Revised, final form, July 1994

Domain Decomposition, Parallel Computing and Petroleum Engineering

Petter E. Bjørstad * Terje Kårstad †

Abstract

A prototype black oil simulator is described. The simulator has a domain-based data structure whereby the reservoir is represented by a possibly large number of smaller reservoirs each having a complete local data structure. This design is essential for effective use of preconditioning techniques based on domain decomposition. The chapter describes a splitting technique for the solution of the nonlinear system and an effective implementation of the algorithm on massively parallel computer systems. Most communication is localized and long range communication is kept at a minimum. Results from an implementation of the method are reported for a 16384 processor MasPar MP-2.

1 Introduction

This chapter describes a numerical algorithm for the simulation of flow in porous media. The problem carries substantial economic interest in the petroleum industry. It is widely used for planning purposes, for reservoir management and for prediction of the reservoir performance [3]. Another field of application is the study of ground water flow, in particular, the study of ground contamination by way of pollution simulation models [1]. The main emphasis of our approach is to exploit data locality by using a data structure in which locality in the physical model implies locality in the computer representation. The domain decomposition approach that we follow effectively separates short and long range communication into two distinct phases in each preconditioning step. This makes it easy to analyze the performance of the method and to adjust the notion of locality with respect to the target computer at hand. Our goal is an algorithm that will be flexible, portable and efficient in a distributed memory system.

*Institutt for Informatikk, Høyteknologisenteret, University of Bergen, N-5020 Bergen, Norway. This author's work was supported by NFR under grant no 27625.

†Statoil, P.O. Box 300 N-4001 Stavanger, Norway. This author's work was supported by Statoil.

A simulator based on the algorithms in this chapter should adapt well to hierarchical memory, cache-based single-processor systems, as well as distributed-memory parallel computers with a wide range of performance characteristics.

Even a simple mathematical model describing the flow in porous media is a nonlinear system consisting of an elliptic equation and a formally parabolic convection-diffusion equation. Our algorithms and techniques apply to a much larger class of applications. Application of similar ideas in the semiconductor context using a simple drift-diffusion model is reported in [5]. Here one encounters a system consisting of a Poisson equation coupled to two, formally parabolic, advection-diffusion equations [32].

2 The governing equations

A petroleum reservoir consists of hydrocarbons and other chemicals trapped in tiny pores in the rock. If the rock permits and if the fluid is sufficiently forced, it will flow from its current position in the rock to some other points in the reservoir. By injection of additional fluids and the release of pressure through the production of fluids at wells, the petroleum engineer can adjust the flow rate and modify the mixture of chemicals produced. In the associated reservoir flow problem, we are given the reservoir conditions (pressure and saturation) and the well flow conditions. The main problem is to model the fluid flow, especially to predict the fluid flow into production wells. The fluid flow problem can be expressed in terms of approximate mathematical equations. The physical laws that govern the flow can be derived from volume balance, phase equilibrium and conservation of mass, plus Darcy's Law stating that the fluid flow is proportional to pressure gradients and gravitational potential differences [1]. The constants of proportionality depend on permeability, viscosity and the density of the phases involved. The equations constituting a mathematical model for a reservoir are almost always too complex to be solved by analytical methods, even after many idealizations.

In order to focus on the main ideas of the algorithm we will here consider a simplified *prototype* black oil simulator in two space dimensions. We assume incompressible flow and neglect gravity. One can then reduce the model [16] to an equation for the pressure (here for the phase pressure p of water)

$$-\nabla \cdot \mathbf{A}(s, \mathbf{x}) \cdot \nabla p = q_1(\mathbf{x}, t), \tag{1}$$

and a saturation equation (here for the relative saturation s of water)

$$\phi \frac{\partial s}{\partial t} + \mathbf{v}(\mathbf{x}) \cdot \nabla f(s) - \epsilon \nabla \cdot (\mathbf{D}(s, \mathbf{x}) \cdot \nabla s) = q_2(\mathbf{x}, t). \tag{2}$$

The parameter ϵ is a scaling parameter introduced when the equations are converted to dimensionless form. It reflects the reciprocal of the dimensions of the reservoir and is therefore most often quite small. The source terms q_i depend on the rate of oil production and water injection in the reservoir. The reservoir properties are the permeability tensor

$$\mathbf{A}(s,\mathbf{x}) = (\lambda_o(s) + \lambda_w(s))\mathbf{K}(\mathbf{x}), \tag{3}$$

the fractional flow function of water,

$$f(s) = \frac{\lambda_w(s)}{\lambda_w(s) + \lambda_o(s)}, \tag{4}$$

and the function

$$\mathbf{D}(s,\mathbf{x}) = \lambda_o(s) f(s)\mathbf{K}(\mathbf{x})\frac{dp_c}{ds}. \tag{5}$$

The total velocity $\mathbf{v} = \mathbf{v}_w + \mathbf{v}_o$ is given by Darcy's law as

$$\mathbf{v}(\mathbf{x}) = -\mathbf{A}(s,\mathbf{x}) \cdot \nabla p. \tag{6}$$

The oil and water parameters giving the ratio of the relative and absolute permeability tensors are defined by

$$\lambda_i(s) = \frac{k_i(s)}{\mu_i} \qquad i = o, w$$

where $k_i(s)$ is the relative permeability and μ_i is the viscosity of oil and water $(i = o, w)$. Finally, $\mathbf{K}(\mathbf{x})$ is the absolute permeability tensor, ϕ is the porosity of the rock, and $p_c(s)$ is the capillary pressure. These quantities are known properties. $\mathbf{K}(\mathbf{x})$ is a measured quantity describing how fluid can move through the rock. The capillary pressure is assumed to be a known function of the computed saturation s; the oil phase pressure is then determined from the relation $p_o = p - p_c$. The pressure equation is modeled with $\mathbf{K}(\mathbf{x}) = 0$ near the boundary in order to enforce a no flow condition. Together with an initial known value for the pressure along the boundary, this effectively imposes a Dirichlet boundary condition. The saturation equation is modeled with a normal no flow condition (Neumann condition), but the injection wells contribute isolated Dirichlet conditions.

3 The Solution Strategy

There are many different approaches for solving the equations. One can treat the complete nonlinear system using Newton's method [3] or decouple the system and iterate between the pressure equation and the saturation equation. The pressure equation is solved with the saturation fixed, and then the saturation equation is solved with the velocity (pressure) fixed. This splitting is not supported by a rigorous mathematical theory; rather it is heuristically motivated by the different behavior of the pressure equation and the saturation equation [1].

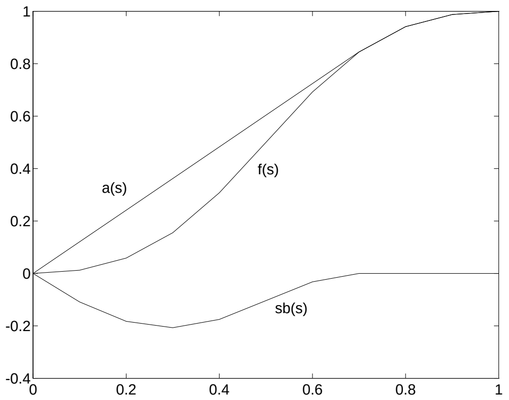

FIG. 1. *The plot shows a typical fractional flow function $f(s)$, and also its splitting $f(s) = a(s) + sb(s)$, where $a(s)$ is the convex envelope of $f(s)$.*

The remainder of this section derives the appropriate equations that are numerically solved at each time step. This derivation is a bit technical and may be skipped by the reader whose primary interest is the application of domain decomposition techniques to partial differential equations. Note that the two resulting equations (1) and (12) have a general form similar to what is encountered in many other applications.

Our method decouples the equations and is based on the work of Espedal and Ewing [27]. Both equations should be solved implicitly in order to allow the use of large time steps. However, in order to handle both the first and second order terms in (2), we begin by splitting off a purely hyperbolic part

$$\phi \frac{\partial s}{\partial t} + \mathbf{v}(\mathbf{x}) \cdot \nabla a(s) = 0. \tag{7}$$

For a convex fractional flow function $a(s)$ the solution to the Riemann problem (7) (given $\mathbf{v}$) is either a continuous solution, called a rarefaction, or a discontinuous solution, called a shock. A common procedure for solving such problems is to use time-stepping along the characteristic curves [20, 29, 37]. Figure 1 shows the shape of a typical fractional flow function $f(s)$ for an oil-water problem. We see that $f(s)$ is not convex. Since the procedure just described depends on convexity we must introduce a splitting of the function. Following [27] (see also [18, 38]), we split the fractional flow function $f(s) = a(s) + sb(s)$, where the convex part $a(s)$ appears in (7). The characteristic system associated with Equation (7) is given by

$$\begin{aligned} \frac{ds}{d\tau} &= \phi \frac{\partial s}{\partial t} + \mathbf{v}(\mathbf{x}) \cdot \nabla a(s) = 0 \\ \frac{dt}{d\tau} &= \phi \end{aligned} \tag{8}$$

$$\frac{d\mathbf{x}}{d\tau} \quad = \quad a'(s)\mathbf{v}(\mathbf{x}).$$

The source term q_2 in (2) models injection wells. These can be described as point sources and adequately taken into account by specifying that $s = 1$ at the corresponding coordinates. We can now use the first equation in (8) and rewrite Equation (2) as

$$\frac{ds}{d\tau} + \mathbf{v}(\mathbf{x}) \cdot \nabla s b(s) - \epsilon \nabla \cdot (\mathbf{D}(s, \mathbf{x}) \cdot \nabla s) = 0. \tag{9}$$

From the last equation in (8) we can find a first approximation for the saturation at a new time τ^{n+1} by tracing the characteristic curve back (upstream) to time level τ^n. In the following, let $s(\mathbf{x})$ denote the saturation at the new time level τ^{n+1} while $s(\bar{\mathbf{x}})$ is evaluated at the previous time τ^n. In light of (7), the saturation $s(\mathbf{x})$ at time level τ^{n+1} should equal $s(\bar{\mathbf{x}})$ at the time τ^n provided that $\bar{\mathbf{x}}$ and $\mathbf{x}$ lies on the same characteristic. That is, given our current coordinate $\mathbf{x}$ we determine $\bar{\mathbf{x}}$ from the single nonlinear equation

$$\bar{\mathbf{x}} \quad = \mathbf{x} - \int_{\tau^n}^{\tau^{n+1}} \mathbf{v}(\mathbf{x}) a'(s(\mathbf{x})) d\tau. \tag{10}$$

This step can be carried out independently for all grid points in the model. The operations are purely local, but in this process one must be able to access the data related to other grid points that are on the same characteristic in the upstream direction. Using the second relation in (8) we approximate the term $\frac{ds}{d\tau}$ in (9) with

$$\frac{\phi}{\Delta t}(s(\mathbf{x}) - s(\bar{\mathbf{x}})). \tag{11}$$

Inserting (11) in (9) and avoiding the nonlinearities in $b(s, \mathbf{x})$ and $D(s, \mathbf{x})$ by using the approximate value $s(\bar{\mathbf{x}})$ from the previous time step, we obtain a linear equation for the saturation $s(\mathbf{x})$ at the new time level

$$s(\mathbf{x}) + \Delta t \phi^{-1} \mathbf{v}(\mathbf{x}) \cdot \nabla (b\, s(\mathbf{x})) - \epsilon \Delta t \phi^{-1} \nabla \cdot (\mathbf{D} \cdot \nabla s(\mathbf{x})) = s(\bar{\mathbf{x}}). \tag{12}$$

To summarize, our solution procedure involves three main computational tasks at every time step: first the solution of the pressure equation (1), next the solution of (10) in order to determine $s(\bar{\mathbf{x}})$, and finally the solution of (12) to find the saturation $s(\mathbf{x})$ at the new time level.

4 Distributed Data Structure

Our choice of data structure for the computer realization of the algorithm is guided by the following considerations:

- Data locality. Independent of any particular computer architecture and specific algorithm, this concept has been established as a primary guideline for high performance, efficient algorithms.

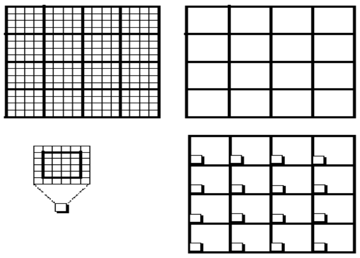

FIG. 2. *The domain on the left, the distributed data structure on the right. Each local data structure contains a complete subdomain with its boundary. In addition, there is one grid point from a coarser discretization in the local structure.*

- Domain decomposition. To be described in the next section, our iterative algorithms depend on an effective preconditioner based on domain decomposition. This representation of the equations restricted to subdomains specifies precisely what data is needed in the local data structures.

- Communication requirements. The explicit distributed memory model defines the necessary communication precisely. In contrast, the data movements are most often implicit in a (virtual) shared memory or cache-based computer. The data structure and the algorithm will together define both the volume and structure of data communication. Keeping this in mind, we will try to separate short and long range communication, minimize the total volume, and balance this with the required volume of computation.

Based on this, it is very natural to consider a (distributed) data structure where the domain (reservoir) is partitioned into many subdomains. Each subdomain carries the entire data structure of a reservoir. In fact, each subdomain is equipped with enough data, including material properties and boundary data, to enable a processor to carry out a complete reservoir simulation. The boundary data is therefore duplicated in subdomains that are neighbors. The stiffness matrices for equations (1) and (12) are assembled and stored as part of the local data structure for each subdomain.

Our algorithm as outlined in the previous section has three basic steps. Since we use preconditioned Krylov space methods for steps 1 and 3, the data structure must be able to support multiplication of vectors by the stiffness matrices as well as efficient application of subdomain-based preconditioners. The matrix vector product needs attention since the global stiffness matrix is *different* from the sum of the local matrices because of the overlap between subdomains. (Since the local data structures store interior boundary information redundantly.) In addition, each subdomain data structure will store at least one grid point belonging to a coarser, global grid. This is an important part of the overall algorithm [Cai, this volume] and the local data structure must therefore also support restriction of its local data to this coarse grid point and interpolation of a value from the coarse data point to all appropriate local data points. Figure 2 gives a schematic picture of the local data structure that naturally is associated with a (possibly virtual) processor.

5 The Numerical Algorithm

With a suitable data structure in place, we can now describe the numerical algorithm. Our focus will be on the main steps with emphasis on data locality and, in particular, on what non-local data movements are necessary. The reader should keep in mind that the procedure can be carried out on one processor (in a loop over local subdomains) or on a parallel set of processors, in the limiting case with as many processors as there are subdomains.

Since we use domain decomposition algorithms extensively we now introduce a particular version, the Additive Schwarz Method (ASM). An extensive literature on this subject exists, see for example [6, 11, 12, 13, 14, 21, 23, 24] and in the references these articles. Domain decomposition algorithms are usually used as sophisticated preconditioners. In fact, the additive method that we describe below may not even converge if used as a stand alone iteration, and is therefore always used together with an outer iterative Krylov space method.

The ASM method was originally motivated by the famous alternating iteration of H.A. Schwarz [39], but it can also be viewed as a generalization of the block Jacobi iteration [26]. Define a restriction matrix R_i by

$$u_i = R_i u$$

where u_i is a grid function defined in the local data structure i. Here u is the global reservoir grid function and R_i consists of zeros and ones in such a way that it only selects the components that belong to subdomain i. The locally stored part of the stiffness matrix is similarly given by $A_i = R_i A R_i^T$. Our preconditioner will be built from the inverses of these local matrices.

Let $B_i = R_i^T A_i^{-1} R_i$. The first part of the preconditioner is then given by

$$B = \sum_{i=1}^{N} B_i \tag{13}$$

The solution of elliptic (diffusion dominated) problems depends on both local and long range effects, as can be easily seen from the structure of the relevant Green's function for such problems. An effective preconditioner must therefore incorporate a mechanism for global exchange of information between all subdomains at every iteration. If not, the number of iterations will necessarily be bounded from below by $O(1/H)$, roughly the number of subdomains spanning the nondimensionalized domain, where we assume that a typical subdomain has diameter H. If the subdomains communicate only with their nearest neighbors (via the shared overlap) then this bound is the time it takes before the influence of one subdomain has propagated across the full domain. This may be acceptable with relatively few subdomains. We are interested in having thousands of subdomains for two reasons. The work per subdomain decreases when the local problems are smaller and the degree of parallelism increases. We therefore introduce a coarse discretization A_c of the domain, with element size equal to the previously defined subdomains. Consistent with previous notation, let R_0^T be the interpolation from the coarse grid to the original (fine) grid and let the transpose R_0 define the corresponding restriction. With

$$B_0 = R_0^T A_c^{-1} R_0, \tag{14}$$

the complete two-level preconditioner is

$$B = \sum_{i=0}^{N} B_i.$$

This class of domain decomposition preconditioners can be shown to have optimal convergence properties in the sense that the number of iterations are bounded independently of the discretization parameters h (of the fine grid), H (of the coarse grid) and of the ratio H/h as long as the overlap between subdomains is fixed independently of h. More information on the theoretical properties of the method can be found in the references listed above. In our implementation, we violate the assumption on the overlap size, by keeping only one grid line of overlap independent of h. This does lead to a modest increase in (outer) iterations, but extensive experience [7, 8, 40] and theoretical results [25] indicate that this may still be advantageous when the amount of data stored and the duplicated computation in the overlap area is taken into account. That is, a small overlap tends to minimize the overall computational time.

We next discuss the calculations that must be carried out at each time level of the simulation. Our focus is on the data locality issue; a more detailed description of the algorithm and its implementation is given in [34].

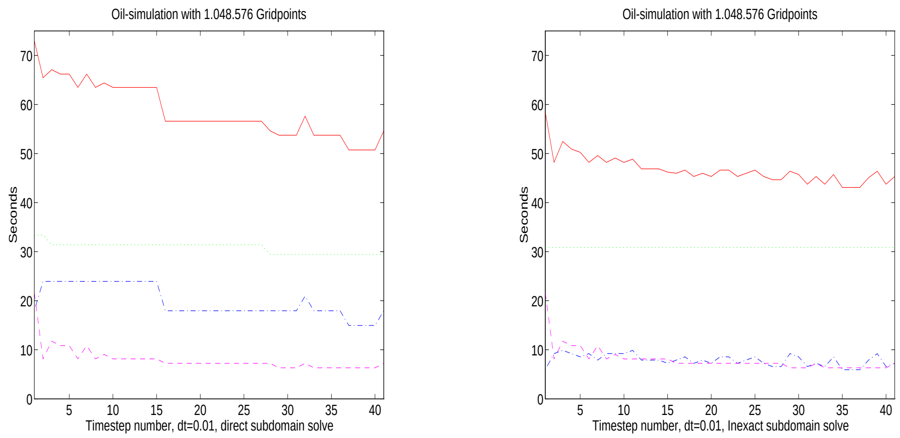

FIG. 3. *Elapsed time for one iteration of the algorithm broken down into the three main steps. The problem has a million grid points and runs on the full MP-2. In the left plot the pressure equation takes about ten seconds, the saturation equation about 20 seconds, the hyperbolic equation about 30 seconds, the top curve shows the total time. A direct method is used for the subdomain solution. The same information appears in the rightmost plot, but an iterative method is used. The time step is fixed at 0.01 in both cases.*

5.1 Solution of the pressure equation

The pressure equation is discretized using linear (P_1) finite elements. The velocity is computed using (6). A detailed description of this procedure can be found in [38]. An alternative technique is a mixed finite element method where both the pressure and the velocity is directly approximated to second order [28].

The first computational task is the calculation of the permeability tensor (3), which is a completely local computation. In the reservoir problem this coefficient is often characterized by having large variations and jump discontinuities. Next, we compute the stiffness matrix for (1) and the corresponding right hand side vector; again carried out independently in each local subdomain. This calculation is repeated for the coarse (subdomain size) triangulation of the domain, again within each local data structure.

The resulting linear system is symmetric, positive definite and the appropriate Krylov space method is Conjugate Gradients [30]. The matrix vector product is a completely local calculation. We precondition the system using the ASM method described above. The local problems

B_i $,i = 1, 2, \ldots,$ are solved directly or approximately using a fixed, small number of SSOR iterations (Symmetric Successive Over-Relaxation [30]). In a case where the number of grid points in a single subdomain exceeds, say, a hundred, one should implement a simple multigrid-based iteration in order to keep the complexity of this step as low as possible.

The application of (13) to a vector requires communication between the nearest neighbor data structures when the local boundary values are updated. The coarse grid matrix (14) requires exchange of data between all subdomains. We use a fixed number of multigrid V-cycles [10, 31] for this purpose. This gives us a very structured long range communication pattern. Previous work [8, 40] on this type of equation has shown that the volume of long range data exchange can be balanced with the local data communication between neighbor domains by adjusting the H/h ratio. Perhaps equally important, the ASM preconditioning is very robust with respect to large variations in the permeability [8]. Under certain circumstances, it is possible to prove that ASM preconditioners will give a rate of convergence that is independent of jumps in the coefficients; see [22] and the references therein.

Similarly, the dot-products in the conjugate gradient iteration require non-local interaction of data. The relative importance of the three kinds of data movement that we encounter in this algorithm depends on the computer architecture at hand. Specific models of this cost is normally easy to establish when the computer is known.

The last part of this computational step determines the total velocity $\mathbf{v}$ from the computed pressure. We note that this is a local calculation. A careful procedure based on control volumes and flux calculations is described in [38].

5.2 Solution of the hyperbolic equation

The numerical solution of (10) is based on the assumption of a smooth velocity field that can be approximated locally by a constant. In order to satisfy this, the time interval $[\tau^n, \tau^{n+1}]$ is broken into several subintervals. The procedure then evaluates (10) starting with the maximal value of the slope of $a(s)$ (see Figure 1). If the final value of $s(\bar{\mathbf{x}})$ at time τ^n is consistent with our upstream integration along a characteristic (i.e., the value is unchanged) then we stop, otherwise the correct saturation is higher and we repeat the procedure doing bisection using smaller values of $a'(s)$.

This algorithm is carried out for all grid points within each local subdomain, but the calculations may, depending on the subdomain size and the length of the time step, access data along the characteristic across a sequence of nearby subdomains. Furthermore, this path may be quite different in different parts of the reservoir. The resulting data movement becomes unstructured.

5.3 Solution of the saturation equation

Our last computational step is to solve (12). This equation is also treated using an ASM preconditioner and several computational steps are similar to those described for the pressure equation. The main difference (and difficulty) is the presence of the first order convective term $\Delta t \phi^{-1} \mathbf{v} \cdot \nabla(b(\mathbf{x})s(\mathbf{x}))$. The equation is discretized with finite elements, using rectangular elements and bilinear basis functions. A Petrov-Galerkin formulation is used to better represent the convective term. More details on this can be found in [4, 19, 33, 15, 35]. The equations are non-symmetric and the Krylov space method BiCGstab [41] was found to perform satisfactory. For relatively small subdomains a direct subdomain solver is competitive and such a solver provides a more robust preconditioner and leads to a smaller number of outer iterations than using an inexact, iterative solver. We have observed cases (with large time steps) where this choice was necessary for convergence; the inexact subdomain solver would only succeed with a smaller time step. Our preferred inexact solver for the nonsymmetric case is based on the nested factorization described in [2], a popular method in the reservoir engineering community.

The nearest neighbor communication and the need for dot-products are similar to the situation discussed for the pressure equation. The main difference is the role of the long range component (14) of the preconditioner. The previous hyperbolic step should (if our splitting is successful) essentially move the oil-water front to its new location. The saturation equation that we solve here serves as a correction in order to fully represent the fractional flow function (4) and account for the diffusive term. If ϵ in (12) is small then we find very satisfactory convergence without using the coarse space, thus eliminating the need for long range data movements in this step. On the other hand, if ϵ is relatively larger, the situation is similar to the pressure equation and a global exchange of data in the reservoir is needed. The situation is complicated by several factors. The two most important are the splitting of the fractional flow function, which changes the relative size of the convective and the diffusive term as a function of the saturation, and the variable distance the (oil-water) front moves in one time step, relative to the size of the subdomains. We have found it difficult to design a coarse space component (14) of the preconditioner that works satisfactorily in all cases. A more refined implementation that provides non-local coupling *only* in the areas of the reservoir that have significant diffusion may be needed.

6 Parallel Computing Aspects

The design of our data structure and the structure of the solution algorithm make a parallel implementation feasible. The algorithm has been described as a sequence of computational steps operating on a (possibly) large number

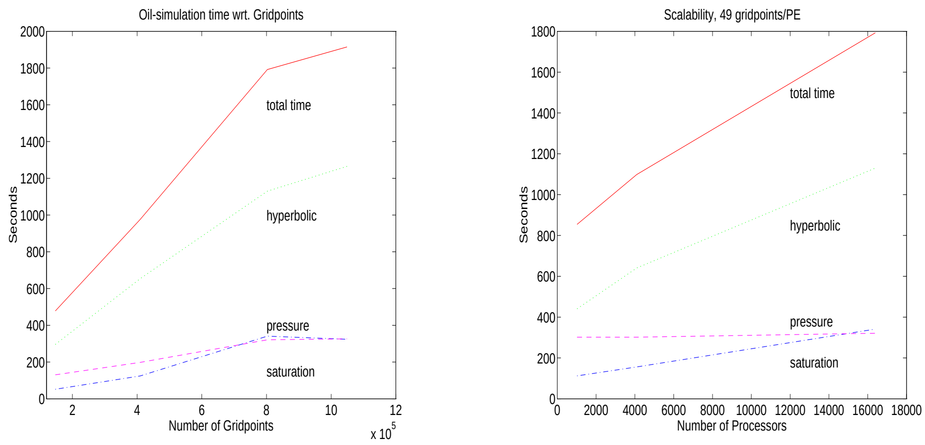

FIG. 4. *The figure shows how the total simulation time (40 time steps) scales. To the left we change the size of the subdomain grid from* 2×2 *to* 8×8 *corresponding to an increase in the number of grid points by a factor 16. The size of the machine is 16384 processors. In the rightmost plot we keep the subdomain grid fixed at* 7×7 *per processor, but scale the size of the machine (and therefore also the size of the problem) by a factor 16.*

of local data structures. Since the access of non-local data is identified and isolated in the description, we can easily analyze the communication overhead for a given parallel computer. An important consideration in domain decomposition algorithms is the size (and the number) of the individual subdomains. We prefer to discuss this somewhat independently of the number of processors on a target parallel machine. One can often think in terms of having a one to one mapping between subdomains and virtual processors. The assignment of virtual processors to actual processors in a parallel computer may then be considered as a separate problem. Still, the number of subdomains should normally be at least equal to the number of processors available. There is a volume-to-surface relation between the computation and the communication when applying the first level of our preconditioner (13). If the algorithm used inside a subdomain has a computational complexity growing faster than linearly in the number of unknowns, small subdomains are strongly favored. Aside from the increased volume of local communication, this must also be balanced with the increase in size of the coarse grid problem (14). On massively parallel machines the coarse grid problem is a significant computation. In this case, it often is advantageous to solve the coarse grid problem separately from the subdomain problems, by applying a parallel algorithm for its solution. One may also use a hybrid multiplicative additive version of the preconditioner since the two steps will be sequential anyway [Cai, this volume].

The unstructured communication encountered when solving (10) poses a very different problem, in particular, on computers that have a highly regular interconnection network. First, observe that it also depends on the size of the subdomains discussed above. A larger subdomain size will reduce or at least localize the data access in this task, since the oil-water front will tend to stay within the same or at most move to a neighboring subdomain during one time step. We also note the influence of the step size in time; decreasing the time step will have the same effect as making the subdomains larger. The computer simulations reported here used a time step that sometimes moved the oil-water front as far as five subdomains in one step. The algorithm we described for step (10) is therefore not very suitable for a highly structured machine like the MasPar MP-2.

These observations indicate a way to partly circumvent the problem. We can try to subdivide our time step. Instead of searching upstream over the full interval $[\tau^n, \tau^{n+1}]$, we take a shorter step starting from, say, $\tau^{n+1/k}$. As soon as the approximate saturation $s(\bar{\mathbf{x}})$ has been found by following the upstream direction, we can update this intermediate value of s at time $\tau^{n+1/k}$ and repeat the process (starting from $\tau^{n+i/k}$ $i = 2, \ldots, k$) until we obtain the approximate value $s(\bar{\mathbf{x}})$ at time level τ^{n+1}.

To what extent is the overall algorithm discussed in this chapter scalable? The definition of a scalable algorithm for a complex simulation like this is far from easy. Intuitively, one would like to ask, for a fixed number of subdomains per processor, each of a fixed size, what happens if one doubles the size of the problem by doubling the size of the machine? An unchanged elapsed computing time would be one (ideal) definition of scalability. (That is for fixed memory size, check if the computing time scales with the number of processors.) However, because of the hyperbolic step, an increase in the spatial resolution will unavoidably increase non-local communication unless the time step is reduced as well. This effect will tend to increase the elapsed time for the simulation. On the other hand, even with implicit schemes that may be able to maintain long time steps, if the spatial resolution increases it is often natural to also increase the resolution in time. This will increase the elapsed time for a given simulation violating our scalability since the time-stepping does not reflect increased memory usage. A more restricted measure would be to consider the elapsed computing time for a single simulation time step with suitable assumptions on the relative resolution in space and time. In this situation the entire algorithm should behave similarly to the pressure solver. It can be seen from Figure 4 that our pressure solver scales with the number of processors up to at least 16384 on our MP-2. Thus, such a scalability result seems within reach, but it may not be what the reservoir engineer asks for: *constant time to solution.*

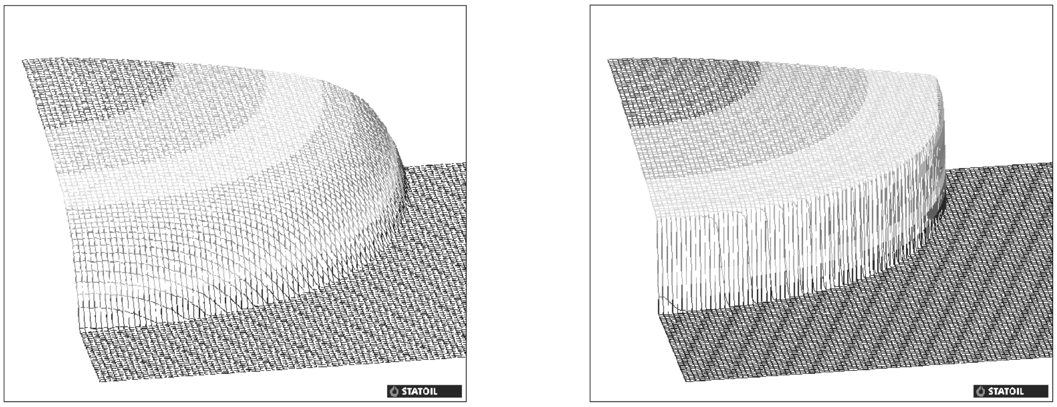

FIG. 5. *The saturation profile after 15 time steps (with* $\Delta t = 0.018$*) when 32 percent of the oil is produced. The left plot shows a simulation using 1024 processors with* $\epsilon = 0.01$ *and* $h = 1/(6 * 32)$*. The right plot shows the same simulation using 16384 processors with* $\epsilon = 0.0001$ *and* $h = 1/(6 * 128)$*.*

7 Results and Discussion

We have tested an implementation of this *prototype* simulator on a 16384 processor MasPar MP-2 computer having one gigabyte of distributed memory. The machine is a SIMD computer with a peak speed of approximately 2.3G 64 bits floating point operations per second. It has a high capacity mesh interconnect providing communication to a processor's 8 nearest neighbors called the XNET and a general 3-stage interconnect called the ROUTER for arbitrary (unstructured) communiction. A more detailed description of the machine can be found in [9, 17, 36].

The simulation reported here is a standard industry benchmark with parameters given in [34]. There is one injection well and one production well and the oil-water front is sweeping across the domain. We assign one subdomain to each processor and consider the performance of the algorithm. First, in Figure 3 we measure the elapsed time per time step of the three main computational steps. We use a grid with more than one million grid points, corresponding to a little less than one hundred grid points per subdomain. This is close to the memory limitation of the machine, given all the general data that must be stored and that scales with the number of grid points in the simulation. Observe the difference in time when using a direct solver for the subdomains due to the more complex saturation equation, while the times are very similar when the inexact iterative method is used. The figure is for $\epsilon = 0.001$, and there is no coarse grid component in the preconditioner for the saturation equation. We notice that more than half of the time is spent in our second computational step, the integration along characteristics.

This is directly related to communication time as we use the slow, but very general, ROUTER hardware to handle the unstructured communication. We have not yet tested the substepping alternative discussed in the previous section. Presumeably, such a change in the algorithm would enable us to use the much faster XNET communication. The overall elapsed time for one time step is between 50 and 60 seconds, giving a total simulation time of about 30 minutes.

In Figure 4 we show how the simulation time scales with the number of grid points and also how a change in the size of the machine affects the elapsed time. The simulation is performed with a fixed time step of 0.01 independent of the resolution in space; this should be kept in mind when interpreting the results. The first figure shows the increase in simulation time as we increase the size of the problem (by increasing the subdomain size). We observe about a factor of four in increased simulation time as the problem size is scaled up by a factor of sixteen. This shows the increased computational efficiency as the relative time spent on communication (and computational overhead) decreases. In the other half of the figure we scale the problem and the machine, keeping the amount of data per processor fixed. As discussed earlier this implies that each processor gets a smaller and smaller piece of the reservoir to compute. Notice the perfect scaling of the pressure solver at about 300 seconds. We have $\epsilon = 0.001$ and do not use the coarse grid in the saturation solver. This shows by the increase in the time for this part of the simulation. Overall, the simulation time increases by a factor of two when we scale the problem and machine by a factor of sixteen.

In summary, the design and testing of this algorithm shows that a domain based decomposition of an oil reservoir simulator is a promising approach for the development of such codes for massively parallel computers. The paradigm has flexibility to balance the computation and communication time on a wide range of computer architectures. Many issues remain. The design of a more adaptive coarse grid operator for the saturation equation is needed. We believe that our step 2, following streamlines in parallel, can be substantially improved. Further generalizations of the method to handle the compressible case, three dimensions and three phases should be investigated. We believe that some of the guiding principles when developing this algorithm can be useful also in other and more general situations. Practical developments of parallel, portable code, based on concepts like the ones presented here, is still a formidable challenge.

References

[1] M. B. Allen, G. A. Behie, and J. A. Trangenstein, *Multiphase Flow in Porous Media*, Springer-Verlag, 1988. Lecture Notes in Engineering.

[2] J. R. Appleyard and I. M. Cheshire, *Nested factorization*, November 1983. Presented at the Seventh Reservoir Simulation Symposium, San Francisco, SPE

12264.

[3] K. Aziz and A. Settari, *Petroleum reservoir simulation*, Elsevier Applied Science Publishers, 1979.

[4] J. W. Barrett and K. W. Morton, *Approximate symmetrization and Petrov-Galerkin methods for diffusion-convection problems*, Comp. Meth. in Appl. Mech. and Eng., 45 (1984).

[5] P. Bjørstad, J. W. M. Coughran, and E. Grosse, *Parallel domain decomposition applied to coupled transport equations*, in Proceedings from the Seventh International Conference on Domain Decomposition Methods in Scientific and Engineering Computing, D. E. Keyes and J. Xu, eds., Providence, 1994, AMS. Penn State University, October 1993.

[6] P. E. Bjørstad, *Multiplicative and Additive Schwarz Methods: Convergence in the 2 domain case*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1989, SIAM.

[7] P. E. Bjørstad, R. Moe, and M. Skogen, *Parallel domain decomposition and iterative refinement algorithms*, in Parallel Algorithms for PDEs, Proceedings of the 6th GAMM-Seminar held in Kiel, Germany, January 19–21, 1990, W. Hackbusch, ed., Braunschweig, Wiesbaden, 1990, Vieweg-Verlag.

[8] P. E. Bjørstad and M. Skogen, *Domain decomposition algorithms of Schwarz type, designed for massively parallel computers*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Philadelphia, 1992, SIAM.

[9] T. Blank, ed., *The MasPar MP-1 architecture*, IEEE, February 1990. Proceedings of IEEE Compcon Spring 1990.

[10] W. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[11] X.-C. Cai, *An additive Schwarz algorithm for nonselfadjoint elliptic equations*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1990, SIAM.

[12] ——, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.

[13] X.-C. Cai and O. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 243–258.

[14] ——, *Multiplicative Schwarz algorithms for some nonsymmetric and indefinite problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.

[15] M. A. Celia, I. Herrera, and S. Kindered, *A new numerical approach for the advective-diffusive transport equation*, Tech. Report 7, Comunicaciones Tecnicas del Instituto Geofisica, (UNAM), Mexico, 1987.

[16] G. Chavent, G. Cohen, and J. Jaffra, *Discontinuous upwinding and mixed finite elements for two-phase flows in reservoir simulation*, Comp. Meth. in Appl. Mech and Eng., 47 (1984), pp. 93–118.

[17] P. Christy, ed., *Software to support massive parallel computing on the MasPar MP -1*, 1990. Proceeding of IEEE Compcon Spring 1990, IEEE, February 1990.

[18] H. K. Dahle, *Adaptive Characteristic Operator Splitting Techniques for Convection-Dominated Diffusion in One and Two Space Dimension*, PhD thesis, Dep. of Appl. Math., University of Bergen, Norway, 1988.

[19] L. Demkowicz and J. T. Oden, *An adaptive characteristic Petrov-Galerkin finite element method for convection-dominated linear and non-linear parabolic problems in one space variable*, Comp. Meth. in Appl. Mech. and Eng., 55

(1986).

[20] J. Douglas and T. F. Russel, *Numerical methods for convection-dominated diffusion problems based on combining method of characteristic with finite elements or finite difference procedures*, SIAM J, 19 (1982), pp. 871–885.

[21] M. Dryja, *An additive Schwarz algorithm for two- and three-dimensional finite element elliptic problems*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1989, SIAM.

[22] M. Dryja, M. Sarkis, and O. B. Widlund, *Multilevel Schwarz methods for elliptic problems with discontinuous coefficients in three dimensions*, Tech. Report 662, Courant Institute, New York University, March 1994.

[23] M. Dryja and O. B. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.

[24] ———, *Additive Schwarz methods for elliptic finite element problems in three dimensions*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Philadelphia, 1992, SIAM.

[25] ———, *Domain decomposition algorithms with small overlap*, Tech. Report 606, Department of Computer Science, Courant Institute, May 1992. To appear in SIAM J. Sci. Stat. Comput.

[26] ———, *Some recent results on Schwarz type domain decomposition algorithms*, Contemporary Mathematics, 157 (1994), pp. 53–61. In proceedings from the sixth international symposium on domain decomposition.

[27] M. S. Espedal and R. E. Ewing, *Characteristic Petrov-Galerkin subdomain method for two-phase immiscible flow*, Comp. Meth. in Appl. Mech. and Eng., 64 (1987).

[28] M. S. Espedal, R. E. Ewing, and T. F. Russel, eds., *Mixed methods, operator splitting, and local refinement techniques for simulation on irregular grids*, September 1990. Proceeding of 2nd European Conference on the Mathamatics in Oil Recovery 11-14, Arles.

[29] R. E. Ewing, T. F. Russel, and M. F. Weeler, *Convergence analysis of an approximation of miscible displacement in porous media by mixed finite elements and a modified method of characteristic*, Comput. Meth. Appl. Mech Engrg., 47 (1984).

[30] G. H. Golub and C. F. V. Loan, *Matrix Computations*, John Hopkins Univ. Press, 1989. Second Edition.

[31] W. Hackbusch, *Multigrid Methods and Applications*, Springer, Berlin, 1985.

[32] K. Hess, J. P. Leburton, and U. Ravaioli, *Computational electronics: Semiconductor transport and device simulation*, Kluwer, 1991.

[33] C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Cambridge University Press, 1987.

[34] T. Kårstad, *Massively parallel algorithms in reservoir simulation*, PhD thesis, Department of Informatics, University of Bergen, September 1993.

[35] K. W. Morton, *Finite Element Methods for None-Self-Adjoint Problems*, Springer-Verlag, 1982. Lecture Notes in Mathematics 965.

[36] J. Nickolls, ed., *The design of the MasPar MP-1, a cost effective massive parallel computer*, IEEE, 1990. Proceedings of IEEE Compcon Spring 1990.

[37] T. F. Russel, ed., *Finite elements with characteristics for two-component incompressible miscible displacement*, 1982. Proceedings from the sixth SPE

symposium on reservoir simulation SPE 10500, New Orleans.

[38] O. Sævareid, *On local Grid Refinement Techniques for Reservoir Flow Problems*, PhD thesis, Dep. of Appl. Math., University of Bergen, Norway, 1990.

[39] H. A. Schwarz, *Gesammelte Mathematische Abhandlungen*, vol. 2, Springer, Berlin, 1890, pp. 133–143. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, volume 15, 1870, pp. 272–286.

[40] M. D. Skogen, *Parallel Schwarz methods*, PhD thesis, University of Bergen, Dept. of Informatics, Høyteknologisenteret, N-5020 Bergen, Norway, 1992.

[41] H. A. van der Vorst, *Bi-CGSTAB: A fast and smootly converging variant of Bi-CG for solution of non-symmetric limear systems*, SIAM J. Sci. Statist. Comput., 13 (1992).