The Full Domain Partition Approach for Parallel Multigrid on Adaptive Grids

William F. Mitchell*

Abstract

The combination of adaptive refinement, multigrid and parallel computing for solving partial differential equations is considered. In the full domain partition each processor contains a partition of the grid plus the minimum number of additional coarse elements required to cover the whole domain. A parallel multigrid algorithm using the full domain partition is presented. The method obtains multigrid rates of convergence while communicating between processors only twice per V-cycle.

1 Introduction

The numerical solution of partial differential equations (PDEs) is the most computationally intensive part of solving mathematical models with many important applications. For this reason, much research has been performed to find faster methods to solve PDEs at higher resolution. On sequential computers, multilevel adaptive methods, i.e. methods that combine adaptive grid refinement and full multigrid, have been shown to have optimal efficiency for many classes of PDEs [1, 2, 4, 6, 8, 9]. However, effective implementation of these techniques on *parallel* computers is still not understood. Considerable research has been done to parallelize the individual components, but combining these results to form a parallel multilevel adaptive method is still in the research phase.

In this paper we consider a parallel multilevel adaptive method in the context of a Single Program Multiple Data (SPMD) programming model in a high-latency/low-bandwidth parallel architecture such as a cluster of workstations. In this environment the communication cost is high compared to the computation cost, so the frequency and size of messages between the processors must be kept small. We will focus on the finite element discretization of a two dimensional PDE using triangular elements, but the concepts presented here should be extendible to other solution techniques.

In the usual SPMD approach to parallelizing a PDE solver, the domain is partitioned into a number of sets equal to the number of processors. (In this paper the term "processor" should be interpreted as "virtual processor", since it is often advantageous to have each physical processor assume the role of more than one "processor".) Each processor is responsible for the data and computations of one set from the partition, and usually also contains shadow copies of grid entities that are outside but near its partition. The processors periodically update the shadow information through messages sent from the owner of the data to the processors that have shadow copies.

In the typical approach, shadow vertices are created for the vertices that are immediate neighbors of the vertices on the boundary of the partition. This provides a local copy

*National Institute of Standards and Technology, Gaithersburg, MD.

of the data required by computations on this processor. In the *full domain partition* (FuDoP) approach the shadow vertices extend over the full domain with the number of them decreasing exponentially as the distance from the boundary of the partition increases.

In Section 2 the FuDoP and FuDoP with extended overlap are defined. Section 3 presents a multigrid algorithm that uses FuDoP to achieve nearly the same rate of convergence as the sequential algorithm while sending messages only twice per V-cycle. Numerical computations examining the multigrid rate of convergence are presented in Section 4.

2 FuDoP Distribution

The Full Domain Partition (FuDoP) is a method of distributing a partitioned adaptive grid over the processors of a parallel computer. In particular, it determines how much additional grid should be placed on each processor. The approach is briefly described here. A more detailed description and analysis of the approach can be found in [7].

A nonuniform grid is generated by an adaptive refinement algorithm (see, for example, [5]) and partitioned into a number of sets equal to the number of processors. Each processor contains the triangles in one partition of the grid. Then the FuDoP distribution adds the minimum number of additional triangles required to cover the whole domain. The additional triangles are obtained from the coarser refinement levels. The resulting grids are as if the processor performed adaptive refinement with zero as the error indicator outside the region of its partition, so triangles outside that region are refined only as required for compatibility. (A triangulation is said to be compatible if the intersection of two triangles is either empty, a common vertex or a common side.) In fact, this is a reasonable way to implement the generation of FuDoP grids. Some triangles and vertices may be replicated over several processors, but each has a unique owner which contains the definitive data should there be any discrepancies. Figure 1 illustrates an example of FuDoP grids. The top part of the figure shows an adaptive grid that has been partitioned into four sets, with the colors indicating the owners of the vertices. The remaining four parts of the figure show the FuDoP grid that is contained on each of the four processors.

Another interpretation of FuDoP is through domain decomposition with overlapping subdomains. Figure 2 illustrates a FuDoP grid on the red processor separated into refinement levels. The non-red triangles around the boundary are triangles owned by other processors. The grids illustrated in this figure are the grids of the multigrid solution method. When viewed level-by-level it is clear that, on each level, the additional triangles of the FuDoP grid form a thin boundary on the sides that are adjacent to other partitions. Thus it looks like domain decomposition with overlapping subdomains on a level-by-level basis. This is similar to a method proposed by Brandt and Diskin [3] for parallel multigrid with finite differences on a uniform grid.

In its simplest form the FuDoP approach does not, however, provide the overlap of traditional overlapping Schwartz domain decomposition methods [10]. The overlap is traditionally defined by including all vertices that are neighbors of vertices in the subdomain. Two levels of overlap is achieved by including all vertices that are neighbors of neighbors, and higher levels of overlap are defined similarly. Figure 3 illustrates a single level of the FuDoP grid for the "red" processor, and the same section of the complete grid. It can be seen that the FuDoP grid is not guaranteed to contain all the neighboring points, as several neighboring points owned by the "green" processor are missing. To achieve higher levels of overlap, the FuDoP approach can be extended by enforcing the requirement that

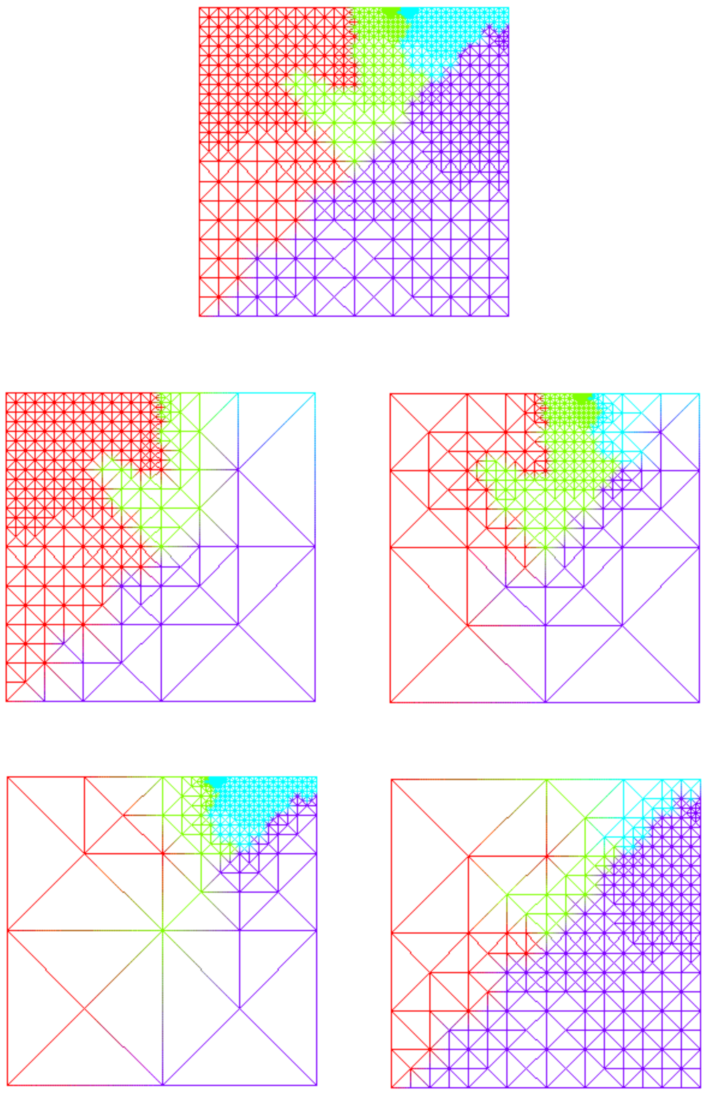

FIG. 1. *An adaptive grid partitioned for four processors and the corresponding FuDoP grids.*

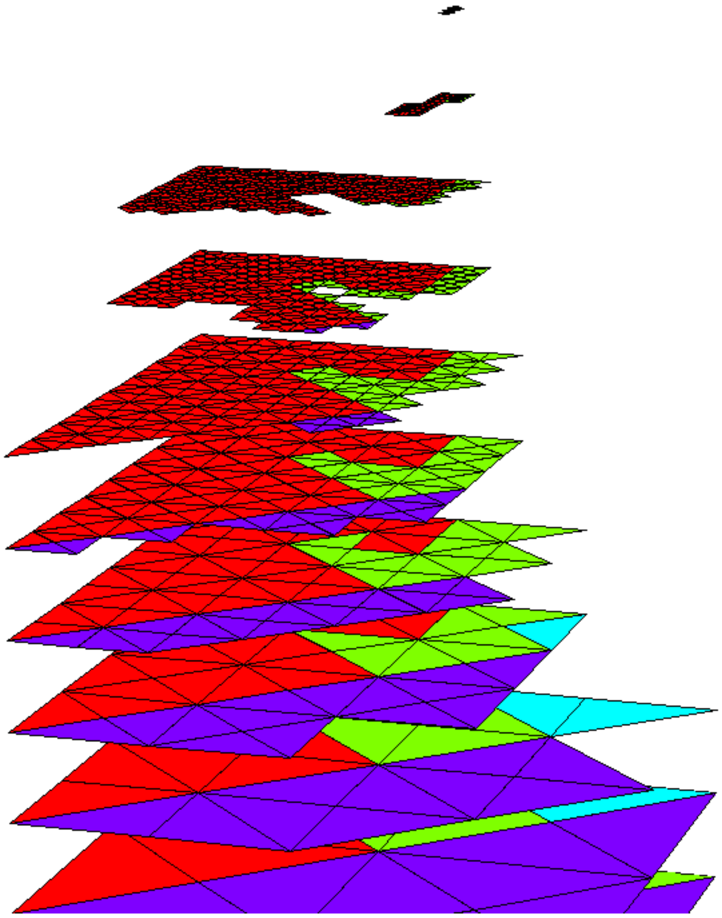

FIG. 2. *The individual levels of a FuDoP grid.*

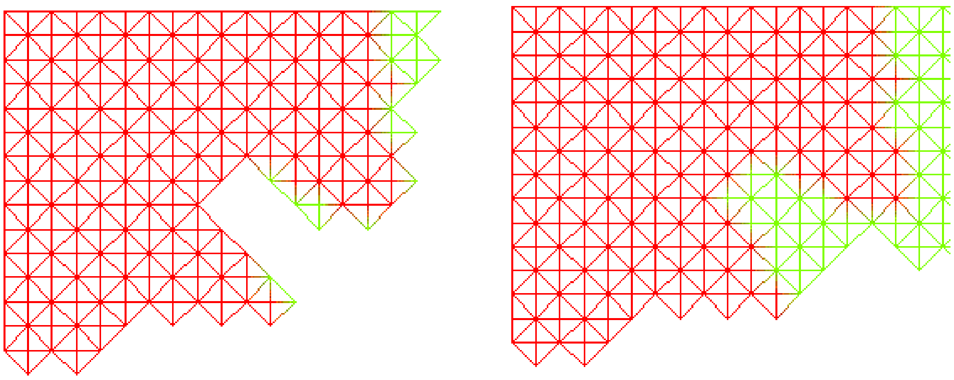

FIG. 3. *Illustration of missing vertex neighbors.*

all vertices within a distance k of the partition be included. We refer to this as FuDoP with k levels of extended overlap, FuDoP(k). The nonextended FuDoP is said to have 0 levels of overlap, FuDoP(0), even though a considerable amount of overlap exists. The extended overlap can be generated by refining triangles to generate the vertices within k steps of the boundary, and any triangles required for compatibility.

3 Multigrid with FuDoP

In this section the use of FuDoP for parallelizing the multigrid method in [6] is described. This is a FAS (full approximation scheme) multigrid method [2] based on the hierarchical basis [11] over grids that were adaptively refined using the newest node bisection of triangles [5].

The sequential form of one V-cycle of the algorithm can be briefly described as follows. The l^{th} grid in the sequence of grids is obtained by using the first l levels of refinement in the adaptive grid. When working on level l, the points that are in grid $l-1$ are referred to as black points, and those that are in grid l but not in grid $l-1$ are referred to as red points. With the rows and columns ordered such that the black points come before the red points, the equations formed by using a two level hierarchical basis can be written

$$\begin{bmatrix} A_{11} & A_{12}^T \\ A_{12} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Note that the coarse-coarse block, A_{11}, of the two level hierarchical matrix is the nodal matrix of the coarse grid. With relaxation referring to a Gauss-Seidel sweep, the V-cycle can be described as:

Algorithm Sequential V-Cycle

1. relaxation at red points
2. convert from nodal basis to two level hierarchical basis
3. recursively solve the coarse grid problem $A_{11}x_1 = b_1 - A_{12}^T x_2$
4. convert back to nodal basis
5. relaxation at red points
6. relaxation at black points that have a red neighbor

On the coarsest grid, the equation is solved exactly.

The FuDoP grids provide a means of adapting this algorithm for parallel computation. Since the FuDoP approach provides each processor with a compatible, hierarchical adaptive grid that covers the whole domain, the multigrid algorithm can be run on each processor in parallel. Obviously, some level of communication will be necessary, since the individual FuDoP grids do not contain sufficient resolution over the entire domain. Numerical studies (see Section 4) indicate that it is sufficient to send messages only twice during a V-cycle to maintain nearly the same rate of convergence as the sequential algorithm. These messages occur at the top and bottom of the V-cycle.

Recall that each vertex is owned by one processor, which has the definitive solution value at that vertex. Other processors may compute slightly different values because of the lack of resolution in the grid around the vertices they do not own. It is, therefore, important for processors to periodically send the solution values of the vertices it owns to the processors that have a copy of that vertex. This occurs at the bottom (before solving exactly on the coarsest grid) and top (after the relaxation on the finest grid) of each V-cycle.

In order to keep the parallel algorithm as close to the sequential algorithm as possible, information about the modifications to the right hand side must also be sent. The term $A_{12}^T x_2$ in the definition of the coarse grid problem contains the information that allows the coarse grid solver to produce fine grid accuracy. With the FuDoP grid, many of the fine grid points contained in the support of a hierarchical basis function associated with a coarse grid point are missing, because they are owned by a different processor. This leads to missing columns in A_{12}^T, and a less accurate representation of the fine grid information in the right hand side of the coarse grid problems. To reduce this effect, each processor keeps track of the changes in the right hand side of equations that it owns and which are copied on other processors. At the bottom of the V-cycle, these changes are sent to the other processors. This provides the correct right hand side during the upward half of the V-cycle. The effect of the changes during the downward half of the V-cycle are ignored to reduce communication.

So the modifications to algorithm Sequential V-Cycle for the parallel V-cycle algorithm are:

1. Keep track of the modifications to the right hand side due to $A_{12}^T x_2$.
2. Send those modifications and the updated solution values at the bottom of the V-cycle.
3. Send the updated solution values at the top of the V-cycle.

4 Numerical Results

Numerical computations were performed to examine the rate of convergence of the multigrid method with FuDoP. Tables 1 and 2 present the results for uniform grids of approximately 1000 and 64000 vertices, respectively, using from 1 to 16 processors and with extended overlaps of 0, 1 and 2. Laplace's equation was solved on the unit square with Dirichlet boundary conditions defined such that the solution is $u(x, y) = x + y$. The finite element method with linear elements solves this problem exactly, so convergence of the algebraic error can be examined without interference from the discretization error. The initial guess for the solution is set to 1.0 for all internal vertices and the energy norm of the error is measured. Ten iterations of the V-cycle are performed. After each cycle the energy norm of the error is measured and the contraction factor, i.e. the ratio of the error after the cycle to the error before the cycle, is computed. The reported value is the maximum of the ten computed contraction factors.

Tables 3 and 4 present similar results for nonuniform grids. Here the boundary conditions for Laplace's equation are piecewise linear across the top boundary, with a discontinuity in the derivative at $x = 0.7$, and zero on the other three sides. This problem generates the adaptive grid of Figure 1. Since the exact solution is not known, the algebraic error is computed by comparing the solution after each cycle with the solution obtained by fifteen V-cycles.

The following observations can be made from these tables:

1. There is no significant difference between the small grid and the large grid, illustrating the independence of the multigrid rate of convergence with respect to grid size.
2. The rate of convergence with FuDoP(0) deteriorates with the number of processors, but is still bounded away from 1.0 as a function of grid size.

Table 1

Contraction factors for a uniform grid with 1089 vertices.

	Processors				
Overlap	1	2	4	8	16
FuDoP(0)	.091	.177	.180	.277	.287
FuDoP(1)	.091	.099	.102	.099	.099
FuDoP(2)	.091	.099	.100	.099	.098

Table 2

Contraction factors for a uniform grid with 66049 vertices.

	Processors				
Overlap	1	2	4	8	16
FuDoP(0)	.095	.189	.203	.189	.244
FuDoP(1)	.095	.108	.110	.109	.107
FuDoP(2)	.095	.109	.111	.108	.114

Table 3

Contraction factors for an adaptive grid with about 1000 vertices.

	Processors				
Overlap	1	2	4	8	16
FuDoP(0)	.090	.254	.271	.272	.296
FuDoP(1)	.090	.095	.099	.102	.115
FuDoP(2)	.090	.095	.100	.103	.115

Table 4

Contraction factors for an adaptive grid with about 60000 vertices.

	Processors				
Overlap	1	2	4	8	16
FuDoP(0)	.093	.276	.276	.285	.285
FuDoP(1)	.093	.096	.101	.117	.148
FuDoP(2)	.093	.099	.109	.105	.126

3. FuDoP(1) has nearly the same rate of convergence as the sequential algorithm, and does not degrade with the number of processors.

4. FuDoP(2) does not provide any significant improvement over FuDoP(1).

5. There is little difference between the rate of convergence on nonuniform grids and uniform grids.

5 Conclusions

The Full Domain Partition with extended overlap is an easily generated extension of the partitions of an adaptive grid that, level by level, provides a small overlap with the other partitions. When used with a multigrid method, it provides a natural domain decomposition with overlapping subdomains on each level. Numerical computations demonstrate that, with FuDoP(1), a parallel implementation of a standard multigrid method for elliptic partial differential equations can obtain nearly the same rate of convergence as the sequential method, independent of the number of processors, with only two communication steps per V-cycle. Theoretical analysis of the convergence rate is currently under investigation.

References

[1] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, volume 15 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, 1994.

[2] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333-390.

[3] A. Brandt and B. Diskin, *Multigrid solvers on decomposed domains*, in Proceedings of the Sixth International Conference on Domain Decomposition Methods, A. Quarteroni, ed., AMS, Providence, 1994, pp. 135-155.

[4] S. F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, volume 6 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, 1989.

[5] W. F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Soft., 15 (1989), pp. 326–347.

[6] W. F. Mitchell, *Optimal multilevel iterative methods for adaptive grids*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 146–167.

[7] W. F. Mitchell, *The full domain partition approach to distributing adaptive grids*, presented at Grid Adaptation in Computational PDEs: Theory and Applications, 1996, submitted to proceedings.

[8] M.-C. Rivara, *Design and data structure of fully adaptive, multigrid, finite-element software*, ACM Trans. Math. Soft., 10 (1984), pp. 242–264.

[9] U. Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, volume 13 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, 1993.

[10] B. Smith, P. Bjørstad and W. Gropp, *Domain Decomposition*, Cambridge University Press, Cambridge, 1996.

[11] H. Yserentant, *On the multi–level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.