

Parallel Computing in the Black Oil Model

J. Liu, Z. Chen, R. E. Ewing, G. Huan, B. Li, and Z. Wang

ABSTRACT. In this paper we present applications of numerical black oil reservoir simulators. The fully implicit solution scheme is utilized for the solution of the coupled governing equations of the black oil model, the Newton-Raphson technique is employed in the linearization of these equations, the finite difference and control volume function approximation methods are used to discretize them, and a parallel preconditioned ORTHOMIN (orthogonal minimum residual) iterative procedure is exploited to solve the linear system arising from the discretization of the linearized equations.

1. Introduction

As the reservoir simulation technology gets more advanced, there are now new requirements from reservoir engineering. These requirements include very fine grids for the development of new oil/gas fields and for the management of old oil/gas fields and the shortening of history match processes and of the entire simulation time, for example. On one hand, the fine grids lead to the size of a large-scale simulation model in terms of millions of unknowns. On the other hand, the shortening of history match and simulation time requires fast and accurate algorithms for solving large-scale systems. Scalar computer machines are already being limited by the speed of light in their capabilities. Vector machines have introduced important new concepts into large-scale computing. Some simplified models in reservoirs can be efficiently implemented on the vector machines. However, a major avenue for real speed advances in computing is the great potential of parallel architectures. In this paper we discuss parallel computing in the black oil model used for numerical reservoir simulation.

2. The Black Oil Model

We briefly review the black oil model. The black oil model is a simplified compositional model. In this model, it is assumed that the hydrocarbon components can be divided into methane and a heavy oil component in a stock tank at the

1991 *Mathematics Subject Classification.* 76S05, 76T05; Secondary 35K60, 35K65.

Key words and phrases. reservoir simulation, control volume function approximation method, finite difference method, black oil model, OpenMP, linear solvers, preconditioners, parallel computing.

This research is supported in part by NSF Grants DMS-9972147 and INT-9901498.

©0000 (copyright holder)

standard pressure. It is also assumed that no mass transfer occurs between the water phase and the other two phases (oil and gas) and no volatile oil exits.

Let ϕ and K denote the porosity and permeability of a porous medium $\Omega \subset \mathbb{R}^3$, s_α , μ_α , p_α , \mathbf{u}_α , B_α , and $K_{r\alpha}$ be the saturation, viscosity, pressure, volumetric velocity, formation volume factor, and relative permeability of the α phase, $\alpha = w, o, g$, respectively, and R_{so} be the gas solubility. Then the mass conservation equations of the black oil model are [1, 5]

$$-\nabla \cdot \left(\frac{\rho_{WS}}{B_w} \mathbf{u}_w \right) + q_w = \frac{\partial}{\partial t} \left(\phi \frac{\rho_{WS}}{B_w} s_w \right)$$

for the water component,

$$-\nabla \cdot \left(\frac{\rho_{OS}}{B_o} \mathbf{u}_o \right) + q_o = \frac{\partial}{\partial t} \left(\phi \frac{\rho_{OS}}{B_o} s_o \right)$$

for the oil component,

$$-\nabla \cdot \left(\frac{\rho_{GS}}{B_g} \mathbf{u}_g + \frac{R_{so} \rho_{GS}}{B_o} \mathbf{u}_o \right) + q_g = \frac{\partial}{\partial t} \left[\phi \left(\frac{\rho_{GS}}{B_g} s_g + \frac{R_{so} \rho_{GS}}{B_o} s_o \right) \right]$$

for the gas component, where $\rho_{\beta S}$ is the density of the β component at standard conditions (stock tank), $\beta = W, O, G$, q_α is the mass flow rate of the α phase at wells, and

$$q_g = q_g^G + q_o^G.$$

The volumetric velocity of the α phase is represented by Darcy's law

$$\mathbf{u}_\alpha = -\frac{KK_{r\alpha}}{\mu_\alpha} \nabla \Phi_\alpha, \quad \alpha = g, o, w,$$

where the potential Φ_α of the α phase is given by

$$\Phi_\alpha = p_\alpha - \rho_\alpha \tilde{g} D, \quad \alpha = w, o, g,$$

ρ_α represents the density of the α phase, \tilde{g} is the gravitational constant, and D is the depth function. The saturations of the water, oil, and gas phases satisfy the constraint

$$s_w + s_o + s_g = 1.$$

Furthermore, the phase pressures are related by the capillary pressures p_{cow} and p_{cgo} :

$$p_{cow} = p_o - p_w, \quad p_{cgo} = p_g - p_o.$$

Finally, the mass flow rates of wells are given by Peaceman's formulas [3]

$$\begin{aligned} q_o &= \sum_{k=1}^{N_w} \sum_{m=1}^{M_{wk}} \frac{2\pi \Delta z_{k,m}}{\ln(r_{e,k,m}/r_{c,k}) + s_{k,m}} \frac{KK_{ro} \rho_{OS}}{\mu_o B_o} [p_{bh,k} - p_o - \rho_o \tilde{g}(D_{w,k} - D)] \delta_{k,m}, \\ q_w &= \sum_{k=1}^{N_w} \sum_{m=1}^{M_{wk}} \frac{2\pi \Delta z_{k,m}}{\ln(r_{e,k,m}/r_{c,k}) + s_{k,m}} \frac{KK_{rw} \rho_{WS}}{\mu_w B_w} [p_{bh,k} - p_w - \rho_w \tilde{g}(D_{w,k} - D)] \delta_{k,m}, \\ q_g^G &= \sum_{k=1}^{N_w} \sum_{m=1}^{M_{wk}} \frac{2\pi \Delta z_{k,m}}{\ln(r_{e,k,m}/r_{c,k}) + s_{k,m}} \frac{KK_{rg} \rho_{GS}}{\mu_g B_g} [p_{bh,k} - p_g - \rho_g \tilde{g}(D_{w,k} - D)] \delta_{k,m}, \\ q_o^G &= \sum_{k=1}^{N_w} \sum_{m=1}^{M_{wk}} \frac{2\pi \Delta z_{k,m}}{\ln(r_{e,k,m}/r_{c,k}) + s_{k,m}} \frac{KK_{ro} R_{so} \rho_{GS}}{\mu_o B_o} [p_{bh,k} - p_o - \rho_o \tilde{g}(D_{w,k} - D)] \delta_{k,m}, \end{aligned}$$

where $\delta_{k,m} = \delta(\mathbf{x} - \mathbf{x}_{k,m})$ (the Dirac delta function at $\mathbf{x}_{k,m}$), N_w is the total number of wells, $M_{w,k}$ is the total number of perforated zones of the k th well, $s_{k,m}$, $\Delta z_{k,m}$, and $\mathbf{x}_{k,m}$ are the skin factor, segment length, and central location of the m th perforated zone of the k th well, $r_{c,k}$ denotes the wellbore radius of the k th well, $r_{e,k,m}$ is the drainage radius of the k th well at the grid block in which $\mathbf{x}_{k,m}$ is located, and $p_{bh,k}$ is the bottom hole pressure of the k th well at datum $D_{w,k}$.

The model is completed by specifying boundary and initial conditions. In this paper we consider no flow boundary conditions

$$\mathbf{u}_\alpha \cdot \mathbf{n} = 0, \quad \alpha = w, o, g, \quad \mathbf{x} \in \partial\Omega,$$

where \mathbf{n} is the outward unit norm to the boundary $\partial\Omega$ of the reservoir domain Ω . The initial conditions depend on the state of a reservoir. When all gas dissolves into the oil phase, there is no gas phase present, i.e., $s_g = 0$. In such a case, the reservoir is called at the undersaturated state. If all three phases co-exist, the reservoir is referred to as at the saturated state. At the undersaturated state, we use $p = p_o$, s_w , and p_b as the unknowns, where p_b is the bubble point pressure. The corresponding initial conditions are

$$p(\mathbf{x}, 0) = p^0(\mathbf{x}), \quad p_b(\mathbf{x}, 0) = p_b^0(\mathbf{x}), \quad s_w(\mathbf{x}, 0) = s_w^0(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$

At the saturated state, we employ $p = p_o$, s_w , and s_o as the unknowns. In this case, the initial conditions become

$$p(\mathbf{x}, 0) = p^0(\mathbf{x}), \quad s_w(\mathbf{x}, 0) = s_w^0(\mathbf{x}), \quad s_o(\mathbf{x}, 0) = s_o^0(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$

The numerical discretization methods in space of the governing equations used in our black oil reservoir simulators include finite difference, control volume finite element, and control volume function approximation methods [2, 5]. The control volume finite element and function approximation methods are based on unstructured grids. The backward Euler method is used to carry out the temporal discretization. At each time step, we obtain a nonlinear system. The Jacobian of the nonlinear system is evaluated in order to apply Newton-Raphson's method. Then we update the Jacobian with the information of wells. The information about the rates of injection and production wells needs to be handled properly [5]. Within each Newton-Raphson iteration, a large-scale sparse linear system must be solved. For petroleum simulation problems with a number of gridblocks of order 100,000, about 80% – 90% of the total simulation time is spent on linear solvers. So solving a linear system efficiently is crucial to numerical simulations of the black oil model. The loops for a numerical reservoir simulator are of the following form:

Time period
 Time-stepping
 Newton-Raphson iterations
 Linear solver

3. Parallel Speedup and Amdahl's Law

There are two basic gauges measuring parallel computations: speedup and efficiency. Let T_1 be the time spent by a program running on a single processor and T_n be the time spent when the program runs on n processors, or more precisely, in n processes or threads. Then speedup S_n and efficiency E_n are defined by,

respectively,

$$S_n = \frac{T_1}{T_n}, \quad E_n = \frac{S_n}{n}.$$

Usually only some parts of a program can be parallelized. Let $0 < p < 1$ be the fraction of the program's code that can be parallelized. The remaining fraction $1 - p$ of the code must run sequentially. Amdahl's law indicates that the ideal speedup is given by

$$S_n(p) = \frac{1}{(p/n) + (1 - p)},$$

where n is the number of processes being used in parallelization. The following graph predicts what speedup can be achieved when 2, 4 or 8 processes are used.

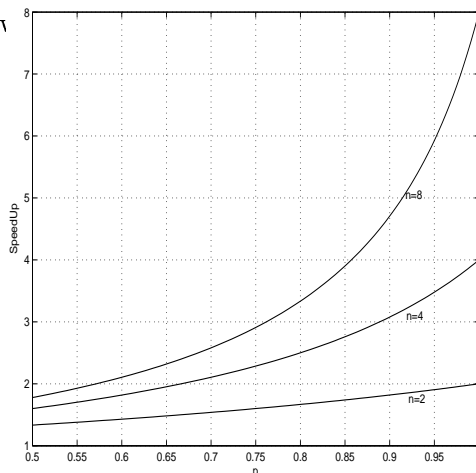


Fig. 1: Ideal speedup for $n = 2, 4, 8$.

In some special cases, superlinear speedup ($S_n > n$) can be observed. This may be due to better cache use in a parallelized program than in its sequential version, but is generally not expected in realistic parallel computing.

4. Parallel Computing Environment

There are a variety of parallel systems; each has its own advantages and disadvantages. Loosely speaking, they fall into one of two categories: shared-memory and distributed-memory. Accordingly, two different programming approaches are adopted by parallel application developers: MPI (Message Passing Interface) for distributed-memory systems and OpenMP (Multiple Processing) for shared-memory systems.

On a traditional symmetric multiprocessor (SMP) system, all processors have equal access to the flat memory through a shared bus. Bus contention becomes a severe bottleneck as the number of processors grows. This obvious drawback makes it hard for SMPs to scale well beyond 8 or 16 processors.

To overcome this difficulty, nonuniform memory access (NUMA) architectures have been designed; those providing cache coherency are called ccNUMA. Commercial examples include SGI Origin 2000 and Compaq Alpha GS 320.

A typical NUMA system consists of a collection of processing modules. Each module consists of one or more processors with internal/local (shared) memory. An individual processor has one or more levels of cache exclusively associated with it.

Processing modules are linked by a high-speed interconnect. A processor can access remote memory located on another processing module through the switched-based interconnect. In this way, individual memories together comprise the system's globally addressable memory. In addition, cache-coherent mechanism assumes responsibility not only for fetching and storing data, but also for ensuring consistency among copies of a data item.

As a successful representative of ccNUMA systems, SGI Origin 2000 is devised as a hypercube, where each processing module typically consists of 2 MIPS R10000 processors, connected by a hub [8]. Each processor has two levels of cache. All modules are connected through high-speed routes. On SGI Origin 2000, remote memory access is 2 to 9 times slower than local memory access, depending on the size of the system, more accurately, the hops taken when accessing remote memory. The sophisticated design of SGI Origin 2000 keeps a logarithmic growth of latency in remote memory access with the growth of the system's size and attains excellent scalability.

SGI Origin 2000 supports data allocation at the granularity of a physical page (usually 16 KB). It provides hardware and software mechanisms to maximize data locality. Under the default "First-Touch" placement policy, memory is allocated to the first process to access the page. Through dynamic page migration, data can also be moved to their optimal locations.

OpenMP has emerged as a viable programming paradigm for shared-memory parallel systems [7]. It consists of compiler directives and library routines that can be inserted into Fortran, C or C++ codes to realize fork-join parallelism. OpenMP directives can be used to declare parallel regions, to distribute computational tasks among threads, and to synchronize threads. Work-sharing directives spread loop iterations, or divide the work into a set of parallel sections. Mechanism for coordinating threads includes barriers, locks, and single-threaded execution. Data may be shared among threads, or may be private to a thread. OpenMP directives can be used to specify explicitly data placement. Moreover, OpenMP is supported by compilers from numerous vendors and provides code portability across shared-memory parallel systems from different vendors. These features of OpenMP ease parallel programming and hence make it widely accepted by users.

Nevertheless, developing efficient parallel applications is a time-consuming and costly task that requires considerable expertise. MPI programming requires global program analysis to design appropriate strategies for data decomposition, work distribution, and process communications. OpenMP programmers must carefully select and reorganize, if necessary, loops that are to be executed in parallel, in order to distribute work evenly, avoid false sharing of data, and make good use of cache. Contention conditions must be detected and removed, and process synchronizations must be minimized.

Numerical experiments in our current study are conducted on an 8-processor SGI Origin 2000 installed with Irix 6.5.15 and OpenMP.

5. Linear Solvers and Preconditioners

As seen in previous sections, within each Newton-Raphson iteration, we have to solve a large-scale linear system. The coefficient matrix of the linear system is sparse but not symmetric. The band structure of the matrix has also been spoiled by wells that perforate into many gridblocks or by irregular gridblock structure. For

such general sparse systems, Krylov subspace methods, such as BiCGSTAB (Bi-Conjugate Gradient Stabilized), GMRES (Generalized Minimum Residual), ORTHOMIN (Orthogonal Minimum Residual), are usually employed as linear solvers. A preconditioner is often used in conjunction with the iterative solver to accelerate its convergence.

Preconditioners play an important role in linear solvers. Usually the construction of a preconditioner is outside of the iterations of its associated linear solver, whereas the preconditioner is applied to the residual vector (at least once) within each iteration of the linear solver. When designing or choosing a preconditioner for a linear solver, we take into account the following factors:

- Good acceleration, which results in less iteration numbers for the solver;
- Less computation when applying the preconditioner to residual vectors;
- The preconditioner has a clear, say, sparse or triangular, structure and demands less in storage;
- The procedure to construct the preconditioner is not complicated and requires less computation.

Obviously, the simplest preconditioner is the identity matrix and the best one is the inverse of the coefficient matrix. All practical preconditioners sit in between. Trade-offs among the above factors are usual in practice and preconditioners remain to be an active research area in numerical linear algebra.

In our present study, we focus on the ORTHOMIN method coupled with an incomplete LU factorization (ILU) preconditioner. ORTHOMIN was first introduced by petroleum engineers [9], and ILU proves robust as a preconditioner for ORTHOMIN. A flow chart of the sequential ORTHOMIN is presented below, where k_{max} and tol are the preset maximum number of iterations and residual tolerance, respectively.

```

k = 0; x0 = 0; r0 = b
while (k < kmax and ||rk||2/||b||2 < tol)
  k = k + 1
  uk = U-1(L-1rk-1)
  vk = Auk
  do (1 ≤ j ≤ k - 1)
    αjk = (qj, vk)/(qj, qj)
  enddo
  pk = uk
  qk = vk
  do (1 ≤ j ≤ k - 1)
    pk = pk - αjkpj
    qk = qk - αjkqj
  enddo
  βk = (qk, rk-1)/(qk, qk)
  xk = xk-1 + βkpk
  rk = rk-1 - βkqk
  Compute ||rk||2
enddo

```

ILU means that we look for an approximate matrix M to the coefficient matrix A in the sense that

$$A = M + E, \quad M = LU.$$

An exact LU factorization of A will generally result in two dense factors. However, some nonzero entries in the lower- and upper-triangular matrices can be discarded since we are looking only for an approximate matrix. In other words, we fill in nonzero entries in L and U according to prescribed thresholds or levels. If L and U have the same sparsity pattern as A , then we are doing the ILU(0) factorization, which is the simplest case and also commonly used in industrial settings.

6. Parallelization of Solvers and Preconditioners

For numerical simulations of the black oil model, the majority of the program's time is spent on the linear solver and its associated preconditioner. The linear solver and preconditioner are relatively independent from other parts of the program, and therefore can be isolated without much difficulty. So we choose them as our entry point for parallelization. In this section, we explore parallelism within the linear solver and preconditioner and propose some parallelization strategies for ORTHOMIN.

There are roughly five types of operations related to linear solvers and preconditioners:

1. *Vector updates*: These are mostly SAXPY operations, i.e., scalar a times vector x plus vector y : $y = ax + y$;
2. *Inner products*;
3. *Matrix-vector multiplication*: Here the matrix is usually the sparse coefficient matrix of the linear system;
4. *Action of preconditioner*: The action of a preconditioner on residual vectors is often in the form of solving one or more linear systems;
5. *Construction of preconditioner*: The construction procedure of the preconditioner is outside of the iterations of the linear solver.

The first two operations belong to the category of fine-grained parallelism and are readily parallelized. Let's take the inner product as an example. Each process/thread computes its local inner product (LIP) of two corresponding segments of two vectors; then a reduction operation is performed to assemble LIPs. On shared-memory machines, this is fulfilled through OpenMP directives. On distributed-memory systems, the assemble operation incurs some communications.

For the sparse matrix-vector multiplication, we parallelize the operation by spreading out the matrix and the vector to several processors via DISTRIBUTE directive in OpenMP and the default "First-Touch" memory allocation policy.

When designing parallel linear solvers and preconditioners, we have to consider all relevant factors comprehensively. For instance, a simple diagonal (also called pointwise Jacobi) preconditioner is easy to parallelize, but its acceleration turns out to be disappointing.

For ORTHOMIN, the construction procedure of the ILU preconditioner is basically a Gaussian elimination process, which is by nature sequential. But we construct L^{-1} instead of L , so later the forward substitution in the action of the preconditioner is readily parallelizable. The upper triangular factor U is chosen as the upper triangular part of the coefficient matrix A , which is simple but has proved stable by our numerical results.

The action of the ILU preconditioner consists of one forward substitution and one backward substitution. Since L^{-1} is already constructed outside of the linear solver, the forward substitution can also be parallelized through OpenMP directives.

The backward substitution remains sequential. Parallelization of this part needs some advanced techniques like matrix reordering and will be addressed in our future work.

7. Numerical Experiments

We carry out numerical experiments on a shared-memory machine, SGI Origin 2000 with 8 250MHz processors and 4G memory; each processor has 32KB L1 cache for instructions, 32KB L1 cache for data, and 4MB L2 cache. The linear solver ORTHOMIN in our simulation program has been partially parallelized using OpenMP directives.

For each time step, we allow 5 Newton-Raphson iterations. The stopping criterion used for ORTHOMIN is $\|r\|_2/\|b\|_2 \leq 10^{-4}$, where r and b are the residual and right-hand side, respectively. Since our ORTHOMIN method can be restarted adaptively, k_{max} is taken as 10 to alleviate memory demand.

Our model problem sits in a 3-dim domain of size $10,000ft \times 10,000ft \times 100ft$. Two uniform meshes consisting of $100 \times 100 \times 6$ and $100 \times 100 \times 3$ gridblocks are generated. Here the z -axis points downwards. There are 2 wells in this domain. One injection well sits in gridblock (5, 5, 1) and one production well sits in gridblock (95, 95, 6) (or (95, 95, 3)). For each gridblock, 3 unknowns are needed. So there are 60,000 (or 30,000) gridblocks and about 180,000 (or 90,000) unknowns. This model problem was first shown in [6] with a mesh of only $10 \times 10 \times 3$ gridblocks.

TABLE 1. Speedup results for ORTHOMIN

Mesh Size	100x100x3	100x100x6
Program Execution Time of 1 CPU (min)	1887	4742
Program Execution Time of 8 CPUs (min)	630	1639
Whole Program Speedup	3.0	2.9
Solver Execution Time of 1 CPU (min)	1623	4054
Solver Execution Time of 8 CPUs (min)	389	1016
Solver Speedup	4.2	4.0

References

- [1] Z. Chen, Formulations and numerical methods for the black oil model in porous media, *SIAM J. Numer. Anal.* **38** (2000), 489–514.
- [2] Z. Chen, G. Huan, and B. Li, Modeling 2D and 3D horizontal wells using CVFA, *Communications in Mathematical Sciences* **1** (2002), in press.
- [3] D.W. Peaceman, Interpretation of well-block pressures in numerical reservoir simulation, SPE 6893, 52nd Annual Fall Technical Conference and Exhibition, Denver, 1977.
- [4] B. Li, Z. Chen, and G. Huan, Control volume function approximation methods and their applications to modeling porous media flow I: The two-phase flow, *Advances in Water Resources*, to appear.
- [5] B. Li, Z. Chen, and G. Huan, Control volume function approximation methods and their applications to modeling porous media flow II: The black oil model, *Advances in Water Resources*, submitted.
- [6] A. S. Odeh, Comparison of solutions to a three-dimensional black-oil reservoir problem, *Journal of Petroleum Technology*, January, 1981, 13–25.
- [7] OpenMP Architecture Review Board, *OpenMP Fortran Application Program Interface*, Version 2.0, Nov. 2000, <http://www.openmp.org>.

- [8] Origin 2000 and Onyx2 performance tuning and optimization guide, SGI Document Number 007-3430-003.
- [9] P. K. W. Vinsome, Orthomin, An iterative method for solving sparse banded sets of simultaneous linear equations, Paper SPE 5729 presented at the SPE-AIME Fourth Symposium on Numerical Simulation of Reservoir Performance, Los Angeles, CA., Feb. 1976.

INSTITUTE FOR SCIENTIFIC COMPUTATION, TEXAS A&M UNIVERSITY, COLLEGE STATION, TX
77843-3404

E-mail address: jliu@isc.tamu.edu

DEPARTMENT OF MATHEMATICS, BOX 750156, SOUTHERN METHODIST UNIVERSITY, DALLAS,
TX 75275-0156

E-mail address: zchen@mail.smu.edu

INSTITUTE FOR SCIENTIFIC COMPUTATION, TEXAS A&M UNIVERSITY, COLLEGE STATION, TX
77843-3404

E-mail address: richard-ewing@tamu.edu

DEPARTMENT OF MATHEMATICS, BOX 750156, SOUTHERN METHODIST UNIVERSITY, DALLAS,
TX 75275-0156

E-mail address: huan@golem.math.smu.edu

DEPARTMENT OF MATHEMATICS, BOX 750156, SOUTHERN METHODIST UNIVERSITY, DALLAS,
TX 75275-0156

E-mail address: bli@mail.smu.edu

DEPARTMENT OF MATHEMATICS, BOX 750156, SOUTHERN METHODIST UNIVERSITY, DALLAS,
TX 75275-0156

E-mail address: zwang@mail.smu.edu