Exploiting OpenMP Efficiency for Multiphase Flow Simulation
in Heterogeneous Porous Media

E. Abreu[1] **and R. B. Seixas**[1,2] *

[1] **IMPA – Instituto Nacional de Matemática Pura e Aplicada, RJ 22460-320, Brazil**

[2] **TECGRAF/PUC-Rio – Grupo de Tecnologia em Computação Gráfica da PUC-Rio, RJ 22453-900, Brazil**

Abstract.

This paper presents an efficient OpenMP implementation based on an operator splitting algorithm for the numerical simulation of multi-dimensional three-phase convection-diffusion transport problems in heterogeneous rock. The nonlinear partial differential equations modelling such a three-phase flow through porous media takes into account variable porosity and permeability fields. In our numerical procedure a locally conservative central difference scheme is used for the approximation of the nonlinear system of hyperbolic conservation laws modelling the convective transport of the fluid phases. This scheme is combined with locally conservative mixed finite elements for the numerical solution of parabolic and elliptic problems associated with the diffusive transport of fluid phases and the pressure-velocity problem. Numerical experiments indicate that the use of large time-splitting relations with OpenMP implementation might translates into a significant reduced computational effort to produce numerical results within a given accuracy requirement.

Keywords: OpenMP, HPC, Operator Splitting, Multiphase flow, Porous media

1. Introduction

This paper discusses an OpenMP (Open Multi-Processing) implementation based on an operator splitting technique for the numerical solution of a highly nonlinear system of differential equations modelling three-phase flow through multi-dimensional heterogeneous porous media. Three-phase flow in porous media is important in pure sciences as well as in technological activities. Examples include enhanced oil recovery (14), geological CO_2 sequestration (22; 26), and radionuclide migration from repositories of nuclear waste (19).

We choose to use an OpenMP to parallelize the code because our need to improve performance and, at the same time, keeping the clarity of the original. Thus, OpenMP emerge as reliable alternative as it is just a set of compiler directives with library routines for parallel application that greatly simplifies writing multi-threaded programs.

We stress that our focus in writing this paper was to give some insights about OpenMP applications for non-experts researchers and students, outside of the HPC research arena, who intend to parallelize existing codes or to develop new ones in a simple manner, with no excessive effort by the user.

---

*Corresponding author. Email: {eabreu,rbs}@impa.br

OpenMP is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran on many architectures, including Linux, Unix and Microsoft Windows platforms (see, e.g., (27; 24)). Essentially, it consists of a set of compiler directives, library routines, and environment variables that influence run-time behavior.

Additionally, OpenMP is an implementation of multithreading, a method of parallelization whereby the master "thread" (a series of instructions executed consecutively) "forks" a specified number of slave "threads" with a task that is divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors. The section of code that is meant to run in parallel based on OpenMP is marked accordingly with a preprocessor directive that will cause for each thread runs subtasks such as "#pragma omp parallel for" provided that the formed section is completely verified and prepared before it is initialized.

We will show that the inclusion of compiler OpenMP directives in the code are quite simple, not requiring significant changes in the structure of the original program. One can use incremental parallelism, working in a portion of the program at one time, and use OpenMP directives as comments in the case of sequential compilers.

The authors are aware about other options for code parallelization such as classical MPI (Message Passing Interface), and more recently CUDA (Compute Unified Device Architecture), in which both require significant and non-trivial programming changes to go from a serial to a parallel version as well as its inherent complexity for debugging.

In our splitting procedure we solve elliptic, hyperbolic, and parabolic differential equations sequentially, using state-of-the-art numerical methods specifically tailored to such types of equations. We remark that it would have been very difficult, if not impossible, to employ such state-of-the-art numerical schemes had we attempted to solve the original system by standard implicit procedures (see, e.g., (25; 13)). Moreover, any implicit procedure would produce considerably more expensive computational solutions since large linear and nonlinear problems, which do not appear in the splitting scheme, would have to be considered (see, e.g., (3; 4) and references therein).

Other approaches for solving numerically three-phase flow systems can be found in (8; 6; 7; 20; 16). The rest of this paper is organized as follows. In Section 2 we introduce the model for three-phase flow in heterogeneous porous media considered in this work. In Section 3 we discuss our OpenMP operator splitting strategy for solving the three-phase flow system. In Section 4 we present a set of computational solutions for the model problem considered here. Additional results with OpenMP implementation are presented in Section 5. Our concluding remarks are given in section 6.

2. Three-phase flow system

We refer the reader to (25; 8; 2) for a detailed description of the derivation of the phase formulation of the governing equations of three-phase flow.

For concreteness, we consider two-dimensional flow of three immiscible incompressible fluid phases in a porous medium. The phases will be refereed to as water, gas, and oil and indicated by the subscripts w, g, and o, respectively. We assume that there are no internal sources or sinks. Compressibility, mass transfer between phases, and thermal effects are neglected. We assume that the three fluid phases saturate the pores; thus, with S_i denoting the saturation (local volume fraction) of phase i, $\sum_i S_i = 1$, $i = g, o, w$. Consequently, any pair of saturations can be chosen to describe the state of the fluid (15; 8; 2).

We shall work with the saturations S_w and S_g of water and gas, respectively; note that

$S_o = 1 - S_w - S_g$. Then, the equations describing conservation of mass of water and gas are:

$$\frac{\partial}{\partial t}(\phi(\mathbf{x})S_w) + \nabla \cdot (\mathbf{v}f_w(S_w, S_g)) = \nabla \cdot \{K(\mathbf{x})\lambda_w[(1 - f_w)\nabla p_{wo} - f_g\nabla p_{go}]\}, \quad (1)$$

$$\frac{\partial}{\partial t}(\phi(\mathbf{x})S_g) + \nabla \cdot (\mathbf{v}f_g(S_w, S_g)) = \nabla \cdot \{K(\mathbf{x})\lambda_g[(1 - f_g)\nabla p_{go} - f_w\nabla p_{wo}]\}, \quad (2)$$

$\mathbf{x} \in \Omega \subset \Re^3$, $t \geq 0$. The diffusive terms in the right hand side of saturation equations above take into account the capillary pressure differences $p_{ij} = p_i - p_j$, $i \neq j$, where p_i is the pressure in phase i, and p_{ij} are typically assumed to depend solely on the saturations of fluid phases.

Here, $K(\mathbf{x})$ and $\phi(\mathbf{x})$ are the absolute permeability and the porosity of the porous medium, respectively. We define the mobility of phase i, $i = w, g, o$, as $\lambda_i(S_w, S_g) = k_i/\mu_i$, which is given in terms of the phase relative permeability k_i and phase viscosity μ_i. The fractional flow function of phase i is defined by $f_i(S_w, S_g) = \lambda_i/\lambda$, whereby $\lambda = \sum_i \lambda_i$, $i = g, o, w$.

The saturation equations (1)-(2) are coupled with the following pressure-velocity system

$$\nabla \cdot \mathbf{v} = 0, \quad \mathbf{v} = -K(\mathbf{x})\lambda(S_w, S_g)\nabla p_o + \mathbf{v}_{wo} + \mathbf{v}_{go}, \quad (3)$$

$\mathbf{x} \in \Omega \subset \Re^3$, $t \geq 0$, associated with the three-phase flow model, where $\mathbf{v}_{wo}$ and $\mathbf{v}_{go}$, due to capillary pressure differences, are defined by $\mathbf{v}_{io} = -K(\mathbf{x})\lambda_i(S_w, S_g)\nabla p_{io}$, $i = w, g$.

Boundary and initial conditions for (1)-(3) must be imposed to complete the definition of the mathematical model; in particular, S_w and S_g must be specified at the initial time $t = 0$. The boundary conditions will be introduced in next Section 3.. The fluid flows are computed in a bounded two-dimensional reservoir $\Omega = [0, X] \times [0, Y]$ in our examples below.

3. Operator splitting and OpenMP for three-phase flow

Operator splitting techniques constitute one of the several bridges between numerical and functional analysis. In numerical analysis, they represent algorithms intended to approximate evolution equations accurately in a computationally efficient fashion. In functional analysis, they are used to prove estimates, existence and representation theorems. The survey article (9) discusses both uses and points to a large bibliography. See also (17; 18) for recent developments in the analysis of time-splitting errors for one-dimensional, nonlinear, convection-diffusion problems.

Operator splitting techniques aiming at computational efficiency have routinely been used in the numerical simulation of reservoir flow problems, particularly in the case of single and two-phase flows. Instead of solving the governing system of differential equations in the form which results directly from the basic conservation laws (supplemented by constitutive relations), the system of equations is rewritten in such a way as to exhibit clearly its mathematical nature. Then distinct appropriate numerical techniques can be orchestrated, within an operator-splitting formulation, to furnish powerful and efficient numerical procedures designed for resolving the sharp gradients and dynamics evolving at vastly different rates which are the hallmarks of reservoir flow problems.

The OpenMP model is based on directives to compilation. Then, the programmer indicates to the compiler which regions should run in parallel in a straightforward manner with respect to MPI and CUDA programing. Indeed, a really nice thing about OpenMP, is that it is much less intrusive by converting the program to using threads.

For existing serial codes the standard approach is to run a parallelism directive over specific and time-consuming portions of the code. By identifying parallelizable OpenMP blocks and loops, the effort is reduced merely in writing the pragmas directives. To ensure that no additional efforts is spent in regions with low time-consuming it is required only a preliminary

analysis of the code in order to introduce appropriate OpenMP directives along the routines of the main program. In some cases, the use of reckless policies might further compromise the high performance of the program.

3.1 A two-level operator-splitting with OpenMP

A serial two-level operator-splitting for three-phase flow equations (1)-(3) was introduced in (2). Aspects of the computational efficiency for serial operator splitting algorithms for multiphase flow in porous media was discussed in (3; 4). Here, we address a parallel operator splitting algorithm based in (2; 3; 4) with an OpenMP implementation.

Thus, for the sake of complementeness, and for convenience of the reader, we reproduce the operator splitting procedure, in which allows us to identify one subsystem of nonlinear hyperbolic conservation laws (associated with convective transport), one parabolic subsystem of equations (associated with diffusive transport), and one elliptic subsystem (associated with the pressure-velocity calculation).

First we split the pressure-velocity calculation from the saturation calculation and then split the saturation calculation into convection and diffusion. We introduce three time steps: Δt_c for the solution of the hyperbolic problem for the convection, Δt_d for the parabolic problem for the diffusion and Δt_p for the elliptic problem for the pressure-velocity calculation: $\Delta t_p = i_1 \Delta t_d = i_1 i_2 \Delta t_c$, where i_1 and i_2 are positive integers, so that $\Delta t_p \geq \Delta t_d \geq \Delta t_c$. Let

$$t^m = m\Delta t_p, \ \ t_n = n\Delta t_d \text{ and } t_{n,\kappa} = t_n + \kappa\Delta t_c, \text{ so that, } 0 \leq \kappa \leq i_2, \text{ and } t_{n,i_2} = t_{n+1}. \quad (4)$$

Given a generic function z, denote its values at times t^m, t_n, and $t_{n,\kappa}$ by z^m, z_n, and $z_{n,\kappa}$. To simplify the description of the operator splitting we also assume each time step to have a single value.

The oil pressure (and Darcy velocity) will be approximated at times t^m, $m = 0, 1, 2, \ldots$. The saturations, S_w and S_g, will be approximated at times t_n, $n = 1, 2, \ldots$; recall that they must specified at $t = 0$. In addition, there will be values for the saturation computed at intermediate times $t_{n,\kappa}$ for $t_n < t_{n,\kappa} \leq t_{n+1}$ that take into account the convective transport of the water and gas but not the diffusive effects. The algorithm will be detailed below.

The initial conditions S_w and S_g at $t = 0$ allow the calculation of $\{p_o{}^0, \mathbf{v}^0\}$. The following is the fractional step algorithm associated with the differential form of three-phase model that should be followed until the final simulation time is reached.

First level

1) Given $S_w^m(\mathbf{x})$ and $S_g^m(\mathbf{x})$, $m \geq 0$, compute $\{p_o^m, \mathbf{v}^m\}$ by Eq. (3), subject to

$$\begin{array}{rllll} \mathbf{v} \cdot \nu = & -q, & \text{for } x = \{0\}, & y \in [0, Y], \\ \mathbf{v} \cdot \nu = & q, & \text{for } x = \{X\}, & y \in [0, Y], \\ \mathbf{v} \cdot \nu = & 0, & \text{for } y = \{0, Y\}, & x \in [0, X], \end{array} \quad (5)$$

where ν is the unit outer normal vector to $\partial\Omega$.

2) For $t^m < t \leq t^{m+1}$, solve the convection-diffusion system (1)-(2), i.e., the saturation equations, with the initial conditions $S_w(\mathbf{x}, t^m) = S_w^m(\mathbf{x})$ and $S_g(\mathbf{x}, t^m) = S_g^m(\mathbf{x})$; $S_w^m(\mathbf{x})$ and $S_g^m(\mathbf{x})$ are evaluated as the final values of the calculation in $[t^{m-1}, t^m]$ for $m > 0$ or the initial saturations for $m = 0$.

Second level

1) Let $t_{n_1} = t^m$ and assume that $\{p_o, \mathbf{v}, S_w, S_g\}$ are known for $t \leq t_{n_1}$.

2) For $n = n_1, \ldots, n_2 = n_1 + (i_1 - 1)$:

    a) For $\kappa = 0, \ldots, (i_2 - 1)$ and $t \in [t_{n,\kappa}, t_{n,\kappa+1}]$ solve the convection system given by:

$$\frac{\partial}{\partial t}(\phi(\mathbf{x}) s_i^\kappa) + \nabla \cdot [\mathbf{E}(t_{n,\kappa}, \mathbf{v}) f_i(s_w^\kappa, s_g^\kappa)] = 0, \quad i = w, g, \tag{6}$$

    with initial and boundary conditions

$$s_i^\kappa(\mathbf{x}, t_{n,\kappa}) = \begin{cases} S_i(\mathbf{x}, t_n), & \kappa = 0, \\ s_i^{\kappa-1}(\mathbf{x}, t_{n,\kappa}), & \kappa = 1, \ldots, i_2 - 1, \end{cases} \quad i = w, g, \tag{7}$$

$$\mathbf{E}(t_{n,\kappa}, \mathbf{v}) f_i \cdot \nu = -q f_i(S_w^L, S_g^L) \cdot \nu, \text{ for } (x, y) \in \{0\} \times [0, Y], \tag{8}$$

    where S_w^L, S_g^L are the water and gas saturations of the injected mixture and q stands for the volumetric injection rate of the two flooding fluids. We note that and $\mathbf{E}(t_{n,\kappa}, \mathbf{v})$ indicates a linear extrapolation operator; it extrapolates to time $t_{n,\kappa}$ the velocity fields $\mathbf{v}^{m-1}$ and $\mathbf{v}^m$.

    b) Set $\bar{S}_i(\mathbf{x}, t_n) = s_i^{i_2-1}(\mathbf{x}, t_{n,i_2}), \; i = w, g$.

    c) Compute the diffusive effects on $[t_n, t_{n+1}]$ by solving the two-component system

$$\frac{\partial}{\partial t}(\phi(\mathbf{x}) S_i) - \nabla \cdot [K(\mathbf{x})\lambda_i \sum_{\substack{i \neq j \\ j=w,o,g}} f_j(\nabla p_{ij})] = 0, \quad i = w, g, \tag{9}$$

    with boundary and initial conditions

$$[K(\mathbf{x})\lambda_i \sum_{\substack{i \neq j \\ j=w,o,g}} f_j(\nabla p_{ij})] \cdot \nu = 0, \quad \mathbf{x} \in \partial\Omega, \quad i = w, g, \tag{10}$$

$$S_i(\mathbf{x}, t_n) = \bar{S}_i(\mathbf{x}, t_n), \quad \mathbf{x} \in \Omega, \quad i = w, g. \tag{11}$$

3) Set $S_i^{m+1}(\mathbf{x}) = S_i(\mathbf{x}, t_{n_2+1}), \; i = w, g$.

3.2 Numerical procedures

We refer the reader to (2) for a detailed description of the discretization of the governing system of equations, Eqs. (3), (5) and (6)-(11). Below we provide the key ideas.

The oil pressure (p_o) and the Darcy velocity ($\mathbf{v}$), Eqs. (3) and (5), are approximated by locally conservative mixed finite elements (see (11; 2)). The linear system of algebraic equations that arises from the discretization of the pressure-velocity problem for p_o is solved by a preconditioned conjugate gradient procedure (PCG) (11; 2).

Locally conservative mixed finite elements are used to discretize the spatial operators in the diffusion system (9)-(11). The time discretization of the latter is performed by means of the implicit backward Euler method (see for details (2)).

To solve the nonlinear hyperbolic conservation laws (6)-(8) we use the nonoscillatory, second order, conservative central difference scheme (23) (see also (2)).

4. Numerical OpenMP results for three-phase flow system

The goal of the numerical experiments reported in this section is to verify the computational OpenMP efficiency for three-phase flow simulation in heterogeneous porous media.

We adopt the model by Corey-Pope (10; 12) for phase relative permeabilities:

$$k_w = S_w^2, \quad k_o = S_o^2, \quad \text{and} \quad k_g = S_g^2. \tag{12}$$

We adopt the following model (21) for capillary pressures given by

$$p_{wo} = 5\epsilon(2 - S_w)(1 - S_w) \quad \text{and} \quad p_{go} = \epsilon(2 - S_g)(1 - S_g), \tag{13}$$

where the dimensionless coefficient ϵ controls the relative importance of diffusive and convective forces. In our numerical experiments we take $\epsilon = 1.0 \times 10^{-3}$.

4.1 Two-dimensional experiments

For three-phase flow, distinct empirical models have been proposed for the relative permeability functions (10; 28; 12), and more recently (16). In addition, it is well known that for some of these models (10; 28; 12), which have been used extensively in petroleum engineering, the 2×2 system of nonlinear hyperbolic conservation laws (the saturation equations (6)-(8)) that arises when capillarity (diffusive) effects are neglected fails to be strictly hyperbolic somewhere in the interior of the saturation triangle (the phase space). This loss of strict hyperbolicity leads to the frequent occurrence of nonclassical shock waves (called transitional or undercompressive shock waves) in the solutions of the three-phase flow model (see (15; 21)). Thus, their accurate computation constitutes a bona fide test for numerical simulators.

The computed fluid flows are defined in a bounded two-dimensional reservoir $\Omega = [0, X] \times [0, Y]$ with aspect ratio $X/Y = 4$, discretized on a uniform grid of 512×128 cells. The spatially variable permeability and porosity fields are defined on a uniform geological grid with 512×128 cells, and have coefficients of variation ((standard deviation)/mean) $CV_k = 1.0$, and $CV_\phi = 0.25$, respectively. The CV might be used as a dimensionless measure of the strength of the heterogeneity of permeability and porosity fields. The viscosities of the fluids are $\mu_o = 1.0$, $\mu_w = 0.5$, and $\mu_g = 0.3$.

In Figures 1-3 gas, oil, and water saturation surface plots are shown as functions of distance for a time of 800 days of simulation. A mixture of fluids (72.1 % of water and 27.9 % of gas) is injected at constant rate along the left horizontal boundary, $x = \{0\}$ and $y \in [0, 128]$, of the computational region and "no-flow" conditions are imposed on the horizontal boundaries, $y = \{0, Y\}$ and $x \in [0, 512]$. Initially, the resident composition of fluid in the reservoir is a mixture of 5% water, 15% gas, and 80% oil.

We remark that for the choice of parameters and initial data described above, a transitional wave appears in the two-dimensional solution of the governing three-phase flow equations (3), (5) and (6)-(11) in this heterogeneous rock formation. In Figures 1-3 this wave is located at 288-352m, in the x direction of flow, with an amplitude ranging from 0.125 to 0.225 (gas), from 0.05 to 0.46 (water) and from 0.415 to 0.73 (oil). For a more detailed discussion of the occurrence of such waves in porous media see (1; 2) and references therein.

For the simulations reported in Figures 1-3 we take, from top to bottom, the following time-step relations $\Delta t_p = \Delta t_d = \Delta t_c$ (as a reference solution, see (2)), $\Delta t_p = \Delta t_d = 100\Delta t_c$, and $\Delta t_p = \Delta t_d = 200\Delta t_c$. In our numerical computations we achieve a time saving of greater than 80 % with respect of that used in the reference solution. An additional performance with OpenMP was obtained: based on the different computer architectures and processors we have used, including even the state-of-the-art Intel's Nehalem processor, we achieve a time saving ranging from 6.86% to 52.23% over the results with the reference solution (with no OpenMP).
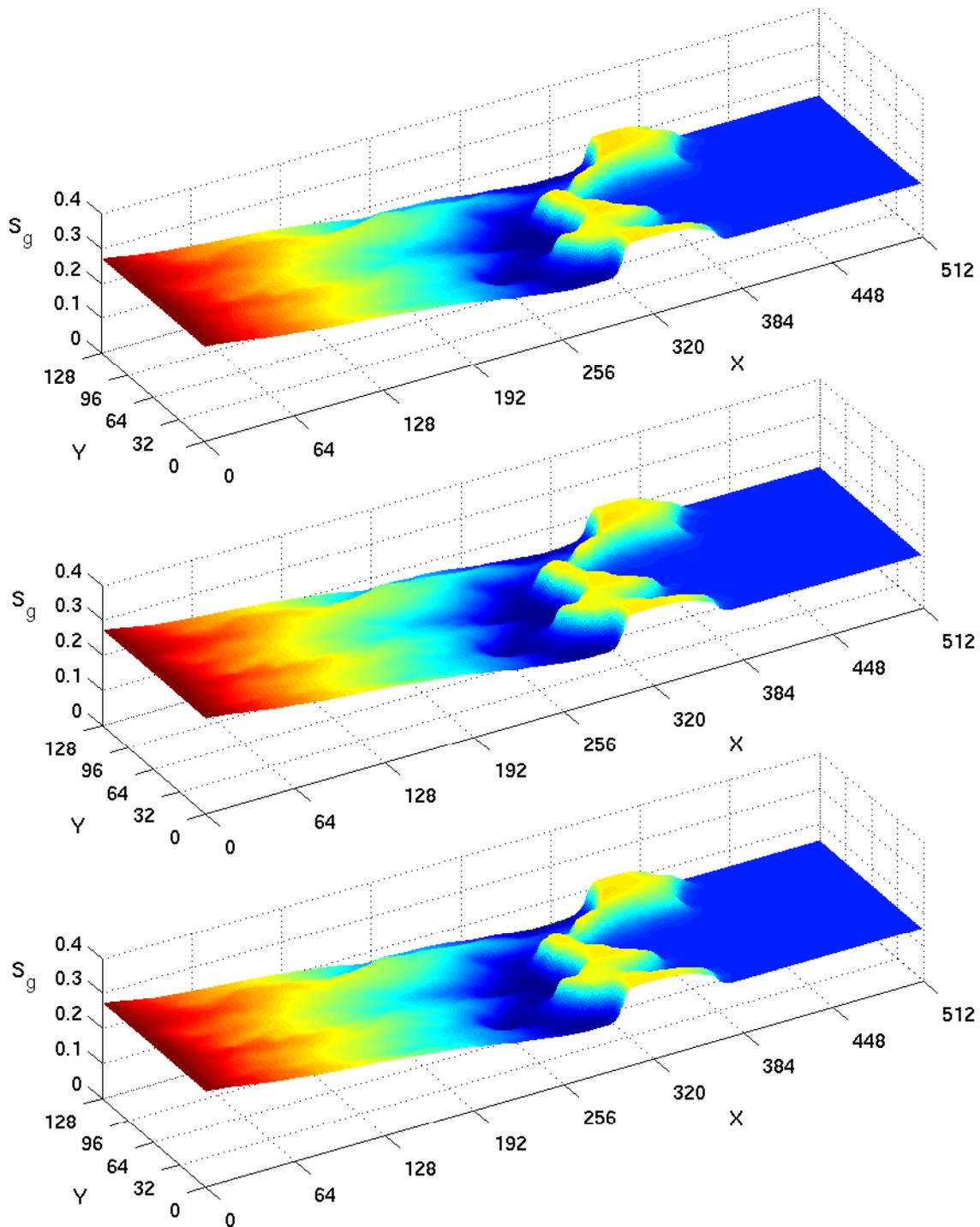
Figure 1: Gas saturation surface plots are shown for a two-dimensional simulation study using OpenMP with an operator splitting algorithm. From top to bottom we consider the following time-step relations $\Delta t_p = \Delta t_d = \Delta t_c$, $\Delta t_p = \Delta t_d = 100\Delta t_c$, and $\Delta t_p = \Delta t_d = 200\Delta t_c$, where Δt_c is determined by a CFL constraint. A transitional shock wave is simulated.
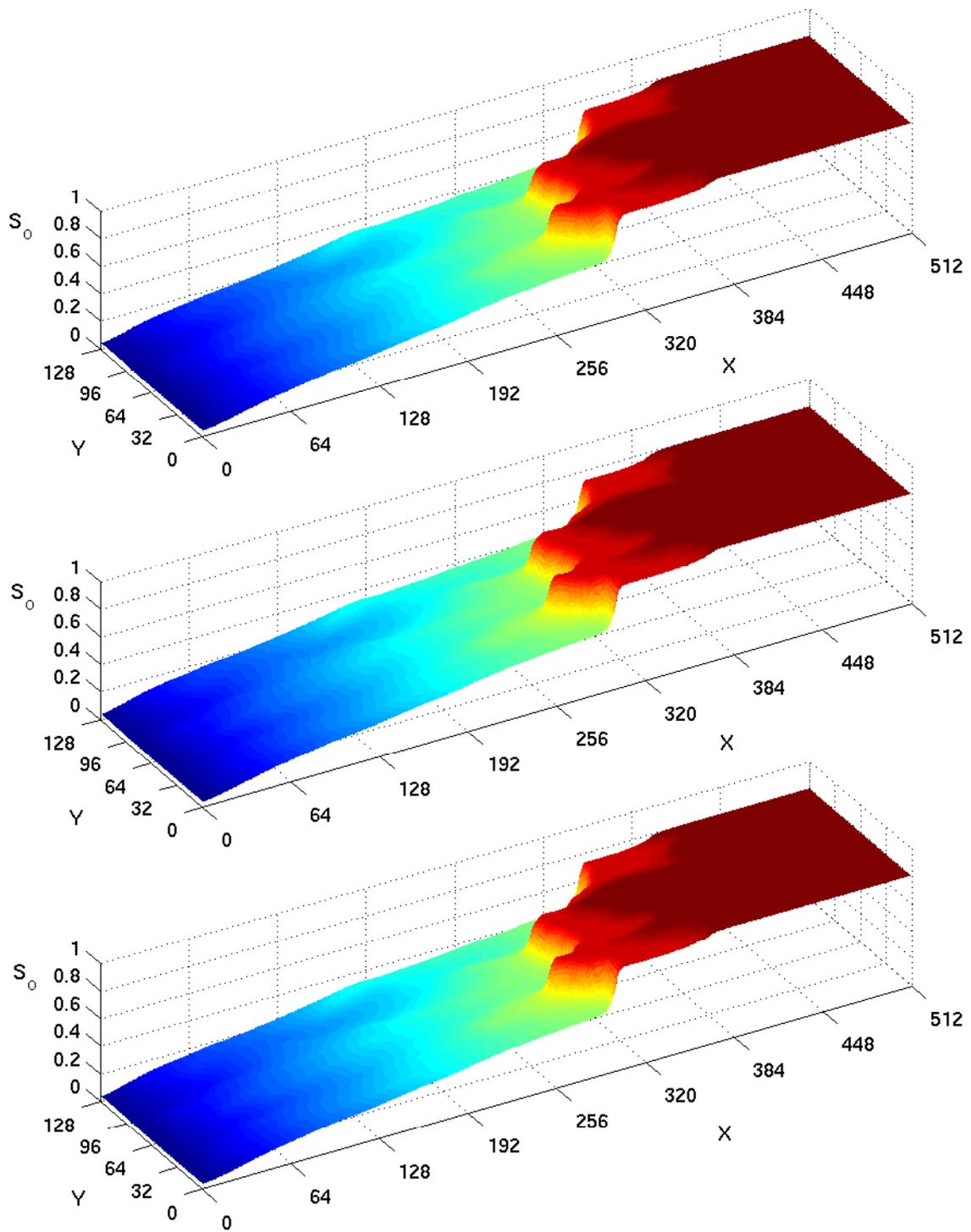
Figure 2: Oil saturation surface plots corresponding to the simulation reported in Figure 1.
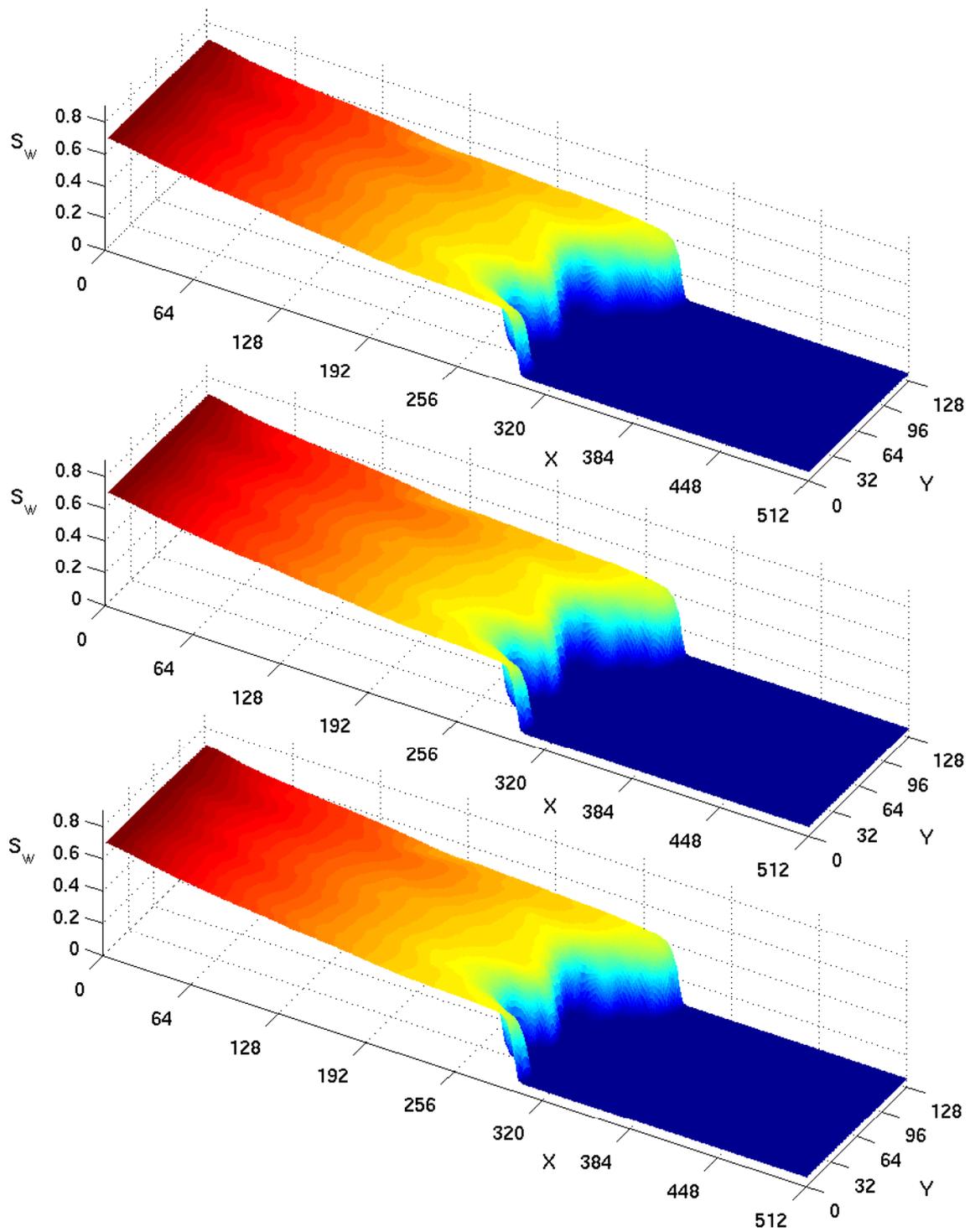
Figure 3: Water saturation surface plots corresponding to the simulation reported in Figure 1.

We have also performed a numerical computation on a two-dimensional reservoir $\Omega = [0, 512] \times [0, 512]$, with aspect ratio $X/Y = 1$ and discretized on a uniform grid of 512×512 cells. In this simulation, reported in Figure 4, we present numerical results with the same parameters and initial data described above, in which a transitional wave appears. We take the following time-step relation $\Delta t_p = \Delta t_d = 200 \Delta t_c$ with an OpenMP implementation.

For the numerical experiments reported in Figures 1-4, we note that even for the largest time-step relation the small and large scale features of the flow are accurately captured. In particular, the sharp gradients of the leading front as well as the viscous fingers due to the interaction between viscous forces and rock heterogeneity were simulated.

Moreover, the combination of the largest time-step relation with an OpenMP implementation might translates into a significant reduced computational effort to produce numerical solutions within a given accuracy requirement.

Figure 5 display results for saturation surface plots of gas (top) and water (bottom) as a function of distance in a two-dimensional reservoir $\Omega = [0, 512] \times [0, 512]$, with aspect ratio $X/Y = 1$ and discretized on a uniform grid of 512×512 cells at the time 650 days of simulation. The viscosities of the fluids are $\mu_o = 2.0$, $\mu_w = 1.0$, and $\mu_g = 0.2$. In this simulation a different mixture of fluids was used: 63 % of water and 37 % of gas is injected at constant rate along the left horizontal boundary, $x = \{0\}$ and $y \in [0, 512]$, of the computational region and "no-flow" conditions are imposed on the horizontal boundaries, $y = \{0, Y\}$ and $x \in [0, 512]$. Initially, the resident composition of fluid in the reservoir is a mixture of 1% water, 1% gas, and 99% oil.

The solutions shown in Figure 5 comprise (from right to left) a "shock" and a composite "shock"-rarefaction wave structure (5), as it would be expected from the associated Riemann solution for this one-dimensional three-phase flow problem in a porous medium.

In addition, the OpenMP algorithm for porous media flow reported in this work resolves with efficiency the "traveling wave" front of the fast leading "shock" wave followed by a "shock"-rarefaction wave, as expected from the analysis of this model (5), yielding a verification of our computations. Here, we also achieve similar computational time saving by comparisons with reliable calculations of a serial code as described above.

5. OpenMP results

The code program for the numerical solution of the three-phase flow system (3), (5) and (6)-(11) has more than ten modules but only three of them have time-consuming calculations and iterations. These modules have been modified by inclusion of OpenMP directives to improve performance. The modules are: "**convection**" for the nonlinear hyperbolic conservation laws (6)-(8), "**diffusion**" for the parabolic equations (9)-(11), and "**pressure-velocity**" for the elliptic equation (3) for oil pressure and Darcy velocity calculations.

It is worth mentioning that our serial original source code has some level of optimization work by means of the Intel VTune Performance Analyzer. It was used as a preliminary tool to identify and assemble some functions and loops in order to reduce data dependencies in the serial operator splitting algorithm as well as to minimize the memory access time in a generic manner. We state that the computational modeling of three-phase algorithm is a work-in-progress in the sense that more physical phenomena must be taken into account (e.g., compressibility and hysteresis) thereby leading to a more complex algorithm to be optimized in advance. Thus, our priority was to attain computational efficiency in a shared-memory architecture and to reach beyond what single core processors can do, but keeping the clarity of the original serial code.
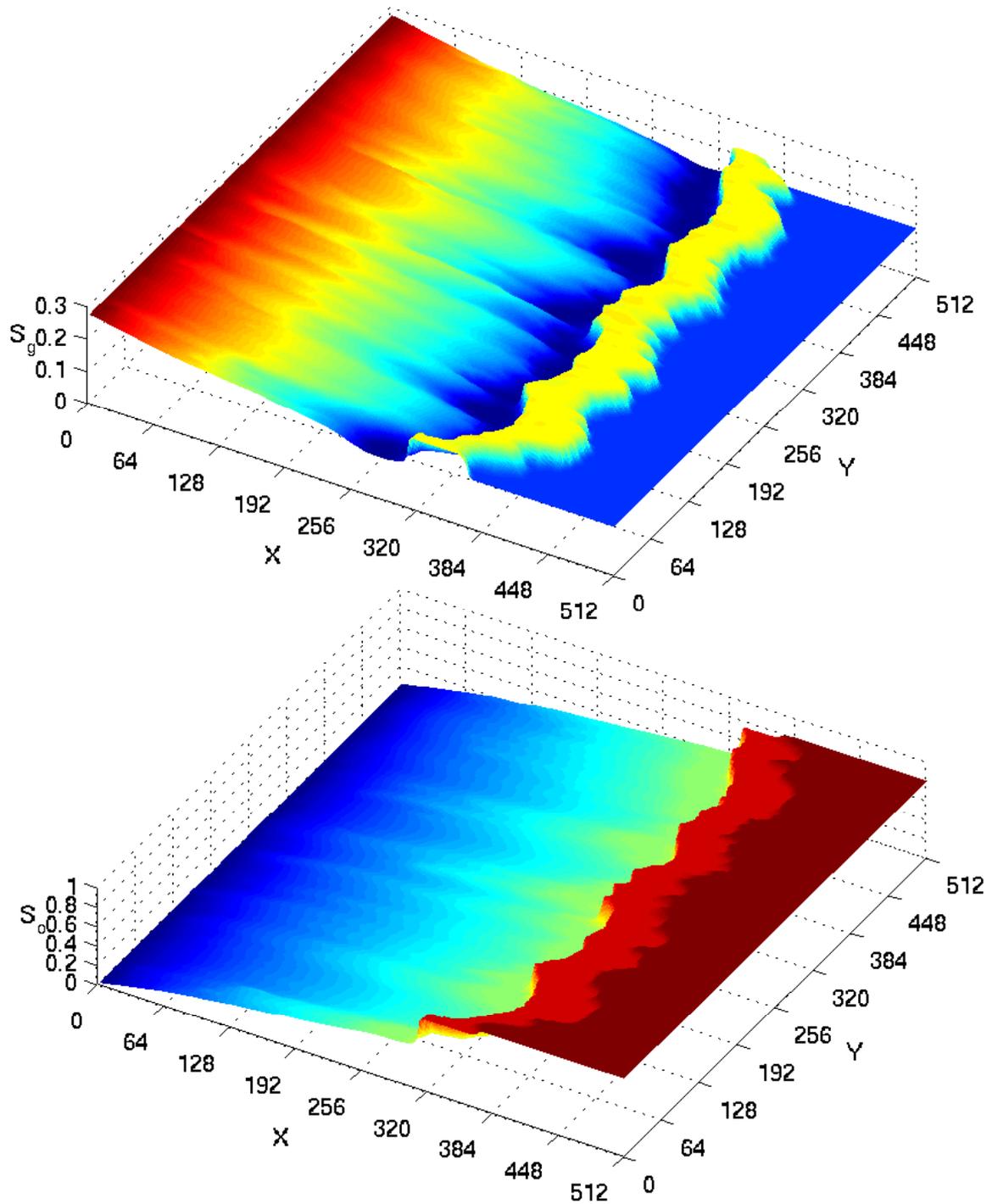
Figure 4: Gas (top) and oil (bottom) saturation surface plots are shown for a two-dimensional 512m × 512m heterogeneous reservoir at 800 days of simulation time. The same parameters and initial data in which a transitional wave appears in the solution of (3), (5) and (6)-(11) was used. According to the results, we have achieved a effective OpenMP performance improvement of 15%.
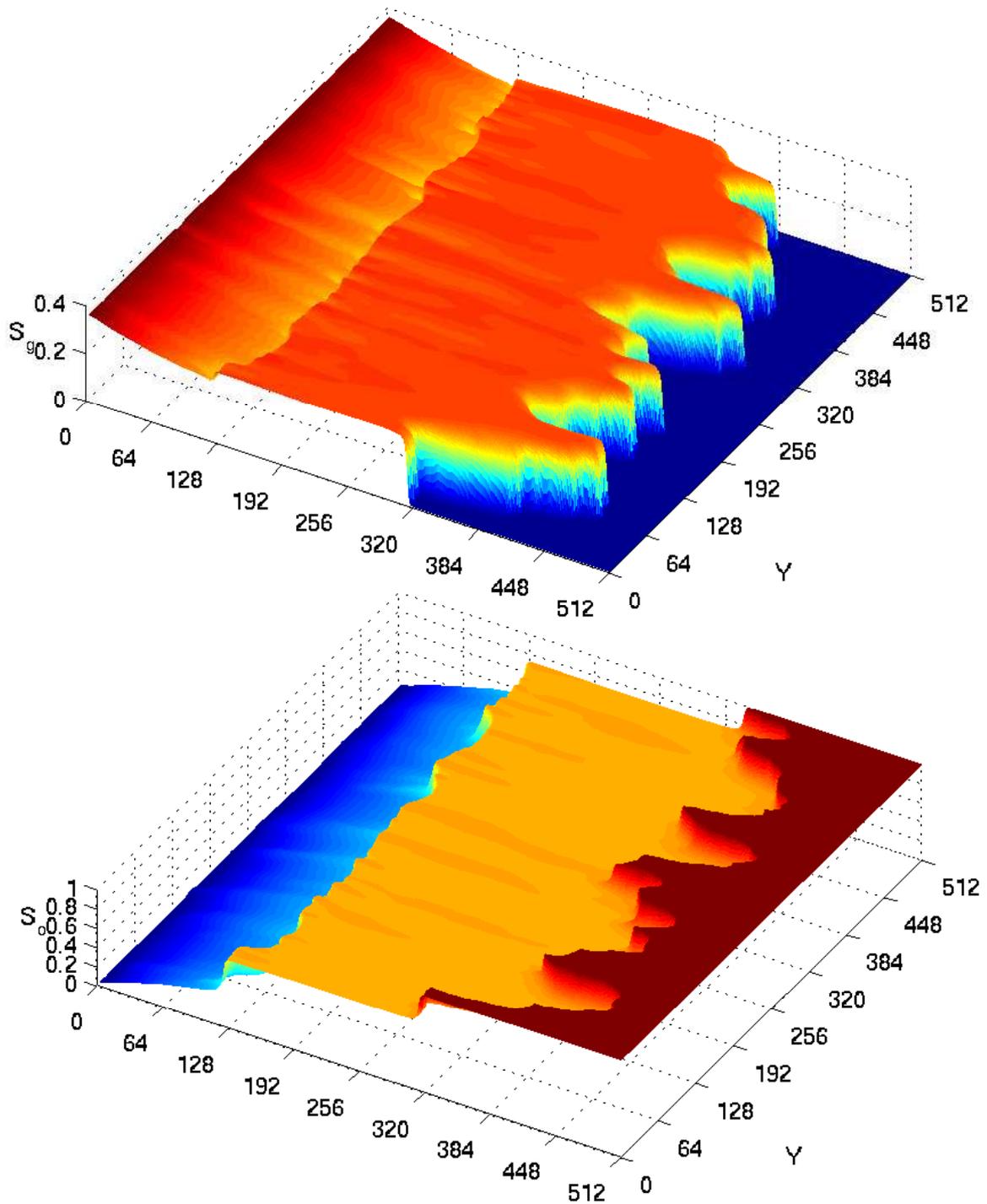
Figure 5: Gas (top) and oil (bottom) saturation surface plots are shown for a two-dimensional 512m × 512m heterogeneous reservoir at 650 days of simulation time in which combines an efficient OpenMP implementation with an operator splitting algorithm.

Following standard OpenMP directions, we get wrong numerical results in our first attempt to include the compiler directive "omp parallel for" in some loops, although with no compilation or execution error messages. Based on our experience with a complex code, a straightforward inclusion of "pragmas" with respect of scope variables, such as "private" and "shared" is not sufficient for an "automatic" OpenMP parallelization.

Others problems we have encountered were due to the OpenMP operator "reduction", when it was applied to non-trivial loops. Besides that, it was also needed include compiler directives, such as "critical" and "atomic" statements, to avoid race conditions. As an example, we show bellow a non-trivial loop with OpenMP pragma:

```
#pragma omp parallel for shared(MAX) private(i,j)
for(i=1 ; i<nx ; i++)
    for(j=1 ; j<ny ; j++){
        a = p[i][j]; b = p[i][j+1];
        c[i][j]=(2.0*a*b)/(a+b);
 #pragma omp atomic
        if (c[i][j] > MAX) MAX = c[i][j];
        }
```

It is worth to mention, however, that a more careful analysis of some loops with respect to both consistency and data coherency, reveal that inclusion of OpenMP directives might cause worse runtimes. In our case, this was clearly observed in computers with a small L2 cache (see Table 1). For instance, we note that the negative speedup for LORENZ can be explained because of small clock and cache size of its processor.

Table 1: Intel Multi-Core Machines

| Computer | 100 days | | | 1000 days | | |
	serial	omp (#cores)	speedup %	serial	omp (#cores)	speedup %
NEHALEM	0.22	0.18 (2)	21.87%	5.41	5.07 (8)	6.86%
RYGAR	0.43	0.29 (3)	47.76%	10.45	6.86 (8)	52.23%
LORENZ	0.62	0.40 (2)	53,45%	9.39	10.27 (2)	-8.52%
TEMPEST	1.24	0.82 (32)	50.83%	21.80	16.09 (32)	35,44%

In order to better evaluate the impact of using OpenMP, we have performed tests by means the use of different computer architectures and processors for the numerical solution of the three-phase flow equations. We also used both *Intel C/C++ Compiler* 11.0 and *Gnu GCC* 4.1.2. In all computers are running CentOS 5.3 as Operational System. The computers used were:

NEHALEM – Intel Xeon X5560 2.80GHz, 8MB cache, 12GB, 8 cores.
RYGAR – Intel Xeon X5450 3.0GHz, 6MB cache, 32GB, 8 cores.
LORENZ – Intel Core 2 6400 2.13GHz, 2MB cache, 4GB, 2 cores.
TEMPEST – AMD Opteron 8350 1GHz, 512KB cache, 64GB, 32 cores.

The first numerical test was done with two differents simulation times: 100 and 1000 days. The objective here was to evaluate how many cores could reach a lower time and a better speedup. As shown in Table 1, for a problem with simulation time of 100 days the lowest time was reached with a few cores. On the other hand, for a problem with a simulation time of 1000 days the lowest time was reach with all cores. The exception was the machine with a AMD Opteron (TEMPEST), where the two best times was reached with all cores. In Table 1, all

simulation times are in hours with the number of cores in parentheses. The values shown in Table 1 is a result of an ensemble average with 10 simulations to avoid sampling errors.

Figure 6 shows the best speedup values obtained in our numerical simulations of the three-phase flow system. Due to the nonlinear characteristic of the differential equations, we remark that simulations times 100 and 1000 days could not be use to evaluate in more details the scalability of our OpenMP implementation on the different computer architectures listed above.
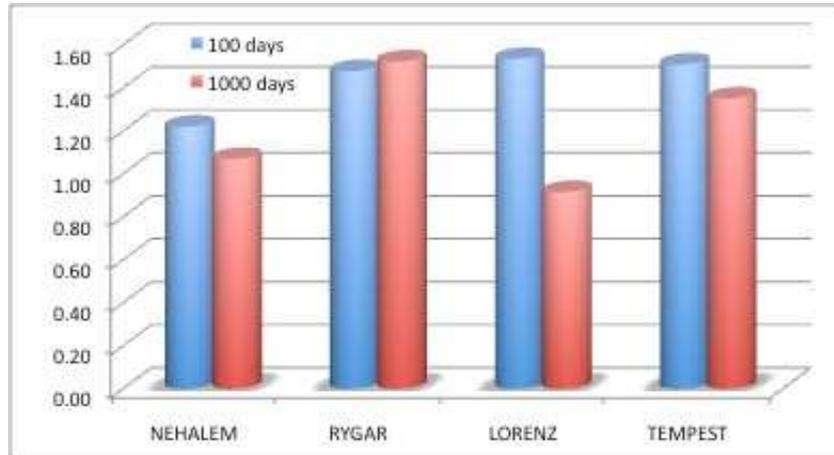

Figure 6: Best speedup values.

In the second test we were concerned to evaluate the use of OpenMP with the *Intel C/C++ Compiler Version* 11.0 and *Gnu C/C++ Compiler Version* 4.1.2. The **TEMPEST** machine was not considered since it has a AMD processor. For the *Intel C/C++ Compiler* we used the following optimizations flags "-openmp -ipo -fast -pc80 -fPIC" and for the *Gnu C/C++ Compiler* we used "-O2 -fopenmp".

Table 2 shows wall clock time results (i.e., the actual time taken by a computer to complete a task), and Figures 7 and 8 show, respectively, wall time comparative studies with respect to architecture, and Intel and Gnu compilers. We note that the lack of TEMPEST wall clock time measures with Intel compiler shown in the Table 2, and Figures 7 and 8, are due to absence of optimization options for AMD processors.

Table 2: Intel versus Gnu Compiler

Computer	Intel ICC			Gnu GCC		
	OpenMP	Serial	SpeedUp	OpenMP	Serial	SpeedUp
NEHALEM	5.06	5.41	6.91 %	6.68	8.83	32.19 %
RYGAR	6.86	10.44	51.75 %	8.90	11.12	24.94 %
LORENZ	10.27	9.39	-8.56 %	11.90	13.57	14.03 %

Based on the results reported in Table 1-2 and Figures 7-8, one can easily notice that the Intel compiler is faster then the Gnu compiler on all architectures. This speedup was 63.15% in NEHALEM, 6.48% in RYGAR, and 44.43 % in LORENZ.

The high performance level obtained by the state-of-the-art NEHALEM processor with the Intel compiler can be better explained by the fact that there is no optimization with a GNU compiler for this particular processor yet.

Although an apparently low parallel efficiency obtained from results of Table 2 for NE-HALEM, RYGAR, and LORENZ machines (about 15 %), we stress that inclusion of OpenMP directives in our code produce satisfactory results with minimum of code changes.
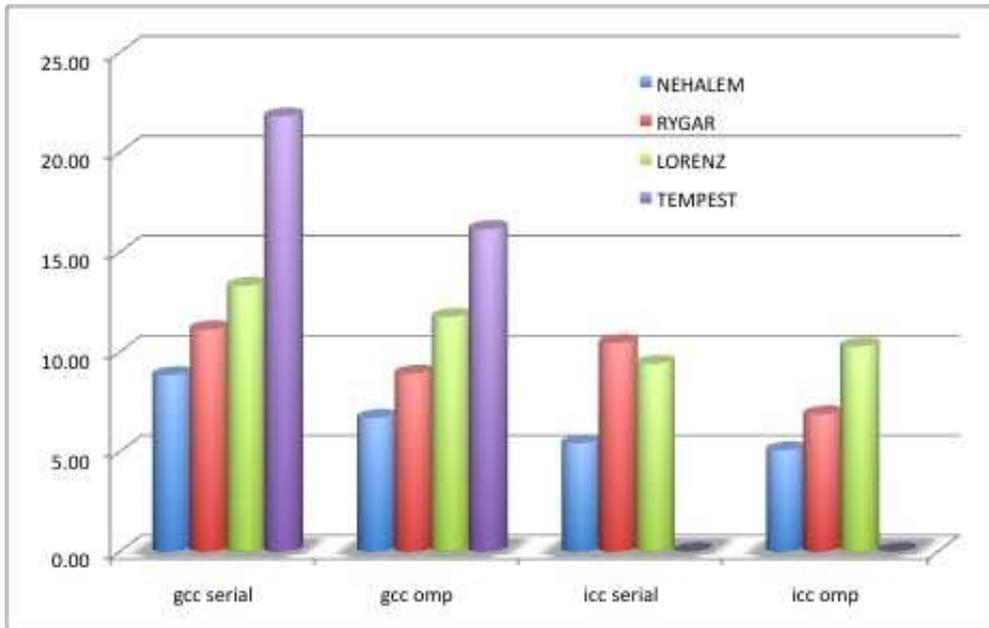
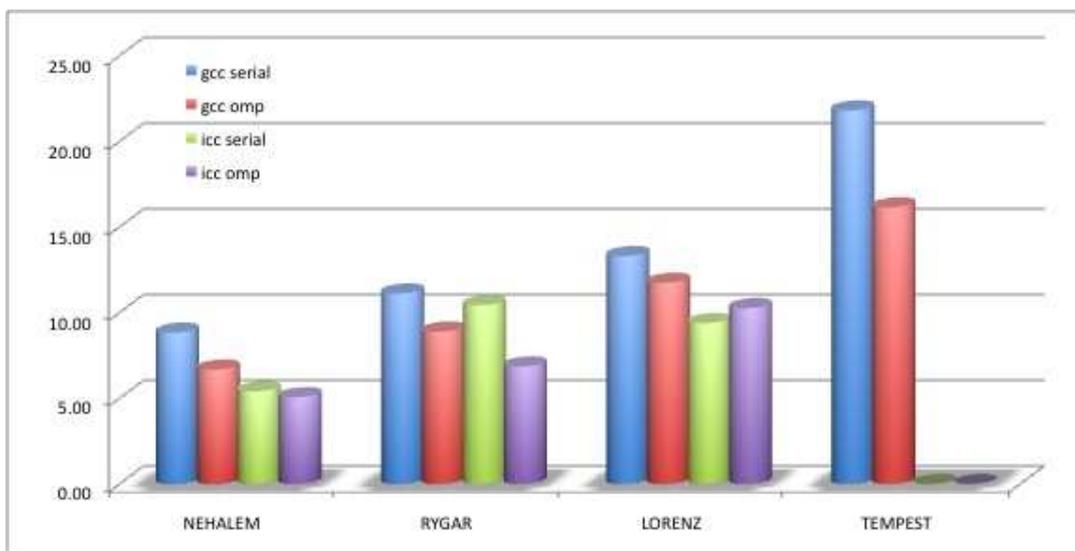**Figure 7:** Wall time results by compiler.

Figure 8: Wall time results by architecture.

For instance, we get reductions of 6.91% (21 minutes over 5 hours) for NEHALEM machine and of 36% (3 hours and 35 minutes over 10 hours and 26 minutes) for RYGAR as well. These results in both cases represent an effective performance since the time-consuming of our computations is the order of days. We remark that the NEHALEM machine is the state-of-the-art of an Intel processor.

6. Conclusions

OpenMP coding on the multicore computer architecture seems to give a start of a beneficial transition from sequential to parallel computation for non-experts researchers and students, outside of the HPC research arena, who intend to parallelize existing codes or to develop new ones in a simple manner, with no excessive effort by the user.

We have verified, based on the obtained results in this work that a speedup can be easily achieved by means of a portable and natural OpenMP parallelization with minimum changes to the serial version.

The reduced hours in an OpenMP programing environment allow us to produce an efficient, high level, and reliable parallel code and this is something that should be highlighted since the data structure of our serial code exhibit levels of complexity.

Exploiting the combination of OpenMP efficiency with an operator splitting technique it has shown that one can get a significant performance for the numerical solution of three-phase flow system through heterogeneous petroleum reservoirs engineering.

As future work, it is expected an application of this numerical simulator to a scientific investigation of the up scale problem for three-phase, immiscible flow in heterogeneous porous media to understand the interplay between nonlinearity and heterogeneity imposed by the geology for applications to petroleum engineering and hydrology by means of large-scale, although expensive and time-consuming, reliable Monte Carlo simulations based on a stochastic modelling approach of the flow equations.

From a computational point of view, the authors believe that is needed an in-depth optimization work to increase efficiency in order to avoid or reduce low speedups results as those shown in Table 2.

REFERENCES

[1] Abreu, E., Furtado, F., Marchesin, D., and Pereira, F., 2004, Transitional Waves in Three-Phase Flows in Heterogeneous Formations. In: C. T. Miller, M. W. Farthing, W. G. Gray and G. F. Pinder (Eds) *Computational Methods for Water Resources, Series: Developments in Water Science*, **I**, 609–620.

[2] Abreu, E., Douglas, Jr.J, Furtado, F., Marchesin, D., and Pereira, F., 2006, Three-Phase Immiscible Displacement in Heterogeneous Petroleum Reservoirs. *Mathematics and computers in simulation*, **73**(1-4), 2-20.

[3] Abreu, E., Douglas, J., Furtado F., Pereira F., 2008, Operator Splitting Based on Physics for Flow in Porous Media. *International J. of Computational Science*, **2**(3), 315–335.

[4] Abreu, E., Douglas, J., Furtado F., Pereira F., 2008, Operator Splitting for Three-phase Flow in Heterogeneous Porous Media. *Communications in Computational Physics*, **6**, 72-84.

[5] A. Azevedo, A. Souza, F. Furtado, and D. Marchesin, 2009, The riemann solution for three-phase flow in a porous medium, *in Proceeding of the 12th International Conference*

on Hyperbolic Problems: Theory, Numerics and Applications, Maryland, College Park, USA, 2009, Center for Scientific Computation and Mathematical Modeling (CSCAMM).

[6] Berre, I., Dahle, H. K., Karlson, K. H., and Nordhaug, H. F., 2002, A streamline front tracking method for two- and three-phase flow including capillary forces. *Contemporary Mathematics: Fluid flow and transport in porous media: mathematical and numerical treatment*, **295**, 49–61.

[7] Chen, Z., and Ewing, R. E., 1997, Fully-discrete finite element analysis of multiphase flow in ground-water hydrology. *SIAM J. on Numerical Analysis*. **34**, 2228–2253.

[8] Chen, Z. and Ewing, R. E., 1997, Comparison of various formulation of three-phase flow in porous media. *J. Computational Physics*, **132**, 362–373.

[9] Chorin, A.J., Hughes, T.J.R., McCraken, M.F., and Marsden, J.E., 1978, Product Formulas and Numerical Algorithms. *Comm. Pure Appl. Math.*, **31**, 205–256.

[10] Corey, A., Rathjens, C., Henderson, J., and Wyllie, M., 1956, Three-phase relative permeability. *Trans. AIME*, **207**, 349–351.

[11] Douglas, Jr.J., Furtado, F., and Pereira, F., 1997, On the numerical simulation of water-flooding of heterogeneous petroleum reservoirs. *Computational Geosciences*, **1**, 155–190.

[12] Dria, D.E., Pope, G.A, and Sepehrnoori, K., 1993, Three-phase gas/oil/brine relative permeabilities measured under CO_2 flooding conditions. *SPE 20184*, 143–150.

[13] R. E. Ewing, 1983, Problems arising in the modeling of processes for hydrocarbon recovery. In R. E. Ewing, editor, SIAM, *Frontiers in Applied Mathematics*, **1** Philadelphia.

[14] Gerritsen, M.G., and Durlofsky, L.J., 2006, Modeling fluid flow in oil reservoirs. *Annual Review of Fluid Mechanics*, **37**, 211–238.

[15] Isaacson, E., Marchesin, D., and Plohr, B., 1990, Transitional waves for conservation laws. *SIAM J. on Mathematical Analysis*, *21*, 837–866.

[16] Juanes, R., and Patzek, T. W., 2004, Three-Phase Displacement Theory: An Improved Description of Relative Permeabilities. *SPE Journal*, **9**(3), 302–313.

[17] Karlsen, K. H, and Risebro, N. H., 2000, Corrected operator splitting for nonlinear parabolic equations. *SIAM J. on Numerical Analysis*, **37**(3), 980-1003.

[18] Karlsen, K. H, Lie, K.-A., Natvig, J. R., Nordhaug, H. F., and Dahle, H. K., 2001, Operator splitting methods for systems of convection-diffusion equations: nonlinear error mechanisms and correction strategies. *J. of Computational Physics*, **173**(2), 636–663.

[19] Lekhov, A.V. and Shvarov, Y.V., 2002, On the rate of radionuclide migration in groundwater. *Water Resources*, **29**(3), 244–251.

[20] Li, B., Chen, Z., and Huan, G., 2003, The sequential method for the black-oil reservoir simulation on unstructured grids. *J. of Computational Physics*, **192**, 36–72.

[21] Marchesin, D. and Plohr, B. J., 2001, Wave structure in WAG recovery. *SPE Journal*, **56480**(6) no.2, 209-219.

[22] S. A. Mathias, P. E. Hardisty, M. R. Trudell and R. W. Zimmerman, 2009, Approximate solutions for pressure buildup during CO_2 injection in brine aquifers. *Transp. Porous Media*, **79**, pp. 265-284.

[23] Nessyahu, N., and Tadmor, E., 1990, Non-oscillatory central differencing for hyperbolic conservation laws. *J. of Computational Physics*, **87**(2) 408–463.

[24] OpenMP, *www.openmp.org*, accessed in june, 2009.

[25] Peaceman, D. W., 1977, Fundamentals of Numerical Reservoir Simulation. Elsevier, Amsterdam.

[26] Pruess, K., 2003, Numerical simulation of leakage from a geological disposal reservoir for CO_2, with transitions between super- and sub-critical conditions. In *TOUGH Symposium 2003, Lawrence Berkeley National Laboratory, Berkeley, California, May 12-14*, 1–8.

[27] R. Rabenseifner, G. Hager, G. Jost, R. Keller, 2008, Hybrid MPI and OpenMP Parallel Programming. *International conference on high performance computing, networking, storage and analysis*, 89 pages.

[28] Stone, H. L., 1970, Probability model for estimating three-phase relative permeability. *Petrol. Trans. AIME*, **249**. *JPT*, **23**(2), 214–218.