Robust Additive Schwarz Methods – Parallel Implementations and Applications

Thesis for the Degree of Doctor Scientiarum

Eero Vainikko

Department of Informatics
University of Bergen

March 1997

Acknowledgments

This thesis presents my work for the degree *doctor scientiarum* at the University of Bergen. It was carried out during my employment at the Department of Informatics, University of Bergen in the period from 1993 to 1996. This work was supported by The Royal Norwegian Council for Scientific and Industrial Research (NTNF) under grant STP.27625 and *para//ab*.

I would like to thank my advisor Petter E. Bjørstad for his guidance and support during my work on this thesis. His ideas, neverending enthusiasm and discussions with him have helped me to complete this work.

I would also like to thank Randi Moe, Maksymilian Dryja and Rudi Olufsen for their support and interesting time together during the resarch. I would also like to thank all my colleagues and friends at *para//ab* and at the Department of Informatics for all encouragement, discussions, support and help that made the work more enjoyable for me.

Finally, I would like to thank my family for help and understanding during my struggle with this work in Bergen.

Tartu, March 1997
Eero Vainikko

Abstract

Domain decomposition methods have become quite attractive within the applied computational sciences and in the scientific computing community. In this thesis we will focus on parallel implementation of domain decomposition methods and application of the methods to different type of problems. We are dealing with the Additive Average method with new coarse spaces. The method has many good properties: it is independent of coefficient jumps across subdomain boundaries, it can handle unstructured grids in two and three dimensions, it has a low computational complexity and good parallel properties. We compare the Additive Average method with other Schwarz methods with and without the subdomain overlap. Examples of application of the methods come from simulations in the oil industry, environmental research and technical design. We carried out numerous computer experiments on different parallel architectures and tested various solution strategies.

Contents

Chapter 1

Introduction

1.1 Overview

This thesis is based on four published papers and one submitted paper referred to as Papers A – E and included in the appendix.

Paper A **Domain Decomposition Techniques in Parallelization of the 3-dimensional FRONTSIM code.**
The paper was published in Parallel Programming and Applications: Proceedings of the Workshop on Parallel Programming and Applications: Proceedings of the Workshop on Parallel Programming and Transputer Conference (NTUG '95), P. Fritzon and L. Finmo eds., Linkőping University, Department of Computer and Information Science, IOS Press, May 1995, pp. 97 - 110.
Basic domain decomposition methods with overlapping subdomains are used for solution of the pressure equation in the parallel version of the FRONTSIM reservoir simulation code. Additive and multiplicative Schwarz methods are implemented together with different subdomain solver techniques.

Paper B **Additive Schwarz Methods without Subdomain Overlap and with New Coarse Spaces.**
The paper was published in Domain Decomposition Methods in Sciences and Engineering, R. Glowinski, J. Périaux, Z. Shi and O. B. Widlund eds., John Wiley & Sons, 1996. Proceedings from the Eight International Conference on Domain Decomposition Methods, May 1995, Beijing, John Wiley & Sons, February, 1997
Two new methods are introduced: the Additive Average method and the method of Diagonal Scaling. Convergence estimates are

proved and numerical experiments are provided for a sequential implementation of the methods. The methods are compared with the Additive Schwarz method with overlapping subdomains

Paper C **Parallel implementation of a Schwarz Domain Decomposition Algorithm.**
The paper was published in Lecture Notes in Computer Science, Jerzy Wasniewski ed., Springer, 1997, Proceedings from Para96, Aug. 1996, Lyngby.
A parallel implementation of the Additive Schwarz method is described. A Multiplicative Average Schwarz method is introduced. Numerical experiments on parallel computers are carried out.

Paper D **Robust Additive Schwarz Methods on Unstructured Grids**
The paper was submitted to Domain Decomposition in Science and Engineering, P. E. Bjørstad, M. Espedal and D. Keyes, eds. Proceedings from the 9th International Conference on Domain Decomposition Methods, Bergen, June 1996, John Wiley & Sons, 1997.
The Additive Average method is used for solution of elliptic partial differential equations on unstructured meshes. The ways of dividing irregular regions into subdomains are discussed. Numerical experiments with different unstructured grids are carried out. Possibilities for improvement of the coarse bilinear form are discussed.

Paper E **Courant Elements for the Inverse Problem in the Filtration Coefficient Identification.** The paper was published in Lecture Notes in pure and applied Mathematics, Finite Element Methods: fifty years of the Courant method, M. Křížek, P. Neittaanmäki and R. Steinberg eds., Marcel Dekker, 1994, pp. 451 – 459.
The paper deals with the problem formulation and with the discrete solution using the Conjugate Gradient method without preconditioning. In this summary the Additive Schwarz method and the method of Diagonal Scaling are used as preconditioners in the numerical solution of the problem. Results of the computer experiments are presented.

The rest of this summary describes the published work and provides more detail and additional results that could not be included in the papers. In particular, we describe our software development in more detail.

1.2 A model problem

We consider the following problem which we write in variational form; it will be the main model problem in this thesis:

Find $u^* \in V(\Omega)$ such that

$$a(u^*, v) = f(v), \ \forall v \in V, \tag{1.1}$$

where $V = V(\Omega)$ is an appropriate Sobolev space. In most part of this thesis we consider the bilinear form

$$a(u, v) = \sum_{i=1}^{N} \int_{\Omega_i} \rho_i \nabla u \cdot \nabla v dx, \tag{1.2}$$

where $\Omega = \bigcup_{i=1}^{N} \Omega_i$ and the right hand side $f(v)$ is defined by

$$f(v) = \int_{\Omega} f v dx.$$

We assume that ρ_i are positive constants that can have big jumps across the substructure boundaries. We consider also the case with no jumps, the Poisson equation where $\rho_i = 1, \ \forall i$. We assume that there is a Dirichlet boundary condition on $\partial\Omega$.

Consider a finite element space V^h of piecewise linear (or bilinear) continuous functions defined on a triangulation and vanishing on $\partial\Omega$. The discretized problem is of the form:

Find $u \in V^h(\Omega)$ such that

$$a(u, v) = f(v), \ \forall v \in V^h(\Omega). \tag{1.3}$$

The stiffness matrix A and the right hand side vector b are obtained from (1.3) by taking $u, v = \varphi_k$, $k = 1, ..., n$ where $\varphi_k \in V^h(\Omega)$ are basis functions of $V^h(\Omega)$.

1.3 The Problem

Let us consider a general problem: the numerical solution of a partial differential equation that is defined on a domain $\Omega \subset R^d$, $d = 2, 3$. One would like to discretize this problem to find a numerical approximation to the continuous solution of the problem. There are several different approaches for doing this, such as the finite difference method, the finite element method and the finite volume method. In this thesis we will consider only the finite element method (FEM), but many of the ideas that are described on this basis can be used with the other discretization schemes as well. As a result of discretization, one obtains a system of linear equations of the form

$$Ax = b. \tag{1.4}$$

The matrix A is usually sparse, depending on the problem type it can be symmetric or unsymmetric and it is normally very large. For the solution of the sparse linear system a common practice is to use some iterative method. Some of the best iterative methods are Krylov subspace methods such as the Conjugate Gradient (CG) method [18] in the case of a symmetric positive definite matrix A, or CGS [30], BiCGSTAB [33], GMRES [27] and QMR [10] otherwise. Typically, since the condition number of the matrix A increases with the dimension of the system, some preconditioning technique should be used. Throughout this thesis we consider only problems with a symmetric positive definite matrix A. We will therefore use the Preconditioned Conjugate Gradient (PCG) method as our Krylov subspace accelerator.

Today, with the development of parallel computer systems there is a natural need for efficient solver techniques that can take advantage of these architectures. Domain decomposition methods are preconditioning techniques that introduce parallelism into the solution process together with a substantial reduction of the number of Krylov subspace iterations. The original idea of domain decomposition dates back to the work by H. A. Schwarz [28] more than 120 years ago. Using an iterative procedure he proved the existence of solution to the Poisson equation in the case of a domain composed from a square and a disk that overlapped. There are a large number of references related to overlapping Schwarz methods. Here we will only mention a few: P.L.Lions [24, 25] reformulated the classical method in terms of orthogonal projections. M.Dryja and O.B.Widlund studied the domain decomposition methods in the case of very many subdomains [8] and developed a general mathematical framework [9] for analyzing these methods. P.E.Bjørstad and M.D.Skogen [2] implemented the domain decomposition methods on parallel computers. For a more complete list of references, see the annual Proceedings from the Domain Decomposition Conferences [11, 5, 6, 12, 20, 26, 21, 13, 1].

There are many different approaches to derive domain decomposition methods. One possibility is to start with the matrix representation (1.4) – taking into account the sparse banded structure of the matrix. The matrix is decomposed into possibly overlapping subblocks and a preconditioner is built up using the subblock system solvers. This is a universal technique in the sense that there is no need to know anything about the geometry of the underlying problem. But this is also a disadvantage since it may be difficult to find optimal splittings of the original stiffness matrix in more complicated cases.

The second approach is based on the underlying PDE: We split the domain Ω, into N overlapping or non-overlapping subdomains $\Omega_i, i = 1, 2, ..., N$. Denote by Ω_{ih} the nodal points of Ω_i resulting from the discretization. We consider the case of a conformal grid, that is, the gridpoints on the interfaces between the different subdomains coincide. We introduce local solvers on each substructure Ω_{ih} that together serve as a preconditioner for the whole system. In the case of overlapping subdomains, after each subdomain solution, also the corresponding neighbors' values on the interface should be updated. These updates can be performed in a multiplicative or additive manner. In the multiplicative Schwarz methods after the local solution on the subdomain the local residual is computed and the residuals on the overlapping areas of the neighbor subdomains are replaced with the new values. Such a process will impose some sequentialities in the parallel algorithm. Also, when using the PCG method, some additional steps are needed to maintain the symmetry of the preconditioned matrix (see Paper A for details.) In the case of an additive Schwarz method all the values after the local solution on the overlapping areas get added. This gives a fully parallel algorithm but normally results in about twice as many PCG iterations as in the multiplicative case.

Usually, the bigger the overlap, the lower the number of global iterations. But, with increasing overlap, the amount of information to be exchanged between the subdomains and the size of the subproblems are also increased.

In the case of non-overlapping methods one needs an interface solver technique. There are different techniques to do this (see *e.g.* [3]). In Section 3.2 we describe the Average Schwarz methods, introduced in Paper B, the interface solver of which also acts as the coarse problem solver.

The coarse problem techniques, introduced in Paper B, are useful to reduce the total number of iterations. This is especially important in the case of a large number of subdomains as the information about the residual in one end of the domain will require many iterations to travel across the subdomains to the other end. With the classical coarse space, we need a coarse triangulation of Ω with step size H. We introduce a standard FE discretization on the coarse mesh Ω_H and define restriction and interpolation operators between the coarse and the fine FE spaces. During the preconditioning step the residual is restricted to the coarse space. After the solution of the coarse

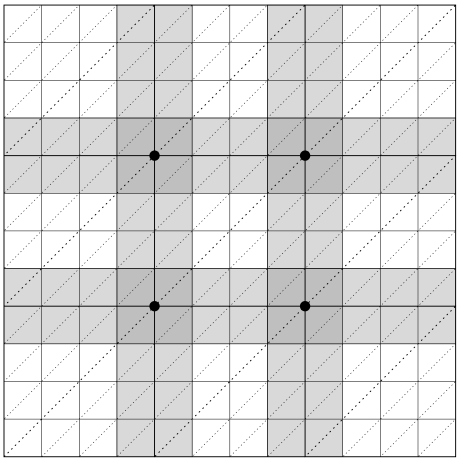

Figure 1.1: A regular domain with triangulation split into 9 subdomains with overlap (shaded areas). and coarse nodes (dark bullets).

problem we interpolate it back to the fine space. See Paper A for more details.

There is a possibility to use multiple levels of coarse spaces in the domain decomposition methods [29, 7]. The resulting multilevel methods are closely related to the multigrid methods (see *e.g.* [4, 23, 34]). In this thesis we consider only methods with two levels.

1.4 Discretization schemes and mesh partitioning

Here we give an overview of our discretization schemes based on the FEM together with the different methods for partitioning the domain into subdomains.

At first consider the case of a regular domain Ω – a unit square in R^d, $d = 2, 3$. In this case it is straightforward to define both the fine and the coarse meshes to be regular with stepsize $h = 1/n$ and $H = 1/N$ respectively. We consider the case of linear finite elements defined on triangles or bilinear elements defined on quadrilaterals in the case $d = 2$ (see Figure 1.1). In the case $d = 3$ we consider trilinear elements based on the solid structures (see Figure 1.2). The dark bullets in Figures 1.1 and 1.2 denote the positions of the coarse grid nodes. Note that the overlap shown in Figure 1.1 with the shaded areas, may have a different width or may even be missing. If instead of the Dirichlet boundary condition we have the Neumann boundary condition

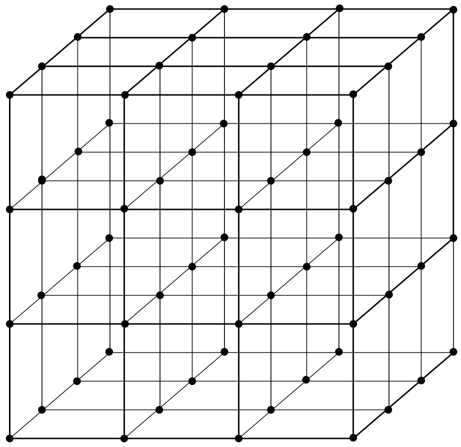

Figure 1.2: 3D domain split into subdomains with coarse grid nodes (dark bullets) in the case of Neumann boundary conditions.

on Γ, a part of the boundary $\partial\Omega$, these coarse grid nodes will appear also on Γ.

Domain decomposition methods are based on splitting the domain into overlapping or non overlapping subdomains. If the domain Ω and/or the triangulation is irregular the task of dividing the region into subregions is not trivial. In this case we use special graph partitioning software such as Chaco-2.0 [17] or Metis-2.0 [19] to obtain the partitioning as discussed in Paper D.

1.5 Conclusion

In this chapter we gave a short introduction to the thesis. We started with the description of the structure of this thesis giving a brief overview of the papers. The main model problem was introduced. We gave some background of domain decomposition methods and mentioned different possibilities how to derive them. Further we discussed briefly different types of domain decomposition methods depending on the existence of the overlap and the coarse grid. Finally, we gave an overview of our discretization scheme (based on FE) and mesh partitioning strategies.

Chapter 2

Overlapping Schwarz and Industrial Reservoir Simulation

FRONTSIM[1] is a software package for petroleum reservoir simulation. The parallel FRONTSIM code was developed as a part of the EC-ESPRIT III / EUROPORT2 project. Paper A gives an overview of the domain decomposition techniques used for the solution of the pressure equation in the case of incompressible flow. The permeability coefficient in the reservoir can vary in the space coordinates in the range from 0 up to 10,000. The discretization of the problem gives a symmetric positive definite matrix A. The reservoir is divided into overlapping subdomains, the overlap consisting of at least 4 gridblocks (which corresponds to 3 nodes when crossing a subdomain boundary). We assume a regular splitting into subdomains. There are two types of wells present in the reservoir: injection wells which are used for pumping water into the reservoir and production wells. For simplicity we assume that the subdomains have no neighbors in the z-direction so the wells (which we assume to be vertical) do not cross the subdomain boundaries. We assume also that the wells do not appear in the subdomain overlaps. There is a Neumann boundary condition on all of $\partial\Omega$.

The parallel FRONTSIM code was one of the first commercial codes where domain decomposition methods were used. The special value of this project was to verify the use of classical domain decomposition methods on realistic industrial problems. Many computer experiments were carried out to find the best strategies for practical problems. Some of the experimental results are presented in the paper.

We will first give a short overview of domain decomposition methods in the case of overlapping subdomains.

[1]FRONTSIM is developed by Technical Software Consultants AS

2.1 One-level methods

2.1.1 The Additive Schwarz method

Assume that the region Ω has been divided into N overlapping subdomains (see Figure 1.1). We consider the discretized problem in matrix form (1.4). To solve the problem efficiently one should find a good preconditioner to the matrix A such that the condition number of the matrix BA is small, which means that $B \sim A^{-1}$. In the one-level additive Schwarz method the preconditioner is of the form $B = B_1 + B_2 + \cdots + B_N$, where $B_i = R_i^T(R_iAR_i^T)^{-1}R_i$, $i = 1, 2, ..., N$. Here the restriction matrix R_i when applied to the entire vector x in (1.4) gives the components $x_i = R_ix$ that belong to subdomain number i. Applying the matrix B_i is equivalent to a subproblem solution on subdomain number i. In a practical implementation of the additive Schwarz method all the subdomain solutions are performed simultaneously and the solutions on the overlapping areas are added. Some communication is needed to perform this addition if the neighbor subdomain resides on a different processor. A more detailed overview of the communication in the Schwarz methods is given in Section 4.4.

The additive Schwarz method can also be derived from the multiplicative version of the algorithm by leaving out the steps that have dependencies.

2.1.2 The Multiplicative Schwarz method

The preconditioner in the multiplicative Schwarz method in the case of two subdomains has the form: $B = B_1 + B_2 - B_2AB_1$. In practice this means that between each subdomain solution we must compute the local residual which replaces also the values of the neighbors on the overlapping areas. Thus, the subproblem solution of all the subdomains that share some overlap should be performed sequentially. Parallelism can be introduced only by using some coloring scheme as shown in Figure 2.1 where $n_c = 4$ colors are used. Note that in the case of multiple subdomains also in the z-direction (which was not the case in the parallel FRONTSIM) 8 colors must be used. Subdomain solution for all the subdomains with the same color can then be run in parallel. If one uses the PCG method which need the preconditioner to be symmetric, some additional steps should be performed to maintain symmetry. Thus, the sequence of colors in the subdomain solution process can be like: 1–2–3–4–3–2–1. The number of PCG iterations in the multiplicative Schwarz method is often about two times smaller, but more work is performed at each iteration step.

To get an efficient implementation of the multiplicative Schwarz method

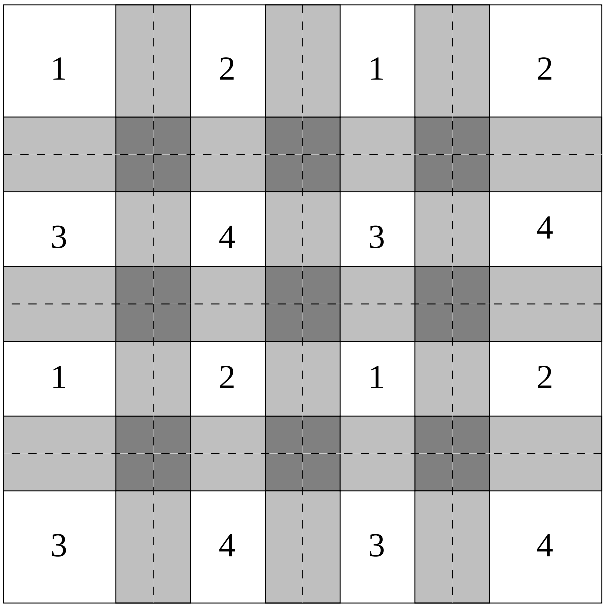

Figure 2.1: Domain decomposition for 4-color multiplicative Schwarz method.

on a parallel computer one should assign at least n_c subdomains with different colors to each processor. Otherwise the processor would be idle during the solution of the subdomain problems with the missing color.

2.2 Two-level methods

One-level domain decomposition methods have a disadvantage when applied in the case of a large number of subdomains; the solution components with low frequencies converge too slowly. The local subdomain solvers adapt the solution quickly to the local pattern while the global smoothing process still is traversing through the subdomains stepwise at each PCG iteration. If the region Ω is divided into $N \times N$ subdomains, it can take N iterations for the information about a substantial residual value at some part of Ω to make an influence on the whole region. Therefore, some global information transfer would help to reduce the number of PCG iterations. We introduce a coarse grid with an additional solver together with interpolation and restriction operations. In this thesis we are dealing with two different approaches for introducing the coarse solver techniques. The first approach which is used in the parallel FRONTSIM code, is based on the existence of a coarse triangulation. The second approach, described in the following sections, is not

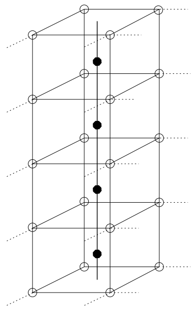

Figure 2.2: The well nodes are denoted by the dark bullets.

depending on the existence of a coarse triangulation and is therefore more general.

Assume that together with the fine triangulation E_h there exist coarse triangulation E_H of Ω. We consider a regular splitting of a regular region as shown in Figures 1.1 or 1.2. The coarse nodes appear in the cross-points of subdomain interfaces such that there are 4 coarse nodes per inner subdomain in the case $d = 2$ or 8 coarse nodes in the case $d = 3$. The coarse grid nodes appear on $\partial\Omega$ only if there is a Neumann boundary condition there. We define the finite element space V^H on the coarse grid exactly in the same way as the space V^h was defined on the fine grid.

We also need interpolation and restriction operators to make use of the coarse space techniques. We define the operator $I_0 = R_0^T$ as a linear[2] interpolation operator from the coarse grid nodes to the fine grid. In FRONTSIM the well nodes are situated in the middle of each block as shown in Figure 2.2. During the writing of Paper A the well nodes were not taken into account during the restriction and interpolation operations. It now appears that the use of also these values can reduce the total number of PCG iterations with up to 10%.

The next question that arises is: how to build up the stiffness matrix on the coarse grid? We describe two possibilities. The first possibility is to assemble the coarse matrix exactly in the same way as it was done on the fine level. This can be easily implemented in simple cases like the Poisson equation. It is more difficult if the coefficients are heterogeneous since the data must be scaled down to the coarse grid. The second possibility which we used in the parallel FRONTSIM is to build up the coarse matrix using the method called Galerkin approximation

$$A_0 = R_0 A R_0^T. \tag{2.1}$$

[2]In the case $d = 2$ we use a bilinear interpolation operator with quadrilateral elements. In the case $d = 3$ we use trilinear interpolation.

This can be performed using matrix multiplication techniques. But if the interpolation and restriction operators do not exist in matrix form (only the operations themselves are implemented) the columns a_k of the coarse matrix A_0 can be computed from

$$a_k = R_0 A R_0^T e_k,\ k = 1, ..., N,$$

where N is the number of coarse nodes and e_k is the k'th unit vector.
Define $B_0 = R_0^T A_0^{-1} R_0$. Then the preconditioner for the two-level additive Schwarz method can be written as

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

Let us now consider the multiplicative Schwarz method. It is possible to implement the coarse solver step in an additive or multiplicative manner (see Paper A for more details). Again, note that special care should be taken to maintain symmetry of the preconditioning step in the PCG method.

2.3 Inexact subproblem solvers

In order to reduce the cost of one PCG iteration one should consider an inexact solver for the subproblem solution. Computing the subdomain and coarse problem solutions inexactly increases the number of total iterations while the total time of the solution decreases. We tried various iterative subproblem solvers like Symmetric Gauss-Seidel (SGS), Incomplete Cholesky factorization and the multigrid (MG) method to find out the best strategies and choices in the different cases. We most often used the SGS method for inexact subdomain solutions. Usually, 2–3 SGS iterations give the best results in overall computation time.

2.4 Numerical results

Here we give some benchmarking results of the parallel FRONTSIM code that did not appear in Paper A. In Table 2.1 we give some benchmarking results on a cluster of 8 DEC ALPHA workstations connected with an *fddi* network. We tested different domain decomposition methods and inexact subdomain and coarse grid solvers applied to the test case of a reservoir with homogeneous geology consisting of 128x128x30 grid cells (491.520 grid cells

fine/coarse	1-level		2-level	
subdomain solver	Additive	Multipl.	Additive	Multipl.
Inc.Chol.	239 (31:11)	134 (35:56)	52 (19:01)	27 (18:18)
SGS	289 (43:12)	123 (44:39)	49 (18:51)	31 (21:54)
Inc.Chol. / Multigrid	—	—	47 (16:52)	25 (17:58)

Table 2.1: Iteration counts and solution times (min:s) for different domain decomposition methods and subdomain solvers in the case of a reservoir with homogeneous geology split into 11x11 subdomains.

Fine solver	Coarse solver	#it. (min:s)
Inc.Chol.	Multigrid	53 (23:0)
Inc. Chol.	Inc.Chol.	57 (23:50)
SGS	SGS	65 (27:21)

Table 2.2: Iteration counts and solution times (min:s) for reservoir with heterogeneous geology consisting of 128x128x40 grid cells. The reservoir is divided into 11×11 subdomains.

in total). The reservoir is located at a depth of 2000 m, with a constant discretization of 10x10x2.1 m in each grid cell. The production wells are located in the four corners of the reservoir and the injector is in the middle. The production in each well is 718 m^3/day at reservoir pressure, *i.e.* a total production of 2872 m^3/day. One time step is 10 years. The reservoir is divided into 11×11 subdomains. The purpose was to test the different Schwarz methods as well as finding the best strategies for choosing inexact solvers. In the first two rows of Table 2.1 the subdomain and coarse grid solvers are of the same type. In the third row the subdomain solver is an Incomplete Cholesky factorization while the multigrid solver is used on the coarse level. As expected, introduction of the coarse grid problem gives a significant reduction of the iteration count and of the total solution time. The best iteration count appears with the 2-level multiplicative Schwarz method, whereas timings are best with the 2-level additive method. Here we conclude that good results can be obtained when the reservoir is split into relatively small subdomains using Incomplete Cholesky as the subdomain solver and a multigrid method as our coarse grid solver.

In Table 2.2 we give test results in the case of a similar reservoir as in the previous example with the exception that the reservoir has heterogeneous geology. The permeability tensor is in the range $[1, 1000]$ in the x-direction and $[490, 510]$ in the y and z-direction. The grid size is $128 \times 128 \times 40$ grid cells, *i.e.* in total 655.360 grid cells. We see a similar behavior of the condition number and of the pressure equation solution time as in the case with the homogeneous geology.

2.5 Conclusion

This chapter served two main purposes: we described an industrial application that was successfully solved using domain decomposition methods and we gave an overview of classical domain decomposition methods in the case of overlapping subdomains. The Additive and the Multiplicative Schwarz methods were described. We described also two-level domain decomposition methods that are based on the existence of the coarse triangulation. We reviewed some details of our parallel reservoir simulator implementation. Several inexact subproblem solution methods were reported. The best strategies with the inexact subproblem solvers in the parallel reservoir simulation were discussed. Finally we presented some numerical experiments that we carried out during the benchmarking of the parallel FRONTSIM code.

We conclude that domain decomposition methods are suitable preconditioning techniques for large industrial application solution on parallel computers. These methods open a way for considering larger problem sizes using less time for computation using parallel platforms.

Chapter 3

New Additive Schwarz Methods

In Paper B we introduce the Average Schwarz method, a method without overlap between the subdomains and a special solver technique on the subdomain interfaces. The method can be efficiently implemented on parallel computers. The method is compared with another non-overlapping domain decomposition technique with a classical coarse space called the Additive Diagonal Scaling method.

3.1 The Additive Diagonal Scaling method

3.1.1 A One-level Additive Diagonal Scaling method

Consider the case when there is no overlap between neighbor subdomains. All the nodes on a substructure can be divided into two groups: inner boundary nodes (the nodes which are shared with some other substructure) and inner nodes. In the Additive Diagonal Scaling method the subdomain solution is associated only with the inner nodes while on the boundary nodes we use a diagonal preconditioner. We denote by $\widetilde{R}_i$ the restriction matrix that applied to the entire vector x in (1.4), returns the components $x_i = \widetilde{R}_i x$ that belong to the set of inner nodes of the i-th subdomain. We define

$$\widetilde{B}_i = \widetilde{R}_i^T(\widetilde{R}_i A \widetilde{R}_i^T)^{-1}\widetilde{R}_i,\ i = 1, ..., N. \tag{3.1}$$

In the Additive Diagonal Scaling method all the boundary nodes on the interfaces are treated as independent subdomains. Let n_B be the total number of nodes on the interfaces between all the subdomains. We denote by

$S_i,\ i = 1, ..., n_B$ the restriction operator that returns the i-th interface node and let

$$D_i = S_i^T \alpha_{ii}^{-1} S_i,$$

where α_{ii} is the diagonal element of the global stiffness matrix A corresponding to the i-th interface node. We can think of a node on an interface as a subdomain consisting of the single node. The solution process of such a one-node subdomain corresponds to dividing with the corresponding diagonal element in the stiffness matrix A. The preconditioner of the method has the form

$$\widetilde{B} = \sum_{i=1}^{N} \widetilde{B}_i + \sum_{j=1}^{n_B} D_j. \tag{3.2}$$

This method has interesting behavior relative to the quasi-monotonicity property in the distribution of the jumps in the coefficients ρ_i. We will discuss these issues in the next section where we add the standard coarse space to the method.

3.1.2 The Additive Diagonal Scaling method with a coarse space

We introduce the coarse space in the same way as in the case of a two-level additive Schwarz method (Section 2.2). Thus, the preconditioner can be written in the form

$$\widehat{B} = B_0 + \widetilde{B}.$$

We decompose the space V^h into

$$V^h = V_{-1} + V_0 + V_1 + ... + V_N,$$

where, for $i = 1, ..., N$, $V_i = H_0^1(\Omega_i) \bigcap V^h$ and zero outside of Ω_i. The coarse space $V_0 = V^H$, is a space of piecewise linear functions on the coarse triangulation which vanish on $\partial\Omega$ (we assume that the coarse triangulation with triangular elements is shape regular). The diagonal space V_{-1} is the restriction of V^h to $\Gamma = \bigcup_i \partial\Omega_i$. Functions in this space vanish at all nodal points outside Γ.

The bilinear form $b_i(u, v) : V_i \times V_i \to R$, for $i = 0, 1, ..., N$, is given by

$$b_i(u,v) = a(u,v),\ u,v \in V_i, \tag{3.3}$$

and for $i = -1$

$$b_{-1}(u,v) = \sum_{i=1}^{N} \rho_i h_i^{d-2} \sum_{x \in \partial\Omega_{ih}} u(x)v(x).$$

Consider the case of large jumps in the coefficient. The method of diagonal scaling is extremely sensitive to the coefficient jump ordering – the property of quasi-monotonicity.

The distribution of the coefficients is said to be quasi-monotone if there exists a monotone path from all subdomains to the subdomain having the largest coefficient, traversing through the faces of the subdomains that have a node in common, see Figure 1.1 in Paper B.

The condition number $\kappa(\widehat{B}A)$ is estimated by

$$\kappa(\widehat{B}A) \leq C \cdot \mathbf{max}_i \begin{cases} \frac{H_i}{h_i}, & \Omega \subset R^d, d = 2,3 \text{ when } \rho_i \text{ is quasi-monotone} \\ \frac{H_i}{h_i} \log \frac{H_i}{h_i}, & \Omega \subset R^2, \text{ when } \rho_i \text{ is not quasi-monotone} \\ (\frac{H_i}{h_i})^2, & \Omega \subset R^3, \text{ when } \rho_i \text{ is not quasi-monotone,} \end{cases}$$

where C is a positive constant independent of h_i and H_i and of the jumps of ρ_i. (See Theorem 2 in Paper B for proof).

3.2 Average Schwarz methods

Average Schwarz methods belong to the group of non-overlapping domain decomposition methods. The good properties of the methods are:

- more general problems can be solved,
- independence of the coefficient ρ_i jump distribution,
- simplicity of implementation and good parallel performance.

Additive and multiplicative variants of the Average Schwarz method are under consideration.

3.2.1 Formulation

We partition the space V^h into $N+1$ subspaces

$$V^h = V_0 + V_1 + ... + V_N$$

where $V^i = H_0^1(\Omega_i) \bigcap V^h$ and zero outside of Ω_i $(i = 1, ..., N)$. We denote the nodal points of $\partial\Omega_i$ and Ω_i by $\partial\Omega_{ih}$ and Ω_{ih}, respectively. Let n_i denote the number of nodes on $\partial\Omega_{ih}$. We define the coarse space V_0 by $V_0 = Range(I_A)$, the range of an interpolation-like operator I_A. For $u \in V^h \cap \overline{\Omega}_i$ we define $I_A u$ as follows:

$$I_A = \begin{cases} u(x) & x \in \partial\Omega_{ih} \\ \overline{u}_i & x \in \Omega_{ih} \end{cases}, \tag{3.4}$$

where

$$\overline{u}_i = \frac{1}{n_i} \sum_{x \in \partial\Omega_{ih}} u(x), \tag{3.5}$$

the average of the nodal values of u on the boundary of Ω_i.

Define by $\widehat{h}_i$ the diameter of the smallest element touching the boundary $\partial\Omega_i$:

$$\widehat{h}_i = \mathbf{inf}_{j: e_j \cap \partial\Omega_i} h_j \tag{3.6}$$

For each V_i, $i = 1, ..., N$ we define the bilinear form $b_i(u, v)$ by (3.3). The bilinear form on the coarse space is defined by

$$b_A(u, v) = \sum_i \rho_i \widehat{h}_i^{d-2} \sum_{x \in \partial\Omega_{ih}} (u(x) - \overline{u}_i)(v(x) - \overline{v}_i). \tag{3.7}$$

Let A_A be the matrix corresponding to the bilinear form (3.7). We define

$$B_A = R_A^T A_A^{-1} R_A,$$

where $R_A = I_A^T$.

3.2.2 The Additive Average method

The preconditioner in the case of the Additive Average method can be written as:

$$\overline{B} = B_A + \sum_{i=1}^{N} \widetilde{B_i},$$

where $\widetilde{B_i}$ are from (3.1). The condition number $\kappa(\overline{B}A)$ can be estimated with

$$\kappa(\overline{B}A) \leq C \cdot \mathbf{max}_i \frac{H_i}{\widehat{h}_i}$$

(see Theorem 1 in Paper B).

The Additive Average method is easy to implement on parallel computers. In Section 4 we give an overview of the parallel implementation of the method. The method is natural to use in the case when the coefficient ρ is constant throughout each subdomain having jumps only on the subdomain boundaries. But the method can be used also in the case when the coefficient is not constant in the subdomains. One such application to the filtration coefficient identification problem is given in Section 6.

3.3 Comparison with the classical method

The Average methods differ form the classical domain decomposition methods in various ways:

Coarse triangulation is not necessary in the Average methods while two-level classical domain decomposition methods need it. This opens a possibility of using the Average methods in more general cases with regions of complex geometry without loosing the power of two-level domain decomposition methods.

Computational complexity of the Additive Average method is smaller as can be seen in Table 1 of Paper D. This is true both for the number of arithmetic operations as well as for communication. The Additive Average method is

simpler to implement. The simplicity has its roots mostly in the way the restriction and the interpolation operators are defined. In the case of the classical two-level Additive Schwarz method we use linear or often (bi/tri)linear interpolation schemes that need either more computer storage or more arithmetic operations. At the same time, the

total number of PCG iterations is smaller in the case of the classical method, see Tables 1.2 and 1.3 in Paper A. Both the classical method and the Additive Average method are

independent of the jumps in the coefficient. This is proved in Paper B for the Additive Average method (Theorem 1). For the classical method no such proof is known but the fact is true from experiments. As we can see in Figures 1.3 and 1.4 of Paper B, the existence of a quasi-monotone path in the coefficient jumps does not affect the convergence rates if we use exact solvers. But with inexact subproblem solvers the Additive Average method is

more efficient. This can be observed in Figure 3 of Paper C. There we can see that if the jump coefficient is not quasi-monotone distributed the Additive Average method wins more and more in total computation time as subdomains and processors increase. This is caused by the fact that the classical Additive Schwarz method with overlapping subdomains needs to smooth the effects of the coefficient jumps within each subdomain. This causes the inexact solvers to become less efficient. In the Additive Average method the jumps do not appear in the subdomain solutions. In the case of quasimonotone jumps the Additive Average method wins in time but not as much as in the other case.

But perhaps the most important advantage for the Additive Average method is that it can handle unstructured grids – there is no need for existence of a coarse triangulation in order to be able to use the Additive Average method. We will discuss this issue more in Section 5.

3.4 The Multiplicative Average Method

In the Additive Average method all the subdomain solves and the coarse problem solution can be run in parallel. There is still a possibility to make some steps sequentially in order to reduce the number of iterations, this leads to *the Multiplicative Average Method.*

In this case, we begin with the coarse solver followed by the subdomain solvers with an additional residual calculation between the two. Since we use the PCG method, the preconditioner should remain symmetric. Thus, we must perform one additional residual calculation followed by the coarse solver to maintain symmetry. The preconditioning step can be written as follows:

$$\begin{aligned} v &\leftarrow B_A r \\ v &\leftarrow v + \widetilde{B}(r - Av) \\ v &\leftarrow v + B_A(r - Av). \end{aligned}$$

Compared to the Multiplicative Schwarz method in the case of overlap (Section 2.1.2), the Multiplicative Average method has the advantage that coloring is not needed. Subdomain solutions can run completely in parallel.

3.5 Conclusion

In this chapter we presented new Additive Schwarz methods with no subdomain overlap between subdomains. The most important results were about the Average Schwarz methods given in the Additive and in the Multiplicative versions. The independence of the coefficient jump distribution of the methods was reported. In order to compare the Average methods with some other method without the subdomain overlap we introduced the Additive Diagonal Scaling method. On contrary to the Additive Average Schwarz method which uses the Average coarse space the coarse problem of the Additive Diagonal Scaling method is based on the classical coarse space. Condition number estimates of the preconditioned stiffness matrix for the both methods were given. The Additive Diagonal Scaling method proved to be sensitive to the quasi-monotonicity property in the distribution of coefficient jumps.

We gave a comparison between the classical domain decomposition method with the subdomain overlap and the Additive Average Schwarz method. The Additive Schwarz method has a number of benefits like the simplicity of implementation and possibility of usage in the case of irregular subdomain structures. The Average methods are also more efficient in practice, the larger number of the total PCG iterations is compensated with far less computational complexity of one preconditioning step.

Chapter 4

Parallel Algorithms

In this section we give a more detailed overview of the way we have implemented the algorithms and data structures. The implementation scheme can be used in general cases on parallel computers. In our parallel implementation we have tried to use the data structures that make it possible to minimize the overall computer memory use, communication size and startups both during the assembly and during the solution process. We describe also temporary arrays in detail that are used for building up the data structures. The parallel implementation follows an SPMD parallel programming model where we assign several special tasks to processor zero. We use the MPI (Message Passing Interface, [14]) for communication. The programs were written in the ANSI C programming language.

4.1 Initialization and stiffness matrix assembly

Here we describe the data structures and operations needed in the fine grid level for initialization of the parallel algorithms in the domain decomposition methods.

4.1.1 Triangulation

We assume that there is available a triangulation E_Ω of the region Ω into the finite element structures $e_k \in E_\Omega$. We will not go into details on how the triangulation is obtained as this can be very different in particular practical cases. In the case of a regular domain (like the unit square) the triangulation can be a quite straightforward task to implement by introducing a regular mesh in the whole domain. In the case of the irregular domain the problem of finding a good triangulation can be quite a complex task and some special mesh generation and triangulation software should be used.

Our data structures for representation of the triangulation in the computer memory consists of two arrays:

- The floating-point array `X` with coordinate values of nodes in the triangulation. We use one array of length d*`n` where $d = \{2, 3\}$ is the problem dimension and `n` stands for the total number of nodes in the triangulation. The coordinates of the `i`-th node are stored in the subsequent d elements beginning with `X[`d`*i]`. This is a good way to represent the array as all the coordinate values of one particular node are typically requested at the same time. This is therefore good for data reuse and cache optimization.

- The integer array `Triangle` with values from the set $\{1, 2, ..., n\}$ representing the node numbers that belong to the triangles[1]. Similarly, information about the `i`-th triangle is stored in the locations `Triangle[i*e,i*e+1,...,i*e+e-1]` where `e` denotes the number of nodes belonging to that element (*e.g.* `e`= 3 in the case of a triangle, 4 in the case of a quadrilateral and 8 in the case of a solid element). Somehow we must also mark the nodes in the triangulation which belong to the outer boundary $\partial\Omega$ where the Dirichlet or Neumann boundary conditions are defined. This can be implemented without additional storage requirement by adding `n` to each node s that belong to the boundary $\partial\Omega$. The node numbers can then be calculated by `Triangle[j]%n`, where `%` denotes the operation of taking modulus.

4.1.2 Dividing the region into subdomains

We consider the case when the region Ω and the triangulation E_Ω can be irregular. Thus, the task of dividing the region into subdomains is not trivial. In Section 5 we describe how it is possible to divide the region Ω into subdomains using special graph partitioning software.

As a result of the partitioning we define the array `P[1,...,n`$_t$`]` with values in the set $\{1, 2, ..., N\}$, which gives for each triangle the partition number it belongs to. We consider here the case of no or minimal overlap between the neighboring subdomains. We assume that one process can have multiple subdomains. For assembling the local stiffness matrix A_i the process that is responsible for subdomain i needs information about

[1] We call all elements triangles, also quadrilaterals and 3-D solid elements.

the triangles $\{e_k : \mathbf{P}e_k = i\}$ together with one additional layer of triangles around partition i. For this purpose, at first we introduce an array `NodeOwners`$_k$`[0,1,...,MaxNumberOfOwners]` for each node k in the mesh. According to the array `Triangle` all the nodes in triangle $i\,(i = 1, ..., \mathbf{n}_t)$ get the partition number `P[`i`]` represented in the array `NodeOwners`$_k$. Then the array `Triangle` is examined for the second time such that if one of the nodes in a triangle has an owner a listed that does not appear in some of the triangle's other nodes, the nodes missing the owner a get it listed with the negative value $-a$ denoting that the node belongs to the subdomain a outer layer. This extra layer of triangles is needed to be able to assemble the local part A_i of the global stiffness matrix A without additional communication in the later stages of the parallel assembly process.

Now we describe the information gathering process for subdomain j, $(j = 1, 2, ..., N)$. Based on the arrays `Triangle`, `NodeOwners` and `X` we create the arrays `LNodeOwners`$_j$, `LTriang`$_j$ and `LX`$_j$ corresponding to the local values of the subdomain j. We use a compact storage scheme for representing the arrays `LNodeOwners`$_j$ consisting of one additional integer array pointing to the starts of the information about each local node as the number of owners of different nodes varies. We list only the node's other owners than j in the array `LNodeOwners`$_j$ (the owner j itself is obvious); if a value in the array `NodeOwners` is negative, it is ignored unless it equals to $-j$. The array `LTriang`$_j$ is selected out of the triangles in the array `Triang` whenever some owners of the triangle nodes is equal to j. The array `LX`$_j$ is formed from the array `X` elements for which there exist a node owner equal to $\pm j$. Local indices are used in the arrays `LTriang`$_j$ and `LX`$_j$, the mapping from local to global indices is preserved to be able to form the global solution from the local ones afterwards.

4.1.3 Load balancing and distribution of jobs

The arrays `LNodeOwners`$_j$, `LTriang`$_j$ and `LX`$_j$ on process zero are distributed to other processes. The number of available processors N_P is often smaller than the number of subdomains N. In this case some load balancing scheme is appropriate to use to spread the work evenly among the processors taking possibly into account also the processor capacities. If N_P is large enough it can be preferable to assign a single subdomain to each processor and the coarse solver to a separate one. Together with the arrays above some additional information is also needed on the processes like the right-hand-side, the porosity coefficient in the case of a heterogeneous reservoir simulation or the ground water levels in the case of a filtration coefficient identification problem.

4.1.4 Local matrix assembly

Assembly of the local matrices A_j $(j = 1, ..., N)$ is performed locally on all processes. Based on the information in the arrays $\texttt{LTriang}_j$ and $\texttt{LNodeOwners}_j$ the process decides which nodes belong to the outer boundary $\partial\Omega$ (where there is a boundary condition), which nodes belong to the interfaces and also identifies the neighbor subdomains. Also, the inner nodes are marked. In preparation for nearest neighbor communication the index numbers of the nodes that belong to the interface between subdomains i and j are stored in the integer array $\texttt{Slize}_{ij}$. The following question may be addressed: how can we guarantee that $\texttt{Slize}_{ij}\texttt{[k]}$ and $\texttt{Slize}_{ji}\texttt{[k]}$ refer to a unique node in the global mesh without communication of the node coordinates between subdomains i and j? The answer is that this is guaranteed without any communication due to the existence of the global numbering of nodes. If two nodes appear in sequence in the global numbering they will appear in the same order also in the local numbering of the nodes.

The local stiffness matrix A_j is assembled in the standard way as in a serial FEM implementation. Special care is taken only with the nodes on the interfaces as it should be possible to compute correctly the global matrix-vector multiplication operation Ax with minimal communication. We have multiple representations of nodes on the interfaces. It is assumed that before the operation Ax the values of the vector x in different representations are equal: $x\texttt{[Slize}_{ij}\texttt{[k]]} = x\texttt{[Slize}_{ji}\texttt{[k]]}$. The operation $y = Ax$ is performed in such a way that the sum $y\texttt{[Slize}_{ij}\texttt{[k]]} + y\texttt{[Slize}_{ji}\texttt{[k]]}$ would give the right value on the interface nodes. Thus, if there exist more than one representation of an element of the global matrix A within the local matrices A_j, $j = 1, 2, ..., N$, the one with the largest subdomain number is chosen to perform the corresponding multiplication operation. The other subdomains get that part of the result only after the summation. Such duplicate elements in the local matrices are marked as inactive for the Ax operation.

In the Additive Schwarz methods without overlap the subdomain solvers are applied only to the subdomain inner nodes and thus, all the elements in the matrix A_j in the rows and columns corresponding to the interface nodes are marked inactive for subdomain solution.

We use a compact storage scheme for all matrices in the implementation storing only nonzero entries of matrices in corresponding floating-point arrays. Information about the nonzero locations is stored separately in integer arrays.

4.2 Coarse matrix assembly

The coarse problem in the Average method has the following form:

$$\begin{pmatrix} A_a & A_{ab} \\ A_{ab}^T & A_b \end{pmatrix} \begin{pmatrix} u_a \\ u_b \end{pmatrix} = \begin{pmatrix} 0 \\ v_b \end{pmatrix} \tag{4.1}$$

where we have introduced one extra unknown per subdomain in the vector u_a corresponding to the average values $\overline{u}$ from (3.5). The subscripts a and b denote average and boundary respectively. The vector u_b has one component for each node on all subdomain boundaries.

Denote by $n_B(i)$ the number of boundary nodes of substructure i ($i = 1, ..., N$), inclusive the outer boundary nodes where we have a Dirichlet boundary condition. Let $n(i)$ be the number of all nodes in the substructure, here we do not count the outer boundary nodes. The $N \times N$ matrix block A_a is diagonal with entries

$$(A_a)_{ii} = C_i \rho_i n_B(i),\ i = 1, ..., N. \tag{4.2}$$

The value of the component

$$C_i = C(\widehat{h}_i) = c_0 \widehat{h}_i^{d-2}$$

in (4.2) depends on $\widehat{h}_i$ only in the case $d = 3$. The scaling factor $c_0 = 1$ in the case $d = 3$ and in the case $d = 2$ with quadrilateral elements. But when using triangular elements, the value $c_0 \approx 1.7$ gives somewhat better convergence rate of the method. This was discovered experimentally.

Denote by $n_S(j)$ the number of subdomains that share node j ($j = 1, ..., n_b$) on the substructure boundaries. Let r_{jk} ($k = 1, ..., n_S(j)$) be the index of the corresponding substructure. The $n_b \times n_b$ block A_b is also diagonal with entries

$$(A_b)_{jj} = \sum_{k=1}^{n_S(j)} C_{r_{jk}} \rho_{r_{jk}}. \tag{4.3}$$

The block A_{ab} contains the nonzero entries

$$(A_{ab})_{ij} = -C_i \rho_i,\ i = 1, ..., N,\ j = 1, ..., n_b \tag{4.4}$$

in such a way that the first block equation of (4.1) just expresses the relation (3.5) between the average and the boundary values.

Note that there is no need to store the matrices A_{ab} and A_b explicitly as all their entries corresponding to the boundary nodes that belong to the same set of substructures have the same value[2]. Instead, we store only one floating point value for each combination of boundary node owners and a pointer for each boundary node to this set of values. The submatrices A_a, A_b and A_{ab} are calculated in parallel and stored locally. During the calculation the ρ value of the neighbor subdomains is needed. This is obtained through a nearest neighbor communication scheme.

4.3 Coarse problem solution

For the solution of the coarse problem in domain decomposition the residual vector $r = \{r_{ij} : i = 1, ..., N,\ j = 1, ..., n(i)\}$ is restricted onto the coarse space. After obtaining the solution of the coarse problem it is interpolated back to the fine space. In the Average Schwarz method the special structure of the matrix in (4.1) is exploited. Denote by $K_i = \{k_1, ..., k_{n_K(i)}\}$ the set of indices of inner nodes of substructure i. Let v_b be the vector of boundary nodes from r. The restriction operation in the Average Schwarz method can be expressed as follows:

$$(v_a)_i := \sum_{j \in K_i} r_{ij},\ i = 1, ..., N, \tag{4.5}$$

$$v_b := v_b - A_{ab}^T A_a^{-1} v_a. \tag{4.6}$$

For the matrix in the system (4.1) it is easy to compute the Schur complement

$$S = A_a - A_{ab} A_b^{-1} A_{ab}^T. \tag{4.7}$$

The matrix S is $N \times N$ symmetric and sparse with a nonzero diagonal and entries at the positions mirroring the pattern of the substructure neighborhood property. Computation of S can be performed in the beginning and it is independent for each substructure, thus we can run it in parallel. After forming S locally it is sent to the coarse problem solver process.

[2]This is true if we assume that the coefficient ρ is constant in each substructure.

On each coarse step we solve the system

$$S u_a = -A_{ab} A_b^{-1} v_b, \tag{4.8}$$

which involves some transform operations expressed at the right hand side of (4.8). This transformation can also run in parallel. As a result, each substructure sends only one floating-point value to the coarse solver. After the solution of (4.8) the values of the vector u_a are ready for the inner nodes meaning that the coarse system solution as described above has incorporated the interpolation-like operator I_A. The boundary nodes get their values after the back-transform operation

$$u_b = A_b^{-1}(v_b - A_{ab}^T u_a), \tag{4.9}$$

which run in parallel.

4.4 Communication

In the parallel implementation of domain decomposition methods it is important to organize communication between processes in a way that reflects the underlying algorithmic and architectural features. In our implementation we try to minimize the communication volume and the number of startups by packing different type of information into a single message when possible. Sometimes it is relevant to reorganize an algorithm to postpone certain communication directives in order to get faster implementations. We use the Message Passing Interface (MPI) [14] as the communication library. The MPI implementation itself is optimized for specific architectures and one need know only the standard library calls without going into the details of a specific architecture to optimize the corresponding code.

4.4.1 Communication types

There are three types of the communication stages in the implementation of domain decomposition methods: the preparatory, the PCG and the concluding communication stages. At the preparatory stage all the initial data about the whole domain and the subdomains is communicated from process zero to other processes. This is organized by packing all kinds of information needed by other processes into longer messages.This information includes general information about the domain and the method to be used during the solution but also information about the grid, the coordinates of the gridpoints, the right hand side, the location of the subdomain in the whole domain etc. During the concluding stage of communication the solution is gathered to process zero for output or postprocessing. In the rest of this section we describe communication in the PCG iteration loop.

We give an overview of the communication model and the differences between the different domain decomposition methods. The Preconditioned Conjugate Gradient method can be written as

Algorithm 1 *PCG*

begin
  $\mathbf{r} \leftarrow \mathbf{b}, \mathbf{x} \leftarrow 0$
  repeat
    $\mathbf{z} \leftarrow P\mathbf{r}$
    update^{I} $\mathbf{z}$
    $\rho_2 \leftarrow \rho_1$
    $\rho_1 \leftarrow \langle \mathbf{r}, \mathbf{z} \rangle$
    if (first iteration)
      $\mathbf{p} \leftarrow \mathbf{z}$
    else
      $\beta \leftarrow \rho_1 / \rho_2$
      $\mathbf{p} \leftarrow \mathbf{z} + \beta \mathbf{p}$
    endif
    $\mathbf{s} \leftarrow A\mathbf{p}$
    update^{II}$\mathbf{s}$
    $\alpha \leftarrow \rho_1 / \langle \mathbf{p}, \mathbf{s} \rangle^{*}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{p}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha \mathbf{s}$
    update* $\mathbf{r}$
  until convergence
end

Domain decomposition procedures (interpolation and restriction operators, fine and coarse solvers) appear in the preconditioning step $\mathbf{z} \leftarrow P\mathbf{r}$ of the algorithm. Here we consider the case of Schwarz methods of additive type .

Communication appears in the following steps:

Vector update operations. The **update** operations in Algorithm 1 reads that one should perform the **update**^{I} and **update**^{II} operations or only the **update***depending on the method. The **update**^{I} and **update**^{II}

denote the subdomain nearest neighbor communication in the case of the classical Additive Schwarz method while the **update*** is the nearest neighbor communication in the case of the Additive Average method. The operation means updating the values on the subdomain interfaces Ω_{ih}. Corresponding values on the interfaces are added. Superscript $*$ in the operator $\langle\cdot,\cdot\rangle^*$ means that the local inner product is performed in a slightly different way in the case of the Average Schwarz method since it is performed before the **update*** operation. The **update*** operation is delayed until the inner node sums of the updated $\mathbf{r}$ is already available. In the Average coarse system solution these sum values are needed by the neighboring subdomains. In our implementation in order to reduce the communication startups the sum values are appended to the messages sent during the **update*** operation.[3]

Inner products. All the subdomain processes compute the inner product locally and the values are summed by using a standard call in the MPI.

Preconditioning. During the preconditioning step we need communication between the subdomain processes and the coarse solver.

4.4.2 Process communication strategy

In the case of multiple subdomains on a single process there are possibly many subdomains on processor k that need to communicate with their nearest neighbor subdomains on the processor l. To reduce the number of communication startups all communication between processors k and l is packed into a single message instead of sending each of the small messages separately. For this purpose we form the list `NeighProc` of all process numbers that the current process has neighboring subdomains. And for each element `NeighProc[k]` we create another list $\texttt{SubdPairs}_k$ containing pairwise the subdomain numbers of the neighbors. Based on these lists it is easy to pack together and unpack the messages between the processors.

4.5 Conclusion

In this chapter we discussed in more detail our implementation of the domain decomposition methods described in this thesis. The compact storage scheme

[3]It is possible to use a similar delay strategy to reduce the number of updates to only one in the case of the classical Additive Schwarz method as well. But this also doubles (in the case with minimal overlap) the message lengths and makes the algorithm implementation more complex. We did not implement it in such a way.

is used whenever possible in the implementation. We described the ways of dividing the region into subdomains. If the region is not regular we use the special graph partitioning software for the splitting. We described how based on the subdomain numbers for each triangle in the triangulation we form the subtasks and send them to other processors. We explained also how to build up structures for the nearest neighbor communication in the best way that would not need extra communication. There are no computationally complex operations on process zero that would cause a bottleneck in the implementation with an exception of the graph partitioning in the non regular case. With the availability of also parallel versions of the graph partitioning software in the future, this bottleneck can be avoided.

We gave an overview on how we build up the coarse matrix with the Average Schwarz methods. The special value of the implementation is that at first the coarse matrix is built up in parallel and then it is sent to the processor that is responsible for the coarse problem solution. We described the way the coarse problem is solved.

Further we discussed the implementation of communication procedures. We described various stages of the PCG loop where communication appears. The implementation with the Average Schwarz methods differs from the classical domain decomposition method implementation. The total number of words communicated and the number of communication startups is less with the Average Schwarz methods. It is possible to reduce the number of startups by the delay in communication rearranging some operations in the algorithm with the Average Schwarz methods. Our implementation allows multiple subdomains on a single process. Messages between different subdomains on the same pair of processors are packed together to reduce the number of communication startups.

Chapter 5

Unstructured Grids

The possibility of using domain decomposition methods with unstructured grids and discretizations has attracted much attention in the scientific community. In Paper D we give a short overview of possible solutions to this type of problems.

Consider a region $\Omega \subset R^d$, $d = 2, 3$. We deal with two possible cases:

- The region Ω is irregular and/or the triangulation of the region is irregular. In this case it is normally impossible to find a coarse triangulation of the region in order to define subdomains matching the triangulation. In practice such situations arise quite often as most of the problems are defined on regions with complex geometry in 2 or 3 dimensions.

- The region Ω itself can be regular but for some reasons like underlying material properties in the region advise us to choose irregular subdomains. We will give a real life example of such a situation.

If we do not have the coarse triangulation the classical two-level Additive Schwarz method cannot be used. But we can use the Average Schwarz methods for solution of such problems as they do not need the coarse triangulation. We will give a more detailed overview of the practical application of the methods.

5.1 Solving large problems on irregular structures

Suppose we want to use a domain decomposition method for solution of a partial differential equation on Ω. The first task is to split the region into N subdomains Ω_i. If the region is large and complex then the task is not

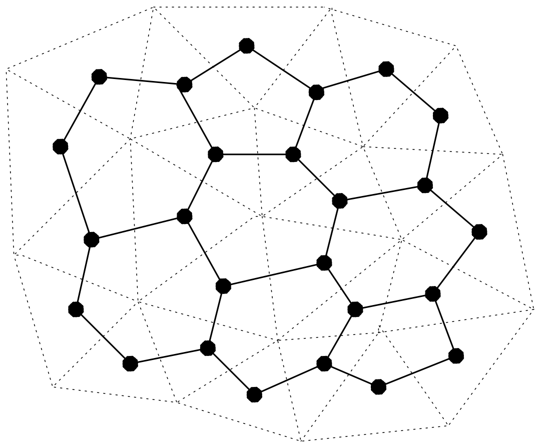

Figure 5.1: Preparing a dual graph based on the triangulation for the graph partitioning software.

trivial. In the case $d = 2$ we could plot the region and split it using the graphical view of it. But in the case $d = 3$, normally, this would be too complex. Fortunately, we have another solution: we let the computer to do the job for us. A special mesh partitioning software can be used. We use two packages for this purpose: Chaco-2.0, developed by Bruce Hendrickson and Robert Leland at Sandia National Laboratories [17] and Metis-2.0 developed by George Karypis and Vipin Kumar at the University of Minnesota [19].

At first we form a dual graph with vertices corresponding to the elements in the triangulation and having edges between adjacent elements (see Figure 5.1). The dual graph and the number N of desired subdomains are given as input to the graph partitioning software. As a result of the partitioning each triangle gets a unique partition number. The resulting subdomains are almost equal in size and therefore it is possible to balance the load equally in the case of parallel implementation of the domain decomposition method. Though our experience shows that care should be taken with the partitions obtained in such a way. In certain situations it may sometimes happen that resulting subdomains are not connected. This would increase the condition number $\kappa(\overline{B}A)$ in the case of the Additive Average method and should be avoided. We use a simple algorithm to check for the connectivity of partitions. A simple graph recoloring algorithm is used if a non connected component of some partition is found. This may lead the sizes of subdomains slightly out of the balance but the number of the PCG iterations is smaller. The graph checking algorithm needs a single pass of the graph.

In Table 1.1 of Paper D we give test results with an unstructured mesh around an airfoil. The graph is partitioned using Metis-2.0. The Additive

N	Dofs	Galerkin	Additive Average	Addit. Av. with weighting
3	15606	33 (64.7)	34 (77.1)	33 (69.5)
15	61484	60 (139)	72 (270)	64 (165)
63	244047	90 (259)	126 (666)	99 (334)

Table 5.1: Comparison of three different methods with exact subproblem solvers on a scaled version of the airfoil mesh.

Average method is used to solve the problem. We refined the mesh to obtain bigger test cases. The tests were run on a T3E and on an Origin-2000 computer using two symmetric Gauss-Seidel sweeps as our inexact solver. In Table 5.1 we give the number of the PCG iterations and the condition number $\kappa(BA)$ for three different methods on the same meshes with exact subproblem solvers. We call here Galerkin the Average Schwarz method with the coarse grid matrix obtained in a similar way to (2.1) using I_A from (3.4) as the interpolation and the transpose $R = I_A^T$ as the restriction operator. Note that the resulting coarse matrix is very big since all the nodes on subdomain interfaces belong to the coarse space. In the last column of Table 5.1 we give results for the Additive Average method with weighting. It appears that with irregular meshes the Additive Average method can be improved with introducing to bilinear form (3.7) special weights reflecting the grid structure. We describe this possibility of improvement of the Additive Average method below in Section 5.3 in more detail.

5.2 Coefficient jumps with irregular structure

With jumps in the coefficient it is usually preferable to partition the mesh so that the jumps appear on subdomain boundaries. For this purpose it is possible to set weights to the edges in the dual graph. The more equal the coefficient is in two neighbor elements the smaller the weight is associated with the edge in the graph. With such weighting the jumps will tend to appear on the substructure boundaries after partitioning. But our experience showed that the mesh partitioning software available today is not capable to solve such a problem with sufficient quality. As a side effect we often get nonconnected components of subdomains and the coefficient jumps can still cross subdomains. Big coefficient jumps in subdomains make the Additive Average method and also the classical domain decomposition methods with inexact subproblem solvers less efficient. We would welcome further research in this area hopefully together with people from the area of graph partitioning.

Let us return to the oil reservoir simulation problem discussed in Paper A. In practice, the coefficient used in the pressure equation in the oil reservoir simulation varies in a wide range, usually from 0 up to 10000. There are often big jumps in the coefficient as the reservoir may consist of different materials with sharp interfaces between the soil with different properties. Assume that we can find subregions Ω_i in which the coefficient is relatively constant. In such cases the Average methods are very suitable if it is only possible to detect the subregions. In Table 1.2 of Paper D we give results in the case of a reservoir discretized into $55 \times 80 \times 100$ gridblocks with two permeability values 1 or 1000 (see Figure 1.3 in Paper D). Since the graph weighting strategy described above does not yet give satisfactory results we divide the dual graph into two parts according to the coefficient value in each gridblock. The two separate subgraphs are then partitioned separately with the graph partitioning software. In such a way we guarantee that the coefficient jumps appear only on the subdomain boundaries. The Additive Average method is applied to the resulting region consisting of 510 subdomains. This is compared with the Additive Average method and the Additive Schwarz method with the regular splitting into the subdomains of the region. We observe that the most efficient method is the first one giving the smallest iteration count and the smallest condition number. As waited, the Additive Average method does give poor results with regular subdomains in this case. Note that we use an inexact subproblem solver here. With an exact subproblem solver the classical Additive Schwarz method would show a better iteration count than the Additive Average method. But in order to reduce the total computation time we use inexact solver in practical cases.

5.3 Possibilities of improvement of the coarse bilinear form

When testing the Additive Average method on irregular grids we observed that the method does not show as small iteration count and condition number as in the regular case. We started to study the reasons of such a behavior more carefully and can give a simple recipe now that improves the convergence rates considerably with irregular meshes. One of the reasons of the worse behavior of the method in the irregular case is the following: The elements of stiffness matrix A are evenly of the same size throughout the whole region with regular grid. On the contrary, with the irregular grid the elements in some parts of matrix A can differ considerably from the elements in a different part of the matrix. Suppose that elements sizes of submatrix A_i

differ considerably from the element sizes of submatrix A_j. With the Additive Average method elements of the coarse matrix associated with both subdomains i and j are about the same or are equal if both subdomains have the same number of nodes common with neighbors. In the Average coarse grid matrix obtained with the Galerkin procedure (see Section 5.1) the elements corresponding to subdomains i and j differ considerably in size. Experiments show that the Additive Average method with the Galerkin coarse grid matrix gives much better iteration counts and the condition number of the preconditioned stiffness matrix. This gave us an idea to modify the bilinear form of the Additive Average method according to the element sizes of matrix A in the same way as in the case of the Galerkin coarse grid matrix. For this purpose we replace bilinear form (3.7) with

$$b_A(u,v) = \sum_i \rho_i \widehat{h}_i^{d-2} \sum_{x \in \partial\Omega_{ih}} c_0(x)(u(x) - \overline{u}_i)(v(x) - \overline{v}_i), \tag{5.1}$$

where

$$c_0(x_k) = c_1 + c_2(a_{kk} - a_{reg})/a_{reg}.$$

We denote here by a_{kk} the diagonal element of stiffness matrix A on the node x_k and by a_{reg} we mark the diagonal element of matrix A with the regular grid. With $d = 2$ and triangular elements we have $a_{reg} = 4.0$ and the best convergence is obtained in the case $c_1 = c_2 \approx 1.7$ (which was found out by experiment). Note that in the implementation of the method we modify appropriately formulas (4.2), (4.3) and (4.4). We use the average value of $c_0(x)$ on each subdomain when appropriate. In Table 5.1 we compare the three methods on the airfoil mesh. We observe that using the weights in the Additive Average method with large grid sizes the reduction of the condition number is more significant.

One of the open questions is: whether the convergence improves also with weights on subdomain inner nodes? Results of an experiment show that the Average method with the Galerkin coarse grid matrix improves with weights on inner nodes but we have not yet detected the similar behavior with the Additive Average method. Our tests show so far that the same type of weights as on the boundary nodes do not improve the convergence. Research in this particular subject will continue.

5.4 Conclusion

In this chapter we studied possibilities of using the domain decomposition methods with irregular meshes. If we do not have the coarse triangulation the classical Additive Schwarz method cannot be used. We described the possibility of using the graph partitioning software for dividing the region into subdomains. The graph partitioning software works well in the case of homogeneous coefficient in the region. If we have jumps in the coefficient a different strategy should be used by first dividing the region into subregions according to coefficient values. The graph partitioning software does not yet give satisfactory results with weighted graphs. We discussed two test examples with irregular meshes and the possibility of improvement of the Additive Schwarz method by introducing weights to the coarse bilinear form. We pointed out some questions that still need further research to obtain a better insight than we have gained up to time of writing of this thesis.

Chapter 6

An inverse problem

Parameter identification problems belong to the class of inverse problems. Suppose we know the solution of an elliptic differential equation and based on that solution we want to obtain some information about material properties in the region. The filtration coefficient identification problem is based on measurements of the ground water levels in some region and an estimate of the coefficient is desired. The problem is known to be ill-posed (see [32]) and therefore more difficult to handle. In Paper E the discretized filtration coefficient identification problem is solved with the CG method without preconditioning. Domain decomposition techniques for this problem are reported in [22] where minimization technique is used. In this part of the thesis we introduce preconditioners based on the Additive Schwarz methods discussed in the previous chapters. This part of the work has not yet been published.

6.1 Problem formulation

The filtration coefficient identification problem is an important application in environmental research. With an estimate of this coefficient in a given area it is possible to carry out numerical simulations that can be of value in complex environmental decisions. For ill-posed problems some kind of regularization technique must usually be used. In Paper E we used the Tikhonov regularization method [31]. Here we use a truncated CG iteration technique and apply it in the preconditioned case.

We take advantage of the same discretization strategies as in Paper E. In the paper we present results for the filtration coefficient identification with real data. Here we evaluate the new domain decomposition methods on the model problem with small measurement errors in the data.

In the filtration coefficient identification problem the bilinear form of the problem has the form:

$$a(v, w) = \int_{\Omega} (\nabla u \cdot \nabla v)(\nabla u \cdot \nabla w)\, dx, \tag{6.1}$$

where $u = u(x)$ is the known function of ground water levels in the region Ω. After solution of the discrete problem:

Find $v^* = \sum_{i=1}^{n} c_i w_i \in S_h$ such that

$$a(v^*, w_h) = f(w_h) \;\; \forall w_h \in S_h, \tag{6.2}$$

the filtration coefficient is then computed from

$$c_{filt} = \sum_{i=1}^{n} c_i \nabla u \cdot \nabla w_i, . \tag{6.3}$$

The stiffness matrix A is obtained from (6.2) in a standard way (see Paper E). The condition number of A is big since the bilinear form (6.1) is dependent also on the gradient of u. This makes the iteration count in the case of the CG method without preconditioning very large. To reduce the condition number some preconditioner should be used. For that purpose we have implemented the Average Schwarz methods for solution of the filtration coefficient identification problem. The implementation is quite similar to the model problem in Section 1.2. While assembling the local matrices A_i, $i = 1, ..., N$, also the average value G_i of the expression $|\nabla u(x, y)|^2/2$ over all triangles of Ω_i is calculated,

$$G_i = \frac{1}{2n_t(i)} \sum_{e_j \in \Omega_{ih}} (\frac{\partial u(x_j, y_j)}{\partial x})^2 + (\frac{\partial u(x_j, y_j)}{\partial y})^2, \tag{6.4}$$

where $n_t(i)$ is the number of triangles in substructure Ω_{ih} and the point $(x_j, y_j) \in e_j$ is in the the center of the triangle e_j.

We introduce in a similar way to Sections 3.1.1 and 3.2.1 the coarse spaces for the one-level Additive Diagonal Scaling method and the Additive Average method. The coarse matrix (4.1) in the Additive Average method is assembled in similar way as in Section 4.2 with the exception that we replace the values ρ_i with G_i. Thus we have instead of formulas (4.2,4.3,4.4) the following:

$$(A_a)_{ii} = C_i G_i n_B(i), \; i = 1, ..., N, \tag{6.5}$$

$$(A_b)_{jj} = \sum_{k=1}^{n_S(j)} C_{r_{jk}} G_{r_{jk}}. \tag{6.6}$$

and

$$(A_{ab})_{ij} = -C_i G_i,\ i = 1, ..., N,\ j = 1, ..., n_b. \tag{6.7}$$

The method is more effective in the case when $\nabla u(x, y)$ does not change very much within each subdomain as can be seen from equation (6.4).

6.2 Numerical results

We consider a problem similar to the model problem in Section 3.1 of Paper E. The functions $c_{filt} = a(x_1, x_2)$ and $u = u(x_1, x_2)$ are defined by

$$\begin{aligned} a(x_1, x_2) &= e^{x_1 + x_2} + \sin 2(x_1 - x_2), \\ u(x_1, x_2) &= 10[(x_1 - 1/4)^2 + (x_2 - 1/3)^2], \end{aligned}$$

and Dirichlet boundary conditions are defined all around the unit square (the coefficient is identifiable due to the fact that there is only outflow present through the boundaries, see [32] for details.) The solution of the problem is known. Note that in the point $(1/4, 1/3)$ the gradient ∇u is zero and the method is known to give the identified coefficient with big error in such a point. This is caused by the fact that the bilinear form (6.1) is zero there. Note that in Paper E the zero point is outside the region in the model problem but a Neumann boundary condition is used to maintain the identifiability of the coefficient. We do not consider that case here.

Suppose the ground water levels u can be measured with some error δu. The filtration coefficient identification problem is extremely sensitive to the measurement errors, especially in the case of small stepsizes in the grid. This is because the bilinear form (6.1) depends on the gradient ∇u. In the case of a small stepsize h the error influences the gradient value more dramatically, especially around the gradient zero point. This may cause a failure of the discretized method and some kind of smoothing procedure must be applied to the input data. In our computer experiments we do not make the smoothing.

We present results of the filtration coefficient identification in three cases: the relative measurement error is up to 0% (no error), 0.1% or 1.0%. We computed the filtration coefficient using expression (6.3) after each PCG iteration to compare it with the known exact coefficient. We then noted on

#blocks	#Subd.	δu (%)	#it.(cond.#)	Min. error	it.
16x16	4x4	0	132 (697)	4.22e-02	41
		0.1	132 (698)	4.37e-02	40
		1.0	133 (712)	9.94e-02	36
32x32	8x8	0	341 (2770)	1.48e-02	194
		0.1	366 (4403)	3.60e-02	108
		1.0	375 (4735)	2.62e-01	111
64x64	16x16	0	883 (25839)	5.52e-03	467
		0.1	880 (25957)	6.73e-02	273
		1.0	1216 (33618)	7.82e-01	356
128x128	32x32	0	2097 (121638)	1.92e-03	1074
		0.1	2127 (139007)	1.86e-01	570
		1.0	failed		

Table 6.1: CG method without preconditioning. Filtration coefficient identification with measurement error. The minimal average error and the iteration # of its achievement are given in the last column.

which iteration step the computed filtration coefficient differs least from the real solution. The solution of ill-posed problems with iterative methods gives us a regularization possibility by truncation of the last iterations when the solution itself does not improve any more. This has been reported in the case of the CG method (see [15], [16]). In the CG iterations the solution components that correspond to the small eigenvalues associated with noise in the data tend to be smoothed last. This is paving the way to the regularization. We were interested to study this property also in the case of PCG methods with our Schwarz preconditioners.

In Tables 6.1, 6.2 and 6.3 we give the results of the computer experiment for the CG method without preconditioning, with the Additive Average method and with the method of Diagonal Scaling. Exact subproblem solvers were used. The subdomain sizes were kept constant (4x4 gridblocks) while the whole unit square was divided into different number of blocks. Note that in the 4th column we give the iteration count and the condition number of the preconditioned matrix after the convergence while in the last column the number denotes the iteration number where at average the best filtration coefficient was recovered. We see that the bigger the measurement error δu is the sooner the lowest average error is detected. We see also that with measurement error refining the grid does not always give better solution. In the case of 128x128 grid the 1% error is already too large and the methods do not converge any more. At the same time the preconditioned methods reduce

#blocks	#Subd.	δu (%)	#it.(cond.#)	Min. error	it.
16x16	4x4	0	55 (75.5)	4.54e-02	33
		0.1	55 (75.4)	4.74e-02	17
		1.0	56 (74.7)	1.10e-01	16
32x32	8x8	0	111 (295)	1.48e-02	49
		0.1	106 (296)	3.60e-02	53
		1.0	123 (342)	2.51e-01	15
64x64	16x16	0	209 (1106)	5.55e-03	152
		0.1	223 (1102)	6.76e-02	140
		1.0	419 (3322)	4.72e-01	5
128x128	32x32	0	402 (4407)	1.93e-03	230
		0.1	530 (6355)	1.50e-01	15
		1.0	failed		

Table 6.2: Additive Average method. Filtration coefficient identification with different measurement error. Exact subproblem solvers. The minimal average error and the iteration # of its achievement are given in the last column.

#blocks	#Subd.	δu (%)	#it.(cond.#)	Min. error	it.
16x16	4x4	0	52 (67.7)	4.54e-02	33
		0.1	53 (67.3)	4.68e-02	20
		1.0	52 (60.3)	1.03e-01	20
32x32	8x8	0	120 (317)	1.45e-02	55
		0.1	120 (316)	3.60e-02	53
		1.0	128 (343)	2.63e-01	34
64x64	16x16	0	256 (1273)	5.55e-03	169
		0.1	257 (1254)	6.76e-02	132
		1.0	433 (3743)	7.84e-01	204
128x128	32x32	0	525 (5458)	1.93e-03	460
		0.1	555 (5984)	1.86e-01	174
		1.0	failed		

Table 6.3: The method of Diagonal Scaling. Filtration coefficient identification with measurement error. Exact subproblem solvers. The minimal average error and the iteration # of its achievement are given in the last column.

#blocks	#Subd.	Solver type	#it.(cond.#)	Min. error	it.
32x32	4x4	exact	97 (241)	3.61e-02	79
		2 SGS	104 (249)	3.61e-02	81
64x64	8x8	exact	172 (663)	6.76e-02	106
		2 SGS	185 (711)	6.76e-02	115
128x128	16x16	exact	421 (6008)	1.81e-01	21
		2 SGS	451 (6501)	1.80e-01	24

Table 6.4: Additive Average method. Filtration coefficient identification with a measurement error of 0.1%. The exact and approximate (2 SGS iterations) subproblem solvers are used. The minimal average error and the iteration # of its achievement are given in the last column.

substantially the number of iterations needed to obtain the filtration coefficient estimates. We conclude that the Additive Average and the method of Diagonal Scaling are suitable for the filtration coefficient identification problem.

In Table 6.4 we compare the exact and the approximate subproblem solver in the Additive Average method. The subdomain sizes are 8x8 here. We see that the usage of the approximate solvers is very reasonable as the condition number and the iteration counts do not grow much compared with the exact solvers.

6.3 Conclusion

The filtration coefficient identification problem is an ill-posed problem which is difficult to solve since the solution is extremely sensitive to errors in input data. In this chapter we described an application of the Schwarz methods discussed in this thesis to the inverse problem. Since the bilinear form of the filtration coefficient identification differs from the bilinear form of our model problem, the coarse matrix is formed in a slightly different way with the Additive Average Schwarz method . We made some numerical experiments with different levels of input data error to investigate the behavior of the PCG iterations and the regularizing property of the truncated iteration loop. We conclude that the Average Schwarz methods can successfully be used for solution of problems stiffness matrix of which have a big condition number. In this way the problem can be run on parallel computers and the total number of the PCG iterations reduces.

Bibliography

[1] P. E. Bjørstad, M. Espedal, and D. Keyes, eds., *Domain Decomposition Methods in Sciences and Engineering*, John Wiley & Sons, 1997. Proceedings from the Ninth International Conference, June 1996, Bergen, Norway.

[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] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring. II*, Math. Comp., 49 (1987), pp. 1 – 16.

[4] W. L. Briggs, *A Multigrid Tutorial*, SIAM, 1987.

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

[6] ——, eds., *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1990.

[7] T. F. Chan and B. F. Smith, *Domain decomposition and multigrid algoritms for elliptic problems on unstructures meshes*, Electronic Transactions on Numerical Analysis, 2 (1994), pp. 171 – 182.

[8] M. Dryja and O. B. Widlund, *Some domain decomposition algorithms for elliptic problems*, Tech. Rep. 438 also Ultracomputer Note

155, Department of Computer Science, Courant Institute, 1989. Proceedings of the Conference on Iterative Methods for Large Linear Systems held in Austin, Texas, October 19 - 21, 1988, to celebrate the sixty-fifth birthday of David M. Young, Jr.

[9] ——, *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.

[10] R. Freund and N. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., (1991), pp. 315–339.

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

[12] R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1991. Held in Moscow, USSR, May 21–25, 1990.

[13] R. Glowinski, J. Périaux, Z. Shi, and O. B. Widlund, eds., *Domain Decomposition Methods in Sciences and Engineering*, John Wiley & Sons, February 1997. Proceedings from the Eight International Conference on Domain Decomposition Methods, May 1995, Beijing.

[14] W. D. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, 1994.

[15] P. C. Hansen, *Regularization tools. a Matlab package for analysis and solution of discrete ill-posed problems*, Tech. Rep. Report UNIC-92-03, Danish Computing Center for Research and Education, Building 305, Technical University of Denmark, DK-2800, Lyngby, Denmark, June 1992.

[16] ——, *Experience with regularizing CG iterations*, Tech. Rep. Report UNIC-94-02, Danish Computing Center for Research and Education,

Building 305, Technical University of Denmark, DK-2800, Lyngby, Denmark, May 1994.

[17] B. Hendrickson and R. Leland, *The Chaco user's guide. Version 2.0*, Tech. Rep. SAND94-2692, Sandia National Laboratories, Albuquerque, NM 87185-1110, July 1995.

[18] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand., (1952), pp. 409–436.

[19] G. Karypis and V. Kumar, *METIS, unstructured graph partitioning and sparse matrix ordering system. Version 2.0*, Tech. Rep. University of Minnesota, Department of Computer Sciences, Minneapolis, MN 55455, August 1995.

[20] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1992. Held in Norfolk, VA, May 6–8, 1991.

[21] D. E. Keyes and J. Xu, eds., *Seventh International Conference of Domain Decomposition Methods in Scientific and Engineering Computing*, AMS, Providence, RI, 1995.

[22] K. Kunisch and X.-C. Tai, *Nonoverlapping domain decomposition methods for inverse problems*, in Domain Decomposition Methods in Sciences and Engineering, P. E. Bjørstad, M. Espedal, and D. Keyes, eds., John Wiley & Sons, 1997. Proceedings from the Ninth International Conference on Domain Decomposition Metods, June 1996, Ullensvang.

[23] B. Lee and M. R. Trummer, *Multigrid conformal mapping via the szegö kernel*, Electronic Transactions on Numerical Analysis, 2 (1994), pp. 22 – 43.

[24] P. L. Lions, *On the Schwarz alternating method. I.*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Philadelphia, 1988, SIAM.

[25] ——, *On the Schwarz alternating method. II.*, in Domain Decomposition Methods, T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1989, SIAM.

[26] A. Quarteroni, Y. A. Kuznetsov, J. Périaux, and O. B. Widlund, eds., *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, vol. 157, AMS, Providence, RI, 1994.

[27] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., (1986), pp. 856–869.

[28] 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.

[29] B. Smith, P. E. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, 1996.

[30] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., (1989), pp. 36–52.

[31] A. N. Tikhonov, *On solving incorrectly posed problems and method of regularization*, Dokl. Acad. Nauk USSR, 151 (1963), pp. 501–504.

[32] E. Vainikko and G. Vainikko, *Some numerical schemes for the identification of the filtration coefficient*, Acta et comment. univ. Tartuensis, 937 (1992), pp. 90 – 102.

[33] H. van der Vorst, *Bi-CGSTAB: A fast and smoothy converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., (1992), pp. 631–644.

[34] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, 1992.

APPENDIX

Paper A:

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

Paper B:

Additive Schwarz Methods without Subdomain Overlap and with New Coarse Spaces

Paper C:

Parallel Implementation of a Schwarz Domain Decomposition Algorithm

Paper D:

Robust Additive Schwarz Methods on Unstructured Grids

Paper E:

Courant Elements for the Inverse Problem in the Filtration Coefficient Identification