite grids track the moving front.

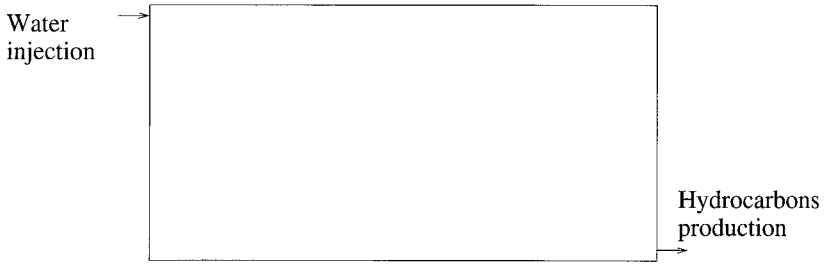

FIG. 13. Boundary conditions for rectangular example.

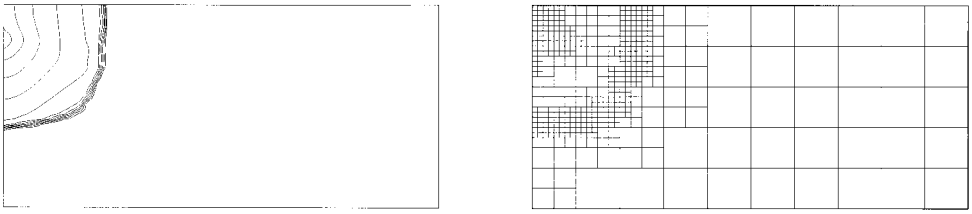

FIG. 14. Water saturation contours and composite grid for the rectangular example, time = 30.0 days, number of cells = 434.

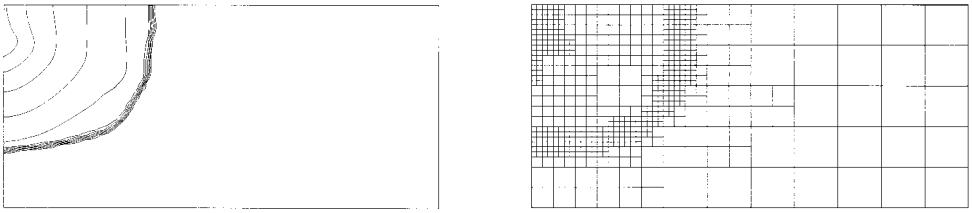

FIG. 15. Water saturation contours and composite grid for the rectangular example, time = 50.0 days, number of cells = 545.

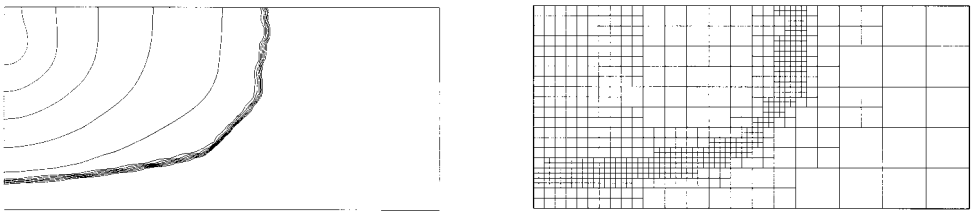

FIG. 16. Water saturation contours and composite grid for the rectangular example, time = 100.0 days, number of cells = 674.

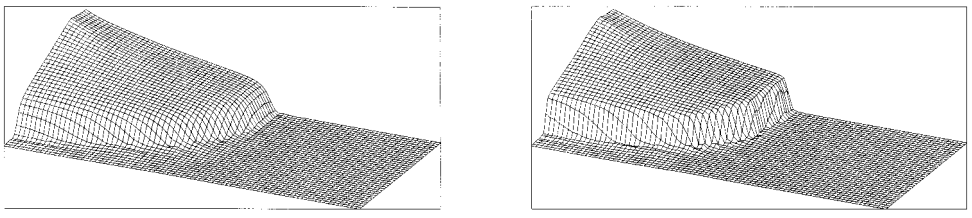

FIG. 17. Comparison between upwind (left) and MUSCL (right) schemes on the composite grid at time = 80.0 days.

TABLE VI. Run statistics of the rectangular example.

Times (t)	Number of cells		Normalized CPU time	
	Composite	Finer	Composite	Finer
20.0	359	80×40	0.93	28.00
30.0	434	80×40	1.74	29.12
50.0	545	80×40	2.46	30.55
70.0	584	80×40	2.52	31.43
100.0	674	80×40	4.21	31.77

Figure 17 shows a comparison of the saturations, at time 80 days, computed by the upwind and MUSCL schemes. Note that the interface water–oil (shock) is better localized by the second scheme.

Table VI shows the performance of the adaptive method. Note that the gain in CPU time is relatively more important than that of the corner example, because the shock front is more detached.

VII. CONCLUSION

In this article, numerical approximation of two-phase incompressible problems is studied. The aim is to develop an algorithm to obtain composite grid and associate solutions with the smallest computational cost. However, the flexibility of the proposed adaptive method, which allows switching the cells of different types, increases the possibility of getting a refinement only near sharp fronts and of limiting the refinement area.

An IMPES method is applied. The basic method used in the spatial discretization involves a convergent finite volume scheme to approach the pressure equation and monotone finite difference schemes to discretize the saturation equation. In each step time, the construction of the composite grid was obtained in several steps. This permits us to reduce the number of cells at successive levels and to improve localization of sharp fronts.

The results seem to confirm that, with the criteria developed here, the adaptive method is extremely attractive for two-phase flow problems.

References

1. R. E. Ewing, ''Mathematical modeling and large-scale computing in energy and environement research,'' in *The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics,* R. E. Ewing, K. I. Gross, and C. F. Martin, Eds., Springer Verlag, New York, 1986, pp. 45–59.
2. T. Hermitte, ''Maillage adaptatif en espce dans le simulations de gisements pétroliers,'' Ph. D Thesis, Université de Provence Aix-Marseille I, 1993.
3. D. W. Stroustrup, *The C++ Programming Language,* Addison–Wesley, New York, 1991.
4. K. Aziz and A. Settari, *Petroleum reservoir simulation,* Applied Science, London, 1979.
5. P. A. Forsyth, ''Adaptive implicit criteria for two-phase flow with gravity and capillary pressure,'' *SIAM J. Sci. Stat. Comp.* **10,** 227–252 (1989).
6. C. Liu, Z. Liu, and S. McCormick, ''An efficient multigrid scheme for elliptic equations with discontinuous coefficients,'' *Comm. Appl. Numer. Methods* **8,** No. 9, 621–631 (1992).

7. R. E. Ewing, D. Lazrarov, and P. S. Vassilevski, ''Local refinement techniques for elliptic problems on cell-centered grids,'' *Technical Report 16, Inst. Sci. Comp.,* University of Wyoming, 1988.
8. R. E. Ewing, B. A. Boyett, D. K. Babu, and R. F. Heinemann, ''Efficient use of locally refined grids for multiphase reservoir simulation,'' *SPE 18413,* (1989).
9. H. C. Yee, ''Construction of explicit and implicit symmetric TVD schemes and their applications,'' *J. Comp. Phys.* **68,** 151–179 (1987).
10. A. Harten, ''High resolution schemes for hyperbolic conservation laws,'' *J. Comp. Phys.* **49,** 357–393 (1983).
11. B. Van Leer, ''Towards the ultimate conservative scheme IV, a new approach to the numerical convection,'' *J. Comp. Phys.* **23,** 276–299 (1977).
12. B. Van Leer, ''Towards the ultimate conservative scheme III, upstream-centered finite difference schemes for ideal compressible flow,'' *J. Comp. Phys.* **23,** 263–275 (1977).
13. J. L. Randall, ''Numerical methods for conservation laws,'' in *Lectures in Mathematics,* Birkhauser, 1990.
14. P. L. Roe, ''Approximate riemann solvers, parameter vectors, and difference schemes,'' *J. Comp. Phys.* **43,** 357–373 (1981).
15. M. Saad, ''An accurate numerical algorithm for solving three-phase flow in porous media,'' *Technical Report 96004,* Mathématiques Appliquées de Bordeaux, Université de Bordeaux I, January, 1996.
16. I. Babuska, J. Chandra, and J. E. Flaherty, Eds., *Adaptive Computational Methods for Partial Differential Equations,* Volume 16 of *Proceedings in Applied Mathematics,* College Park, Maryland, February 1983. U. S. Army Research Office, SIAM.
17. I. Babuska and W. C. Rheinboldt, ''*A posteriori* error analysis of finite element solutions of one-dimensional problems,'' *SIAM J. Numer. Anal.* **18,** 565–589 (1981).
18. M. B. Bieterman and I. Babuska, ''The finite element method for parabolic equations, I: *A posteriori* error estimation, II: *A posteriori* error estimation and adaptive approach,'' *Numerische Mathematik* **40,** 339–371 and 373–406 (1982).
19. R. E. Bank, ''A multilevel iterative method for nonlinear elliptic equations,'' in *Elliptic problem solvers,* in M. Schultz, Ed., Academic Press, New York, 1981.
20. Z. Cai, F. Le Gland, and H. Zhang, ''An adaptive local grid refinement method for nonlinear filtering,'' *Rapport de Recherche 954,* IRISA-INRIA, 1995.
21. U. Rüede, *Mathematical and Computational Techniques for Multilevel Adaptive Methods,* Volume 13 of *Frontiers in Applied Mathematics,* SIAM, Philadelphia, 1993.