Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 201 (2004) 224–237
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

# Parallel computation for reservoir thermal simulation of multicomponent and multiphase fluid flow

Yuanle Ma [a], Zhangxin Chen [b,*]

[a] *INET, Tsinghua University, Beijing 100084, PR China*
[b] *Department of Mathematics, Center for Scientific Computation, P.O. Box 750156, Southern Methodist University, Dallas, TX 75275-0156, USA*

Received 27 October 2003; received in revised form 16 April 2004; accepted 14 May 2004
Available online 2 July 2004

**Abstract**

We consider parallel computing technology for the thermal simulation of multicomponent, multiphase fluid flow in petroleum reservoirs. This paper reports the development and applications of a parallel thermal recovery simulation code. This code utilizes the message passing interface (MPI) library, overlapping domain decomposition, and dynamic memory allocation techniques. Its efficiency is investigated through simulation of two three-dimensional multicomponent, multiphase field models for heavy oil crudes. Numerical results for these two simulation models indicate that this parallel code can significantly improve capacity and efficiency for large-scale thermal simulations.


*AMS:* 35K60; 35K65; 76S05; 76T05

*Keywords:* Petroleum reservoirs; Parallel computing; Thermal recovery; MPI; Domain decomposition

## 1. Introduction

The rapid development of parallel computers can overcome the limitations of problem size and space resolution for reservoir simulation on a single-processor machine. In the past decade, the total number of gridblocks employed in a typical reservoir simulator has increased from thousands to millions. This is particularly due to the advent of the most prevalent parallel computers, distributed-memory machines, which can be made up of hundreds to thousands of processors. Most parallel computing techniques in the petroleum industry have been developed for reservoir simulation using the black oil and compositional models (see, e.g. [2,6]). These two models involve the mass conservation equation, Darcy's law, and mass interchange between fluid phases, i.e., they describe isothermal flow. Parallel computing algorithms for the

* Corresponding author. Tel.: +214-768-2506.
*E-mail addresses:* mayl@tsinghua.edu.cn (Y. Ma), zchen@mail.smu.edu (Z. Chen).

0021-9991/$ - see front matter 
doi:10.1016/j.jcp.2004.05.014

thermal model which describes non-isothermal flow of multicomponent, multiphase fluids have not been available. The thermal model involves more physics and is more complex than the black oil and compositional models. Not only does it introduce one more unknown (temperature or energy), it also introduces greater nonlinearity and coupling in the governing equations. Hence numerical simulation for this model is far more complicated than that for other two simpler models.

Thermal methods, particularly steam drive and soak, make up the largest share of the enhanced oil recovery (EOR) projects in petroleum industry and have experienced rapid growth since the early 1970s. Steam methods currently account for nearly 80% of the EOR oil in USA [8], for example. Thermal flooding has been commercially successful for the past 30 years. Thus the development of parallel computing techniques for large-scale thermal simulation has become increasingly important.

In this paper, we report on our work in parallelization of a serial code for the non-isothermal flow of multicomponent, multiphase fluids in three-dimensional petroleum reservoirs. This serial code has been widely used in many oilfields. It can be exploited for designing oil recovery schemes of old and new fields, examining history match, predicting production, and studying residual oil distribution [11]. Its parallel version is developed using the message passing interface (MPI) library. MPI is a standard procedure for message passing that allows data communication between different processors. The parallel implementation first divides a simulation domain into a number of smaller domains, each of which corresponds to a processor. The nonlinear governing equations describing the flow system are then discretized in space and time and solved on each smaller domain. These smaller domains overlap so that the communication between neighboring domains (i.e., processors) occurs over the overlapping regions.

Parallel computing is more useful for thermal methods than for all other methods used in the EOR projects due to their displacement mechanisms. The thermal methods rely on several displacement mechanisms to recover oil, such as viscosity reduction, distillation, miscible displacement, thermal expansion, wettability changes, cracking, and lowered oil–water interfacial tension. For most applications, the most important is the reduction of crude viscosity with increasing temperature. The four basic approaches to achieve this mechanism are hot water flooding, steam soak, steam drive, and in situ combustion. In a steam soak (stimulation or huff'n puff), for example, steam is introduced into a well, and then the well is returned to production after a brief shut-in period. In this approach, all the wells do not necessarily have the same operating schemes. The well operating schemes significantly affect the control and choice of time step sizes (and even space step sizes). The step sizes are small in the regions where wells are very active (particularly where a well is in the shut-in and production period), and can be large in the regions where the wells are not active (or where a well is in the injection period). Thus the step sizes at different locations of wells can be very different. The domain decomposition and parallel computing techniques provide an excellent scenario in which different step sizes in different regions can be efficiently used and controlled.

The significant enhancement of computational efficiency in our parallel thermal code is demonstrated through simulation of two field flow problems. These problems simulate three-dimensional models of multicomponent, multiphase flow involving heavy crudes. Simulation results carried out on different parallel computers indicate that the parallel code provides very accurate history matches and that this code achieves superlinear speed-up. We have been developing, refining, and testing this parallel code since 1998, and have had extensive experience in its development and applications in the petroleum industry. In this paper, we summarize our code development and application experience in the past 5 years.

The rest of this paper is organized as follows. In Section 2, we briefly review the differential equations governing thermal recovery simulation. Then, in Section 3, we describe our parallel code. The domain decomposition method, date communication, load balancing, time step size controlling, and linear equation solvers are described. In Section 4, we present two simulation examples. Finally, we conclude with a few remarks.

## 2. The governing equations

The parallel code evolves from a serial code for thermal simulation of multicomponent, multiphase fluid flow in petroleum reservoirs. This serial code has been widely used in many oil fields, and has been developed based on the displacement mechanisms of thermal methods: (a) reduction of crude viscosity with increasing temperature, (b) change of relative permeabilities for greater oil displacement, (c) vaporization of connate water and parts of crudes for a miscible displacement of light components, and (d) high temperatures of fluids and rock to maintain high reservoir pressure. It can model such important physical factors and processes as

- viscosity, gravity, and capillary forces,
- heat conduction and convection processes,
- heat losses to overburden and underburden of a reservoir,
- mass transfer between phases,
- effects of temperature on physical property parameters of oil, gas, and water,
- rock compression and expansion.

The governing equations for the thermal model include the mass and energy conservation equations, Darcy's law, and the mole fraction, saturation, and capillary pressure constraint equations [1]:

(1) The mass conservation equation

$$\sum_{j=1}^{N_p}\left(\int_V \frac{\partial}{\partial t}(\phi s_j \rho_j \chi_{ij})\mathrm{d}V + \int_S (\rho_j \chi_{ij} v_j)\mathrm{d}S\right) + q_i = 0, \quad i = 1,2,\ldots,N_c,$$

where $\phi$ is the rock porosity, $\rho_j$, $s_j$, and $v_j$ are the density, saturation, and surface flow velocity of phase $j$, $N_c$ and $N_p$ are the numbers of compositions and phases, respectively, $\chi_{ij}$ is the mole fraction of composition $i$ in phase $j$, $q_i$ is the source item of composition $i$, and $V$ is a typical volume with surface $S$.

(2) The energy conservation equation

$$\int_V \frac{\partial}{\partial t}\left(\phi \sum_{j=1}^{N_p} \rho_j s_j u_j + (1-\phi)\rho_{\mathrm{rock}} C_p (T - T_{\mathrm{int}})\right)\mathrm{d}V + \int_S (q_{i,\mathrm{h}} + q_{i,\mathrm{c}})\mathrm{d}S + Q_{i,\mathrm{c}} + Q_{i,l} = 0,$$

where $u_j$, $\rho_{\mathrm{rock}}$, $C_p$, $T$, $T_{\mathrm{int}}$, $q_{i,\mathrm{h}}$, $q_{i,\mathrm{c}}$, $Q_{i,\mathrm{c}}$, and $Q_{i,\mathrm{h}}$ are, respectively, the inner energy of phase $j$, rock density, rock heat capacity, reservoir temperature, initial temperature, enthalpy flow velocity, heat flow between adjacent volumes, heat source item, and heat loss to overburden and underburden.

(3) Darcy's law

$$\mathbf{v}_j = -\frac{1}{\mu_j}\mathbf{k}_j\left(\nabla p_j - \rho_j g \nabla z\right), \quad j = 1,2,\ldots,N_p,$$

where $\mathbf{k}_j$, $p_j$, and $\mu_j$ are the effective permeability, pressure, and viscosity for phase $j$, $g$ is the gravitational acceleration, and $z$ is the depth.

(4) The mole fraction constraint equation

$$\sum_{i=1}^{N_c} \chi_{ij} = 1, \quad j = 1,2,\ldots,N_p.$$

(5) The saturation constraint equation

$$\sum_{j=1}^{N_p} s_j = 1.$$

(6) The capillary constraint equation

$$p_{cj1} = p_j - p_1, \quad j = 2, 3, \ldots, N_p.$$

There are $N_c + 3N_p + 1$ equations ($N_c$ mass equations, an energy equation, $N_p$ Darcy's laws, $N_p$ mole fraction equations, a saturation equation, and $N_p - 1$ capillary equations) for the same number of unknowns ($N_c$ mole fractions, $N_p$ saturations, $N_p$ velocities, $N_p$ pressures, and temperature). The differential system is closed with a proper set of boundary and initial conditions. In computations, Darcy's laws for the fluid velocities and the capillary constraints are substituted into the mass conservation equations so the unknowns $\mathbf{v}_j$ and $p_k$ can be eliminated ($j = 1, 2, \ldots, N_p$, $k = 2, 3, \ldots, N_p$). Then the total number of equations and unknowns is reduced to $N_c + N_p + 2$ ($N_c$ mass equations, $N_p$ mole fraction equations, an energy equation, and a saturation equation for the unknowns: $N_c$ mole fractions, $N_p$ saturations, pressure, and temperature). If there exist $M$ grid points, a system of $M(N_c + N_p + 2)$ algebraic equations need to be solved. For a simulation problem with the order of 10,000 grid points, the computational cost and storage memory would be enormous. Thus parallel computing techniques are more appropriate.

## 3. Parallel implementation

Due to the fact that 60–80% of the computational time is spent on the assembly and solution of a linear system of algebraic equations, a prevailing strategy in reservoir simulation is to parallelize only this part, i.e., the linear solver part. However, this strategy may not be effective. First, the model scale is limited by the size of accessible memory of the CPU. This difficulty becomes prominent in a parallel environment with a PC or workstation cluster. Also, most preconditioners for linear solvers used in reservoir simulation are based on incomplete LU factorization, which is by nature sequential. While various techniques, such as the use of parallel approximate inverses, have been introduced to parallelize these preconditioners, additional computations are needed. Thus, to improve really the efficiency of a simulation code, a global parallel scheme must be employed.

In this paper we exploit the domain decomposition method for the parallel implementation, i.e, we divide a simulation domain into a number of smaller domains. Each of these smaller domains corresponds to a processor, that is, each processor performs the entire flow simulation on its corresponding domain, which includes initialization, PVT and rock property data calculation, space and time discretization, Newton–Raphson iteration, solution of linear systems, and data output.

### 3.1. Domain decomposition

The domain decomposition method is a technique for solving a partial differential problem based on a decomposition of the spatial domain of the problem into a number of smaller domains [4]. In general, this method can be classified as an overlapping method or a non-overlapping method. The overlapping method is generally easier to describe and implement [4]. It is also easier to achieve an optimal convergence rate using this method, and it is often more robust. But additional work is needed on the overlapped regions. Furthermore, if the coefficients of a differential problem are discontinuous across inter-boundaries, the extended subdomains have discontinuous coefficients, which makes their solution more problematic. On the other hand, the non-overlapping method requires solution of interface problems at interfaces of the subdomains. In this paper we employ the overlapping technique.

Let $\{\Omega_i\}_{i=1}^N$ be a non-overlapping division of the reservoir domain $\Omega$. Each subdomain $\Omega_i$ may be chosen as elements from a coarse finite element partition $T_H$ of $\Omega$ with mesh size $H$. Next, each $\Omega_i$ is extended to $\hat{\Omega}_i$,

consisting of all points in $\Omega$ within a distance of $\delta$. We use matched grids here, i.e., the grids in neighboring subdomains are matched.

In this paper, we use the block-centered finite difference method in space with harmonically averaged coefficients (or, equivalently [15], the mixed finite element method using the Raviart–Thomas [14] and Nedelec [13] spaces of lowest-order on rectangular parallelepipeds) and the backward Euler scheme in time for the discretization of the governing equations stated in the previous section. The discretized equations are then solved simultaneously and implicitly, i.e., in a fully implicit manner. These nonlinear equations are solved using the Newton–Raphson iteration.

The domain decomposition method applied to a parabolic problem is simpler than for an elliptic problem. To see this, we consider the parabolic equation

$$c\frac{\partial u}{\partial t} + L(u) = f, \tag{3.1}$$

where $L$ is a linear differential operator of second order. After discretizing in space by finite differences and in time by the backward Euler scheme (with time step $\Delta t$), this equation becomes

$$c^{n+1}\frac{u^{n+1} - u^n}{\Delta t} + \mathscr{L}^{n+1}u^{n+1} = f^{n+1}, \tag{3.2}$$

where $u^n = u(x, t^n)$ at the $n$th time level and $\mathscr{L}^{n+1}$ is the matrix corresponding to the discretization of $L$ in space. At every time step $t^{n+1}$, Eq. (3.2) gives an elliptic problem

$$(c^{n+1}I + \Delta t\mathscr{L}^{n+1})u^{n+1} = c^{n+1}u^n + \Delta t f^{n+1}, \tag{3.3}$$

where $I$ is the identity matrix. Now, we see that the matrix of this system has a much better condition number $\mathscr{O}(1 + \Delta t/h^2)$, instead of the condition number of $\mathscr{O}(h^{-2})$ for a typical elliptic problem, where $h$ is the mesh size of a fine partition $T_h$ of $\Omega$ over which we seek an approximate solution to $u$.

Let $\mathscr{L}^{n+1}$ be an $M \times M$ matrix, and let $\hat{M}_i$ be the number of interior nodes in $\hat{\Omega}_i$. For each $\hat{\Omega}_i$, $i = 1, 2, \ldots, N$, let $\hat{I}_i$ represent the indices of the nodes lying in the interior of $\hat{\Omega}_i$. Now, let $R_i$ indicate the $M \times \hat{M}_i$ matrix (with entries of 1's and 0's) that restricts a vector $w$ of length $M$ to a subvector $R_i w$ of length $\hat{M}_i$ by choosing the subvector having indices in $\hat{I}_i$ (corresponding to the interior nodes in $\hat{\Omega}_i$). $R_i$ is called the restriction matrix, and the transpose $R_i^{\mathrm{T}}$ of $R_i$ is referred to as an interpolation or extension matrix. $R_i^{\mathrm{T}}$ extends subvectors of length $\hat{M}_i$ on $\hat{\Omega}_i$ to vectors of length $M$ using extension by zero to the rest of $\Omega$. Analogously, $R_H$ and $R_H^{\mathrm{T}}$ represent the restriction and extension matrices associated with the coarse grid $T_H$. Now, the local submatrices are given by $L_i^{n+1} = R_i(c^{n+1}I + \Delta t\mathscr{L}^{n+1})R_i^{\mathrm{T}}$.

The usual additive Schwarz methods based on overlapping domain decomposition are defined by

$$(\mathscr{S}_1^{n+1})^{-1} = \sum_{i=1}^{N} R_i^{\mathrm{T}}(L_i^{n+1})^{-1}R_i, \tag{3.4}$$

or

$$(\mathscr{S}_2^{n+1})^{-1} = \sum_{i=1}^{N} R_i^{\mathrm{T}}(L_i^{n+1})^{-1}R_i + R_H^{\mathrm{T}}(L_H^{n+1})^{-1}R_H. \tag{3.5}$$

Method (3.5) involves a coarse grid correction.

It can be shown [3] that there exists a constant $C$ such that if $\Delta t \leqslant CH^2$, the condition number $\mathrm{cond}((\mathscr{S}_1^{n+1})^{-1}(c^{n+1}I + \Delta t\mathscr{L}^{n+1}))$ is bounded by a constant $C_1$ independent of $\Delta t$, $H$, and $h$. For a larger $\Delta t$, $\mathrm{cond}((\mathscr{S}_2^{n+1})^{-1}(c^{n+1}I + \Delta t\mathscr{L}^{n+1}))$ possesses the same property. Hence, if $\Delta t \leqslant CH^2$, method (3.4) is effective.

However, if $\Delta t$ is larger, a coarse grid correction must be employed to maintain a constant rate of convergence. Similar results hold for multiplicative Schwarz methods [3].

The numerical simulation of fluid flows in petroleum reservoirs has some special features. Each grid-block must contain at most one injection or production well. Thus, in the usual case where there are hundreds of wells in a reservoir, the coarse grid cannot be coarse enough for our domain decomposition approach, that is, $H$ is often small. In addition, due to the peculiar displacement mechanisms of thermal methods mentioned in Section 1, it would require tremendous computational time to carry out the numerical simulation over the entire reservoir domain. Thus method (3.5) is not so effective. On the other hand, when $H$ is small, the condition $\Delta t \leqslant CH^2$ for method (3.4) is too restrictive, particularly for long time simulation. In this paper we exploit a simplified domain decomposition method, which uses the parabolic structure of problem (3.1). Indeed, for such a problem, the so-called Green's function decays exponentially in space, which implies that any local error has almost no influence away from its original position [7].

We write (3.2) as

$$(c^{n+1}I + \Delta t\mathscr{L}^{n+1})(u^{n+1} - u^n) = \Delta t(f^{n+1} - \mathscr{L}^{n+1}u^n).$$

Thus, at the time step $t^{n+1}$, we solve the equation

$$(c^{n+1}I + \Delta t\mathscr{L}^{n+1})\Delta u = \Delta f^{n+1} \equiv \Delta t(f^{n+1} - \mathscr{L}^{n+1}u^n), \tag{3.6}$$

where $\Delta u = u^{n+1} - u^n$. Now, the simplified domain decomposition method is constructed as follows, for $i = 1, 2, \ldots, N$,

$$\begin{array}{ll} (c^{n+1}I + \Delta t\mathscr{L}^{n+1})\Delta\hat{u}_i = \Delta f^{n+1} & \text{in } \Omega_i, \\ (c^{n+1}I + \Delta t\mathscr{L}^{n+1})\Delta\hat{u}_i = 0 & \text{in } \hat{\Omega}_i \setminus \Omega_i, \\ \Delta\hat{u}_i = 0 & \text{on } \partial\hat{\Omega}_i. \end{array} \tag{3.7}$$

The boundary condition for $\Delta\hat{u}_i$ on $\partial\hat{\Omega}_i$ can also be of the Neumann type. Obviously, these boundary conditions are not correct, but the associated error will occur only in a small neighborhood of $\partial\hat{\Omega}_i$. Thus $\Delta\hat{u}_i$ may accurately approximate $\Delta u$ in $\Omega_i$.

A major problem in this approach is to determine where the boundary $\partial\hat{\Omega}_i$ is located. As observed in [12], a moderate amount of overlapping gives very good results. In fact, using a maximum principle argument, one can prove the error estimate [9]

$$\|\Delta u - \Delta\hat{u}\|_{H^1(\Omega)} \leqslant Ce^{-\delta/\Delta t}\|\Delta f^{n+1}\|_{L^2(\Omega)},$$

where $\|\cdot\|_{H^1(\Omega)}$ and $\|\cdot\|_{L^2(\Omega)}$ are the $H^1$- and $L^2$-norms, respectively, and $\delta$ is the overlapping width. Therefore, with reasonably large overlapping (or small time steps), the solution $\Delta\hat{u}$ to (3.7) approximates $\Delta u$.

### 3.2. Load balancing

In parallel computing, one should try to distribute the work load equally on all processors. In practice, it is difficult to achieve a load balance close to the optimum. Fortunately, in reservoir simulation, there are several guidelines for distributing the work load. First, the grid blocks should be evenly distributed among the processors with not only approximately the same number of internal blocks, but also roughly the same number of external blocks per processor. Second, if natural faults exist in a reservoir, these faults should be used as the inter-boundaries between subdomains. Some of the PVT and rock property data are discontinuous across the faults, and there is no data communication across them. Third, all the subdomains should contain the same number of wells. The well-operating schemes must be also taken into account for

load balancing. A well can be an injector or producer. In the thermal modeling, a well can be both, and the injection, production, and shut-in periods need be considered in distributing the work load. Among these three guidelines, the last should be respected the most.

### 3.3. Data communication

Message passing between processors is an essential component of parallel computing. It can take two forms: blocking (synchronous) and non-blocking (asynchronous). Which form is to be used depends on the characteristics of data to be transferred. In reservoir simulation, according to their time variant characteristics, the communication data are divided into three basic types, the static data, the slow transient data, and the fast transient data. The data describing the geometric model of a reservoir and rock property parameters are the static data. Essentially, these data do not change in the simulation. At a time step in the iteration process, the values of pressure, temperature, and saturation are the slow transient data. These data need to be recorded at certain times to restart a computation. All others are fast transient data. In particular, those that are frequently transferred over the overlapping regions are of this type. The blocking communication mode is used to transfer the static and slow transient data, and the non-blocking communication mode is adopted to transfer the fast transient data to reduce communication overhead and improve communication efficiency.

### 3.4. Time step size and communication time control

As discussed in Section 1, the time step sizes on different subdomains can be different. To ensure that the well data of all production periods can be safely loaded and that a simulation process is stable and accurate on each processor, the step size $\Delta t_i^n$ on the $i$th subdomain $\Omega_i$ can be chosen using an adaptive control strategy developed in [5] that possesses these properties, $i = 1, 2, \ldots, N$.

To synchronize the computational processes on different processors and to pass messages efficiently between processors at certain times, the $n$th communication time is controlled as follows:

1. Predict the communication time $t_i^n$ for the $i$th subdomain, $i = 1, 2, \ldots, N$.
2. Determine the $n$th synchronic communication time $t^n$ by

$$t^n = \min\{t_1^n, t_2^n, \ldots, t_N^n\}.$$

3. Find the $n$th communication time $t_i^n$ for the $i$th subdomain: $t_i^n = t^n$.

While the minimum time level approach is recommended here, we point out that the maximum and weighted time level approaches can be also utilized. From our experience, when a domain decomposition approximately achieves a load balance, these three approaches do not differ much. The approach adopted here generates the most accurate solutions.

### 3.5. Linear equation solution

After the spatially and temporally discretized nonlinear equations are assembled for each subdomain at each time step, these equations are solved using the Newton–Raphson iteration. At each Newton iteration, a linear system of algebraic equations is solved using a preconditioned iterative algorithm. In reservoir simulation, this system has peculiar structures. The coefficient (stiffness) matrix is sparse but non-symmetric and indefinite. While sparse, its band structure is often spoiled by wells that perforate into many gridblocks. For such a general system, Krylov subspace methods, such as BiCGSTAB (bi-conjugate gradient stablized), GMRES (generalized minimum residual), and 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. In this paper the linear solver is based on ORTHOMIN [17], with an incomplete LU factorization (ILU) preconditioner; the more robust algorithm GMRES [16] can be also used, and it is under investigation.

## 4. Numerical tests

We use two field-scale models [10] to test the efficiency of our parallel thermal simulation code. We carried out the numerical tests on three parallel computers to which we have direct access, a PC cluster with 8 processors, Dawn 1000A, and Dawn 2002.

Although numerical tests are not presented on a parallel machine with a larger number of processors, we are confident that observations similar to those in this section can be made for larger machines.

### *4.1. Example 1*

This simulation model comes from the design of development and production schemes for a real oil field. This field started to operate on May 1, 1987. The grid blocks used for the simulation are $155 \times 62 \times 6$, the step size in the $x$- and $y$-directions is 35 m, and it is 5 m in the $z$-direction. It has six geological layers, and there are 290 wells. Initially, the oil saturation in the oil rich region is 0.66, and the average oil saturation in the entire reservoir is 0.58. The initial pressure and temperature are 9.2 Mpa and 37 °C, respectively. At 30 °C, the oil viscosity is 3700 cP. The density of crude oil is 977 kg/m$^3$. The no-flow boundary condition on the external boundary of the reservoir is utilized in this and other examples.

The choice of the time step size is based on an adaptive control strategy [5], and is of the order of several days. The linear solver is based on ORTHOMIN [17] with incomplete LU factorization preconditioners, and the fully implicit scheme in time is utilized. The oil reservoir has three large faults. The parallelization strategy in the code supports unstructured domain decomposition; these faults are used as the internal boundaries of subdomains so that the data communication between them is reduced. The decomposition of the reservoir into eight subdomains (on a typical layer) is shown in Fig. 1.

The first numerical test was carried out on a linux parallel PC cluster: ROCKETCALC with 8 processors; each processor has Intel Pentium, 2 GHz, 2G RAM, and integrated Intel 100Mbps ethernet. The simulation was run on from 2 to 8 processors by consecutively doubling the number of processors. The daily oil production (m$^3$/day) and cumulative oil production (m$^3$) curves obtained using 2, 4, and 8 processors, respectively, are displayed in Figs. 2 and 3. They indicate that the production curves obtained on the different numbers of processors match very well. The wall clock time and CPU time used for the simulation run on these processors at the final time $T = 3850$ days (May 1987 – December 1997) are

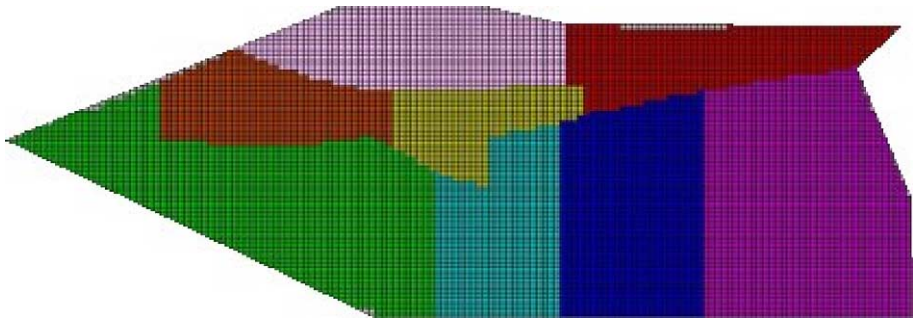

Fig. 1. A partition of $\Omega$ into eight subdomains in Example 1.

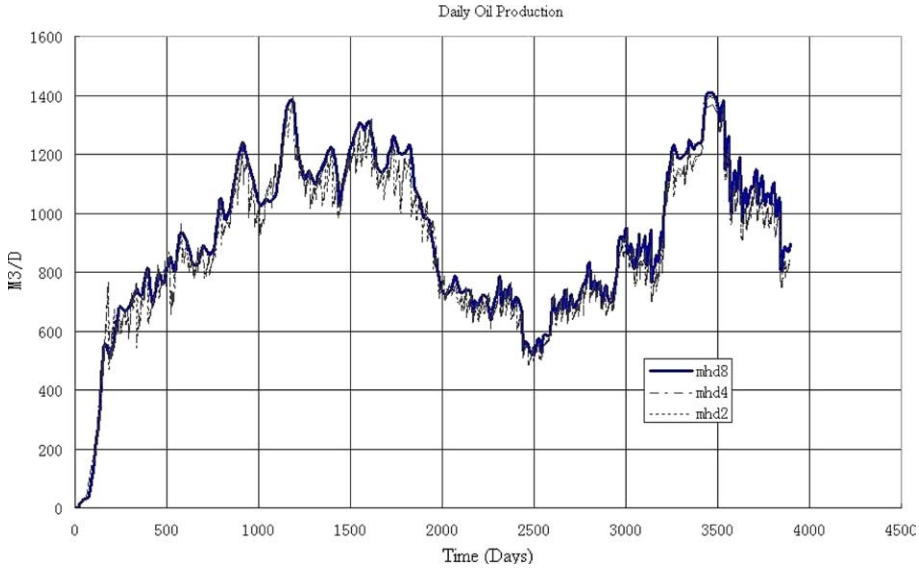


Fig. 2. Daily oil production in Example 1.

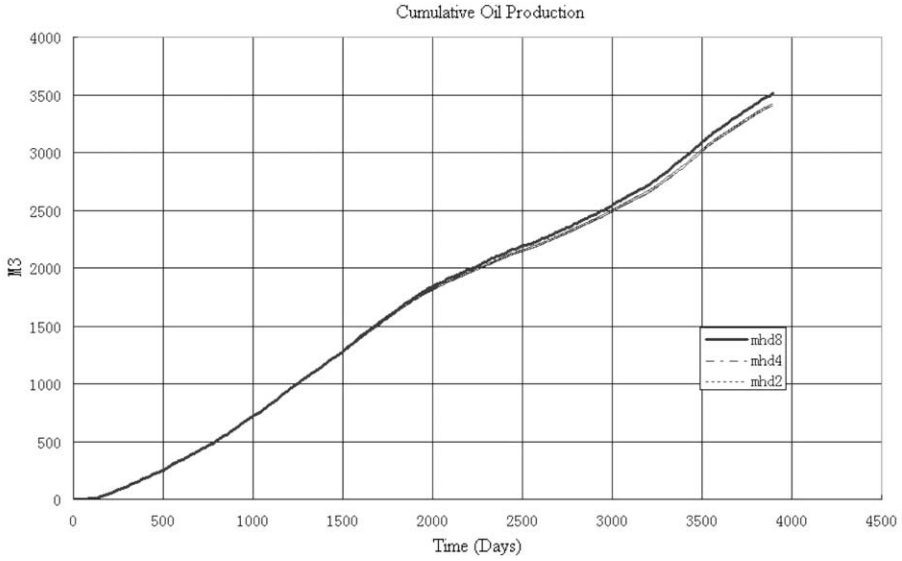


Fig. 3. Cumulative oil production in Example 1.

reported in Table 1. This table shows that doubling the processor number reduces the total execution time by far more than half. This table also shows the speed-up (relative to two processors) for the parallel code. This speed-up is defined based on the performance of two processors as $T_2/T_p$, where $T_p$ is the CPU time using $p$ processors. The speed-ups from 2 to 4 and 8 processors increase by factors of 5.87 and 37.58, respectively. These results clearly indicate that the CPU time is significantly reduced as the number of processors increases.

Table 1
Speedups on different number of processors

| | 2 | 4 | 8 |
|---|---|---|---|
| Wall-clock time (s) | 243,486.3 | 61,260.5 | 12,240 |
| CPU time (s) | 167,824.6 | 28,602.5 | 4465.9 |
| Speed-up | – | 5.87 | 37.58 |
| Number of time steps | 6636 | 3458 | 1827 |

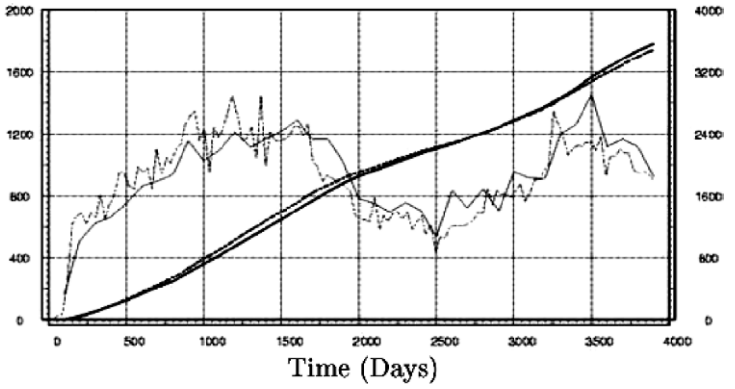


Fig. 4. History match in Example 1: (···) real production, (——) calculated production.

We now tested the same example on a different parallel computer, Dawn 2002. This computer is a cluster with 80 nodes, each node has two CPUs and 2 GB RAM, and each CPU is the IBM power P3. We first exploited the same number of grid blocks, i.e., $155 \times 62 \times 6$. The history match for the daily liquid (water and oil) production (tn/day, the left vertical axis) and cumulative liquid production (kT, the right axis) obtained using 32 processors is given in Fig. 4. The matching error for these curves is within 1.5%, which shows a high accuracy of the parallel code.

The CPU time used for the simulation run on 8, 16, and 32 processors at the final time $T = 3850$ days and the speed-ups ($T_8/T_p$) from 8 to 16 and 32 processors are listed in Table 2. The speed-ups from 8 to 16 and 32 processors increase by factors of 2.67 and 6.50, respectively, and superlinear speed-up occurs. From Tables 1 and 2, we see that the PC cluster achieves better speed-ups.

The number of grids was now doubled in the $x$- and $y$-directions, i.e., the number of grid blocks is now $310 \times 124 \times 6$. The corresponding CPU time for the simulation on 16 and 32 processors at the final time $T = 3850$ days and the speed-up ($T_{16}/T_p$) from 16 and 32 processors are described in Table 3. This table indicates that a similar speed-up is observed even if the grid is refined. As an illustration, the residual oil distribution on the first layer of the reservoir is shown in Fig. 6.

Table 2
Speedups on different number of processors on Dawn 2002

| | 8 | 16 | 32 |
|---|---|---|---|
| CPU time (s) | 42,120 | 15,780 | 6480 |
| Speed-up | – | 2.67 | 6.50 |

Table 3
Speedups with a refined grid on Dawn 2002

| | 16 | 32 |
|---|---|---|
| CPU time (s) | 69,360 | 25,860 |
| Speed-up | – | 2.68 |

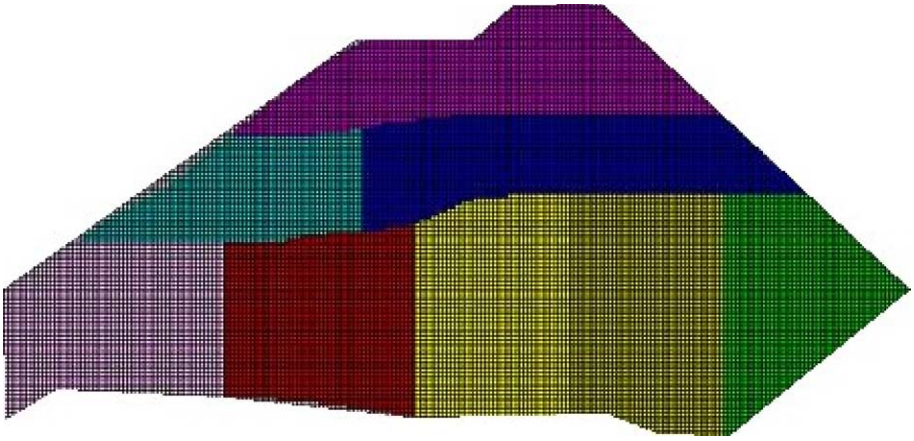

Fig. 5. A partition of $\Omega$ into eight subdomains in Example 2.

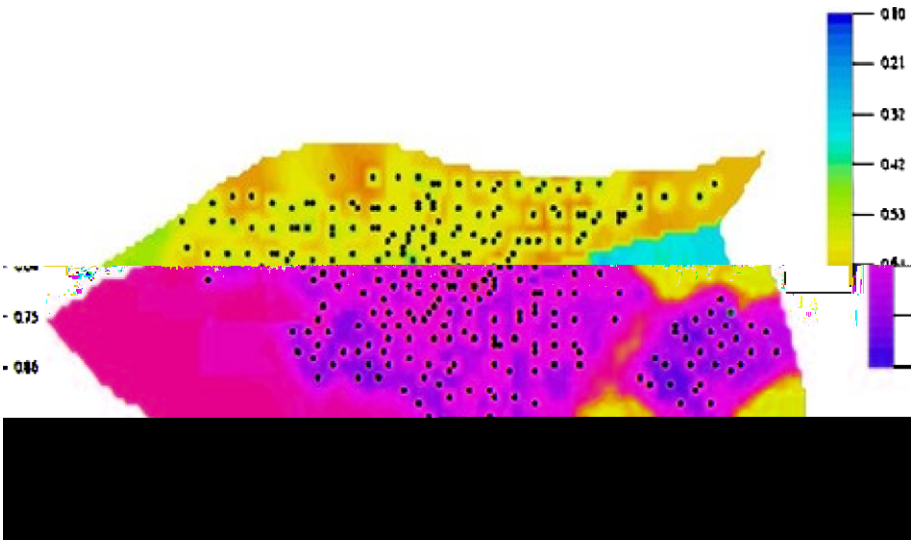


Fig. 6. The residual oil distribution for Example 1.

### 4.2. Example 2

This simulation model comes from the study of development and production schemes for a different oil field. We ran this simulation model for the time period of December 1, 1974 to October 31, 2002. The grid blocks used in this example are $240 \times 120 \times 10$, and the dimensions of the reservoir are $4770 \times 2360 \times 25$ m$^3$,

see Fig. 5, where a decomposition of the reservoir domain into eight subdomains (on a typical layer) is shown. There are 648 wells in this reservoir. Initially, the oil saturation in the oil rich region is 0.73, and the average oil saturation in the entire reservoir is 0.57. The initial pressure and temperature are 3.0 Mpa and 20 °C, respectively. At 20 °C, the oil viscosity is 3000 cP. The density of crude oil is 905 kg/m$^3$.

The simulation was run on Dawn 1000A. This computer is also a cluster with eight nodes, each node has a CPU and 256 MB RAM, and each CPU is an IBM power P2. Superlinear speed-up has been observed for this example. Instead of displaying the CPU time and speed-up, we test the accuracy of the solutions obtained by the parallel code. As an illustration, we show the oil saturation distribution on the fourth layer of the reservoir after 10,196 days in Fig. 7. The history matches for the monthly oil production (tn/month) and cumulative oil production ($10^4$ tn) obtained on this computer are given in Figs. 8 and 9, respectively. The curves predicted by this code match the real ones very well, which indicates that the parallel code provides a very accurate prediction for oil production.

The code's superlinear speed-up is attributed to both the sequential algorithm optimization (i.e, our domain decomposition approach) and the parallel computation, which can be reasoned as follows: In the thermal reservoir simulation, we cannot afford a global, serial simulation on the entire reservoir. Part of the reason is that tremendously small time steps would be used on the entire reservoir and during the entire simulation. As noted, the domain decomposition and parallel computing techniques provide an excellent scenario in which different step sizes in different regions can be effectively made. Also, in contrast to other codes where only the linear solver part is parallelized, our thermal code is globally parallelized from

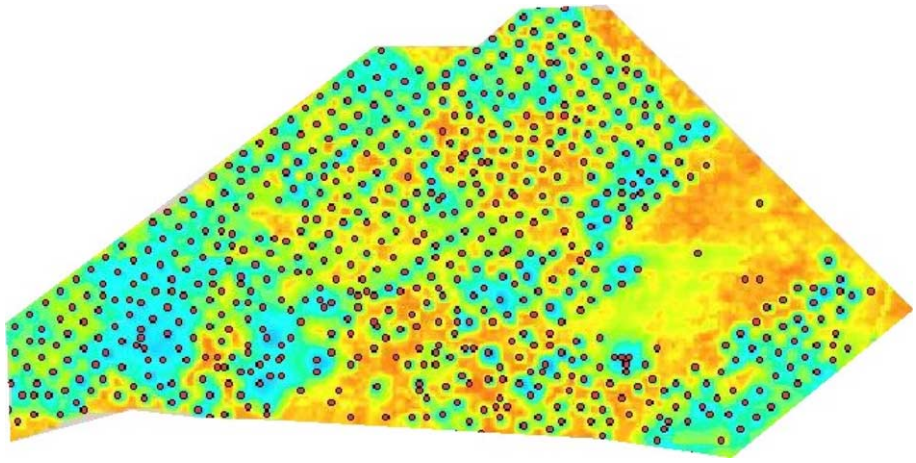

Fig. 7. Oil saturation after 10,196 days for Example 2.

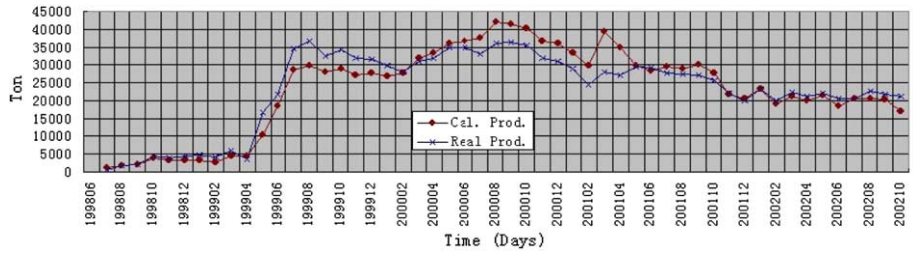


Fig. 8. History match for the monthly oil production in Example 2.

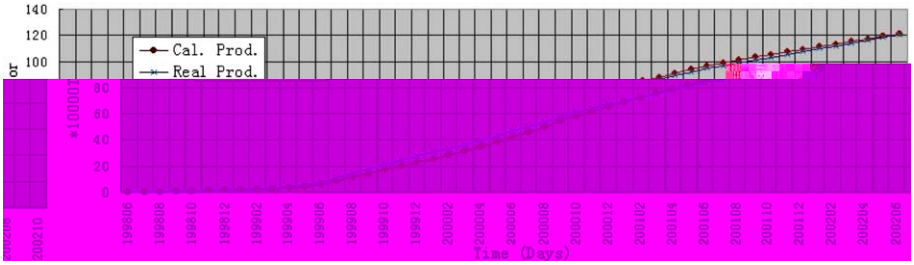


Fig. 9. History match for the cumulative oil production in Example 2.

initialization, PVT and rock property data calculation, space and time discretization, Newton–Raphson iteration, solution of linear systems, to data output. The global parallelization does not have the difficulty of parallelizing preconditioners which are by nature sequential such as the widely used incomplete LU factorization, and in addition to the parallelization of the linear solver part, the parallelization of other parts (particularly on a distributed-memory computer) is also extremely important for a large-scale reservoir simulation. Furthermore, in contrast to other parallel codes in reservoir simulation where the governing flow equations are first discretized in time and the domain decomposition technique is then applied to the resulting elliptic equations, the decomposition technique in our parallel code utilizes the parabolic structure of the governing equations. As discussed earlier, a parabolic problem has a much better condition number than a corresponding elliptic problem, and simplified domain decomposition methods can be developed when the parabolic nature is employed.

## 5. Concluding remarks

Massive parallel computing technology has been applied in our thermal simulation code for applications to large-scale petroleum reservoir simulations. In the parallel code, both computing effort and memory requirements are distributed among and shared by all processors of a parallel computer. This code has been tested on a parallel PC cluster and two Dawn multi-CPU computers. Its performance has been evaluated through simulating two field flow problems. The test results indicate that doubling the processor number reduces the total execution time by more than half and that the code achieves superlinear speed-up. The major benefits of this parallel code are that it allows adequate description of reservoirs and accurate representation of the reservoirs with high resolution in space, that it enhances the speed of the computer simulations, and that more physics can be incorporated into the reservoir models.

## Acknowledgements

The authors thank the referee for his instructive comments and valuable suggestions that led to a significant improvement of this paper. They also want to thank Prof. Ian Gladwell for checking English in this paper.

## References

[1] J. Bear, Dynamics of Fluids in Porous Media, Dover, New York, 1972.
[2] F.J.L. Briens, C.H. Wu, J. Gazdag, Compositional reservoir simulation in parallel supercomputing environment, in: SPE Paper 21214, The 11th SPE Symp. on Reserv. Simul. in Anaheim, CA, February 17–20, 1997.

[3] X. Cai, Additive Schwarz algorithms for parabolic convection–diffusion equations, Numer. Math. 60 (1991) 41–61.
[4] T.F. Chan, T.P. Mathew, Domain decomposition algorithms, Acta Numer. (1994) 61–143.
[5] Z. Chen, G. Huan, B. Li, An improved IMPES method for two-phase flow in porous media, Transp. Porous Media 54 (2004) 361–376.
[6] J.E. Killough, D. Camilleri, B.L. Darlow, J.A. Foster, A parallel reservoir simulator based on local grid refinement, in: SPE Paper 37978, Presented at the 1997 Reservoir Simulation in Dallas, TX, June 1997.
[7] Y. Kuznetsov, New algorithms for the approximate realization of implicit difference schemes, Sov. J. Numer. Anal. Modeling 3 (1988) 99–114.
[8] L.W. Lake, Enhanced Oil Recovery, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[9] P. Le Tallec, Domain decomposition methods in computational mechanics, Comput. Mech. Adv. 2 (1994) 121–220.
[10] X. Ma, D. Ma, personal communication.
[11] Y. Ma, X. Ma, D. Ma, Numerical simulation of large-scale heavy crude reservoir (in chinese), Special Oil Gas Reservoirs 8 (2001) 32–35.
[12] G.A. Meurant, Numerical experiments with a domain decomposition method for parabolic problems on parallel computers, in: R. Glowinski et al. (Eds.), The Fourth Int. Symp. on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1991.
[13] J. Nedelec, Mixed finite elements in $\Re^3$, Numer. Math. 35 (1980) 315–341.
[14] P.A. Raviart, J.M. Thomas, A Mixed Finite Element Method for Second Order Elliptic Problems, Lecture Notes in Math, vol. 606, Springer, Berlin, 1977, pp. 292–31.
[15] T. Russell, M.F. Wheeler, Finite element and finite difference methods for continuous flows in porous media, Chapter II, The mathematics of reservoir simulation, in: R. Ewing (Ed.), Frontiers in Applied Mathematics 1, SIAM, Philadelphia, PA, 1983, pp. 35–106.
[16] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[17] P.K. Vinsome, W. Orthomin, An iterative method for solving sparse banded sets of simultaneous linear equations, in: Paper SPE 5729, Presented at the SPE-AIME Fourth Symposium on Numerical Simulation of Reservoir Performance, Los Angeles, CA, February 1976.