Parallel Computing 12 (1989) 131–144
North-Holland

Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Basic ideas *

L. HART and S. McCORMICK

Center for Applied Parallel Processing and Computational Mathematics Group, The University of Colorado at Denver, 1200 Larimer Street, Denver, CO 80204, U.S.A.

Received October 1987
Revised December 1988

Abstract. Several mesh refinement methods exist for solving partial differential equations that make efficient use of local grids on scalar computers. On distributed memory multiprocessors, such methods benefit from their tendency to create multiple refinement regions, yet they suffer from the sequential way that the levels of refinement are treated. The asynchronous fast adaptive composite grid method (AFAC) is developed here as a method that can process refinement levels in parallel while maintaining full multilevel convergence speeds. In the present paper, we develop a simple two-level AFAC theory and provide estimates of its asymptotic convergence factors as it applies to very large scale examples. In a companion paper, we report on extensive timing results for AFAC, implemented on an Intel iPSC hypercube.

Keywords. Fast adaptive composite grid method (FAC), partial differential equations, asynchronous FAC, distributed memory multiprocessors.

1. Introduction

Developed originally in 1983 [5], the fast adaptive composite grid method (FAC) is a multilevel scheme that nominally uses global and local uniform grids for adaptive solution of partial differential equations. As with other such methods, it provides some parallelism by typically producing several independent refinement regions, but is hampered by the need to handle the refinement levels sequentially. In this paper, we introduce an asynchronous version of FAC, called AFAC, that overcomes this difficulty; AFAC allows for processing of the refinement levels in a parallel mode (simultaneous or asynchronous) without significant loss of performance. The development of AFAC will be based first on viewing FAC as a block Gauss–Seidel method applied to a particular singular system of equations; AFAC will then be introduced as a block Jacobi scheme applied to this system that has been modified to remove the singularity. This version of AFAC is actually the simultaneous one, which is easiest to analyze and most suitable for current applications; the asynchronous version corresponds to chaotic block relaxation of the modified system.

* This work was supported by the Air Force Office of Scientific Research under grant number AFOSR-86-0126 and the Department of Energy under grant number DE-AC03-84ER.

FAC has been described and analyzed both numerically and analytically in earlier papers (cf. [2,3,5,8]). Here we will include material from earlier work only as it is essential to our development of asynchronous versions of FAC, most of which is given in Section 2. In Section 3, we discuss the impediments to full parallelization of FAC and possible cures, leading to AFAC, which is described in Section 4. In Section 5, we mention briefly how AFAC could be implemented on a hypercube; in Section 6, we discuss a few advanced concepts; in Section 7, we develop a two-grid theory for AFAC; and in Section 8, we demonstrate its potential by estimating convergence factors as it applies to very large scale examples.

This paper is a revised and expanded version of one that appeared in the preliminary proceedings of The Third Copper Mountain Conference on Multigrid Methods, April 6–10, 1987. A companion paper [7] will report on AFAC in conjunction with multigrid [1] as the local solver and a special load balancing scheme [6] implemented on an Intel iPSC hypercube.

2. FAC

The purpose of this section is to describe the basic FAC scheme in order to provide a foundation for the subsequent development of AFAC. We will therefore keep the description simple and the discussion to a minimum. We first describe a basic linear two-level process. (Here the term 'level' refers to a collection of uniform grids with the same mesh size.) For simplicity, we restrict our attention to the two-dimensional unit square and to uniform rectangular grids. Most of our discussion will focus on pre-adapted grids, although we briefly mention the self-adaptive case in Section 6.

Let $\mathcal{H}^1$ and $\mathcal{H}^2$ be appropriate spaces of functions defined on the respective closed and open unit square in $\mathcal{R}^2$ and suppose $\mathcal{L}: \mathcal{H}^1 \to \mathcal{H}^2$ is an appropriate linear partial differential operator. To simplify discussion, we have implicitly restricted our attention to linear homogeneous boundary conditions. Given ϕ in $\mathcal{H}^2$, consider the problem

$$\mathcal{L}(\psi) = \phi, \quad \psi \in \mathcal{H}^1. \tag{2.1}$$

Let

$$\Omega^1 = \left\{ \left[\frac{i}{m_x^{(1)}}, \frac{j}{m_y^{(1)}} \right] \middle| i = 0, 1, \ldots, m_x^{(1)}, \; j = 0, 1, \ldots, m_y^{(1)} \right\}$$

be a (*global*) uniform grid defined on the unit square, where $m_x^{(1)}$ and $m_y^{(1)}$ are positive integers. Suppose Ω^2 is a *local* uniform grid with smaller mesh sizes and whose boundaries align with grid lines of Ω^1. Assuming that the mesh sizes of Ω^2 are $1/(pm_x^{(1)})$ and $1/(pm_y^{(1)})$ for some integer $p \geq 2$, then we may write

$$\Omega^2 = \left\{ \left[\frac{i_0}{m_x^{(1)}} + \frac{k}{pm_x^{(1)}}, \frac{j_0}{m_y^{(1)}} + \frac{l}{pm_y^{(1)}} \right] \middle| k = 0, 1, \ldots, m_x^{(2)}, \; l = 0, 1, \ldots, m_y^{(2)} \right\}.$$

Here $m_x^{(2)}$ and $m_y^{(2)}$ are positive multiples of p. Letting $i_1 = i_0 + m_x^{(2)}/p$ and $j_1 = j_0 + m_y^{(2)}/p$, then the southwest and northeast corners of Ω^2 are $[i_0/m_x^{(1)}, j_0/m_y^{(1)}]$ and $[i_1/m_x^{(1)}, j_1/m_y^{(1)}]$, respectively. We assume that $0 \leq i_0 < i_1 \leq m_x^{(1)}$ and $0 \leq j_0 < j_1 \leq m_y^{(1)}$. (See Figs. 1 and 2.)

The presumptions now are: we want to approximate values of the solution, ψ, of (2.1) at points of both Ω^1 and Ω^2; we want the increased *resolution* of Ω^2 to provide a commensurate increase in *accuracy* on Ω^1 as well as Ω^2; we want as much of the computation as possible to consist only of operations on the uniform grids; and we would like a process that could be essentially as efficient as multigrid might be with local grids present. FAC provides this capability by way of the *composite grid*, Ω, whose *interior* points consist of Ω^1_{int} and Ω^2_{int}, the

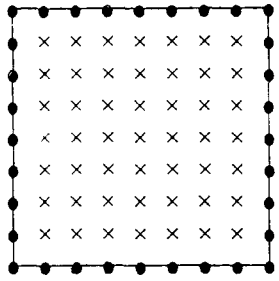

Fig. 1. Coarse grid: $m_x^{(1)} = m_y^{(1)} = 8$, ● boundary points, ⊗ interior points.

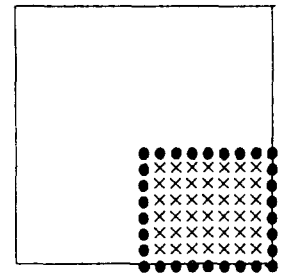

Fig. 2. Patch grid: $m_x^{(2)} = m_y^{(2)} = 8$, $p = 2$, $i_0 = 4$, $j_0 = 0$, $i_1 = 8$, $j_1 = 4$.

respective interior points of Ω^1 and Ω^2. (By the term 'interior' used for grids, we mean all points of the grid except those treated as boundary points.) The increased accuracy and efficiency and the predominance of uniform grid computations are achieved by developing a target composite grid discretization which is solved iteratively by successive processing of global and local levels. The impact of the nonuniformity of the composite grid on FAC processing is felt only during the computation of the composite grid residual at the relative few interface points (Fig. 3).

Specifically, suppose H is the discrete space of functions defined on the composite grid, Ω, and suppose we are given a composite grid operator, L, in $[H, H]$, the space of linear operators mapping H into H. For ease of discussion, suppose L is symmetric positive definite. Here we think of L as a discretization of $\mathscr{L}$ on Ω so that the target problem is now

$$Lu = f, \quad u \text{ in } H, \tag{2.2}$$

where f in H is a discrete approximation to ϕ in $\mathscr{H}^2$. Let H^k be the discrete space of functions defined on Ω^k, $k = 1, 2$. Then, to use FAC, we need several additional operators:

- coarse grid operator L^1 in $[H^1, H^1]$,
- fine grid operator L^2 in $[H^2, H^2]$,
- interpolation operators I_1 in $[H^1, H]$ and I_2 in $[H^2, H]$,
- restriction operators I^1 in $[H, H^1]$ and I^2 in $[H, H^2]$.

Assuming I_k is full rank, suppose that the variational conditions

$$I^k = c_k I_k^{\mathrm{t}}, \qquad L^k = I^k L I_k \tag{2.3}$$

are satisfied for $k = 1, 2$. (These conditions are not always easy nor advisable to obtain in practice; they are assumed here only to simplify the development of the algorithms and facilitate the theory.) Here, superscript t denotes operator adjoint in the Euclidean innerproduct and the c_k are positive constants. I_2 is assumed to be the natural imbedding operator from

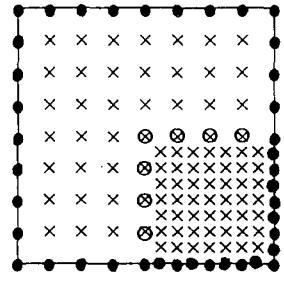

Fig. 3. Composite grid: ⊗ interface points.

H_2 into H so that $I_2 u^2$ is the vector function in H that agrees with u^2 at points of Ω^2 but is zero elsewhere. Now consider the decomposition

$$H = H_1 + H_2 \tag{2.4}$$

where $H_k = I_k H^k$ are imbedded subspaces of H, $k = 1, 2$. This decomposition induces a block representation of L given by

$$L^{\mathrm{B}} = \begin{bmatrix} L_{11}^{\mathrm{B}} & L_{12}^{\mathrm{B}} \\ L_{21}^{\mathrm{B}} & L_{22}^{\mathrm{B}} \end{bmatrix}.$$

Note that L^{B} is a *block representation* of L in the sense that L^{B} is a 2×2 *matrix of operators* $L_{ij}^{\mathrm{B}}: H_j \to H_i$. The L_{ij}^{B} act on subspaces of H; the block vector $\binom{u_1^{\mathrm{B}}}{u_2^{\mathrm{B}}}$ for $u_1^{\mathrm{B}} \in H_i$ corresponds to $u = u_1^{\mathrm{B}} + u_2^{\mathrm{B}}$ in H. More precisely, $L_{ij}^{\mathrm{B}} = Q_i^{\mathrm{t}} L Q_j$ where $Q_i = I_i (I^i L I_i)^{-1} I^i L$ is the *energy orthogonal projection* of H onto H_i. We will show in Section 3 that H_1 and H_2 do not actually partition H (that is, (2.4) is not a direct sum) because they have nontrivial intersection; this means that L^{B} is singular.

Now one cycle of FAC starting with initial guess u in H and right-hand side f in H is denoted by the expression

$$u \leftarrow FAC(u, f).$$

(Here, u is a transient approximation to the solution of (2.2) and left arrow denotes replacement.) The two-grid *coarse-to-fine* cycle of FAC for solving (2.2) is given by the following steps:

Step 1. Find an approximate solution, u^1, in H^1 of the coarse grid equation

$$L^1 u^1 = f^1 \equiv I^1 (f - Lu), \quad u^1 \text{ in } H^1.$$

Make the correction

$$u \leftarrow u + I_1 u^1.$$

Step 2. Find an approximate solution, u^2, in H^2 of the fine grid equation

$$L^2 u^2 = f^2 \equiv I^2 (f - Lu), \quad u^2 \text{ in } H^2.$$

Make the correction

$$u \leftarrow u + I_2 u^2.$$

Let $r = f - Lu$ be the residual at the start of an FAC cycle. Then one cycle of FAC is equivalent to first obtaining an approximate solution of the equation $L^1 u^1 = I^1 r$, then using boundary values at interface points defined by u^1 to solve the fine grid equation, and finally correcting u by u^2 in Ω^2 and u^1 elsewhere. Thus, the first cycle of FAC is similar to many methods that use local grids: solve the global grid equations, then use this solution to provide boundary values and solve the local grid equations. What distinguishes FAC from other methods is the use of a composite grid equation: subsequent FAC cycles are applied to the composite grid *residual* equation. If the operators and approximate grid solvers are designed correctly, then it is this simple feature of FAC that ensures essentially optimal efficiency and accuracy, even in the presence of operator singularities. Note that it is only the composite grid residual computation that involves irregular grid points, and only then at the relatively few interface points. (See [2] for further discussion, including a description of some of the possibilities for choosing the grid and intergrid operators and an analysis of the serial and parallel complexity of FAC. See [5] and [8] for theoretical analyses.)

Multilevel versions of FAC allow choices of how the various levels are scheduled in the cycling process. But to control complexity, it is best to use a *coarse-to-fine*, a *fine-to-coarse*, or a combined *V-cycle* cycling scheme. Let Ω^k be the kth element of an increasingly finer and more local sequence of grids with function spaces H^k, $k = 1, 2, \ldots, q$, let Ω be the composite of these grids with its function space H, and suppose for each $k = 1, 2, \ldots, q$ that we are given $L^k \in [H^k, H^k]$, $I^k \in [H, H^k]$, $I_k \in [H^k, H]$ and $L \in [H, H]$. Assuming that I_k are full rank, suppose the variational conditions (2.3) are satisfied for $1 \leqslant k \leqslant q$. Then the *coarse-to-fine* cycle of multilevel FAC is as follows:

For each $k = 1, 2, \ldots, q$ in turn, let u^k be an approximate solution of

$$L^k u^k = f^k \equiv I^k(f - Lu), \quad u^k \in H^k, \tag{2.5a}$$

and make the correction

$$u \leftarrow u + I_k u^k. \tag{2.5b}$$

This description provides a very convenient and instructive interpretation of FAC, but it is far removed from how it should be implemented. Concept and design should indeed focus on the composite grid, but implementation should avoid its explicit use: computation should be done on the various uniform levels, Ω^k, that constitute Ω; composite grid computations, like the evaluation of $f - Lu$ in the expression for f^k, can be done implicitly on these levels without explicit appeal to Ω. See [3,8] for a more practical description.

3. Parallelization of FAC

The main purpose of this section is to motivate the description of the AFAC algorithms in Section 4. Here we consider ways of modifying FAC to allow for simultaneous or asynchronous treatment of the levels, H^k. More specifically, we will show that FAC can be interpreted as a block Gauss–Seidel scheme on a certain singular block system. While this singularity does not impair FAC, it does prevent the effective use of a (parallel) block Jacobi scheme. We will then consider three ways to modify FAC to effectively ameliorate or eliminate this singularity, thereby paving the way for AFAC. We first make a few general comments on parallelism in adaptive processes.

FAC exhibits a reasonable degree of parallelism, first in that disjoint grids on the same level can be treated asynchronously and second in that the grid solvers (e.g., multigrid [1]) may themselves exhibit parallelism. This is sufficient for large applications on modest-size machines (e.g., where the number of processors is small compared to the number and sizes of grids on each level), especially for shared memory systems. Yet the sequentialness in FAC's treatment of its grid levels can almost paralyze large distributed memory multiprocessors. Indeed, general adaptive grid schemes are faced with the specter that processors assigned to finer grid levels must wait for the coarse-level processors to transmit boundary data, and the coarse-level processors must then wait for the finer ones to transmit their updates. To be truly effective in advanced, large-scale computing environments, the adaptive method must be able to process grid levels simultaneously instead of sequentially, without significant loss in scalar efficiency.

To see how this might be done with FAC, consider the decomposition of the composite grid space, H, into the union of its imbedded subspaces, $H_k = I_k H^k$, according to

$$H = \sum_{k=1}^{q} H_k. \tag{3.1}$$

The composite grid operator L then has a block representation in terms of this decomposition

which we write as

$$L^B = \left(L^B_{ij}\right)_{q\times q} \tag{3.2}$$

Note here that L^B acts on the space $H^B = \{u^B = (u^B_1, \ldots, u^B_q)^t\colon u^B_k \in H_k,\ 1 \leqslant k \leqslant q\}$.

Now (2.5a, b) can be viewed simply as a block Gauss–Seidel process applied to

$$L^B u^B = f^B, \quad u^B \in H^B, \tag{3.3}$$

where $f^B = (f^B_1, \ldots, f^B_q)^t$, $f^B_k = Q^t_k f$, and Q_k is the energy orthogonal projection of H onto H_k, $1 \leqslant k < q$. This would seem to suggest that a block Jacobi version of (2.5a, b) might provide the simultaneity that we need. It does, of course, but unfortunately at the expense of convergence. This stems directly from the fact that (3.1) is generally not a partition: the H_k's have nontrivial intersection. To see the impact of this, let e^1 be a nonzero vector in H^1 with support contained in the interior of Ω^2. Then $I_1 e^1$ is in $H_1 \cap H_2$. Thus, if block Jacobi were used to resolve an error in (3.3) of this kind, it would be resolved on *both* levels Ω^1 and Ω^2 at least, resulting in a disastrous overshoot of the correction. On the other hand, block Gauss–Seidel avoids this singularity of L^B in effect by resolving such components of the error on certain levels and reflecting this resolution in the residual so that successive computations cannot recompute them. For example, in the coarse-to-fine FAC cycle as we have defined it in (2.5a, b), the above error $I_1 e^1$ would be resolved only on Ω^1.

There are three basic approaches to dealing with this singularity of L^B so as to allow for asynchronous versions of FAC. The simplest is

Underrelaxation. Since block Jacobi would tend to overcorrect these singular components, an underrelaxation parameter, ω, could be used. However, we would have to use something like $\omega = 1/q$ since the singularity is q-fold. This would make the resulting process unacceptable slow.

Two effective approaches upon which we will base AFAC are

Fine grid modification. To free the levels up so that they may be handled asynchronously, we could eliminate the singularities in various ways on each finer level Ω^k, $2 \leqslant k \leqslant q$. The simplest and most efficient way may be to use multigrid as the grid solver and to suppress its coarse grid correction. More precisely, this approach consists of first determining a coarse grid approximation on the kth level by applying multigrid to the problem there, but performing relaxation only on the coarse grids in the solution process. Second, the usual multigrid algorithm (*with* fine grid relaxation) is applied to the level k equations and the result is then modified by subtracting from it the coarse grid approximation. This means that we would be solving the finer level k equations in (2.5a) on a complement (in Ω^k) of the range of interpolation from the coarse levels in Ω^k, $2 \leqslant k \leqslant q$. Note that, while this modification would make no essential change in the coarse-to-fine cycle of FAC (provided that the levels were handled sequentially), it would remove the singularity of L^B so that the various levels could be handled in parallel. See Section 4 for a detailed description of AFACf, which is based on this approach.

Coarse grid modification. Probably the least expensive approach is to attempt to remove the singularities on coarse levels. (This cannot generally be done without a real change in the basic algorithm; that is, even if the modified levels were then handled sequentially, changes of this type would amount to essential changes in FAC.) An easy way to do this is to modify the right-hand side $f^k = I^k(f - Lu)$ in (2.5a). To describe one possibility, let $\Omega^k_0 = \bigcup_{j=k+1}^{q} \Omega^k_{\text{int}}$ and let P^k be the orthogonal projector mapping H onto $\{f \in H\colon f = 0 \text{ on } \Omega^k_0\}$, $1 \leqslant k \leqslant q-1$. This means that $P^k f$ is determined by zeroing out f at points of Ω^k_0. Let $P^q = I$. Then we could replace f^k in (2.5a) by $\tilde{f}^k = I^k P^k (f - Lu)$, $1 \leqslant k \leqslant q$. The effect here is to remove interlevel data dependencies and to approximately decouple the levels so that they may be handled

simultaneously or asynchronously. In other words, the $\tilde{f}^k$ may be determined and set at the beginning of each FAC cycle: if the levels were then treated sequentially, the process would amount to a slight variation on FAC; more importantly, we may now treat the levels in parallel. In the next section, we describe AFACc, which is based on this approach.

Remark 3.1. A two-grid exact-solver fine-to-coarse version of FAC would start on the fine grid, Ω^2, producing a zero composite grid residual there. Thus, conventional FAC and both of the modified approaches suggested above have the effect of producing zero residuals on the coarse level refinement regions. In this sense, then, these modifications are natural extensions of FAC.

Remark 3.2. The basic FAC process can be coupled with multigrid in essentially two distinct ways. One, which we call MG-FAC, amounts to a simple modification of the usual multigrid process. Conventional multigrid treats all levels by relaxation and intergrid transfers, whether the grids are local or global, and usually works best when the successive level mesh sizes differ by a factor of two. For treating local grids, the only modification we need to make to multigrid is to incorporate the appropriate composite grid residuals at the interface points in the fine-to-coarse transfers. The problem with MG-FAC is that there is no real effective way to treat the local levels independently. The other approach, which we call FAC-MG, is to use multigrid as the approximate solver in the FAC process. As we shall see, it is FAC-MG that exhibits the parallelism necessary to be effective in a large-scale computing environment. In other words, multigrid methods like MG-FAC cannot effectively capitalize on the presence of local grids in a multiprocessor—the parallelism in relaxation usually depends little on whether or not the grids are disjoint; but FAC's separation of the roles of the composite grid solver and the uniform grid solver (e.g., multigrid) enables asynchronous handling of all levels. Thus, the localness of the levels in FAC is just what gives it its enhanced parallelism.

4. Asynchronous FAC (AFAC)

We will describe the simplest versions of AFAC based on the two approaches described in Section 3. These descriptions will be restricted to the simultaneous forms of AFAC; the asynchronous counterparts are natural extensions of these algorithms that do not require synchronization at the correction step. We start with the coarse grid approach, AFACc, because it is easier to describe.

Specifically, one cycle of the *simultaneous* version of AFACc, applied to (2.2) and starting with initial guess u in H and right-hand side f in H, is written as

$$u \leftarrow \text{AFACc}(u, f).$$

AFACc is defined by the following steps, where we assume that $r = f - Lu$ initially:

Step 1. Define the right-hand sides by $f^k = I^k P^k r$, $k = 1, 2, \ldots, q$.
Step 2. Find an approximate solution, u^k, in H^k of $L^k u^k = f^k$ for each $k = 1, 2, \ldots, q$.
Step 3. Form the correction $u \leftarrow u + I_k u^k$, for each $k = 1, 2, \ldots, q$. Update the residual $r = f - Lu$.

As before, this description and the one that follows are very useful ways to conceptualize AFAC, but they are removed from implementation. Note that, besides the ability to handle the various refinement levels in parallel, there are two essential features of AFACc that distinguish it from FAC: the fine grid boundary values are fixed at the start of the cycle and the coarse grid right-hand sides are set to zero at points 'covered' by further refinement.

To describe AFACf, the scheme based on fine grid modification, we need a few additional definitions. For each $k = 2, 3, \ldots, q$, let $I_{k-1}^k = [H^{k-1}, H^k]$ be the natural interpolation operator that is compatible with I_{k-1} and I_k, that is, I_{k-1} and $I_k I_{k-1}^k$ agree on the subspace of vectors in H^{k-1} with support in $\Omega^{k-1} \cap \Omega^k$ and I_{k-1}^k is zero on the subspace of vectors in H^{k-1} with support on $\Omega^{k-1} \setminus \Omega^k$. Let $I_k^{k-1} = I_{k-1}^{k\prime}$ and $Q_k^{k-1} = I - L^k I_{k-1}^k (I_k^{k-1} L^k I_{k-1}^k)^{-1} I_k^{k-1}$. Here, Q_k^{k-1} is a projector that eliminates the singularity noted in the last section. Specifically, the solution of $L^k u^k = Q_k^{k-1} f^k$ is a projection of $(L^k)^{-1} f^k$ onto a complement of the range of I_{k-1}^k. In fact, the projection Q_k^{k-1} is orthogonal in the *energy innerproduct* defined by $\langle v^k, w^k \rangle_{L^k} = \langle v^k, L^k w^k \rangle$, where $\langle \cdot, \cdot \rangle$ is the Euclidean innerproduct. Let $Q_1^0 = I$.

One cycle of AFACf is represented by

$$u \leftarrow \mathrm{AfACf}(u)$$

and defined by the following steps, assuming $r = f - Lu$ initially:

Step 1. Define the right-hand sides by $f^k = I^k r$, $k = 1, 2, \ldots, q$.
Step 2. Find an approximate solution, u^k, in H^k of $L^k u^k = Q_k^{k-1} f^k$ for each $k = 1, 2, \ldots, q$.
Step 3. Form the corrections $u \leftarrow u + I_k u^k$ for each $k = 1, 2, \ldots, q$. Update the residual $r = f - Lu$.

5. AFAC for distributed memory multiprocessors

In this section we very briefly discuss a few essential principles for implementing AFAC on large scale distributed memory multiprocessors. The prototype system we assume here is a hypercube with relatively many processors (i.e., the number of processors is roughly comparable to the total number of points in Ω), although the comments relate to more modest applications as well.

An essential ingredient for implementing adaptive methods on such machines is an efficient *load balancer*. This is in fact what motivated the development of the multilevel load balancing technique (MLB [6]), which is an efficient and itself highly parallelizable method for balancing work on a distributed memory multiprocessor system. Accounting for both arithmetic and communication costs, MLB partitions the underlying domain into regions that are automatically assigned to individual processors. The basic idea behind MLB for, say, two-dimensional rectangular domains is first to form a discrete arithmetic cost density function on this domain. Using the global communication features of a hypercube, MLB next determines horizontal lines of the partition, producing a number of balanced strips, then partitions each strip into an appropriate number of balanced rectangles. Prescribed communication costs are incorporated by manipulating the original density function in a way that involves local averages of neighboring horizontal strips. A critical aspect of MLB is that these operations are done by individual processors themselves; these processors are actually balancing their own loads.

Application of MLB to AFAC may be done in a *stack* fashion. By this we mean that the discrete cost density function could just be a measure of the density of Ω, the composite grid. This is reasonable when multigrid is used as the approximate solver in AFAC because the arithmetic cost of processing a particular subdomain of Ω is presumably a linear function of the number of points in this subdomain. An important point here in terms of complexity is that Ω can be represented and treated with a few parameters (e.g., level mesh size and corner pointers); explicit treatment of Ω is not really necessary.

We use the term stack to distinguish between partitioning of Ω by *levels*, where one or a number of processors would be assigned to each Ω^k. Because most of the arithmetic operations in AFAC are incurred predominantly by the local solver, the stack approach is potentially more

effective for many applications. The sequel paper [7] will study the performance of AFAC using this level processor assignment and load balancing approach. This has the advantage that if multigrid is used as the approximate grid solver, then the only interprocessor communication necessary in Step 1 of AFAC is that required by multigrid. Careful implementation can limit this to communication between neighbors at most two hypercube pathlengths away, and only then during relaxation (cf. [1]).

6. Comments

6.1. *'Optimal' algorithms*

To achieve optimal complexity in a serial computing environment, it is not enough to have a basic algorithm that converges quickly to the solution of the discrete problem. The number of necessary iterations of such an algebraic method would grow with problem dimension because the convergence criterion would have to become more severe to stay in tune with increased resolution. Yet, in a manner similar to that for multigrid, incorporating the basic FAC cycles in a nested iteration scheme eliminates this dependency. First, we adopt the philosophy that an error in the approximation, u, of the solution, u^*, of (2.2) is acceptable if its norm is the order of the discretization error. More specifically, suppose that $\|\cdot\|$ represents some norm on H and on $\mathscr{H}$ and suppose we have a natural mapping $\mathscr{I}: H \to \mathscr{H}$. Let ψ^* be the solution of (2.1). Then we say that u is an *acceptable approximation* to u^* if

$$\|u^* - u\| \leqslant c\,\|\psi^* - \mathscr{I}u^*\| \tag{6.1}$$

for some constant c near unity. Second, instead of solving (2.2) directly, we start by solving a coarser composite grid problem which is constructed by coarsening each of the constituent levels, Ω^k, nominally by increasing each of their mesh sizes by a power of two (see Fig. 4). (This process may involve extending the domain of the coarser Ω^k a little to ensure that its boundary coincides with points on the coarsened level $k-1$.) Continuing in this way, we would produce a *nested* sequence of composite grids $\Omega^{(1)}, \Omega^{(2)}, \ldots, \Omega^{(s)}$, the finest $\Omega^{(s)} = \Omega$ being our target grid for (2.2). The idea of *nested iteration* is then to use AFAC to solve $\Omega^{(l)}$ in turn, starting first with $l = 1$. Only after the errors on $\Omega^{(l)}$ are deemed acceptable is the solution approximation, $u^{(l)}$, on $\Omega^{(l)}$ interpolated to start the basic AFAC iteration on $\Omega^{(l+1)}$. The underlying premises here, which are true in typical applications, are that the computation on all levels $\Omega^{(l)}$, $l < s$, is inexpensive compared to that on $\Omega^{(s)}$ and that an acceptable approximation on level l is very nearly acceptable on level $l+1$. This implies that only a very small number of AFAC cycles would be required on each level. (Note that the first premise is valid only when the number of processors is not extremely large. Specifically, if there are as many processors as

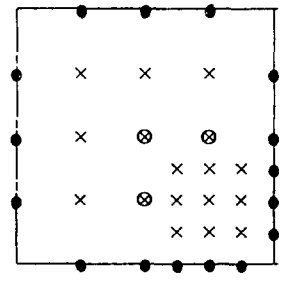

Fig. 4. Coarser composite grid: $m_x^{(1)} = m_y^{(1)} = 4$, $m_x^{(2)} = m_y^{(2)} = 4$.

there are grid points in $\Omega^{(l)}$, say, then it is generally wasteful to start processing on the coarser composite grids $\Omega^{(1)},\ldots,\Omega^{(l-1)}$.)

For a complexity analysis of this type of algorithm used in conjunction with FAC, see [2].

6.2. Basic solvers: Multigrid

The question arises as to what should the cycling strategy be in the multigrid solver used for each grid (see (2.5a)). Since processing on coarser grids can be relatively expensive on multiprocessors, W-cycles and FMG-cycles are generally prohibitive in cost. Fortunately, for many applications there seems to be little reason to use anything but a V-cycle, which has minimal parallel complexity. To see this, note that FMG-cycles are needed only for ensuring convergence to within truncation error. But this role is taken over by FAC's nested iteration scheme. In other words, a basic multigrid V-cycle is fully compatible with a basic AFAC cycle; nested iteration should be incorporated in the outer AFAC steps, not the inner multigrid cycles.

6.3. Nonlinearities

If the partial differential operator $\mathscr{L}: \mathscr{H}^1 \to \mathscr{H}^2$ is nonlinear, then we rewrite (2.1) as

$$\mathscr{L}(\psi) = \phi, \quad \psi \in \mathscr{H}^1. \tag{6.2}$$

Extension of FAC and AFAC to such problems is simply a matter of defining what we mean by a coarse grid correction equation, which we do as follows. Let $L: H \to H$ be a discretization of $\mathscr{L}$ on Ω and u be the present approximation. Let $L^k: H^k \to H^k$ be the level Ω^k discretization of $\mathscr{L}$ and define $\tilde{L}^k: H^k \to H^k$ by

$$\tilde{L}^k(u^k) = L^k\big((I^k u) + u^k\big) - L^k(I^k u).$$

Then to extend FAC to the solution of (6.2), we simply replace (2.5a) by

$$\tilde{L}^k(u^k) = f^k \equiv I^k\big(f - L(u)\big), \quad u^k \in H^k.$$

For AFACc, we use $I^k P^k(f - L(u))$ instead of $I^k(f - L(u))$ here. For AFACf, we instead use $Q_k^{k-1} I^k(f - L(u))$. The equations on each level can be solved using an appropriate nonlinear version of multigrid, for example.

6.4. Self-adaptive algorithms

Our discussion has concentrated on the pre-adapted grid case, but there is no principle impediment to a self-adaptive version of AFAC. Self-adaptive versions have been developed for FAC (cf. [3]); the same local mechanisms used in these codes can be developed for AFAC and implemented in a multiprocessing system like a hypercube. The two main adjustments that this should precipitate are that MLB, or some other load balancing scheme, would be used occasionally to ameliorate imbalance in the evolving loads, and nested iteration would be implemented as a natural consequence of the refinement strategy.

6.5. Other aspects

We have not discussed various other aspects like to what operators these methods apply, what discretization schemes are best, and how these methods behave in temporal and higher spatial dimensions. Many of these aspects are discussed in the existing [2,3,5,8] and forthcoming literature, however, and will not be treated here.

7. Theory

A simple two-level convergence theory can now be developed by interpreting AFACf and FAC as block Jacobi and Gauss–Seidel methods, respectively, then using the fact that the associated 2×2 block matrix is two-cyclic. This allows us to relate the theory for AFACf to the existing theory for FAC.

Consider the two-level (i.e., $q = 2$) version of AFACf defined in Section 4. With the assumptions there still in place, $T = I - I_1(L^1)^{-1}I^1L$ and TL^{-1} are well defined. Let $\rho(\cdot)$ denote spectral radius and define $\delta = \rho(L^1)\rho(TL^{-1})$. Finally, let the energy norm be defined by $\| u \|_L = \langle u, Lu \rangle^{1/2}$.

Theorem 7.1. *The two-level AFACf scheme converges linearly with convergence factor bounded by* $\gamma = (\delta/(1+\delta))^{1/4}$. *That is, if* $u \in H$, $u_{(\text{new})} = AFACf(u, f)$, $e = u^* - u$, *and* $e_{(\text{new})} = u^* - u_{(\text{new})}$, *then*

$$\| e_{(\text{new})} \|_L \leqslant \gamma \| e \|_L .$$

Proof. We first characterize AFACf and FAC as block relaxation methods on a certain 2×2 block system. Specifically, consider the decomposition $H = H_1 + \hat{H}_2$ where $\hat{H}_2 = I_2\hat{H}^2$, $\hat{H}^2 = \{ u^2 \in H^2 \mid Q_2^1 u^2 = u^2 \}$. This is a modification of the decomposition in (2.4). Note that the solution of $L^2u^2 = Q_2^1 f^2$ is the best energy approximation to $(L^2)^{-1}f^2$ in $\hat{H}^2$. Now this decomposition induces a block representation similar to (3.2) which we write as

$$L^{\mathrm{b}}u^{\mathrm{b}} \equiv \begin{pmatrix} L_{11}^{\mathrm{b}} & L_{12}^{\mathrm{b}} \\ L_{21}^{\mathrm{b}} & L_{22}^{\mathrm{b}} \end{pmatrix} \begin{pmatrix} u_1^{\mathrm{b}} \\ u_2^{\mathrm{b}} \end{pmatrix} = \begin{pmatrix} f_1^{\mathrm{b}} \\ f_2^{\mathrm{b}} \end{pmatrix}.$$

It is now straightforward to see that AFACf and FAC are just the respective block Jacobi and block Gauss–Seidel methods applied to this system. But since L^{b} is 2×2 block symmetric and positive definite, it is two-cyclic. Thus, a theorem of Young [9] implies that the spectral radius of the iteration matrix for AFACf is just the square root of that for FAC. But the latter is bounded by the energy convergence factor for FAC, which by Theorem 1 of [5] is bounded in turn by $(\delta/(1+\delta))^{1/2}$. Thus, the spectral radius of the iteration matrix for AFACf is bounded by γ. The proof now follows from noting that this iteration matrix is symmetric in energy, so its spectral radius and energy convergence bound are the same.

Remark 7.2. By monotonicity arguments, it is easy to see that the worst case for δ is achieved when Ω^2 extends over the whole domain. This means that $i_0 = j_0 = 0$, $i_1 = m_x^{(1)}$, and $i_2 = m_y^{(1)}$. (This is an absurd case for either FAC or AFAC because the domains of Ω^1 and Ω^2 agree, so the presence of Ω^1 in the outer FAC process is wasted. Of course, Ω^1 is useful for the multigrid solution process on Ω^2.) By the usual multigrid arguments (cf., [4]), it is easy to show for standard applications (e.g., elliptic problems with full regularity and adequate approximation properties) that δ is bounded above by a constant independent of the meshsizes of either Ω^1 or Ω^2 (and therefore of their ratio, p). This means that AFACf convergence is of optimal order and that AFACf-MV with nested iteration is of optimal complexity.

Remark 7.3. The theory in [8] extends the basic results for FAC in [5] to more complex applications: approximate grid solvers, singular L, and badly graded meshes. By way of the arguments of the above theorem, these results extend immediately to AFACf.

8. Numerical convergence estimates

Several numerical experiments were performed to determine asymptotic convergence factors for AFAC. The tests on AFACc were on Poisson's equation on the unit square with homogeneous Dirichlet conditions, using finite volume discretization for the composite grids and bilinear interpolation and injection for the interlevel transfers. This gives the usual 5-point stencil on uniform regions. The grid solvers were multigrid schemes based on red-black Gauss–Seidel relaxation. Basic V-cycling and full multigrid were used.

Four refinement configurations were tested. All used a mesh refinement ratio (i.e., p, the ratio of mesh sizes on successively finer levels) of 2. Configuration number 1 used 10 levels, each consisting of a 33×33 grid with southwest corner starting at grid point (5, 5) of its parent (the next coarser level). Number 2 is identical except that levels 2 through 10 are 17×17 grids. Number 3 uses 6 levels, each a 65×65 grid, with the southwest corner parent grid location for the successively finer levels at (3, 3), (3, 3), (5, 5), (9, 9), and (17, 17). Number 4 is a simple two-level version of number 1, consisting of a global 33×33 grid with a 33×33 refinement starting at grid point (5, 5).

We first tested AFACc, using both FMV(ν_0, ν_1, ν_2) and MV(ν_0, ν_1, ν_2) as the basic solvers. Here, ν_1 and ν_2 refer to the respective number of red-black Gauss–Seidel sweeps before and after coarse grid correction and ν_0 is the number of V-cycles used on the finest grid. (The coarser levels in FMV used one V-cycle only.) For comparison, we tested FAC/, which is the coarse-to-fine cycle of FAC, using the same solvers. Table 1 depicts the average asymptotic per-cycle convergence factor for each of these tests. These factors were computed by averaging the per-cycle convergence factors of cycles 11 through 20 of 20-cycle test runs, so they provide good estimates of the spectral radii of the iteration matrices.

We make several observations:

(a) FAC is fairly sensitive to the grid solver performance, even showing divergence in the worst case (using one V-cycle with $\nu_1 = \nu_2 = 1$). No doubt the performance is most seriously affected by the use of injection, which ignores certain nonzero residuals in refinement regions. We will see later that this sensitivity vanishes when variational discretizations are used.

(b) AFACc is very stable with respect to grid solver performance. In fact, reasonably good rates are obtained even with the least expensive solvers, while there is essentially nothing to gain by spending much more work in solving each grid equation.

(c) When the least expensive solvers are used, AFACc actually outperforms FAC in all but the two-level case. Further effort spent in the grid solvers then tips the scales in the direction of FAC, but the improvement in performance of FAC is not enough to warrant this added expense.

(d) AFACc's performance is also very stable with respect to the grid configurations. Only in the two-level case is performance significantly different from the others, and even then it is only slightly better.

Next we performed tests on AFACf. To maintain compatibility with the theory, we developed a fully variational setting for Poisson's equation: we used bilinear interpolation, full weighting and the Galerkin operator approximation, yielding the usual 9-point stencil in uniform regions. We used the same test configurations. However, because of the remarkable stability of FAC and AFACf with respect to grid solver accuracy, we include in Table 2 only the results for MV(1, 1, 1) and FMV(1, 1, 1). Again, the factors are averaged over cycles 11 through 20 of 20-cycle test runs. Because of energy-symmetry, the AFACf factors estimate the energy norms of the AFACf iteration matrices.

We observe three striking features from this table:

(a) For variational discretization, there is virtually nothing to gain for either FAC or AFACf by using anything more costly than the simplest grid solver MV(1, 1, 1). This suggests that we

Table 1
Asymptotic convergence factors for AFACc vs. FAC in a nonvariational setting

ν_0	ν_1, ν_2	AFACc-FMV				FAC/-FMV				AFACa-MV				FAC/-MV			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	1, 1	0.39	0.45	0.38	0.26	0.73	0.70	0.57	0.31	0.67	0.68	0.58	0.42	1.9	1.1	3.2	0.98
2	1, 1	0.39	0.38	0.36	0.26	0.56	0.55	0.34	0.11	0.40	0.40	0.35	0.26	0.74	0.72	0.58	0.23
3	1, 1	0.39	0.39	0.36	0.26	0.43	0.43	0.22	0.09	0.38	0.38	0.36	0.26	0.56	0.56	0.35	0.10
6	1, 1	0.38	0.33	0.36	0.26	0.21	0.20	0.10	0.09	0.38	0.33	0.36	0.26	0.27	0.26	0.11	0.09
9	1, 1	0.38	0.33	0.36	0.26	0.14	0.13	0.10	0.09	0.38	0.33	0.36	0.26	0.16	0.13	0.10	0.09
1	2, 1	0.39	0.36	0.36	0.26	0.55	0.53	0.35	0.10	0.51	0.52	0.42	0.25	0.80	0.79	0.62	0.40
2	2, 1	0.38	0.33	0.36	0.26	0.36	0.36	0.17	0.09	0.38	0.35	0.36	0.26	0.55	0.54	0.35	0.10
3	2, 1	0.38	0.33	0.36	0.26	0.25	0.25	0.12	0.09	0.38	0.33	0.36	0.26	0.36	0.36	0.17	0.09
4	2, 1	0.38	0.33	0.36	0.26	0.18	0.17	0.10	0.09	0.38	0.33	0.36	0.26	0.25	0.25	0.12	0.09
1	2, 2	0.39	0.33	0.36	0.26	0.48	0.42	0.31	0.09	0.40	0.39	0.35	0.25	0.89	0.51	0.86	0.20
2	2, 2	0.38	0.33	0.36	0.26	0.21	0.23	0.12	0.09	0.39	0.33	0.36	0.26	0.48	0.42	0.31	0.10

Table 2
Asymptotic convergence factors for AFACf vs. FAC in a variational setting

ν_0	ν_1, ν_2	AFACf-FMV				FAC/-FMV				AFACf-MV				FAC/-MV			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	1, 1	0.22	0.22	0.21	0.22	0.19	0.19	0.19	0.19	0.24	0.24	0.22	0.23	0.19	0.19	0.19	0.19

are doing just about as good as we can do in solving composite grid problems by way of uniform ones.

(b) Perhaps most surprising is the observation that there is virtually no difference between two-level and multi-level rates for either method. This very desirable property, which is rare in multilevel processes, suggests that the components that couple adjacent levels are themselves nearly decoupled by the FAC and AFACf processes.

(c) The observed convergence factor for FAC is about the square of that for AFACf in all cases. This is to be expected, especially for the third case, because of the relationship to the respective Gauss–Seidel and Jacobi block relaxation schemes.

We have also tested AFACf on much larger problems (including the presently absurd cases of six levels of size 1027×1027 and 50 levels of size 33×33) with essentially the same performance results.

References

[1] B. Briggs, L. Hart, S. McCormick and D. Quinlan, Multigrid methods on a hypercube, in: Lecture Notes in Pure and Applied Mathematics **110** (Marcel Dekker, New York, 1988).

[2] L. Hart, S. McCormick, A. O'Gallagher and J. Thomas, The fast adaptive composite grid method (FAC): Algorithms for advanced computers, *Appl. Math. Comp.* **13** (3/4) (1985).

[3] M. Heroux, S. McCormick, M. McKay and J. Thomas, Applications of the fast adaptive composite grid method, in: Lecture Notes in Pure and Applied Mathematics **110** (Marcel Dekker, New York, 1988).

[4] J. Mandel, S. McCormick and R. Bank, Variational multigrid theory, SIAM Frontiers in Applied Mathematics, Vol. III on Multigrid Methods (1987) Ch. IV.

[5] S. McCormick, Fast adaptive composite grid methods, in: K. Böhmer and H.J. Stetter, eds., *Defect Correction Methods: Theory and Applications*, Computing Supplementum **5** (Springer, Wien, 1984) 115–121.

[6] S. McCormick and D. Quinlan, Multilevel load balancing for multiprocessors—an outline, in: S. McCormick, ed., *Prelim. Proc. 3rd Copper Mountain Conf. on Multigrid Methods*, Copper Mountain, CO (1987).

[7] S. McCormick and D. Quinlan, Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Performance results, *Parallel Comput.* **12** (1989) 145–156, this issue.

[8] S. McCormick and J. Thomas, The fast adaptive composite grid method (FAC) for elliptic equations, *Math. Comp.* **46** (1986) 439–456.

[9] D. Young, Iterative methods for solving partial differential equations of elliptic type, Doctoral Thesis, Harvard University, 1950.