Parallelizing Black Oil Reservoir Simulation Systems for SMP Machines

Fabrício A. B. Silva[1,2], Ernesto P.Lopes[3], Eliana P. L. Aude[1], Flavio Mendes[1],Henrique Serdeira[1], Julio Silveira[1]

Núcleo de Computação Eletrônica - NCE[1]
Federal University do Rio de Janeiro
{fasilva, elaude, fmendes, serdeira, julio}@nce.ufrj.br

Graduate Program in Informatics[2]
Santos Catholic University
fabricio@unisantos.br

COPPE[3]
Federal University of Rio de Janeiro
lopes@cos.ufrj.br

Abstract

In this paper we discuss the parallelization for SMP machines of black oil reservoir simulation programs. A reservoir simulator is a sophisticated computer program used to predict the future performance of a reservoir based on its current state and past performance. SMP architectures are composed of homogeneous microprocessors sharing access to the main memory. Due to advancements in microprocessor technology and the shared memory access, the best parallelization strategy for SMP machines may differ significantly from strategies used previously in MPP machines. In this paper we present a parallel implementation optimized for SMP machines of BOAST, a three-phase, three-dimensional black oil simulator. The communication library used was LAM-MPI. We used the domain decomposition technique to parallelize the two more computing-intensive subprograms (routines) of BOAST, which take about 80% of the total execution time of a typical simulation in a SMP machine. The speed-up obtained show that the chosen parallelization strategy has good load balancing and low communication overhead, achieving near-optimal performance in several simulations.

1. Introduction

A reservoir simulator is a sophisticated computer program used to predict the future performance of a reservoir based on its current state and past performance. It is the main computational tool used by petroleum engineers to explore methods for increasing the ultimate recovery of hydrocarbons from a reservoir. The simulator solves the system of partial differential equations describing multiphase fluid flow (oil, water, gas) in a porous reservoir rock.

Simulation of large reservoirs or entire fields containing millions of grid blocks and as many as a thousand wells entails solution of a large set of differential equations. The computer resources required to solve the large set of equations governing multiphase flow in the reservoir can grow rapidly depending on the reservoir size, number of grid blocks and the type of steeping scheme used on the model. The amount of computation involved and the structure of the simulation make this problem suitable for parallel implementation.

The mechanics of the simulations can be described as follows: the reservoir is first divided into segments using X,Y,Z axes. Rock and fluid properties are then assigned to each block to describe the reservoir system. Computations are carried out for all phases in each block at discrete time steps. The results usually consist of production volumes and rates, pressure and saturation distributions, material balance errors, and other process specific information provided at selected time steps.

We based this study on BOAST [3], a multi-phase, three dimensional black oil simulator developed by the US Department of Energy. The BOAST program simulates isothermal, Darcy flow in three dimensions. It assumes reservoir fluids can be described by three fluid phases (oil, gas, and water) of constant composition with physical properties that depend on pressure only. These reservoir fluid approximations are acceptable for a large percentage of the world's oil and gas reservoirs.

Proceedings of the 36th Annual Simulation Symposium (ANSS'03)
1080-241X/03 $17.00 © 2003 IEEE

In this paper we present a parallel implementation of BOAST using MPI [8] as communication library, optimized for SMP machines. The implementation of MPI used on experiments was LAM-MPI [5]. We used the domain decomposition technique to parallelize the two more computing-intensive subprograms (routines) of BOAST, which take about 80 % of the total execution time of a typical simulation in a SMP machine. Results show that the parallelization strategy used minimizes load imbalance and has small communication overhead.

This paper is organized as follows: In section 2 we present a detailed description of BOAST. The main characteristics of SMP architectures are discussed on section 3. The parallelization the BOAST is detailed on section 4. Section 5 shows simulation results and the corresponding analysis. Related work is described in section 6, and section 7 contains our final remarks.

2. Description of BOAST

In our experiments we used a modified version of BOAST, BOAST-VHS, described in [3]. BOAST-VHS employs the implicit pressure - explicit saturation (IMPES) formulation for solving its system of finite-difference equations. The IMPES method finds the pressure distribution for a given time step first, then the saturation distribution for the same time step. Once the pressure equation is solved implicitly, the saturations are explicitly updated. Please see [2,3,7] for a throughout discussion of the set of governing differential equations, including the pressure equation.

The finite difference form of the pressure equation can be expressed as follows:

$$AE_i P_{i+1} + AW_i P_{i-1} + AN_j P_{j+1} + AS_j P_{j-1} + AB_k P_{k+1} + AT_k P_{k-1} + E_{i,j,k} P_{i,j,k} = b_{i,j,k} \quad (1)$$

Where AE, AW, AN, AS, AB, AT, E and b are dense three-dimensional matrices. P is the pressure matrix to be calculated. This equation can be rewritten as:

$$AP = B \quad (2)$$

Where **A** is a heptadiagonal matrix.

BOAST-VHS employs the line-successive, over-relaxation (LSOR) iterative solution technique to solve the system of pressure equations. This method requires less storage and usually is faster for larger problems than direct methods. However, the CPU time for the iterative methods depends on the type of problem to be solved and the selection of the iterative or overrelaxation parameter, which weights the influence of the previous iteration in the computation of the new pressure value.

To solve the linear matrix system of equations Ap = b the LSOR method used in BOAST is started by splitting the coefficient matrix A as follows:

$$A = G + H \quad (3)$$

Based on equation 3 the following iteration is defined:

$$Gp_m = b - Hp_{m-1} \quad (4)$$

For an x-line LSOR (used in BOAST-VHS) the linear matrix system of equation reduces to the following tri-diagonal system:

$$AW_{i,j,k} p_{i-1,j,k} + E_{i,j,k} p_i + AE_{i,j,k} p_{i+1} = b_{i,j,k} - Hp_{i,j,k} \quad (5)$$

This tridiagonal system is then directly solved by BOAST.

3. SMP Architectures

SMP (symmetric multiprocessing) architectures are composed of multiple homogeneous processors, every processor sharing access to a common main memory. The access time to any main memory position is the same for every processor. On SMP systems communications among processors occur normally through the shared main memory.

In symmetric multiprocessing, the processors also share the I/O bus. A single copy of the operating system is in charge of all the processors. SMP systems do not usually exceed 16 processors, although some of the latest machines released by several UNIX vendors support up to 64 processors.

A SMP operating system has only one run queue. All scheduling is done by a common scheduler, which assigns work to processors in symmetric fashion, on the basis of CPU availability.

3. Parallelization of BOAST

Analysis of CPU time consumed by BOAST subprograms showed that three segments of the FORTRAN code (SOLMAT,LSOR and LTRI) account for about 80% of CPU time (table 1). These measurements were made using a serial version of the BOAST-VHS program running in a SMP machine with 4 Intel Pentium III Xeon processors running at 800MHz and

4 gigabytes of RAM memory. The operating system used was Linux.

Table 1. Percentage of the total execution time of the routines SOLMAT and LSOR

Subprogram	Average percentage of total execution time
SOLMAT	60
LSOR/LTRI	20

SOLMAT is the subprogram responsible for computing pressure equation coefficients. LSOR implements the linear successive overrelaxation method and LTRI is called by LSOR and is responsible for solving the tri-diagonal system generated by BOAST. The measurements in table 1 were determinant in defining the strategy of parallelization, and it is worth noting that they may differ depending on the architecture. For instance, the same measurements made on the Cray T3D, for the same fluid property and relative permeability data, differ significantly [11]. In that case the pair of routines LSOR/LTRI were responsible for about 80 % of the total execution time. As a consequence, the strategy of parallelization implemented in this work is different from the one adopted at [10,11], where a MPP machine (Cray) was used.

For the parallelization of BOAST-VHS using MPI we used the domain decomposition method for both the SOLMAT and LSOR subprograms. Domain decomposition method has been conventionally used to solve a linear matrix system of equations (Ap=b) on multiple processors. In this approach, the grid cells of the reservoir system are subdivided onto different processors. Then, the data communication occurs for the boundary grid cells, which are dependent on the grid cells of other processors. Examining the structure of regular grid system used in the finite difference method, the domain can be decomposed using one, two or three-dimensional partitioning. The decomposition method used in this work was one-dimensional partitioning. Is its possible to partition grid cells along X,Y and Z axes. In this paper we present an implementation where the partition is done along the Z-axis, in order to minimize communication overhead. This is due to the way FORTRAN stores an array in memory, in column major order, as exemplified in figure 1 for a three-dimensional array. By decomposing the three-dimensional matrix along the Z-axis, memory-to-memory copies are avoided and the communication overhead is reduced [1]. However it should be stressed that it is possible to partition grid cells along any of the three axes using the strategy proposed in this paper. Partition along the X and Y axes would incur in memory to memory copies before each communication, therefore increasing communication overhead.

(1,1,1)	(2,1,1)	(1,2,1)	(2,2,1)	(1,1,2)	(2,1,2)	(1,2,2)	(2,2,2)

Figure 1. This figure illustrates the way FORTRAN stores a three-dimensional array of dimensions 2 x 2 x 2 in memory. Memory addresses increase from left to right.

Both SOLMAT and LSOR have a main loop which encompasses almost all the computation of the respective subprogram. This main loop is subdivided into three loops, one for each dimension as shown in figure 2.

```
DO 200 K=1,KK
DO 200 J=1,JJ
DO 200 I=1,II
```

Figure 2. Main Loop in subprograms SOLMAT and LSOR

Where KK, JJ and II represent the size in grid blocks of the reservoir along the axis Z, Y and X respectively. In the parallelization of BOAST, each processor runs an instance of the program and the amount of computation of the outermost loop is divided among the processors, as shown in figure 3.

```
DO 200 K=BEGIN,END
DO 200 J=1,JJ
DO 200 I=1,II
```

Figure 3. Main Loop Parallelized

Where BEGIN and END are variables that depend on the MPI rank of each processor. These variables are computed by the parallelized version of BOAST in order to guarantee the evenly distribution of iterations among processors.

In the parallel version of BOAST, each processor runs its own modified copy of the BOAST program. Communication among neighbor processors occurs before any iteration of the iterative solver (for the LSOR subprogram) or at the end of the routine (SOLMAT). There is no need to communicate values between processors at the beginning of each routine, since each

processor enter the routine with the necessary values (matrices and variables) for the subprogram on that processor already updated. Therefore it was possible to minimize communication among processors, reducing communication overhead due to MPI calls. Note that it was not necessary to parallelize the routine LTRI, since it called by LSOR inside its main loop.

4. Simulation Results

In this section we show simulation results for the parallel reservoir simulation system using MPI as communication library. The machine used for measurements was a SMP system with 4 Intel Pentium III Xeon processors running at 800 MHz and 4 gigabytes of RAM memory. The operating system used as Linux. The implementation of the MPI library used was LAM-MPI [5], compiled with the transport *usysv* activated. This option delivers best performance for MPI systems running on SMPs, since all communications among processors in the same node are made through shared memory. Two different examples were simulated:

Example 1 – In this example a two-dimensional pressure depletion production of 10 years from a horizontal well of 1200 ft is simulated. We used the data available in [3] and extended the reservoir in the three dimensions. We have simulated the following grid sizes: 125.000 grid blocks (equivalent to a grid of dimensions 50 x 50 x 50) and 1.000.000 grid blocks (equivalent to 100 x 100 x 100). Due manly to the development of parallel computers, the total number of grid blocks used in a typical simulation increased from thousand to millions in the past decade [14]. That is the main reason why we choose to run simulations with such large number of grid blocks.

For a grid size of 50 x 50 x 50, or 125000 grid blocks, the total execution time for the serial version of BOAST was 803 seconds and for the parallel version was 341 seconds (figure 4). For this simulation, the speed-up of the parallel version over the serial version is 2,35.

When simulating one million grid blocks (figure 5) the results obtained are shown at figure 5. The total execution time for the parallel version was 2930 seconds, and for the serial version was 6283 seconds. The speed-up in this case was 2,14.

We have also measured the amount of time spent in the subprograms SOLMAT and LSOR by both the serial and parallel versions of BOAST. Results for a grid size of 1.000.000 grid blocks are shown in figures 6 and 7.

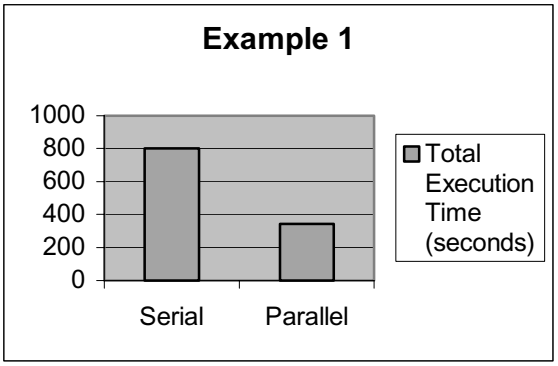

Figure 4 - Example 1, total execution time, 125.000 grid blocks.

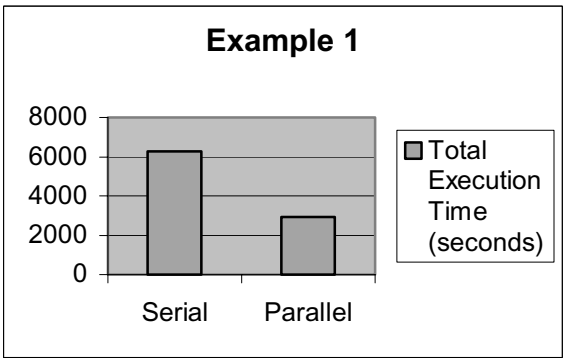

Figure 5 - Example 1, total execution time, 1.000.000 grid blocks.

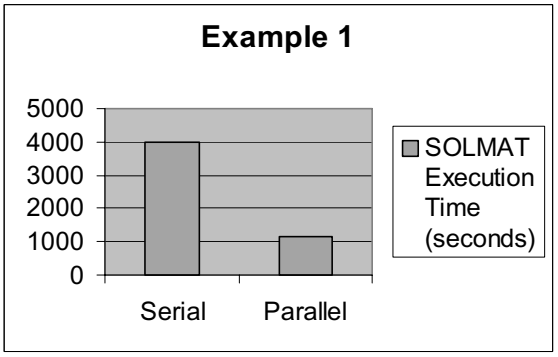

Figure 6 - Time spent in subprogram SOLMAT, example 1, 1.000.000 grid blocks

The time spent in the SOLMAT subprogram for the serial version was 3968 seconds. The parallel version spent 1162 seconds at the SOLMAT subroutine. The speed-up ratio only for this subroutine was 3,41 using four processors.

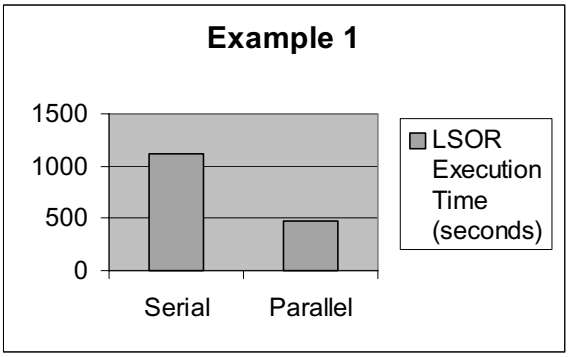

Figure 7 - Time spent in subprogram LSOR, example 1, 1.000.000 grid blocks

As shown in figure 7, the serial version spent 1111 seconds in the LSOR subroutine, while the equivalent time for the parallel version was 474 seconds. The speed-up for this routine was 2,34.

Example 2 – Example 2 consists in the simulation of a three-dimensional homogeneous reservoir model showing pressure depletion production from a 739 ft slant well. Fluid property and relative permeability are also available at [3]. The problem was simulated for three years. The same grid sizes used on example 1 were simulated for example 2.

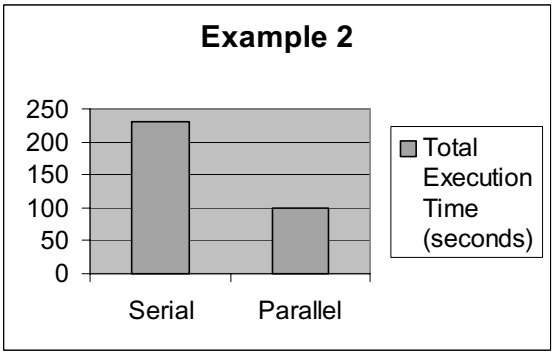

Figure 8 - Example 2, total execution time, grid size 50 x 50 x 50

For a grid size of 50 x 50 x 50 (125.000) blocks, the total execution time for the parallel version was 99 seconds, and for the serial version was 231 seconds, as shown in figure 8. The speed-up was equal to 2,33.

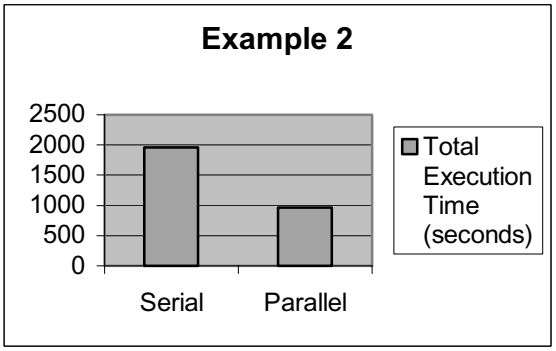

Figure 9 - Example 2, total execution time, grid size 100 x 100 x 100

When simulating a larger reservoir having 1.000.000 grid blocks with the same fluid property and relative permeability data, the total execution time for the serial version was 1955 seconds (figure 9). It was 960 seconds for the parallel version. For this execution, the speed-up was 2,03.

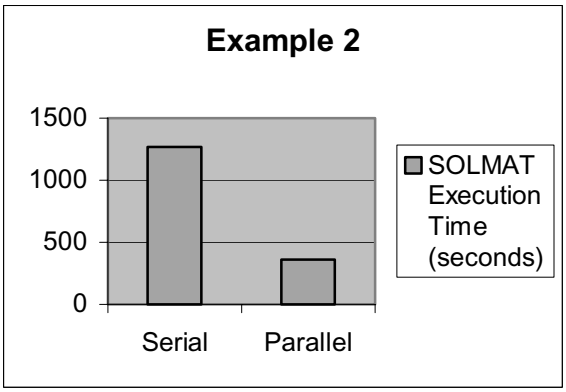

Figure 10 - Time spent in subprogram SOLMAT, example 2, grid size 100 x 100 x 100

We also measured the speed-up relative to the routines SOLMAT and LSOR. In the serial version of BOAST, the time spent executing the routine SOLMAT was 1270 seconds. It was 360 seconds for the parallel version, as shown in figure 10.

For the serial version, the time spent inside the LSOR routine, the other subprogram parallelized, was 233 seconds. It was 116 seconds for the parallel version.

All results generated by parallel versions were checked for accuracy with the ones generated by the serial version. Since we are parallelizing two subprograms that correspond to roughly 80 percent of the total execution time, the maximum achievable speedup in four processors is 2.5. The minimum parallel execution time in four

processors is equivalent to equally dividing 80 percent of total execution time among four processors (considering a linear speed-up), plus the execution time for the code that was not parallelized, considering no communication overhead. It is worth noting that the speed-up obtained for some executions were very near the maximum achievable linear speed-up (in four processors) of 2.5. Two factors are determinant in improving the speed-up of a parallel program: load imbalance and communication overhead. Since the load is perfectly balanced among processors, as each processor runs its own copy of BOAST, results in this section shown that the communication overhead in the parallel version is low.

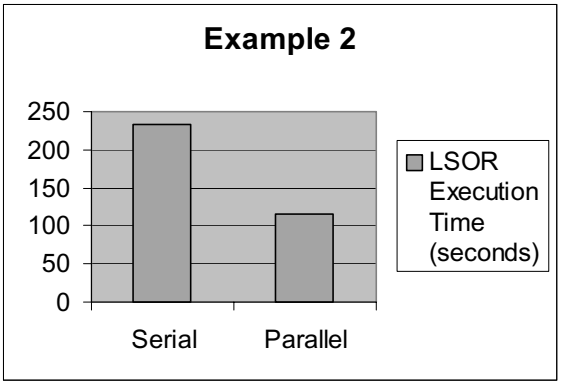

Figure 11 - Time spent in subprogram LSOR, example 2, grid size 100 x 100 x 100

5. Related Work

Kim [7] compared the performance of several iterative solvers applied to the black oil reservoir problem. The conclusion was that the use of LSOR in parallel, by splitting the coefficient matrix A among multiple processors [4], has performance comparable to powerful nonstationary parallel Krylov subspace methods such as TFQMR, which are much more complex to implement. Kim also developed computationally efficient discrete fracture model for fractured oil reservoir simulation.

Lu et al. [6] discuss a multi-block algorithm for an implicit black-oil model for a multi-phase simulator. Numerical experiments shown that the technique proposed was effective and scalable, and it is a promising method when multiple sub-domains are considered.

Wang et al. [10,11] described a parallel implementation of BOAST where the LSOR routine was substituted by a multi-grid [13] solver. They proved that a multi-grid solver can be more efficient in many situations. However, is it interesting to note that the percentage of the total execution time spent in the LSOR routine in their target architecture (Cray T3D) was greater than the time we measured in our SMP machine, for the same relative permeability and fluid property data. In our case, using a SMP machine, the utilization of a multi-grid solver would have smaller impact in the total execution time than it was the case in their work.

Wheeler et al. [12] present a domain decomposition procedure for treating multiphysics and multiscale simulation with multinumerical models. Numerical simulations were presented to illustrate several decomposition strategies. Zang e tal. [14] described the parallelization for the Cray T3E-900 of TOUGH2, a widely used numerical simulator of isothermal flows of multicomponent, multiphase fluids in three-dimensional porous and fractured media.

6. Conclusion

In this paper we described an efficient parallel implementation of a black oil reservoir simulation system for SMP machines. Beside the fact that MPI was used as communication library, the transport used on LAM-MPI is optimized of SMP machines and uses shared memory for communication between processors in the same node. Simulation results show that the speed-ups obtained are near the maximum speed-up possible, which indicates low communication overhead and good load balancing among processors.

New research directions that our group is actively pursuing are: the adaptation of the system to execute in a cluster of workstations; the development of a parallel solver based on multi-grid methods; implementation of multi-block algorithms in the simulator; the development of a new version of the parallel simulator for SMPs where parallelization is implemented through parallel compiler directives (such as, for example, OpenMP[9])

7. Acknowledgements

We would like to thank Finep and CNPq (Brazilian government) for the financial support.

8.References

[1] Aoyama, Y. and Nakano, J. *RS/6000 SP: Practical MPI Programming*, IBM Redbook, 1999

[2] Aziz,K. and Settari, A. *Petroleum Reservoir Simulation*, Applied Science Publishers, London, 1979

[3] Chang, M.M., Sarathi, P., Heemstra, R. J., Cheng A.M., Pautz, J. F., *User's Guide and Documentation Manual For "BOAST-VHS for the PC"*, Topical Report NIPER-542, 1992

[4] Hofhaus, J., Van de Velde, E.F. "Alternating-Direction Line Relaxation Methods on Multicomputers", *SIAM Journal of Scientific Computing(2)*, Society of Industrial and Applied Mathematics, March 1996, pp 454-478

[5] The LAM-MPI team, *http://www.lam-mpi.org/*, 2003

[6] Lu, Q., Peszynka, M. and Wheeler, M. F. A "Parallel Muli-Block Black-Oil Model in Multi-Model Implementation", *Proceedings of the 2001 SPE Reservoir Simulation Symposium*, Society of Petroleum Engineers, Houston, Texas, 2001

[7] Kim, J. G. *Advanced Techniques for Oil Reservoir Simulation: Discrete Fracture Model and Parallel Implementation*, Ph.D. Thesis, University of Utah, 1999

[8] The MPI Forum, *http://www.mpi-forum.org/*, 2003

[9] The OpenMP Forum, *http://www.openmp.org/*, 2003

[10] Wang, K. *Multi-Grid Solution for Petroleum Reservoir Simulation on Distributed Memory Processors*, M.Sc. Thesis, University of Alaska Fairbanks, 1995

[11] Wang, K, Ogbe D. O., Lawal, A. S., Morton D. J. "Multigrid Approach for Petroleum Reservoir Simulation on the Cray T3D", *Proceedings of the CUG Fall 95*, 1995

[12] Wheller, M.F., Arbogast T., Bryant S., Eaton J., Lu., Q., Peszynska, M, Yotov, I. "A Parallel Multiblock/Multidomain Approach for Reservoir Simulation", *Proceedings of the 1999 SPE Reservoir Simulation Symposium,* Society of Petroleum Engineers, Houston, Texas, 1999

[13] Wesseling, P. *An Introduction to Multi-Grid Methods*, John Wiley and Sons Ltd., England, 1991

[14] Zang, K., Wu Y.S., Ding, C., Pruess, K., Elmroth, E., "Parallel Computing Techniques for Large Scale Reservoir Simulation of Multi-Component and Multiphase Fluid Flow", *Proceedings of the 2001 SPE Reservoir Simulation Symposium,* Society of Petroleum Engineers, Houston, Texas, 2001