Domain Decomposition Techniques in Parallelization of the 3-dimensional FRONTSIM code.

Petter Bjørstad *, Randi Moe *, Rudi Olufsen †, Eero Vainikko *

March 24, 1995

Abstract

We have studied the parallelization methods for the pressure equation in the reservoir simulator FRONTSIM‡ for 3-dimensional reservoirs. In order to obtain a faster simulator and finer resolution, we have studied the use of domain decomposition methods combined with the use of parallel processing. Both the additive Schwarz method and the multiplicative Schwarz method are implemented together with coarse-grid solver techniques. The purpose of the investigation is to find the best strategies for different problem sizes and different approximate solvers. The code is parallelized using the PVM message passing library for communication and has been tested on a cluster of workstations. This effort is part of the EC-ESPRIT III / EUROPORT2 project. Some implementation results and directions for further developments are given.

1 Introduction.

FRONTSIM, developed by Technical Software Consultants A/S (TSC), is a software package for petroleum reservoir simulation. The simulator FRONTSIM [3, 17] handles compressible or incompressible, two-phase, black oil problems for two- or three-dimensional heterogeneous reservoirs. In order to obtain a faster simulator and finer resolution, we have studied the use of domain decomposition methods combined with the use of parallel processing for solving the pressure equation.

The problem is discretized using the finite element method. An important part of the simulator is the solution of the pressure equation of incompressible flow which reads

$$\frac{\partial}{\partial x}(\alpha(\lambda_n + \lambda_w)\frac{\partial P}{\partial x}) = \frac{\partial}{\partial x}((\lambda_n \rho_n + \lambda_w \rho_w) g \frac{\partial D}{\partial x}).$$

Here we solve for the pressure P given a cross-section α, the gravitational acceleration g, the vertical distance D in the reservoir, the wetting and nonwetting phase mobilities λ_w and λ_n, and the corresponding phase densities ρ_w and ρ_n. Discretization of the equation gives a symmetric and positive definite matrix so that e.g. the method of conjugate gradients (CG) can be used to solve the resulting system of linear equations. In order to obtain faster convergence, especially in the case of larger problem sizes, the general strategy is to use an effective preconditioner. The pressure equation is assembled and solved for every timestep of the simulation and this computation typically consumes 50 – 90 % of the total sequential cpu time. Therefore, it is highly preferable to try to perform this computation in parallel.

*Para//ab, Institutt for Informatikk, University of Bergen, Høyteknologisenteret N-5020 Bergen, Norway.

†Technical Software Consultants as Gaustadalléen 21, N-0371 Oslo, Norway

‡FRONTSIM is developed by TSC A/S

The sequential version of FRONTSIM is using incomplete Cholesky factorization with some fill-in as preconditioner in the incompressible case. Domain decomposition is a preconditioner technique that facilitates a parallel solution process and at the same time reduces the number of CG-iterations significantly.

2 Domain Decomposition.

Domain decomposition techniques have received much interest in recent years, *cf. eg.*, Glowinski et al. [10], Chan et al. [4, 5, 6], and Keyes et al. [12]. This is to a large extent due to their potential on multiprocessor computing systems. These methods are based on a subdivision of the computational domain into subdomains of reduced size and sometimes simpler geometrical shape. A preconditioner (an approximation to the original problem) can then be derived by considering the problem restricted to subdomains plus an appropriate coupling mechanism between the new (and smaller) subproblems. An application of the preconditioner therefore involves the solution of a family of subproblems of lower discrete dimension. The resulting methods should almost always be accelerated by some Krylov space method, *ie.,* the Conjugate Gradient method or some appropriate nonsymmetric variant like GMRES or BICGSTAB. If the complexity of the subdomain solver grows faster than linearly in the number of unknowns, then the overall complexity is reduced as we "divide and conquer" the domain. Thus an improved performance can be obtained also in a sequential setting. Naturally, the potential gain is larger when the individual subdomains can be processed in parallel.

The general technique of domain splitting is not new. H.A. Schwarz introduced such a technique in a theoretical context more than hundred years ago [13]. Within computational methods for structures based on finite elements, similar techniques have been used in order to organize block elimination procedures and take advantage of identical substructures. Dryja and Widlund [7] have developed a common mathematical framework for analyzing these methods. A description of domain decomposition algorithms suitable for practical implementation is found in the book by Smith, Bjørstad and Gropp [16].

The domain decomposition methods can be divided into two main categories, methods having overlap between subdomains and non-overlapping variants. The two categories are often referred to as Schwarz type methods and iterative substructuring methods, respectively. In a Schwarz type method the subdomains exchange information through the overlapping nodes, while in a non-overlapping method one usually introduces what is often called an "interface solve" in order to precondition the Schur complement operator that represents the unknowns restricted to an interface (between two subdomains). We will exclusively consider overlapping methods in this paper.

2.1 Schwarz algorithms

The original Schwarz alternating method is an iterative process for the solution of boundary value problems. It consists of solving successively, a similar problem in two subdomains, alternating from one subdomain to the other. In the simplest two domain case as shown in Figures 1 and 2, the domain Ω is divided into two overlapping subdomains $\Omega^{(1)}$ and $\Omega^{(2)}$.

In order to describe the approach using simple linear algebra notation, let us denote the complete discrete linear system to be solved

$$Ax = b.$$

Now consider a basic first order stationary iterative method for the solution of a system

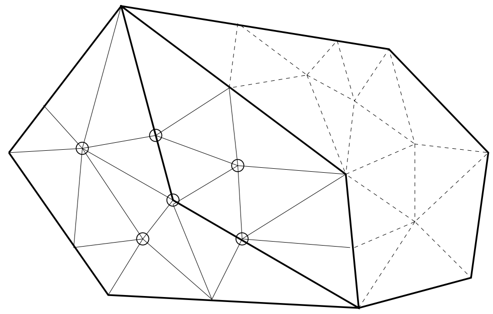

Figure 1: Nodes associated with $\Omega^{(1)}$

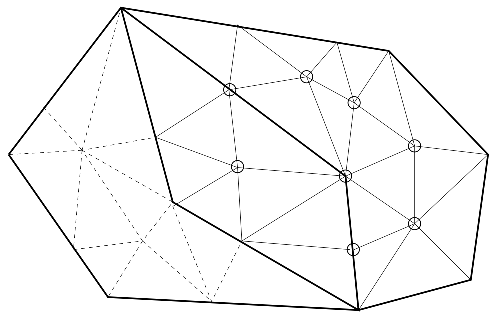

Figure 2: Nodes associated with $\Omega^{(2)}$

of linear equations

$$x^{n+1} \leftarrow x^n + B(b - Ax^n).$$

Here the matrix B is called the preconditioner and it should capture the essential structure of A^{-1} in the sense that the condition number of BA should be considerable less than the condition number of A (see e.g. Golub and Van Loan [11]).

The pressure equation that must be solved in FRONTSIM, has a positive definite and symmetric matrix A. This implies that we can use the conjugate gradient method to accelerate the basic iteration as long as we construct a positive definite and symmetric matrix B.

Algorithm 1 describes the CG method for solving $Ax = b$. We observe that one only needs to supply routines for the computation of the matrix-vector products $q = Ar$ and $s = Bq$ for given vectors r and q. In order to construct an appropriate B, we define the restriction matrix R_i as the matrix that applied to an entire discrete vector x returns the components $x_i = R_i x$ that can be identified with subdomain number i. In the following, we will use subscripts when referring to subdomains while using superscript when referring to iteration parameters. We then define components from which to build our matrix B by

$$B_i = R_i^T (R_i A R_i^T)^{-1} R_i.$$

Notice that the product $B_i x$ will involve a solution of a (smaller) problem restricted to subdomain i, expressed by the term $(R_i A R_i^T)^{-1}$.

In the multiplicative Schwarz method the preconditioner B takes the form

$$B_1 + B_2 - B_2 A B_1$$

in the case of only two subdomains. The calculation of Bx corresponds to one complete sweep of the classical alternating method of Schwarz, starting with an initial guess zero. The method has the same sequential behavior as a Gauss-Seidel iteration.

Algorithm 1 (The conjugate gradient method) .
begin

$x_0 \leftarrow 0$, Compute $b^0 = Bb$, $r^0 \leftarrow b^0$

for $k = 1, \ldots, k_{max}$

if *(k = 1)* **then**

$\beta^1 \leftarrow 0$, $p^1 \leftarrow r^0$
$q^1 \leftarrow Ar^0$

else

$\beta^k \leftarrow \langle Ar^{k-1}, r^{k-1}\rangle / \langle Ar^{k-2}, r^{k-2}\rangle$
$p^k \leftarrow r^{k-1} + \beta^k p^{k-1}$
$q^k \leftarrow Ar^{k-1} + \beta^k q^{k-1}$

endif

Compute $s^k = Bq^k$
$\alpha^k \leftarrow \langle Ar^{k-1}, r^{k-1}\rangle / \langle q^k, s^k\rangle$
$x^k \leftarrow x^{k-1} + \alpha^k p^k$
$r^k \leftarrow r^{k-1} - \alpha^k s^k$

endfor

end

Because of the sequential nature of the classical multiplicative method, it is not ideal for parallel computations. However, there is still some potential for parallel computations as illustrated in Figure 3, where we have divided a square into boxes. The coloring of each box refers to independent subdomains, *ie.,* boxes with the same color (number) can be solved simultaneously. In this case the solution process needs four sequential steps and in each step four subdomains can be solved in parallel. In order to accelerate the multiplicative method by the CG method, extra steps have to be taken to maintain symmetry. In such a case the solution process consists of seven sequential steps, *ie.,* the sequence 1–2–3–4–3–2–1 of colors.

We can change our preconditioner B by eliminating the terms that forces sequential steps. The additive Schwarz method uses

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

where N is the number of subdomains. This change is completely analogous to the relationship between Gauss-Seidel and Jacobi iterations. In the former case, we solve a block system with overlapping blocks, carefully using the most recently updated information in each step. In the latter case, we solve all the block systems (corresponding to all the subdomains) simultaneously before updating with new values. However, the number of iterations needed for convergence is known to be about twice as many as for the multiplicative method (see Bjørstad [1], Skogen [14], and Widlund [18]). Multiplicative and additive Schwarz methods are optimal in the sense that the condition number is independent of the discretization parameter, h, as long as the overlap and the number of subdomains are kept constant. Another very important property that can be shown, is that the rate of convergence does not depend on jumps (or large variations) in the coefficients of the partial differential equation that occur across subdomain boundaries (see Dryja and Widlund [8] and Smith [15]). This is of particular relevance in the solution of pressure equations related to oil reservoir simulations.

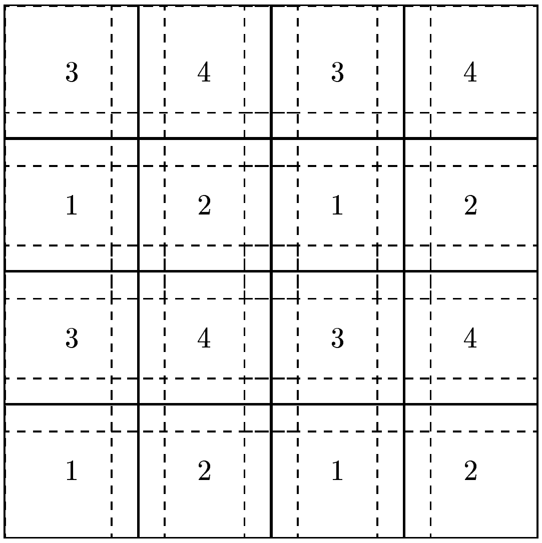

Figure 3: Domain decomposition for a 4-color multiplicative Schwarz method.

When a large number of subdomains is used, some sort of global coupling must be introduced, otherwise the convergence rate will deteriorate rapidly with the number of subdomains. This effect is due to the fact that in each iteration the interaction between different subdomains is only through nearest neighbors. A significant residual at one end of the region may influence the error everywhere, and therefore the number of steps until convergence of any iterative method will depend on the number of subdomains. For example, in the case of a square divided into N boxes, the number of iterations before convergence must be at least of order $N^{1/2}$, the diameter of the domain, as long as no mechanism for global exchange of information is available.

One can alleviate this problem by introducing a coarser discretization of the domain, typically using the diameter H of the subdomains as discretization parameter.

By defining a restriction matrix R_0 by the requirement that its transpose R_0^T should linearly interpolate from the coarse grid back to the set of fine grid points, we can add a term $B_0 = R_0^T(R_0AR_0^T)^{-1}R_0$, to our preconditioner and restore favorable convergence properties that now do not depend on the number of subdomains. This approach can be proved optimal in the sense that the condition number has an upper bound independent of the discretization parameter and of the number of subdomains as long as the overlap is constant. One should note that this approach requires broadcasting of information to all processors at every iteration. In a parallel computing context we therefore need to balance the two requirements of minimizing the amount of communication while still maintaining satisfactory convergence properties. Successful implementations on massively parallel systems have been carried out by Skogen [14].

Two-level methods can be combined with both additive and multiplicative methods. Assuming that B_A and B_M denote the one-level additive and the multiplicative method, respectively, we can obtain different two-level methods like the two-level additive method

$$B = B_0 + B_A,$$

the two-level multiplicative method

$$B = B_0 + B_M - B_0AB_M,$$

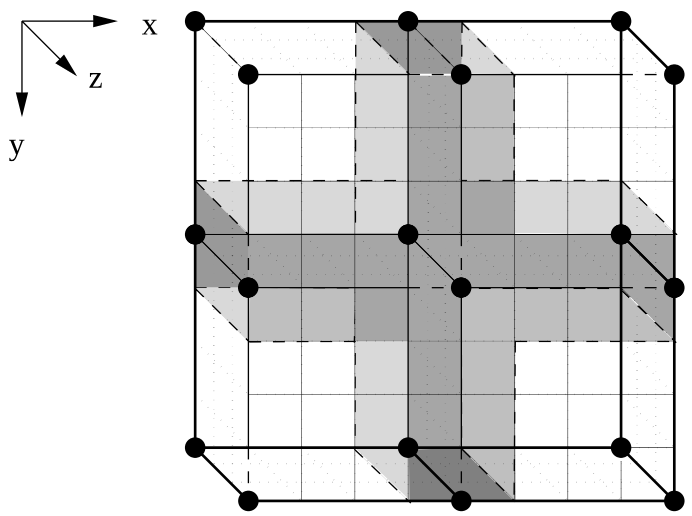

Figure 4: A splitting of the reservoir into 4 subdomains with minimum overlap. The dark bullets denote the coarse grid nodes.

or a hybrid two-level multiplicative method

$$B = B_0 + B_M.$$

Note that in the second case, *ie.,* the two-level multiplicative method, one will need a parallel solver for the coarse grid problem in a parallel implementation as the coarse grid problem and the subdomain problems can not be solved simultaneously.

3 Domain Decomposition in FRONTSIM.

We will in this section look at the details of implementing the domain decomposition methods discussed in Section 2, for solving the 3-dimensional pressure equation in FRONTSIM. In this prototype implementation, the global domain has been divided into n by m subdomains of approximately equal size. The pressure equation in FRONTSIM is defined with Neumann boundary conditions, but the positive definiteness is maintained by prescribing the pressure in at least one point. Note that the interior boundaries of the subdomains will have a zero Dirichlet boundary condition in our Schwarz algorithms. The reservoir has vertical production and injection wells. We assume that the reservoir is split into subdomains in such a way that wells do not pass through subdomain boundaries. Thus, with vertical wells we end up with subdomains with possible neighbors only in the xy plane. Usually, the number of grid blocks needed to model a reservoir is much smaller in the z-direction. The overlap between neighbor subdomains is at least one extra grid block, *ie.,* two neighboring subdomains have at least one common gridplane. Figure 4 shows an example domain divided into four subdomains with this overlap. A slightly larger overlap of two grid blocks has been used in the reported computational results.

Before each invocation of the pressure equation solver, the submatrices A_i of A is assembled element by element in each subdomain. This assembling is a small cost compared to the solution process. We will use the notation A_i for the restriction of A to the subdomain Ω_i.

All elements of the subdomain matrices A_i are present in A. The coarse grid nodes are chosen as shown in Figure 4, *ie.,* as the crosspoints between subdomains. Hence, if the number of subdomains is n in the x-direction and m in the y-direction then the number of coarse grid nodes in the directions x and y is always $n+1$ and $m+1$, respectively. In our assumptions there are always only 2 coarse grid nodes in the z-direction. In addition to the coarse grid node locations we must also construct the restriction operator R_0 defined in Section 2. With our particular choice of coarse grid nodes there are exactly 8 coarse grid nodes per subdomain.

In order to make use of the coarse grid techniques we also need the corresponding coarse matrix A_0. One can either find it explicitly through analogous assembling of the matrix A itself or just calculate it from the formula $A_0 = R_0^T A R_0$. The former method has the possible advantage that the coarse matrix has exactly the same structure as the matrix A itself. In the latter case the matrix A_0 looses some of its sparsity, but it can be generated in an automatic way when the assembling procedure is unavailable for the coarse grid. We use the latter method also because we avoid the upscaling of the geological data. Note that building A_0 requires some communication if the matrix is stored locally on different processors. For each subdomain Ω_i the local part $A_{0,i}$ of the coarse-grid matrix A_0 is calculated. Then all the submatrices $A_{0,i}$ are collected to form the matrix A_0 itself.

3.1 Data Structure.

In the parallel version of FRONTSIM the matrices A_i, the right hand side vectors b_i, and all the other vectors needed in the solution process are located locally with each subdomain. This means that, in principle, it is possible to locate each subdomain to a separate processor. As the matrix A is sparse and symmetric, only the lower triangular part of it is stored. For each subdomain one data array is needed for storing the nonzero elements together with standard additional integer pointers to the location of the nonzeroes. This is a common technique in sparse matrix computations. Note that the data related to an overlapping area is stored with each subdomain it belongs to. This is done primarily to avoid unnecessary communication in a parallel implementation.

3.2 The conjugate gradient iteration.

The conjugate gradient method consists of four major components, *ie.,* matrix-vector multiplication, solution of a linear system, inner products, and vector updates. We will in the following discuss details of these computations in the context of domain decomposition.

- Matrix-vector multiplication (Ar^k in Algorithm 1).
  When we want to compute the global matrix-vector product Ar^k, we compute $A_i r_i^k$, where $A_i = R_i A R_i^T$, on all subdomains Ω_i. This approach produces a correct result for all vector elements except the elements corresponding to the outer layer along the interfaces. However, the correct values for these elements are computed by the neighbor subdomains. We have chosen to compute the product $A_i r_i^k$, and correct the interface elements afterwards by transferring the correct values from the neighboring subdomains. Alternatively, we could have extended the matrix A_i and the vector r_i^k with the missing values. However, there would still be a need for communication of these extra values, - in this case before the matrix-vector multiplication.

- Solution of a linear system (compute $s^k = Bq^k$ in Algorithm 1).
  In the case of the additive Schwarz method, the individual subdomains can be solved independently. The solution process consists of a loop over all sub-domains in any order. Afterwards, the subdomain solutions are added in order to obtain the global solution. In practice, this involves adding the values corresponding to the overlapping regions. Again, we need to transfer values between the subdomains.

- Inner products ($\langle \cdot, \cdot \rangle$ in Algorithm 1).
  The global inner product is built from adding local inner products on each subdomain. However, one has to make sure that the contribution from the overlapping areas is only counted once. In the case of parallel processing, the contribution from each process is finally added up and broadcasted to all processors.

- Vector updates (e.g. $x^k \leftarrow x^{k-1} + \alpha^k p^k$ in Algorithm 1).
  The vector updates can be carried out without any communication assuming that the parts of the vectors corresponding to the overlapping areas are duplicated on every subdomain.

The added overhead before and after the conjugate gradient iteration when introducing domain decomposition is summarized in Algorithm 2.

Algorithm 2 (The pressure solver using the Schwarz method) .

begin
  if first visit **then**
    initiate the domain decomposition data structure
  endif
  for each subdomain Ω_i
    assemble the matrix A_i and the vector b_i
  endfor
  apply the conjugate gradient method given in Algorithm 1
  for each subdomain Ω_i
    update the global pressure vector x^k based on x_i^k
  endfor
end

3.3 Inexact subdomain- and coarse-grid solvers

In order to achieve an efficient implementation one should use an approximate solution technique to form the preconditioners B_i. At the same time the exact solver is used to verify the general properties of the domain decomposition method. We have implemented both an exact solver, using the Cholesky factorization, and approximate solvers. The approximate solvers currently implemented are an incomplete Cholesky factorization without fill-in and the symmetric Gauss-Seidel iteration (SGS), see e.g. Golub and Van Loan [11]. In general, using approximate solvers in the preconditioning step increases the number of CG-iterations but when considering the total running time and memory consumption, it is usually much more efficient than exact solvers. Still, both the SGS and the incomplete Cholesky give good results when the subdomain sizes (and the coarse grid mesh size) are not too large. With larger subdomain sizes multilevel solvers should be used in order to control the overall computational complexity.

4 Parallel implementation.

In this section we present the parallel algorithm for solution of the pressure equation with the two-level additive Schwarz method. With this algorithm incorporated into the solution process, the code is well prepared for a parallel implementation. The extra cost of a parallel implementation is due to communication and synchronization.

The implementation is based on a master-slave model. The master program executes all but the pressure equation solver sequentially. The first time the pressure equation is solved, the master process spawns the requested number of new processes and sends information about the problem sizes to the new processes. Algorithm 3 shows the part of the master program for solving the pressure equation.

Algorithm 3 (The parallel additive Schwarz method - master program) .

begin
  if first visit **then**
    if (more than one processor) **then**
      spawn new tasks
      send time independent data
    endif
    generate the domain decomposition data structure
  endif
  send time dependent data
  for each local subdomain Ω_i
    assemble the matrix A_i and the vector b_i
    generate the matrix $A_{0,i}$
  endfor
  receive $A_{0,i}$ from the other tasks for all non-local subdomains Ω_i
  assemble the matrix A_0
  apply the conjugate gradient method given in Algorithm 1
  for each local subdomain Ω_i
    update the global pressure vector x based on x_i^k
  endfor
  if (more than one processor) **then**
    receive x_i's from the other tasks
    update the global pressure vector x
  endif
end

After the completion of the conjugate gradient iteration, the master process receives the parts of the solution vectors computed by the other processes. The new processes execute the program shown in Algorithm 4.

Algorithm 4 (The parallel additive Schwarz method - slave program) .

begin
  receive time independent data from the master process
  generate the domain decomposition data structure
  for each solve of the pressure equation
    receive time dependent data from the master process
    for each local subdomain Ω_i
      assemble the matrix A_i and the vector b_i
    endfor
    apply the conjugate gradient method given in Algorithm 1
    for each local subdomain Ω_i
      send the vector x_i^k to the master process
    endfor
  endfor
end

64x64x2 grid blocks 8x8 subdomains Subdomain solver	Preconditioning method			
	One-level		Two-level	
	Additive	Multiplicative	Additive	Hybrid multiplicative
Inc. Cholesky	135	67	43	27
SGS	107	55	51	35
Cholesky	26	20	18	13

Table 1: Iterations with a fixed number of grid blocks (64x64x2) in the reservoir and a fixed number of subdomains (8x8).

All processes take an equal part in the computations with the master process as the coordinating process which also handles the interface to the rest of the simulator. The slave processes are hence idle during the saturation solver. Note that in the current implementation, the master process is responsible for the coarse grid solver.

5 Test Results.

We present some preliminary test results from a parallel FRONTSIM based on domain decomposition methods for solving the pressure equation. It should be mentioned that the code is still under rapid development, with the aim of obtaining higher efficiency and reduced memory consumption. Development of more efficient subdomain solvers is also currently in process. The tests have been run on up to four SunSparc 10 workstations, each with 2 processors, connected by an FDDI network. Hence we had up to 8 processors available. The message passing library PVM (Parallel Virtual Machine) [9] was used for interprocessor communication. We are investigating the best strategies for different problem sizes and local solvers.

We have tested 4 different types of domain decomposition methods. The first two methods presented are the one-level multiplicative and additive Schwarz methods, described in Section 2.1. In addition, two-level methods for both the additive and the multiplicative Schwarz methods have been implemented. Here we consider a hybrid two-level multiplicative Schwarz method where we solve the coarse grid problem in parallel with the subdomain problems. The method is multiplicative on the fine grid level and additive with respect to the coarse grid update. See also [2], where similar observations were made.

We consider the 2-phase (water and oil) test case of the reservoir with a homogeneous geological structure. The dimensions of the reservoir are 1000m in the xy plane and 50m in the z-direction. Two injection and two production wells are present, one in each corner grid block of the reservoir. The termination criterion used for all the reported numerical experiments requires the initial residual norm to be reduced by five orders of magnitude.

5.1 Convergence rate

In Table 1 we list the CG iteration counts for the four different domain decomposition techniques mentioned combined with different subdomain solvers. The reservoir dimensions in this test case are 64x64x2 grid blocks with 8x8 subdomains, i.e 8x8x2 grid blocks in each subdomain (excluding the overlap). The iteration count refer to the first timestep of the simulation.

In this case the incomplete Cholesky factorization and the SGS method give fairly similar iteration counts. In all reported results we used 2 complete sweeps of SGS. It should be mentioned that we would have expected a bit greater difference between the number of iterations in one-level multiplicative and additive Schwarz methods when the direct solver

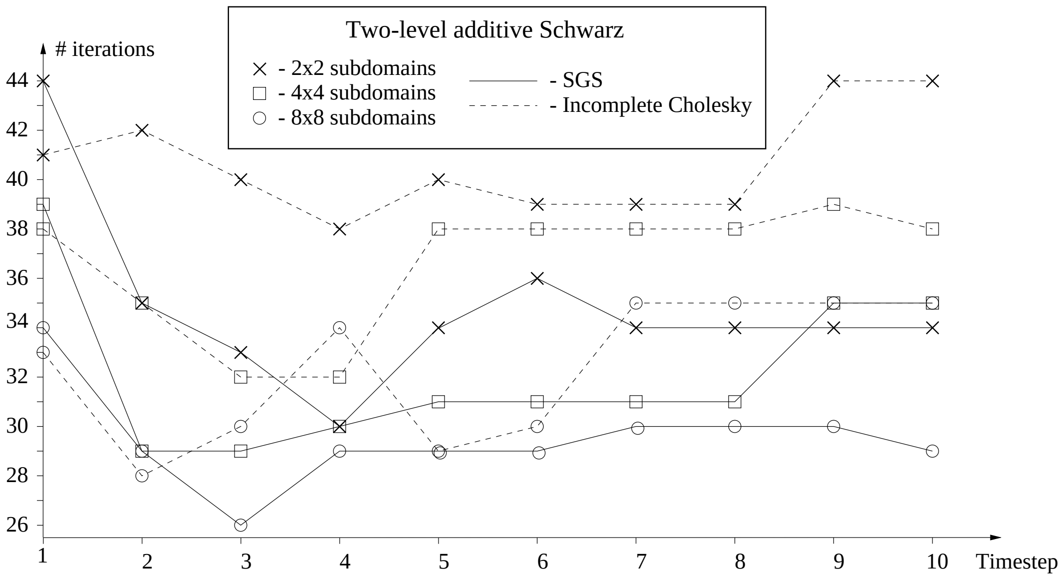

Figure 5: Variations of the iteration counts in 10 timesteps with the fixed size of reservoir (32x32x4 grid blocks) and different number of subdomains.

(Cholesky factorization) was used. The two-level additive Schwarz method gives the most effective results considering time. The multiplicative Schwarz method is better in the iteration count but does at the same time use much more cpu time per iteration. This is due to more computational work needed in each iteration caused by the extra steps required to maintain symmetry as was discussed in Section 2.1. The number of sequential steps can be reduced from 7 to 4 if some Krylov accelerator that does not need the symmetry in the preconditioned matrix, is used.

In Figure 5, iteration counts are given for different numbers of subdomains using the two approximate solvers in the case of a reservoir with a fixed size 32x32x4. The method in use is two-level additive Schwarz. The iteration count is not constant throughout the simulation, and the figure gives the iteration count for the first ten time steps. Incomplete Cholesky and SGS method have about the same number of iterations. We observe that when the number of subdomains increases the number of iterations decreases. This is due to a better balance between the size of the subdomain problems related to the size of the coarse grid problem.

In Table 2 we give iteration counts for the additive Schwarz methods with incomplete Cholesky and SGS as subdomain and coarse-grid solver for a fixed 6x6x4 subdomain size. The number of subdomains is varied and hence, also the dimensions of the reservoir itself. When using the one-level method, the number of iterations increases with the number of subdomains as expected. At the same time, the number of iterations in the two-level method does not increase so quickly, but it does not remain constant as in the case of a direct coarse grid solver. This is due to the coarse grid solver getting more inaccurate as we increase the size of the problem. We conclude that a multilevel method is needed for solving the coarse grid problem in order to keep the work low. This approach is similar to the one in [2] although the target computer system is very different.

5.2 Parallel performance

Figure 6 represents timing results for the two-level additive Schwarz method where we used a different number of processors to solve the problem with a fixed size of 64x64x4. The times represent solving the pressure equation (not including assembling) in one timestep. We see

Additive Schwarz method with fixed 6x6x4 blocks in subdomains				
	Number of iterations			
	Incomplete Cholesky		SGS	
#Subdomains	1-level	2-level	1-level	2-level
2x2	52	31	50	36
4x4	60	37	46	34
6x6	87	36	67	37
8x8	105	39	85	36
10x10	126	40	101	37
12x12	145	44	118	37
14x14	143	47	115	39
16x16	154	49	132	40

Table 2: Number of iterations in the case of a fixed subdomain size (6x6x4) for various number of subdomains and reservoir sizes.

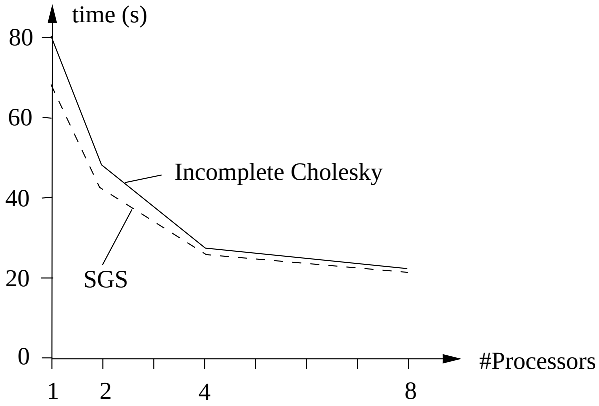

Figure 6: Time in seconds for solution of the pressure equation with 2-level additive Schwarz method for the reservoir of the size 64x64x4 and 8x8 subdomains.

that the parallel efficiency decreases going from 4 to 8 processors as the curves become more horizontal. This can be explained partly by the fact that current load balancing scheme is far from optimal. The processor solving the coarse-grid problem has the same number of subdomains to solve as the other processors. This imbalance becomes more important as the number of subdomains per processor is reduced. At the same time the relative importance of communication to computation increases with the number of processors. Hence, the parallel efficiency will decrease. One should expect that this loss of efficiency will appear later when the problem size is increased. The distribution of subdomains among the processors is done by a rather primitive scheme in the current version and will be improved in order to reduce the amount of communication. In future implementations we expect to see better efficiency when the number of processors is increased.

6 Conclusion

We have studied domain decomposition methods for solving the pressure equation in FRONSIM. We studied altogether four different kinds of combinations of the Schwarz methods — one- and two-level techniques combined with additive and multiplicative Schwarz methods. Both exact and approximate solvers were implemented for solving the subproblems as well as the coarse grid problem. Using coarse grid techniques has shown to be effective in FRONTSIM for improving the convergence rate as well as to reduce the runtime. The lowest number of iterations is achieved by the two-level multiplicative Schwarz while the most effective method in cpu time is the two-level additive Schwarz method.

Further efforts should be made for optimizing the code in order to achieve a sequential performance close to or better than the original code for larger problems. Comparisons of the performance of the sequential and parallel versions will be given in later reports. One of the goals of this project is to be able to solve problems with up to one million grid blocks efficiently on standard workstations. However, the original sequential version of FRONTSIM is not capable of solving problems of this size, and hence comparisons can not be given.

The parallel implementation will also be improved through better load balancing algorithms. Better subdomain solvers will be targeted to different subproblem sizes. This means *eg.,* implementation of a multigrid solver. A promising approach for solving large problems, is to divide the reservoir into a large number of relatively small subdomains. Then one can use the approximate solvers that were presented here for subdomain solvers and the large coarse grid problem can be solved with an effective multilevel method.

References

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

[2] P. E. Bjørstad and M. D. Skogen, *Domain decomposition algorithms of Schwarz type, designed for massively parallel computers*, in Domain Decomposition Methods for Partial Differential Equations, D. E. Keyes, T. F. Chan, G. Meurant, J. S. Scroggs, and R. G. Voigt, eds., SIAM, 1992, pp. 362–375. In the proceedings from the Fifth International Symposium on Domain Decomposition Methods, Norfolk, Virginia 1991.

[3] F. Bratvedt, K. Bradtvedt, C. F. Buchholz, T. Gimse, H. Holden, L. Holden, and N. H. Risebro, *FRONTLINE and FRONTSIM; Two Full scale, Two-phase, Black Oil Reservoir Simulators Based on Front Tracking*, Surv. Math. Ind., (1993), pp. 185–215.

[4] T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., *Domain Decomposition Methods for Partial Differential Equations II*, Philadelphia, 1989, SIAM. Proceedings of the Second International Symposium on Domain Decomposition Methods , Los Angeles, California , January 14 - 16, 1988.

[5] ——, eds., *Domain Decomposition Methods for Partial Differential Equations III*, Philadelphia, 1990, SIAM. Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Houston, Texas, March, 1989.

[6] ——, eds., *Domain Decomposition Methods for Partial Differential Equations IV*, Philadelphia, 1991, SIAM. Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, Moscow, May, 1990.

[7] M. Dryja and O. B. Widlund, *Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems*, in Domain Decomposition Methods for Partial Differential Equations III, T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1990, SIAM. Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Houston, Texas, March, 1989.

[8] ——, *Additive Schwarz methods for elliptic finite element problems in three dimensions*, Tech. Rep. 570, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, N.Y. 10012, June 1991.

[9] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM 3 User's Guide and Reference Manual*, Tech. Rep. ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, May 1993.

[10] R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., *Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, 1988, SIAM. Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Paris, France, January 1987.

[11] G. H. Golub and C. F.Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 1989. Second Edition.

[12] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., *Domain Decomposition Methods for Partial Differential Equations V*, Philadelphia, 1992, SIAM. Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, Norfolk, Virginia, May, 1991.

[13] 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.

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

[15] B. F. Smith, *An iterative substructuring algorithm for problems in three dimensions*, tech. rep., Argonne National Laboratory, 9700 South Cass Ave, Argonne, IL 60439-4844.

[16] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, to be published by Cambridge University Press, Cambridge, 1995.

[17] Technical Software Consultants A/S, Oslo, Norway, *FRONTSIM - A Front Tracking Reservoir Simulator*, 1992.

[18] O. B. Widlund, *On the rate of convergence of the classical Schwarz alternating method in the case of more than two subregions*, tech. rep., Department of Computer Science, Courant Institute, 1989. A revised version in preparation.