Robust Additive Schwarz Methods on Unstructured Grids

Petter E. Bjørstad, Maksymilian Dryja, and Eero Vainikko

1 Introduction

The purpose of this paper is to describe a fully parallel two and three-dimensional computer implementation of the Additive Average Schwarz algorithm first described in [BDV97]. This paper draws heavily on the Ph.D. thesis work of Eero Vainikko [Vai97] and his extensive work required to design and develop a general computer implementation with direct coupling to state of art graph partitioning software such as Chaco [HL95] and MeTiS [KK95].

Our algorithm and its implementation share several interesting properties not always available in more standard domain decomposition algorithms:

- It handles unstructured grids in two and three dimensions
- It allows subdomains to be completely unstructured
- It is fully parallel and portable
- It is robust with respect to discontinuous coefficients across subdomain boundaries
- It can take advantage of inexact solvers.

There has been a strong trend within the applied computational sciences [VSB91] and in the scientific computing community to consider the use of unstructured grids and discretizations [CSZ96]. This approach often offers considerable advantages, in particular, when attacking three-dimensional problems where adaptivity and accuracy requirements may change across the computational domain.

The flexibility offered by unstructured discretizations may come with a cost in terms of more expensive computational procedures and possibly less efficient parallel implementations. Unstructured meshes have been standard in structures calculations based on finite elements since its pioneering start about forty years ago, but always dependent on direct methods of solution. Recently, the study of iterative methods for structural analysis has received more attention partly motivated by the very large problems generated by detailed three-dimensional models.

The most natural approach to domain decomposition algorithms for unstructured problems, the iterative substructure or Schur complement methods [SBG96], can be viewed as a direct line of development from the multilevel superelement technique used to organize a direct factorization method [Prz85]. In both cases the solution procedure reduces the problem to the interfaces of substructures by considering Schur complement matrices. The explicit formation of these matrices is generally avoided when using the iterative substructure approach. These methods belong to the nonoverlapping class of domain decomposition algorithms. Early work with the Neumann-Dirichlet algorithm can be found in [BW86] and more recent, very promising work related to the Neumann-Neumann algorithm can be found in [LT94], [Man93] and [TV97].

In comparison, relatively little has been reported on the use of overlapping Schwarz methods for unstructured grids. In [CSZ96], the authors use a standard (overlapping) Schwarz method and a sequence of non-matching coarser grids. In particular, a regular grid is used at the coarsest level. This choice is at least partly dictated by the standard theory for Schwarz methods which assumes that the coarse grid can be viewed as a finite element triangulation of the domain.

Our proposed algorithm, the Additive Average Method [BDV97] is an additive Schwarz method with an alternative coarse space. The method avoids overlapping subdomains and solves a special linear system on the interface variables. In this respect, the method resembles the iterative substructuring methods. Our algorithm computes average values on each subdomain. The use of average values in order to capture the coarse grid behavior and achieve almost optimal preconditioners for iterative substructuring algorithms was used in the important paper [BPS87] already ten years ago.

This paper is organized as follows. In Section 2, we quickly review the method and its basic properties. The reader should consult [BDV97] to see the proof of convergence. In Section 3, we discuss a computer implementation and show how our algorithm can be coupled to state of art graph partitioning software packages. We do not discuss computational complexity, but refer to [BDV96] where one can also find details on parallel computations with reported performance. In the last section, we report on realistic computational problems with unstructured grids. We investigate subdomain partitioning guided by material properties and report on numerical experiments carried out on different parallel machines.

2 The Method and Some Properties

Let us consider a polyhedral region $\Omega \subset R^d, d = 2, 3$, which has been divided into N subdomains. The subdivision may be based on the information about a coefficient $\rho_i > 0$ which is assumed to be constant in each subdomain but may have big jumps across subdomain boundaries. We consider an elliptic problem in the variational form:

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

$$\sum_{i=1}^{N} \int_{\Omega_i} \rho_i \nabla u^* \cdot \nabla v dx = \int_{\Omega} f v dx \quad \forall v \in V(\Omega), \tag{2.1}$$

where $V(\Omega)$ is an appropriate Sobolev space.

We assume that the region Ω is triangulated into elements $e_k,\ k = 1, 2, ..., n_{t.}$ Let V^h be a finite element space of piecewise linear, continuous functions defined on the triangulation and vanishing on $\partial\Omega$, the boundary of Ω . Our discrete problem reads:

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

$$a(u, v) = f(v) \quad \forall v \in V^h(\Omega), \tag{2.2}$$

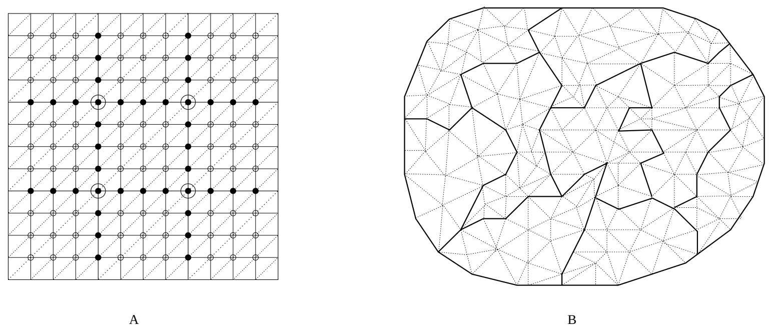
where

$$a(u, v) = \sum_{i=1}^{N} \int_{\Omega_i} \rho_i \nabla u \cdot \nabla v dx, \qquad f(v) = \int_{\Omega} f v dx. \tag{2.3}$$

In this paper we focus on the implementation and early experience with this method applied to unstructured grid problems.

The domain is divided into nonoverlapping subdomains. Figure 1 shows some key differences between a structured and unstructured grid in a domain decomposition context. On a regular grid it is straightforward to define a coarser space However,

Figure 1 An example of a regular and an irregular splitting of a domain.



there are situations where for some essential reason, like *e.g.,* geological structure or material properties there is no natural regular splitting, in fact, one may want to decompose the domain according to problem specific considerations. With our method, we are free to do this as our coarse space construction requires no regular subdomain structure.

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 $u \in V^h$ restricted to $\overline{\Omega}_i$ we define $I_A u$ as follows:

$$I_A u = \begin{cases} u(x) & x \in \partial\Omega_{ih} \\ \overline{u}_i & x \in \Omega_{ih} \end{cases} \tag{2.4}$$

where

$$\overline{u}_i = \frac{1}{n_i} \sum_{x \in \partial\Omega_{ih}} u(x), \tag{2.5}$$

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

We define by h_i the diameter of the smallest element touching the boundary $\partial\Omega_i$:

$$h_i = \inf_{j: e_j \cap \partial\Omega_i} h_j. \tag{2.6}$$

For each $V_i, i = 1, ..., N$ we introduce a bilinear form $b_i(u, v)$ on $V_i \times V_i$ of the form

$$b_i(u, v) = a(u, v). \tag{2.7}$$

An approximate bilinear form on the coarse space can be defined by

$$b_0(u, v) = \sum_i \rho_i h_i^{d-2} \sum_{x \in \partial\Omega_{ih}} (u(x) - \overline{u}_i)(v(x) - \overline{v}_i). \tag{2.8}$$

Denote the matrix obtained form the bilinear form $a(u, v)$ in (2.3) by A and let A_i $i = 1, ..., N$ denote the matrices defined by (2.7) *i.e.,* the restriction of A to the nodes of Ω_i. Similarly, the coarse grid matrix A_0 is computed from (2.8). Together with A_0 we also define another variant of the coarse matrix defined by

$$A_e = \tilde{I}_A^T A \tilde{I}_A \tag{2.9}$$

where $\tilde{I}_A^T$ is a restriction operator defined as the transpose of $\tilde{I}_A$, the matrix representation of I_A in (2.4). The matrix A_e can therefore be explicitly formed using only the definition of I_A. A_0 is an approximation of A_e, having sparse structure and it is therefore easier to use in computations. We call the use of A_e the Galerkin coarse space approximation.

Note that in the Additive Average method all the subdomain solves and the coarse problem solution can be run in parallel. Alternatively, we can define multiplicative variants of the method just as in the standard Schwarz methods. A simple, symmetric variant consists of a coarse solver followed by the subdomain solutions and another coarse solver at the end of the cycle. Residual updates must be carried out between the steps as usual.

When the Additive Average method is combined with a Krylov subspace iterative method, one can prove that the condition number of the resulting iteration matrix is independent of jumps in the coefficient ρ_i across subdomain boundaries and that it depends linearly on the ratio H/h where H is the largest subdomain diameter while

h measures the smallest diameter of an inscribed circle in an element that touches a subdomain boundary. The reader should consult [BDV97] to see a proof of convergence. Furthermore, it can be shown that the condition number depends linearly on the aspect ratio of the subdomains. Thus, in the case of a very long and thin subdomain it would be advantageous to further decompose this into smaller subdomains having a better aspect ratio.

3 Computer Implementation

Implementation of the Average Additive method as a parallel algorithm has several advantages over the standard Schwarz methods with respect to simplicity (no overlap), reduced arithmetic, and reduced communication. These issues are more fully explained in [BDV96].

The parallel implementation follows an SPMD programming model, where we assign several special tasks including the coarse space solver to processor zero. This processor will also handle code parameters and the initial triangulation of the domain that is needed to create the finite element data structures. In the regular case, the splitting of the domain into subdomains is straightforward. In the case of a nonregular domain, processor zero will create a graph where each element e_k is a vertex with edges connecting it to all vertices that correspond to element neighbors. We have coupled our code directly to the Chaco-2.0 [HL95] and the MeTiS [KK95] packages for graph partitioning. Both packages can be used to split the graph (and therefore our domain) into a specified number of pieces. The packages allow us to specify weights on the edges which can be used to supply information about the coefficients ρ_i. Ideally, this could be used to enforce a subdomain splitting where we guarantee that the ρ_i remains (near) constant in each subdomain with all the jumps across subdomain boundaries. In practice, this is somewhat difficult to achieve and more experience with the coupling of domain decomposition software and graph partitioning packages is needed.

The graph partitioning process gives us a discrete function $P : \{e_k, k = 1, ..., n_t\} \to \{1, ..., N\}$ that determines for each vertex (element) which subdomain it belongs to. In order to be able to assemble the stiffness matrix in parallel, subdomain number i is assigned information about its triangles $\{e_k : Pe_k = i\}$ together with one additional layer of the triangles around partition i. For each subdomain, we then send this information to an available processor. Next, we can perform the assembly of the stiffness matrix A in parallel. Each piece of the matrix A is finally stored where it was assembled. Each processor has also been given information about possible other subdomains that may 'own' one of its nodes. With this information the nodes can be split into sets of interior and boundary nodes respectively, and the necessary data structures for nearest neighbor as well as coarse solver communication can be prepared.

4 Numerical Experiments with Unstructured Meshes

Our first example considers an unstructured grid around an airfoil, see Figure 2. We subdivide the element mesh using the graph partitioning packages described

Figure 2 An unstructured mesh around an airfoil. The paper considers a similar mesh and two further levels of refinement.

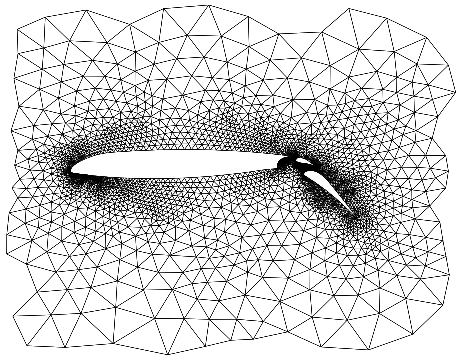


Table 1 Characteristics of the iterative method when applied to a scaled version of the airfoil mesh.

# PE	N	Dofs	$\kappa(BA)$	# Iter	Time/iter T3E	Time/iter Origin
4	3	15606	117	54	0.036	0.019
16	15	61484	636	119	0.030	0.026
28	27	244047	3000	252	***	0.059

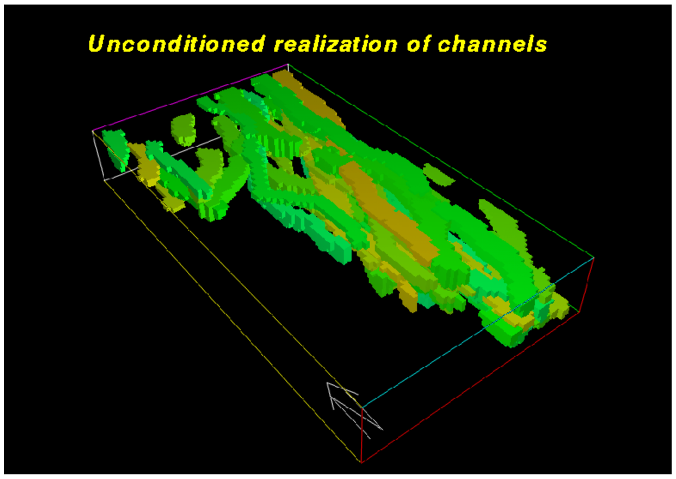
earlier. We further refine the mesh in order to scale the problem keeping roughly the same number of nodes per subdomain. In Table 1 we show iteration counts and condition number estimates for three different cases. We also show the execution time per iteration on a T3E and on an Origin-2000 computer when using only two symmetric Gauss-Seidel sweeps as our inexact solver. Observe that our code is completely unoptimized for these machines and that the execution times could change considerably in the near future. We note that the times are quite similar and that the time on the Origin scales correctly between the second and third row in the table. The T3E did not have enough memory per node to run our largest case.

We see that the condition number of our preconditioned problem does increase unlike the situation for a uniform refinement of a structured grid. A considerable part of this effect is due to our inexact solver, the entry in the second row of the table is reduced from 636 to 270 if we use an exact subdomain solver. This entry is further reduced to 165 by a simple diagonal scaling of the approximate bilinear form in (2.8). We also note that the quality of our graph partitioning (for example the subdomain aspect ratio) may change as we refine the problem and request more subdomains. The ratio H/h of our largest subdomain diameter relative to the smallest element is of the order 10^4 in this example, but tends to decrease as we refine our domain and introduce a larger number of subdomains. We plan to investigate the use of weights in the definition of our interpolation operator I_A as well as a more careful study of the quality of the mesh partitioning in order to further improve the situation.

Our last example is motivated by the oil industry. Figure 3 shows an oil reservoir model obtained from Norsk Hydro. The figure shows two distinctly different rock structures, with very different permeability. The geometry is quite irregular and complex in all three spacial dimensions. The entire domain is discretized into $55 \times 80 \times 100$ blocks and we use the permeability values to split the corresponding graph representation into two separate graphs which in turn are split into subdomains by our graph partitioning software. In this way, we obtain a highly irregular subdomain partitioning precisely tailored to our physical problem. We also split the problem in a completely regular fashion resulting in many subdomains having an internal discontinuous jump (from 1 to 1000) in the parameter ρ_i. In Table 2 we compare the results of four different experiments; our irregular subdomain partitioning combined with the Additive Average method, the same method using a regular splitting, a classical Additive Schwarz preconditioner as well as just running the conjugate gradient iteration without any preconditioning. In the three first cases we use two simple symmetric Gauss-Seidel sweeps as our inexact subdomain solver.

We first observe the large difference between the two first methods due to the internal jumps in the second method. The overlapping Schwarz method is considerably more robust, but still uses about 2.5 times as many iterations each of which is more costly than the iterations in the first method. Finally, we see that the problem requires a very large number of iterations without any preconditioning.

Figure 3 An oil reservoir model.



5 Conclusion

We have described a nonoverlapping additive Schwarz algorithm applied to unstructured grid problems. The method has interesting properties with respect to generality as well as efficient parallel implementations. Repeated refinements of an unstructured grid causes an increase in the condition number of our iteration operator. There may be many different factors contributing to this and we are currently trying

Table 2 Comparison of four different algorithms. The

Method	Decomposition	N	$\kappa(BA)$	# Iter
Additive Average	Irregular	510	694	117
Additive Average	Regular	512	169000	1610
Additive Schwarz	Regular	512	2330	292
No preconditioning	-	-	401000	2746

to identify these in order to further improve the algorithm.

We have demonstrated that an irregular domain decomposition adapted to the problem at hand can result in improved convergence of our iterative algorithm. Further work in progress includes scaling of the bilinear forms and the use of weights in the interpolation in order to improve the robustness when used on highly unstructured meshes. Our code is directly coupled to state of art graph partitioning packages. This interaction deserves more study in order to arrive at a best possible overall solution strategy.

REFERENCES

[BDV96] Bjørstad P. E., Dryja M., and Vainikko E. (December 1996) Parallel implementation of a schwarz domain decomposition algorithm. In Wasniewski J., Dongarra J., Madsen K., and Olesen D. (eds) *Applied Parallel Computing in Industrial Problems and Optimization*. Springer. Lecture Notes in Computer Science volume 1184.

[BDV97] Bjørstad P. E., Dryja M., and Vainikko E. (1997) Additive Schwarz methods without subdomain overlap and with new coarse spaces. In Glowinski R., Périaux J., Shi Z., and Widlund O. B. (eds) *Domain Decomposition Methods in Sciences and Engineering*. John Wiley & Sons. Proceedings from the Eight International Conference on Domain Decomposition Metods, May 1995, Beijing.

[BPS87] Bramble J. H., Pasciak J. E., and Schatz A. H. (1987) The construction of preconditioners for elliptic problems by substructuring, II. *Math. Comp.* 49: 1–16.

[BW86] Bjørstad P. E. and Widlund O. B. (1986) Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.* 23(6): 1093–1120.

[CSZ96] Chan T. F., Smith B. F., and Zou J. (1996) Overlapping Schwarz methods on unstructured meshes using non-matching coarse grids. *Numer. Math.* 73(2): 149–167.

[HL95] Hendrickson B. and Leland R. (July 1995) The Chaco user's guide. Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM 87185-1110.

[KK95] Karypis G. and Kumar V. (August 1995) Metis, unstructured graph partitioning and sparse matrix ordering system. version 2.0. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455.

[LT94] Le Tallec P. (1994) Domain decomposition methods in computational mechanics. In Oden J. T. (ed) *Computational Mechanics Advances*, volume 1 (2), pages 121–220. North-Holland.

[Man93] Mandel J. (1993) Balancing domain decomposition. *Comm. Numer. Meth. Engrg.* 9: 233–241.

[Prz85] Przemieniecki J. S. (1985) *Theory of Matrix Structural Analysis*. Dover

Publications, Inc., New York. Reprint of McGraw Hill, 1968.

[SBG96] Smith B. F., Bjørstad P. E., and Gropp W. (1996) *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press.

[TV97] Tallec P. L. and Vidrascu M. (1997) Generalized Neumann-Neumann preconditioners for iterative substructuring. In Bjørstad P. E., Espedal M., and Keyes D. (eds) *Domain Decomposition Methods in Sciences and Engineering.* John Wiley & Sons. Proceedings from the Ninth International Conference, June 1996, Bergen, Norway.

[Vai97] Vainikko E. (March 1997) *Robust Additive Schwarz Methods – Parallel Implementations and Applications.* PhD thesis, University of Bergen.

[VSB91] Venkatakrishnan V., Simon H. D., and Barth T. J. (September 1991) A mimd implementation of a parallel euler solver for unstructured grids. Technical report, NASA Ames, Mail stop 202A NASA Ames Research Center, Moffett Field, CA 94305. RNR Technical Report RNR-91-024.