CHAPTER 25

On Adaptive Domain Decomposition with Moving Subdomains

Calvin J. Ribbens*

Abstract. Considerable attention has only recently been given to adaptivity in the context of domain decomposition. While most of the work is based on a grid refinement strategy, we discuss here a moving grid approach to adaptive domain decomposition. The performance of two iterative methods (GMRES and Craig's method), and three domain-decomposition preconditioners, is compared in the context of our adaptation scheme. We report on a series of numerical experiments which illustrate the effect of adaptation on the performance of typical domain decomposition methods.

1. Introduction. An important measure of the maturity and relevance of a sub-area in applied mathematics is the degree to which algorithms from that sub-area become standard tools for solving "real-world" problems in science and engineering. Most of the early work in domain decomposition has been on unrealistic model problems—necessarily so, since fundamental theoretical understanding is achieved in this way. However, significant steps are being taken to extend these results and algorithms to more general problems. Instead of considering only two-dimensional, self-adjoint, scalar elliptic equations, work is now focused on non-selfadjoint operators, on systems of equations, and on time-dependent and nonlinear problems. These are important characteristics of realistic problems in science and engineering applications. Good performance in the presence of such characteristics is essential if domain decomposition is to be a method of choice for solving large PDE problems on parallel supercomputers.

Another important component of modern numerical methods for realistic problems is adaptive gridding. Whether due to complicated geometries, or to nearly singular behavior in problem coefficients or solution, the need for grid adaptation is pervasive. Domain decomposition algorithms must allow for the presence of this powerful tool in large-scale PDE calculations. Unfortunately, extensive use of highly irregular grid structures can cause problems for a family of parallel algorithms such as domain decomposition. Parallelism is obviously much easier to exploit in an algorithm featuring extremely uniform computations over uniform data structures. In fact, an important advantage of domain decomposition is that it is a convenient and high-level way to decompose a large problem into many subproblems

* Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061. Research supported by Department of Energy grant DE-FG05-88ER25068.

of similar size. Grid adaptation—particularly extensive use of local grid refinement—can do considerable damage to the parallel performance of a domain decomposition algorithm. Furthermore, convergence analysis of domain decomposition methods generally assumes a uniform decomposition. Irregular grids or decompositions make such analysis very difficult.

Careful studies of adaptation in the context of domain decomposition have only begun to appear. Gropp and Keyes [6, 7] look at local grid refinement in the context of a domain decomposition scheme, restricting the adaptation to conform to a predetermined regular tensor product decomposition of the domain into "tiles". Ewing [5] also requires that the local refinement take place uniformly within a prescribed coarse "macro-mesh". Bramble, et al. [1] consider local grid refinement in connection with a preconditioner based on domain decomposition, although they have not reported performance of a parallel implementation. Each of these papers attests to the difficulties inherent in trying to use grid adaptation in a parallel environment rewarding regularity.

Adaptive grid methods generally take one of two approaches: local grid refinement or grid motion. Hybrid schemes—in which locally refined grids move with the solution of time-dependent problems, for example—are also common. Moving grid methods are logically more regular than local refinement methods, and thus may prove more amenable to parallel implementation. A common way to view moving grid methods is in terms of a function which maps points in a curvilinear (adapted) grid on the problem domain to a uniform rectangular grid on a computational domain. By using this function as a change of variables in the original PDE, one can define an equivalent problem on the computational domain. In this way, irregularity introduced by grid adaptation is located in the transformed PDE, rather than in the grid and its associated data structures. Note that the transformed PDE operator will in general have cross-derivative terms, first derivative terms, etc., even if the original operator is very simple.

It is not possible to adapt as effectively to extremely difficult problems with a strict moving grid approach. Furthermore, many of the largest unstructured grid problems come from grids that are specially designed to conform to an irregular structure such as an airfoil or fuselage; often these grids represent a patchwork or hierarchy of grids. So a certain amount of unstructured computation will always be present, at some level. It seems likely however, that locating a considerable amount of the irregularity due to adaptation in a transformation of the PDE rather than in the grid can only help parallel performance. The idea is to preserve as much structure as possible—in grid, data structure, load balancing, etc.

In this paper we consider adaptation applied at a high level—in the choice of the decomposition itself. We use a moving subdomain strategy, defining the subdomains (rather than individual grid elements) as the image under an adaptive mapping. We then solve an equivalent transformed problem on a uniform decomposition. The result is a convenient and efficient high-level mechanism for adaptively choosing the decomposition. We are also guaranteed a balanced load, without having to resort to complicated load balancing heuristics.

Our goal is to better understand how some typical domain decomposition algorithms behave in the presence of moving grid adaptation. The remainder of the paper is organized as follows. In Section 2 we describe the preconditioned iterative solvers we want to compare. Section 3 defines a parameterized model problem, on which the numerical experiments described in Section 4 are run. We also describe in Section 3 a simple adaptive grid scheme used for these experiments. Concluding remarks appear in Section 5.

2. Preconditioned iterative methods. We briefly recall the details of the domain decomposition schemes under consideration. Since this paper is not a survey of adaptive grid

or domain decomposition methods, but rather about the interaction between such methods, it is hoped that the methods we have chosen are sufficiently representative to allow more general conclusions to be drawn. Moving grid methods can yield transformed PDE operators with very difficult characteristics (strongly non-selfadjoint, rapidly varying coefficients, etc.). Hence, it is important to have a robust combination of iterative method and preconditioner. Experience has shown that applying adaptation can result in considerably slower convergence for some preconditioned methods (see [12]).

We look first at two iterative solvers for general systems of linear equations $Ax = b$, GMRES and Craig's method. In Section 2.3 we describe two variations of a preconditioner Q for these iterative methods, based on domain decomposition. We apply the preconditioner on the right, meaning we solve $AQ^{-1}y = b$, where $Qx = y$. This requires one post-convergence preconditioner solve.

2.1. Generalized minimum residual method. The Generalized Minimum Residual (GMRES) method described by Saad and Schultz [13] is a widely-used iterative solver for nonsymmetric linear systems. It was derived as a generalization of an algorithm for indefinite symmetric systems called MINRES (see Paige and Saunders [10]). GMRES is based on Arnoldi's methods—a Gram-Schmidt-like process for computing an orthonormal basis $\{v_1, \ldots, v_k\}$ for the k dimensional Krylov space $K = \text{span}\{v_1, Av_1, \ldots, A^{k-1}v_1\}$. The initial vector v_1 is taken as $r_0/\|r_0\|_2$, where r_0 is the initial residual. The kth step of GMRES requires one matrix-vector multiply and $O(k)$ vector operations; all previous vectors $v_1, \ldots, v_k$ are needed to compute v_{k+1}. On convergence, the approximate solution x_k is found by solving a $(k+1) \times k$ upper-hessenberg least squares problem for the coefficients of x_k with respect to the Arnoldi basis. Givens rotations are performed during each step of the algorithm, so that only a triangular system solve is required in order to find these coefficients.

In practice the extra $O(kn)$ storage for the Arnoldi vectors, and the $O(kn)$ work required for the vector operations in each step, make "restarting" GMRES an attractive alternative. Hence, "GMRES(m)" refers to an algorithm in which the approximate solution is computed every m steps (unless convergence occurs sooner), and this solution is used as an initial guess for a restart of GMRES. In this way, the number of Arnoldi vectors saved at any one time ranges from 1 to m.

GMRES enjoys several attractive theoretical properties. It can be shown that the kth iterate $x_k = x_0 + z$ computed by GMRES has minimum (two-norm) residual over all z in K. An easy consequence is finite termination—in exact arithmetic, the algorithm is guaranteed to converge in n steps (one hopes for much faster convergence, of course). Furthermore, Saad and Schultz show that GMRES(m) always converges, for any m, if A is positive real. If A is indefinite, GMRES(m) can sometimes stall, failing to make any further progress. (See [8], where very large values of m were needed in order to achieve convergence). Obviously, a good preconditioner is crucial to the efficient performance of GMRES for many systems.

2.2. Craig's method. As a base-line for our comparisons, we also consider Craig's method [4], a classical conjugate gradient scheme applied to a variant of the normal equations. In particular, Craig's method solves $AA^tz = b$, where $A^tz = x$. The iterates x_k minimize the error norm $\|x - x_k\|_2$ over the translated Krylov space

$$x_0 + \text{span}\{A^tr_0, A^t(AA^t)r_0, \ldots, A^t(AA^t)^{k-1}r_0\}.$$

Each iteration of Craig's method requires two matrix-vector products—one each with A and A^t. Preconditioning requires matrix solves with Q and Q^t. Although Craig's method is theoretically applicable to any nonsingular problem, its convergence rate is governed by the

square of the condition number of A, and hence can be extremely slow. Still, with a good preconditioner, Craig's method often represents a "slow-but-sure" approach for difficult problems (see [8], for example).

2.3. Preconditioner. As preconditioner we use two variations of a block triangular approximation to the matrix A. Preconditioners of this kind, sometimes termed Schur Complement preconditioners, are widely used (see [3], [6], and [9], for example). We use a centered nine-point, second-order finite difference discretization, since a five-point stencil does not achieve second-order accuracy for cross-derivative terms. The preconditioner Q is best described in terms of a special ordering of the equations and unknowns. The idea is to order the grid points interior to a subdomain first (one subdomain at a time), then points falling on a single interface between subdomains second, and finally the "crosspoints" at the intersection of two interfaces third. The domain decomposition preconditioner is then defined in terms of approximations to the various equations, using the same ordering. In particular, let

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & \tilde{A}_{22} & A_{23} \\ 0 & 0 & \tilde{A}_{33} \end{bmatrix},$$

where the first block row of A and Q represents interior equations, the second row interface equations, and the third row crosspoint equations. Notice that A_{11} is itself block diagonal, with one block for each subdomain. A_{22} would also be block diagonal, with one block for each interface, except for interactions between unknowns on different interfaces. With a five-point stencil, A_{31} and A_{13} would be zero.

The preconditioner Q is defined as follows. The subdomain interior equations are solved exactly as part of the application of the preconditioner. We use band Gauss elimination with no pivoting. The interface equations are replaced by one of two block diagonal approximations $\tilde{A}_{22}$ described below. The crosspoint equations $\tilde{A}_{33}$ are taken as a coarse grid discretization of the original PDE, involving only the crosspoints. In this way, an application of the preconditioner requires first a crosspoint solve, then independent interface solves, and finally independent subdomain solves. In a parallel distributed-memory implementation, where each processor is responsible for one or more subdomains, the subdomain solves require no communication, the interface solves require communication with neighbors, and the crosspoint solve requires global communication.

Choosing a good interface preconditioner is crucial to the performance of preconditioners such as the one we are considering. The paper by Chan and Keyes [3] compares five different choices for a two-subdomain case. We consider two of those choices here:

Tangential (TANG). The interface equations are replaced by a "tangential" approximation in the preconditioner, where derivatives normal to a given interface are simply assumed to be zero. Gropp and Keyes [6] found TANG to work well in combination with an upper triangular preconditioner such as this.

Interface Probe (PROBE). The idea is to take $\tilde{A}_{22}$ as an approximation of the Schur complement $A_{22} - A_{21}A_{11}^{-1}A_{12}$. We set $\tilde{A}_{22} = A_{22} - E_0$, where E_0 is a diagonal matrix satisfying $E_0 v = A_{21}A_{11}^{-1}A_{12}v$ and $v = [1, \ldots, 1]^t$. E_0 can be thought of as a diagonal approximation to $A_{21}A_{11}^{-1}A_{12}$ in that it has the same row-sum as this matrix. Notice that PROBE requires one initial subdomain solve to compute $\tilde{A}_{22}$. Chan and Keyes report that PROBE is generally the most robust of the five they tested. This method is called the modified schur complement method in [9].

TABLE 1
Iterations and time for Craig's method and GMRES, with $\alpha = 100$, $h^{-1} = 64$, $H^{-1} = 8$.

	CRAIG				GMRES			
β	TANG		PROBE		TANG		PROBE	
0	101	53	337	174	26	11	38	17
1	100	52	361	186	26	11	37	17
2	109	57	398	205	28	12	39	18
3	126	66	449	232	31	13	39	18
4	151	78	498	257	35	15	41	19
5	190	98	578	297	38	17	43	20
6	234	121	694	358	41	19	45	21
7	302	156	836	430	46	22	48	23
8	443	228	1109	570	57	27	54	26
9	963	494	1874	961	75	34	67	30

3. Model problem and adaptation scheme. Having stressed in Section 1 the importance of developing domain decomposition algorithms for realistic problems, we now proceed to define a completely arbitrary model problem for testing purpose. This inconsistency is endured for the moment so that we may first understand the behavior of the methods described above in a controlled, easily tuned, setting. In particular, the numerical experiments summarized in Section 4 are all based on the elliptic equation

$$a(x,y)u_{xx} + u_{yy} = f(x,y),$$

where the coefficient $a(x,y)$ and true solution $u(x,y)$ are given by

$$\begin{aligned} a(x,y) &= \exp\left[-\alpha((x-.1)^2 + (y-.1)^2)\right], \\ u(x,y) &= \exp\left[-100((x-0.1)^2 + (y-0.1)^2)\right] x(x-1)y(y-1). \end{aligned}$$

We impose Dirichlet boundary conditions on the unit square and choose f so that the true solution is as indicated. Note that both the solution and the coefficient of u_{xx} have peaks at $(.1,.1)$, with the strength of the peak in $a(x,y)$ adjusted by the parameter α.

It is not difficult to construct an adaptive mapping for a problem such as this, where the area of difficulty is located near a single point. We use a method (*point a priori* [11]) which moves grid points toward a specified point in the domain, and does so with "strength" β. Larger values of β correspond to a stronger attraction toward the given point. In the experiments reported below, the effect of varying the problem parameters α and β is studied.

4. Numerical results. Tables 1–4 give data summarizing various experiments using the methods described above. We use h to refer to the grid spacing of the global grid; H refers to the resolution of the subdomains (e.g., $H^{-1} = 4$ indicates a 4×4 subdomain decomposition). The iterative solvers are considered to have converged when the initial residual is reduced by 10^{-5}, with an initial guess of zero. We restart GMRES after 50 iterations (i.e., this is GMRES(50)). All data was collected on a DECStation 5000 running Ultrix 4.1, using double precision and the MIPS Fortran compiler *f77*. In the tables, where iterations and time are reported, the left column is iterations, and the right is time in seconds.

Table 1 gives iterations and time to convergence for the two iterative solvers using both versions of the preconditioner, for various choices of the adaptation parameter β. The

TABLE 2
Time for Gauss Elimination, and iterations and time for GMRES, with $\alpha = 100$, $H/h = 4$.

		$\beta = 0$				$\beta = 5$			
h^{-1}	GE	TANG		PROBE		TANG		PROBE	
16	0.13	17	0.30	17	0.28	21	0.38	18	0.32
32	1.41	19	1.65	23	1.95	29	2.60	29	2.57
64	18.09	19	7.47	26	10.33	39	17.30	44	20.08
128	333.25	19	33.72	27	49.25	48	105.47	62	128.30

TABLE 3
Iterations and time for GMRES, with $\alpha = 100$, $h^{-1} = 128$.

	$\beta = 0$				$\beta = 5$			
H^{-1}	TANG		PROBE		TANG		PROBE	
2	5	86	7	95	25	153	21	143
4	31	101	34	112	40	131	41	137
8	36	83	62	147	49	123	70	163
16	27	49	44	93	52	114	70	142
32	19	34	27	50	48	107	62	129
64	11	43	16	53	35	107	45	139

problem size is fixed. It is evident that Craig's method is not competitive, especially when preconditioned with PROBE. The convergence rate of Craig's method is just too slow for these problems. With no preconditioning (not reported here) CRAIG requires nearly n iterations to converge, while GMRES still converges in much less than n iterations. Table 1 also points strongly to a deterioration in convergence for all methods as β grows. Of all the methods, the PROBE preconditioned GMRES seems to suffer the least from increasingly strong adaptation, but its advantage over TANG is small.

In Table 2 we summarize results when the problem size increases, with the relative resolution H/h fixed at 4. We also include time for a direct solve on this problem, using band gauss elimination. The TANG preconditioner is clearly better when there is no adaptation ($\beta = 0$); for $\beta = 5$ the advantage is not nearly as strong. Notice also that the number of iterations remains relatively unchanged as h^{-1} grows if $\beta = 0$, but that it grows steadily if $h^{-1} = 5$. A recent report of Cai, et al. [2] describes a related algorithm that could improve this situation. Their method requires two subdomain solves per iteration, but it achieves a very slow growth in the number of iterations as h is refined (assuming H/h is constant). The theory in [2] requires large H if the problem is strongly non-selfadjoint however.

The effect of varying the number of subdomains, with global grid resolution h^{-1} fixed, is illustrated in Table 3. The results are somewhat counter-intuitive. As expected, the number of iterations first increases with H^{-1} as the subdomains become smaller, but eventually it begins to decrease as the size of the crosspoint system becomes larger. However, one also expects that the total time needed to converge would initially decrease with increasing H^{-1}, reach a minimum, and then begin to increase again. This pattern is not repeated here. Part of the explanation seems to lie in the accidents of where the peak in $a(x, y)$ happens to lie with respect to the decomposition. One of the potential advantages of our adaptive domain decomposition approach is that a point singularity, such as is found in our model problem, may be located in a single subdomain, so that the preconditioner essentially solves this region exactly. If one is not careful however, the singularity may fall on an interface or at a crosspoint, in which case the preconditioner is likely to have difficulty. Note also that

TABLE 4
Iterations and time for GMRES, with $h^{-1} = 128$, $H^{-1} = 32$.

	$\beta = 0$						$\beta = 5$					
α	TANG		PROBE		ILU		TANG		PROBE		ILU	
0	19	34	29	54	74	169	65	136	83	172	146	352
1	19	34	28	52	70	160	57	123	74	152	138	327
10	19	34	28	52	57	135	49	110	64	133	111	263
100	19	34	27	50	57	135	48	107	62	130	105	253
1000	18	32	27	50	58	136	47	104	62	130	104	252

the minimum sequential time is achieved at $H^{-1} = 32$. The results would be different in a distributed-memory environment, where a large crosspoint system is to be avoided. Further investigation of this algorithm on such systems is needed.

Finally, Table 4 shows an interesting effect due to varying α, the parameter that adjusts the strength of the peak in the PDE coefficient of u_{xx}. We see that performance actually improves with increasing α. This may be due to the fact that a larger α in this problem serves more to localize the nearly singular behavior than to increase the strength of singular behavior throughout the problem domain. This allows the preconditioners to perform better. We have included here results for an Incomplete LU Factorization (ILU(0)) preconditioner as well. ILU is one of the most popular preconditioners for Krylov subspace methods in many application areas. We see that the domain decomposition preconditioners are superior for our applications. It is generally easier to parallelize the domain decomposition preconditioners as well.

5. Conclusions. We have studied the convergence behavior of two iterative methods and two domain-decomposition preconditioners, in the presence of moving grid adaptation of various degrees. A moving subdomain adaptive approach is attractive in that it provides a convenient high-level way to achieve a certain amount of adaptation without sacrificing a regular, uniform decomposition. Preconditioners based on domain decomposition can be especially effective in this situation if the subdomains are defined so that areas of difficulty in the problem are located in a single subdomain. If this is possible, then the preconditioner has the advantage of doing essentially an exact solve in that subdomain. A purely matrix based preconditioner such as ILU does not have this advantage.

Of the methods considered here, GMRES with the tangential interface preconditioner is generally best, although there is some evidence that the interface probe preconditioner is more robust in the presence of strong adaptation. The performance of the various methods is still too sensitive to the strength of the adaptation, however. A more robust combination of iterative solver and preconditioner should be found, so that the characteristics of the transformed PDE operator (large non-selfadjoint terms, cross-derivative terms, etc.) do not cause such serious problems. These concerns apply to more than just the adaptive grid framework too, since even without adaptation such problem characteristics occur.

REFERENCES

[1] J. H. BRAMBLE, R. E. EWING, J. E. PASCIAK, AND A. H. SCHATZ, *A preconditioning technique for the efficient solution of problems with local grid refinement*, Comput. Meths. Appl. Mech. Engrg., 67 (1988), pp. 149–159.

[2] X.-C. CAI, W. D. GROPP, AND D. E. KEYES, *Convergence rate estimate for a domain decomposition method*, Tech. Rep. RR-827, Yale University, 1990.

[3] T. F. Chan and D. E. Keyes, *Interface preconditionings for domain-decomposed convection-diffusion operators*, in Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Philadelphia, 1990, SIAM, pp. 245–262.

[4] E. J. Craig, *Iteration procedures for simultaneous equations*, PhD thesis, M.I.T., 1954.

[5] R. E. Ewing, *Domain decomposition techniques for efficient adaptive local grid refinement*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., Philadelphia, 1989, SIAM, pp. 192–206.

[6] W. D. Gropp and D. E. Keyes, *Domain decomposition with local mesh refinement*, Tech. Rep. RR-726, Yale University, 1990.

[7] ———, *Parallel performance of domain-decomposed preconditioned krylov methods for pdes with adaptive refinement*, Tech. Rep. RR-773, Yale University, 1990.

[8] K. M. Irani, M. P. Kamat, C. J. Ribbens, H. F. Walker, and L. T. Watson, *Experiments with conjugate gradient algorithms for homotopy curve tracking*, SIAM J. Optim., (1991). To appear.

[9] D. E. Keyes and W. D. Gropp, *Domain decomposition for nonsymmetric systems of equations: examples from computational fluid dynamics*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, eds., Philadelphia, 1989, SIAM, pp. 321–339.

[10] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[11] C. J. Ribbens, *A priori grid adaption strategies for elliptic pdes*, in Advances in Computer Methods for Partial Differential Equations VI, R. Vichnevetsky and R. Stepleman, eds., New Brunswick, N.J., 1987, IMACS, pp. 102–107.

[12] ———, *A moving mesh scheme for adaptive domain decomposition*, in Unstructured Scientific Computation on Multiprocessors, P. Mehrotra, J. Saltz, and R. Voigt, eds., Cambridge, 1991, MIT Press. To appear.

[13] Y. Saad and M. H. Schultz, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.