CHAPTER 8

An Iterative Substructuring Algorithm for Problems in Three Dimensions

Barry F. Smith*

Abstract. In domain decomposition algorithms with more than a few subdomains, there is a crucial need for a mechanism to provide for global communication of information at each step of the iterative process. The convergence rate will decay rapidly with an increasing number of subdomains if communication is only between neighboring subdomains. For iterative substructuring algorithms (those domain decomposition algorithms that use nonoverlapping subdomains), the method that provides for good global communication in two dimensions does not work well for problems in three dimensions. In this paper we present an alternative approach for providing global communication that works well in three dimensions. Sample theoretical and numerical results are presented.

1. Introduction. In this paper we will discuss an iterative substructuring algorithm designed explicitly for problems in three dimensions. Our algorithm has almost optimal convergence properties for problems in both two and three dimensions. Earlier iterative substructuring algorithms designed for two dimensions have poor convergence properties when applied in three dimensions. We introduce two simple variants of the algorithm and demonstrate some sample numerical and analytic results we have obtained. Our focus is on the techniques used in the construction of the coarse problem that provides for global communication of information at each iteration. This global communication is crucial when a large number of subdomains are used. For other work on iterative substructuring algorithms, we refer to Bramble, Pasciak, and Schatz [2], [3], Dryja and Widlund [5], Keyes and Gropp [8], [9], and Smith [14], [16].

In the next section we introduce the elliptic problems we are interested in solving. This is followed by a brief review of parts of the abstract theory of Schwarz methods. We then introduce the iterative substructuring algorithm and conclude with some preliminary numerical experiments.

* Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439-4844. Email: bsmith@mcs.anl.gov. This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

2. The Problem. We begin with a scalar, second-order, self-adjoint, coercive, bilinear form $a_\Omega(u,v)$ on $\Omega \subset R^3$ and impose a homogeneous Dirichlet condition on $\Gamma_0 \subset \partial\Omega$ and Neumann boundary conditions on $\partial\Omega \setminus \Gamma_0$. In addition, we assume that the underlying elliptic operator has no zero-order terms. We wish to find $u \in H^1_{\Gamma_0}(\Omega)$ such that

$$a_\Omega(u,v) = (f,v), \quad \forall v \in H^1_{\Gamma_0}(\Omega).$$

More generally we are interested in multicomponent problems and eventually problems with non-selfadjoint differential operators. In this paper we restrict ourselves to the scalar self-adjoint class of problems.

We begin with a discretization that satisfies the usual rules for finite element triangulations. Let $V^h(\Omega)$ be the space of continuous, piecewise linear functions on the triangulation. In addition, for the construction of the preconditioner, we assume that the set of elements are partitioned into disjoint substructures Ω_i. We further assume that there exists constants c and C independent of h and H such that for all substructures $cH \leq \text{diam}(\Omega_i) \leq CH$. In the experiments reported here, the substructures will always be brick-shaped, though this is not necessary for the algorithm. The discrete problem is to find $u^h \in V^h(\Omega)$ such that

$$a_\Omega(u^h, v^h) = (f, v^h), \quad \forall v^h \in V^h(\Omega). \tag{1}$$

3. Iterative Substructuring with Many Subdomains. Schwarz methods are iterative methods for the solution of linear systems that arise from the discretization of partial differential equations. The solution space is decomposed into subspaces. Approximations to the solution are updated by projecting the error, in some appropriate inner product that approximates $a(\cdot,\cdot)$, onto these subspaces. The iterative schemes are accelerated by using the conjugate gradient method or a version for nonsymmetric problems such as GMRES. See Dryja and Widlund [6] for details and an extensive bibliography.

The key technical tool in the Schwarz analysis of domain decomposition algorithms is the so-called Lions' lemma, see Lions [10] and Nepomnyaschikh [13]. We let the solution space V^h be decomposed into subspaces V_i^h and let $b_i(\cdot,\cdot)$ be the inner product on V_i^h that approximates $a(\cdot,\cdot)$. Assume that C_0^h is the minimum value such that for all $u^h \in V^h$ there exists a representation $u^h = \sum u_i^h$ with $u_i^h \in V_i^h$, such that,

$$\sum_i b_i(u_i^h, u_i^h) \leq (C_0^h)^2 a(u^h, u^h). \tag{2}$$

Then $1/(C_0^h)^2$ is the smallest eigenvalue of the preconditioned problem. The largest eigenvalue is often easily calculated using some form of strengthened Cauchy-Schwarz inequality, (see Dryja and Widlund [6]) and generally is not much larger then one.

Iterative substructuring algorithms can be constructed using the Schwarz approach, from three basic components: subspaces associated with the interiors of the subdomains, subspaces associated with two subdomains that share a common edge (face in three dimensions), and a coarse subspace that involves the unknowns associated with the subdomain vertices (in three dimensions we may additionally have

subspaces associated with the edges shared by more then two subdomains). We refer the reader to Dryja and Widlund [6] and Dryja, Smith, and Widlund [4] for details on how the solutions from the local problems and the global coarse problem are merged at each iteration to form the approximate solution. The local problems can be solved exactly by, for instance, a sparse solver, or approximately by an iterative method such as multigrid.

In two dimensions the construction of the global coarse space is straightforward. We define the space V_0^h by the values on the subdomain vertices extended as piecewise linear functions on the subdomains. The coarse space contribution to the update to the solution is obtained by projecting the error onto the subpace V_0^h and then interpolating it back to V^h.

In three dimensions, this approach has two fundamental flaws. The first is the expense and difficulty of the piecewise linear interpolation onto the subdomains from the subdomain vertices. The more basic problem is that when nonoverlapping subdomains are used with this coarse subspace, the condition number of the preconditioned problem grows faster than $O(H/h)$. The reason for this growth can be understood by examining the bound for Lions' lemma.

In three dimensions, consider a finite element function u^h that is one at a single finite element node that is a vertex of a subdomain and zero on all other finite element nodes. The energy of u^h is approximately $\int_{\text{supp}(u^h)} 1/h^2 \sim h$. Since V_0 is the only subspace whose elements are nonzero at the vertex nodes, in the decomposition of u^h we must take u_0^h to be the interpolant of u^h. That is, u_0^h is a continuous, piecewise linear function on the subdomains that is one on a single subdomain vertex and zero on all subdomain vertices. Its energy is approximately $\int_{\text{supp}(u_0^h)} 1/H^2 \sim H$. Hence $(C_0^h)^2 \sim H/h$. To summarize, the problem in three dimensions is that interpolating the value of a finite element function from a single node can result in large changes in energy, see Smith [14], [15] for a more technical explanation.

Following an approach of Bramble, Pasciak, and Schatz [3], we construct the new coarse space V_0^h in a way that avoids interpolating the value of finite element functions from a single node. We define the space V_0^h by values on the wirebasket; these values are then extended onto the faces by the average of the values on the edges adjacent to that face. Finally, the values on the interior of the subdomain are obtained by extending the values from the faces and edges as discrete harmonic on the interior. This type of interpolation results in only small, $O(1+\log(H/h))$, changes in the energy; see Dryja, Smith, and Widlund [4].

On the subspace V_0^h we need an accurate, yet easily computed, approximation to the H^1 inner product. We approximate the inner product one subdomain at a time, namely,

$$b_0(u^h, v^h) = \sum_i b_0^{\Omega_i}(u^h, v^h).$$

We assume that for all Ω_i

$$c_i b_0^{\Omega_i}(u^h, u^h) \leq (u^h, u^h)_{H^1(\Omega_i)} \leq C_i b_0^{\Omega_i}(u^h, u^h). \tag{3}$$

Then by summing over i we obtain

$$(\min_i c_i) b_0(u^h, u^h) \leq (u^h, u^h)_{H^1(\Omega)} \leq (\max_i C_i) b_0(u^h, u^h).$$

For a subdomain with no fixed Dirichlet boundary, the null space of $(u^h, u^h)_{H^1(\Omega_i)}$ is the space of constant functions. In order for (3) to hold for $c_i > 0$, it is necessary for $b_0^{\Omega_i}(u^h, u^h)$ to also have the null space of the constant functions. We therefore use

$$b_0(u^h, v^h) = (1 + \log(H/h)) h \sum_i \min_{\bar{w}^{(i)}} \min_{\bar{y}^{(i)}} (\underline{u}_W^{(i)} - \bar{w}^{(i)} z^{(i)})^T I (\underline{v}_W^{(i)} - \bar{y}^{(i)} z^{(i)}).$$

The $z^{(i)}$ is a vector of all ones of the same dimension as $\underline{u}_W^{(i)}$, while $\underline{u}_W^{(i)}$ is the vector of coefficients of the finite element function u^h restricted to the wirebasket of subdomain Ω_i. The $(1 + \log(H/h))$ factor is needed as a technical detail in the proofs. The calculation of a projection onto the space V_0 in the $b_0(\cdot, \cdot)$ inner product chiefly involves the solution of a sparse linear system with one unknown per subdomain; see Smith [14]. The techniques used in the construction of $b_0(\cdot, \cdot)$ are similar to those used by Bramble, Pasciak, and Schatz [3] and Mandel [11], [12].

The main theoretical result obtained for this algorithm is given in Smith [14], cf. also Dryja, Smith, and Widlund [4].

THEOREM 3.1. *When exact interior solvers are used in the algorithm outlined above, the condition number of the resulting system can be bounded independently of the number of subdomains and the jumps in the coefficients of the differential equation between subdomains. The condition number, κ, grows only weakly with the number of unknowns per subdomain. In particular,*

$$\kappa \leq C(1 + \log(H/h))^2.$$

When approximate interior solvers are used, the degradation in the convergence rate is determined by the angle between the space of discrete harmonic functions and the space of approximately discrete harmonic functions; see Börgers [1] and Haase, Langer, and Meyer [7]. The best bounds obtained so far indicate that asymptotically for small subdomain size H and small mesh size h, $\log(H/h)$ multigrid V–cycle sweeps are needed for each approximate solve in order to preserve the overall convergence rate given above. In practice, at least for simple problems, $o(1)$ multigrid V–cycle sweeps are all that are needed; see Haase, Langer, and Meyer [7] and Smith [16].

4. Numerical Results. In this section we report on some numerical results for several model problems. All the calculations have been performed on an Intel iPSC/860 hypercube with 32 processors, each with 16 megabytes of memory. For additional and more detailed reports on the numerical experiments, see Smith [16].

In the first set of experiments we study the growth in the condition number as we refine the mesh. We consider the unit cube with Dirichlet boundary conditions given on one face. The cube is uniformly divided into 64 subcubes. In Table 1 we report on the condition numbers when we precondition with a simple diagonal scaling and the full algorithm discussed above. In addition, we include iteration counts and run

times for a problem with a nontrivial right-hand side. The iterations were stopped after a relative decrease in the l_2 norm of the residual of 10^{-4}. We consider three preconditioners: simple diagonal scaling, the iterative substructuring algorithm with exact interior solvers, and the iterative substructuring algorithm with one multigrid V–cycle as an approximate interior solver.

As expected, the condition number when using diagonal scaling grows in proportion to $(1/h)^2$, while the iteration count for the same preconditioner grows linearly with $1/h$. Also as expected, for the preconditioned problem the the condition number,κ, grows roughly as $(1 + \log(H/h))^2$. What is interesting is that the performance is very similar when either an exact interior solver or merely one multigrid V–cycle is used.

In the second problem we fix the number of unknowns at 857,375 and increase the number of subdomains, see Table 2. This is for a unit cube with Dirichlet boundary conditions and a mesh of $1/h = 96$ in each coordinate direction. We note that as we increase the number of subdomains the condition numbers for the fully preconditioned problem decreases. These problems were run with 32 processors. As the number of subdomains increases the amount of overhead that is related to shifting data between subdomains increases and hence the total time to solve the problem increases. Increasing the number of processors would alleviate this problem. We note that the original problem is relatively well conditioned: $\kappa \sim 3734$. Thus simple diagonal scaling is competitive and, as we can see in Table 2 can actually perform better then the other two preconditioners. Most applications, however, are not this well conditioned.

TABLE 1
Problem 1: Growth in Condition Numbers for 64 Subdomains.

		Preconditioner						
	Number of	Diagonal			Exact	1 V–cycle		
H/h	Unknowns	κ	It.	Time	κ	κ	It.	Time
8	34,848	4,980	129	5.4	16.9	16.9	17	3.9
16	270,400	19,920	262	26.6	27.3	26.7	23	14.1
20	524,880	31,125	325	49.8	31.3	31.0	25	25.3
24	903,264	48,609	388	87.6	34.9	38.8	28	46.8
32	2,130,048	79,682	522	233.4	40.9	51.0	32	130.8

In the third problem we compare some timing results using the algorithm with one multigrid V–cycle as an interior solver and simple diagonal scaling. In this problem we use a region which has the shape of a table with three legs. We prescribe Dirichlet boundary data on the top of the table only. The subdomains are rectangular bricks with aspect ratios of 4:5:20 and there are 244 subdomains. The interior problems are solved approximately with one multigrid V-cycle. These results are reported in Table 3. We observe that the full preconditioner performs much better than diagonal scaling. Both algorithms obtain reasonably good speedups as we increase the number of processors.

TABLE 2
Dirichlet Problem with 857,375 Unknowns

Number of Subdomains	H/h	Preconditioner: Diagonal κ	Diagonal It.	Diagonal Time	Exact κ	1 V-cycle κ	1 V-cycle It.	1 V-cycle Time
27	32	3,734	145	34.33	31.4	32.6	22	44.4
64	24	3,734	145	33.73	31.1	31.3	25	40.2
216	16	3,734	145	63.03	26.0	26.1	23	56.9
512	12	3,734	145	125.74	22.0	22.0	21	111.5

TABLE 3
Problem 3 with 244 Subdomains.

H/h	Number of Unknowns	Number of Processors	Diagonal (time, in sec.)	1 V-cycle (time, in sec.)
8	132,792	Number of Iterations	309	20
		Condition Numbers	19,657	20.8
		8	143.2	54.9
		16	78.2	35.9
		32	48.3	26.6
16	1,030,512	Number of Iterations	617	35
		Condition Numbers	78,486	74.1
		16	427.1	155.1
		32	237.2	88.5
20	2,000,460	Number of Iterations	772	39
		Condition Numbers	122,582	93.9
		32	453.4	157.5

It is of interest to know what part of the algorithm consumes most of the computer time during the solution process. For the problem in Table 1 with 2,130,048 unknowns, we have traced the percentage of the time in various operations when 32 processors were used. For the full preconditioner the bulk of the solution time, 72%, is spent in the multigrid code. The rest is spent in the matrix multiply, 5%; the daxpy, 4%; the inner product, 3%; and the face and coarse solvers, 2%. The time spent on communication between processors can be divided into the time spent on communications related to the inner product, 1%; the matrix multiply, 4%; and the preconditioner, 8%.

For the diagonally preconditioned problem, 25% of the time is spent on the matrix multiply, 22% on the daxpy, 18% on the inner product, and 7% on the diagonal scaling. The percentage of the total time spent on communication for the matrix multiply is 23% and it is 4% for the inner product. The application of the full preconditioner results in a decrease in the proportion of the time spent in interprocessor communication and hence a more efficient use of the machine.

We make the following preliminary conclusions. For the simplest model problems, the iterative substructuring algorithms are not competitive with simple diagonal

scaling, when **exact** interior solvers are used. Despite the decrease in the number of iterations, the large cost of the interior solvers are just too great to overcome the low cost of a diagonal scaling. When one multigrid V-cycle is used to solve the interior problems approximately, the algorithm performs almost as well as with exact interior solvers in terms of the condition number and is much faster. It is possible to find simple problems where the full preconditioner does perform better, in terms of time, than diagonal scaling. On the other hand, for the simplest possible problem, a unit cube with Dirichlet boundary conditions on the entire boundary (see Table 2), simple diagonal scaling beats this particular implementation of the iterative substructuring algorithm.

Much more work is needed to determine the best approach for variable coefficient, multicomponent problems. We expect that one multigrid V–cycle as an approximate solver will not be as effective as for simple problems. In addition, it is imperative that the problems associated with the faces and the coarse problem be adapted to the particular set of equations we are solving. Not only should the bounds depend only weakly on H and h, but they also should depend only weakly on the partial differential equation. Future work will focus on comparisons of different face preconditioners and modifications to the wirebasket problem to take the particular form of the differential equation into account.

REFERENCES

[1] C. Börgers, *The Neumann–Dirichlet domain decomposition method with inexact solvers on the subdomains*, Numer. Math., 55 (1989), pp. 123–136.

[2] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring, I*, Math. Comp., 47 (1986), pp. 103–134.

[3] ——, *The construction of preconditioners for elliptic problems by substructuring, IV*, Math. Comp., 53 (1989), pp. 1–24.

[4] M. Dryja, B. F. Smith, and O. B. Widlund, *Schwarz analysis of iterative substructuring algorithms for problems in three dimensions*, Tech. Rep., in preparation, Department of Computer Science, Courant Institute, 1991.

[5] M. Dryja and O. B. Widlund, *Some domain decomposition algorithms for elliptic problems*, in Iterative Methods for Large Linear Systems, Academic Press, San Diego, California, 1989, pp. 273–291.

[6] ——, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, held in Houston, Texas, March 20–22, 1989, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, PA, 1990, pp. 3–21.

[7] G. Haase, U. Langer, and A. Meyer, *A new approach to the Dirichlet domain decomposition method*, Tech. Rep., Technical University of Chemnitz, 1990.

[8] D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. s166–s202.

[9] ——, *Domain decomposition techniques for nonsymmetric systems of equations*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, PA, 1989, pp. 321–339.

[10] 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., SIAM, Philadelphia, PA, 1988, pp. 1–42.

[11] J. Mandel, *Iterative solvers by substructuring for the p-version finite elment method*, Comput.

Methods Appl. Mech. Engrg., 80 (1990), pp. 117-128.

[12] ——, *Two-level domain decomposition preconditioners for the p-version finite element method in three dimensions*, Int. J. Numer. Methods Engrg., 29 (1990), pp. 1095-1108.

[13] S. V. NEPOMNYASCHIKH, *Domain Decomposition and Schwarz Methods in a Subspace for the Approximate Solution of Elliptic Boundary Value Problems*, Ph.D. thesis, Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk, USSR, 1986.

[14] B. F. SMITH, *A domain decomposition algorithm for elliptic problems in three dimensions*, Tech. Rep. 519, Department of Computer Science, Courant Institute, October 1990.

[15] ——, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Courant Institute of Mathematical Sciences, September 1990. Tech. Rep. 517, Department of Computer Science, Courant Institute.

[16] ——, *A parallel implementation of an iterative substructuring algorithm for problems in three dimensions*, in preparation, Argonne National Laboratory, MCS Division, 1991.