PARALLEL MULTIGRID METHODS

Jim E. Jones

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

Stephen F. McCormick

Program in Applied Mathematics
University of Colorado
Boulder, CO 80309-0526

ABSTRACT

Multigrid methods have proved to be among the fastest numerical methods for solving a broad class of problems, from many types of partial differential equations to problems with no continuous origin. On a serial computer, multigrid methods are able to solve a widening class of problems with work equivalent to a few evaluations of the discrete residual (i.e., a few relaxations). Many research projects have been conducted on parallel multigrid methods, and they have addressed a variety of subjects: from proposed new algorithms to theoretical studies to questions about practical implementation. The aim here is to provide a brief overview of this active and abundant field of research, with the aim of providing some guidance to those who contemplate entering it.

1. Multigrid Basics

The goal of multigrid methods (and the related multilevel and multiscale methods) is the efficient computation of accurate numerical solutions to scientific problems. (We are deliberately vague here in what we mean by a problem because these methods have been successfully used in very diverse applications.) While there is no single multigrid method, there seems to be one philosophy of computation that is common to all multigrid methodologies:

> *"The amount of computational work should be proportional to the amount of real physical changes in the computed solution,"* Achi Brandt's "golden rule" (Brandt, 1977).

If this rule is violated, such as when an iterative method's convergence stalls, it is often due to the mistreatment of solution components of differing scales. For stalling iterative methods, components that are smooth with respect to the given grid are typically slow to converge. Multigrid methods use several interacting grids, or scales of discretization, to account for these varying scales. This allows for rapid convergence, thus restoring the proper balance between computational work and changes in the solution.

It would be premature to begin our brief overview of the vast field of parallel multigrid without even a very cursory look at the general field of multigrid methods. We will be content with a brief glimpse at a few basic multigrid algorithms and leave the rest of the field for the reader's own study. Thankfully, there are a host of excellent resources available to those new to multigrid methods: for introductory level material and references to more advanced topics, see the tutorial (Briggs, 1987); for current information on the field, including an extensive bibliography (Douglas and Douglas) a repository of papers and codes, and current events, access the World Wide Web server MGNet at file://na.cs.yale.edu/pub/mgnet/www/mgnet.html or its anonymous ftp site at na.cs.yale.edu/mgnet.

1.1. V cycle algorithm

Although they have since been applied to various types of partial differential equations, to integral equations, and to equations with no continuous origin (e.g., geodesy and molecular dynamics), multigrid methods were first used for solving elliptic partial differential equations (PDEs). This is the type of problem we consider here, even though much of what we say will also apply to other types of problems.

Let $Lu = f$ denote the PDE to be solved numerically, by which we mean finding a discrete representation u^h that is an accurate approximation to the continuous solution u. The multigrid approach to this problem starts by discretizing the PDE on a fine grid Ω^{h_f} using, say, a finite difference method, resulting in a matrix equation $L^{h_f} u^{h_f} = f^{h_f}$ that must be solved. Multigrid gets its name because, in addition to the fine grid, the method uses coarser grids, $\Omega^{2h_f}, \Omega^{4h_f}, \ldots, \Omega^{h_c}$, which are usually obtained from the finer grids by successively removing every other point in each coordinate direction. This is done recursively until one reaches the coarsest grid

Ω^{h_c}, which typically consists of one or just a few points. Applying the same discretization process that is used on the finest grid to each of the coarser grids results in a series of discrete problems $L^h u^h = f^h, h = h_f, 2h_f, \ldots, h_c$. An important component of multigrid methods is an iterative (relaxation) process, which we denote by $u^h \leftarrow \mathcal{G}(u^h, L^h, f^h)$. Usually this iterative process is fairly simple: typically just point or block Jacobi or Gauss-Seidel relaxation of the equation $L^h u^h = f^h$. Two other important components are the intergrid transfer operators: prolongation (or interpolation), P_{2h}^h, and restriction, R_h^{2h}. Usually, these operators are also of fairly simple form: prolongation may be based on simple linear interpolation and restriction on its transpose. Below is a representation of the multigrid V-cycle algorithm, which we denote by $u^h \leftarrow \mathcal{MG}(u^h, L^h, f^h)$, in its recursive form. It uses $\gamma_1 \geq 0$ relaxations before coarse grid correction and $\gamma_2 \geq 0$ after, with $\gamma_1 + \gamma_2 \geq 1$.

1.(*Coarsest grid solution*) If $h = h_c$, set $u^h \leftarrow (L^h)^{-1} f^h$ and return. Otherwise, go to **2**.
2.(*Pre-smoothing*) Perform $u^h \leftarrow \mathcal{G}(u^h, L^h, f^h)$ γ_1 times.
3.(*Coarse grid correction*) Set $f^{2h} \leftarrow R_h^{2h}(f^h - L^h u^h), u^{2h} \leftarrow 0$, $u^{2h} \leftarrow \mathcal{MG}(u^{2h}, L^{2h}, f^{2h})$, and $u^h \leftarrow u^h + P_{2h}^h u^{2h}$.
4.(*Post-smoothing*) Perform $u^h \leftarrow \mathcal{G}(u^h, L^h, f^h)$ γ_2 times.

When relaxation and coarse grid correction are properly chosen, multigrid methods are generally very efficient solvers. High frequency error is effectively reduced on the fine grid by relaxation, and low frequency error is transferred to coarser grids through the residual equation, resolved there by recursive computation, and effectively reduced on the fine grid by interpolation and correction. Typically, a single V-cycle (with $\gamma_1 = 2$ and $\gamma_2 = 1$, say) reduces the norm of the error between the current approximation and the exact solution by about an order of magnitude. Most important, such reduction factors are assured for virtually any grid size.

To implement a particular multigrid algorithm, there are many choices to be made: prolongation operators, restriction operators, the relaxation process, the fine grid discretization process, and even how to form the coarse grid operators (in some cases, it is important to consider alternatives to defining L^{2h} in the same way L^h was defined). The performance of the method depends critically on these choices. We believe that they should not be made independently, as if ordering a meal à la carte, but rather guided, as much as possible, by a single principle — the fine grid discretization process. For example, especially effective multigrid schemes can be developed naturally in the context of finite volume element (McCormick, 1989) or finite element (McCormick, 1992) methods.

1.2. Full multigrid algorithm (FMG)

The multigrid V-cycle algorithm can be used to calculate approximate solutions of the discrete problem $L^{h_f} u^{h_f} = f^{h_f}$ to any desired accuracy. Of course, the true goal of computation is presumably the solution of the continuous problem $Lu = f$. Efficient calculation of accurate approximations to this problem is the aim of the so-called full multigrid algorithm (FMG) introduced in (Brandt, 1977). The basic idea behind FMG is to combine V-cycle multigrid with nested iteration, using coarser grids to generate "good" initial guesses for finer grids. Below is a representation of the full multigrid algorithm, which we denote by $u^h \leftarrow \mathcal{FMG}(L^h, f^h)$, in its recursive form. It uses $\gamma_0 \geq 1$ V-cycles on each successively finer grid in the nested iteration process.

1.(*Coarsest grid solution*) If $h = h_c$, set $u^h \leftarrow (L^h)^{-1} f^h$ and return. Otherwise, go to **2**.
2.(*Coarse grid initial approximation*) Set $u^{2h} \leftarrow \mathcal{FMG}(L^{2h}, f^{2h})$.
3.(*Fine grid V-cycles*) Set $u^h \leftarrow I_{2h}^h u^{2h}$ and perform $u^h \leftarrow \mathcal{MG}(u^h, L^h, f^h)$ γ_0 times.

As discussed in (Brandt, 1984), depending on the application, the FMG interpolation operator I_{2h}^h may or may not be the same as the prolongation operator P_{2h}^h in the multigrid V-cycle algorithm.

1.3. Computational complexity

While a full development of the theory is well beyond the scope of this paper, a brief discussion of computational complexity will illustrate the appeal of multigrid methods. For the reader interested in multigrid theory there are many articles in the literature; recent publications (Bramble, 1993, Xu, 1992, and Yserentant, 1993) may be good starting points because of their broad scope and large bibliographies. Here we make use of two natural assumptions: in an appropriate norm, the discretization on grid h has $O(h^p)$ accuracy for some positive integer p, and the basic V-cycle algorithm converges with factor bounded above by $\gamma < 1$ for all admissible h.

As mentioned previously, the V-cycle algorithm's goal is to obtain an accurate approximation to the solution of the discrete problem $L^{h_f} u^{h_f} = f^{h_f}$. This is an iterative method, and a natural stopping criterion is to require that the difference between the approximate solution and the true solution to the discrete problem (this difference is called the algebraic error) be comparable to the difference between the true solution to the discrete problem and the true solution to the continuous problem $Lu = f$ (this difference is called the discretization error). When this criterion is met, we will say the discrete system is solved to the level of discretization error. Now each V-cycle has cost of $O(n)$, where n is the number of unknowns in the fine grid system. However, starting from $O(1)$ size error, it is easy to see that this convergence criterion requires $O(\log n)$ V-cycles. Thus, the overall cost of achieving this accuracy by the V-cycle algorithm is $O(n \log n)$.

A follower of Brandt's "golden rule" will be unhappy with the $\log n$ term in the V-cycle's complexity, since true efficiency should mean that the computational work grows linearly with the size of the problem. The reason for this sub-optimal complexity is the V-cycle's limited use of coarse grids: they should be used not only to resolve fine grid error, but also to generate "good" initial guesses on fine grids. This is precisely the two roles that the coarse grids play in the FMG algorithm. The initial coarse grid computation in FMG delivers a fine grid approximation that is already accurate to within a fixed factor of the discretization error, so that just one or a few subsequent V-cycles are enough to converge to the level of discretization error. As a result, FMG's overall's cost to solve to this level is $O(n)$. The constant here and for the V-cycle above is typically just small multiple of the computational work involved in calculating the residual of the discrete problem at each of the n points. For example, the FMG algorithm can solve the 5-point Poisson equation to level of discretization error in about 22 computer operations per grid point (Brandt and Diskin, 1994). We thus like to think of the FMG algorithm as an optimal direct solver for the continuous problem.

2. Parallel Multigrid

In a serial computing environment, multigrid methods are well established as one of the most efficient ways to solve large-scale computational problems. The development of parallel computers with many processors is motivated by the dramatic increase in computing power required to handle the very large-scale computational problems now on the horizon. Thus, multigrid methods and parallel computers are both aimed at solving the same problems, which brings up the question: *Do we need both?* We believe the answer is a definite *yes.*

This belief is based in part on the following considerations. First, for most applications, there are likely to be many more unknowns than available processors, regardless of the size of the computer system. This follows directly from the reasonable premise that demand always taxes capability. Second, in the current generation of parallel computers, the individual processors are fairly powerful compared to their number, a trend that is likely to carry into future generations. Thus, although parallel and serial computing seem to demand quite different considerations, it is likely that the most efficient numerical algorithms for parallel computing will come from the parallelization of algorithms that are already very efficient in the serial environment. In other words, we take the view that challenging problems will always demand very substantial computation on each processor, enough so that the basic algorithm must be efficient in the sequential sense. (This assumes that the individual processors are assigned tasks that are essentially analogous to the global algorithm tasks, as is typical for the problems considered here.) This view is admittedly arguable, so it is important to consider the case where the number of unknowns and processors are comparable (if for no other reason than to understand the options on coarse grids in a parallel multigrid algorithm where this case must arise).

Parallel multigrid algorithms can be loosely classified into two categories. The simplest and most common approach involves no real changes to the basic multigrid algorithm. The desire to exploit parallel computation wherever possible guides the choice of the relaxation process (efficient parallel multigrid algorithms typically use either Jacobi or multicolor Gauss-Seidel relaxation schemes) and the partitioning of grids to processors. We will call this standard approach *parallel computations within levels* to emphasize that the the different grids (or levels) are still processed sequentially. The second approach is radically different and is motivated by the desire to overcome the parallel limitation inherent in the sequential processing of different grids. Adhering to the standard multigrid algorithm, it is not possible to process the coarse and fine levels simultaneously because the coarser grids must wait for the finer grid residuals and the finer grids must wait for coarser grid corrections. When the number of coarse grid unknowns is less than the number of available processors (as will almost always be the case for the coarsest grids, which typically contain only a few points), there will be a large number of idle processors. To overcome this possible problem, several unique algorithm modifications have been suggested, to which we refer as *concurrent multigrid algorithms*.

We close this section with an obvious but often under-appreciated point. The most efficient algorithm is presumably the one that solves the problem in the shortest time. This means that questions about processor utilization, communication to computation ratios, and other issues about parallel performance are secondary. While these questions are often very useful in analyzing the performance of an algorithm, their importance can be overstated. For example, an algorithm with little or no communication requirements and whose parallel performance is perfectly scalable may be, for a particular application and computer, much less efficient than an algorithm that does not "parallelize" well.

2.1. Parallel computations within levels

A simple approach to implementing a multigrid algorithm on a parallel computer is by domain partitioning. This is the approach advocated by Achi Brandt in his earliest (see Brandt, 1981) through most recent (see Brandt and Diskin, 1994) work on parallel multigrid, and the following discussion borrows from this latter reference. Let Ω^{h_f} denote the finest grid of the multigrid algorithm, the grid on which we want to solve $Lu = f$. Assuming, as mentioned above, that the number of grid points in Ω^{h_f} is much larger than the number of processors P, the domain can be partitioned and each of the resulting subgrids $\Omega_i^{h_f}$ of $\Omega^{h_f} = \cup_{i=1}^{P} \Omega_i^{h_f}$ can be assigned to an individual processor. The partition of the domain also naturally induces a similar division of all coarser grids:

$$\Omega^h = \cup_{i=1}^{P} \Omega_i^h, \quad (h = h_f, 2h_f, \ldots, h_c)$$

A multigrid algorithm can then easily be parallelized by having each processor p perform all operations that result in changes to the values at grid points belonging to one of its subgrids Ω_p^h. Note that on the coarser grids, where there are more processors than grid points, some processors will be idle. However assuming that, on the finer levels where most of the computing time is spent, there are more grid points than processors, it can be argued that this very coarse level inefficiency is negligible (for arguments in this vein see (Briggs et al., 1988) and (Degani and Fox, 1995)). These papers also contain numerical results ((Briggs et al., 1988) implement a Poisson solver, (Degani and Fox, 1995) implement a Navier-Stokes solver) which illustrate the efficiency of the multigrid algorithm when ratio of fine grid unknowns to processors is large.

An important point to consider is the problem of communication between processors. (An interesting argument in (Chan and Tuminaro, 1987) shows that standard parallel multigrid is, in some sense, an "optimal" method for solving an elliptic PDE if communication delays are ignored.) Assuming that each processor has its own memory not shared by others, there are several points in a multigrid algorithm, such as in relaxation and residual calculation, where information must be transferred. In (Brandt and Diskin, 1994), the desire to minimize the number of communication episodes, and the amount of information transferred at each, led to various minor (but novel) modifications of the basic multigrid algorithm. The algorithm uses overlapping subgrids and the data transfer between processors occurs only once per cycle. The resulting FMG-like algorithm is able to solve model elliptic problems to the level of discretization error in essentially the same work as needed for solving just once, by FMG, a separate problem on each subgrid.

We close this section by noting that, at first glance, the V-cycle algorithm appears to "parallelize" better than the FMG algorithm because it spends relatively less time on the coarser grids, where the idle processor problem can occur and the communication-to-computation ratio is greater. However, several researchers (see Matheson, 1994 and Tuminaro and Womble, 1993) have compared the performance of V-cycle multigrid and FMG using various computational models designed to reflect actual machine performance. These studies indicate that, for most applications, FMG is still more efficient, in terms of total computational time to achieve discretization-level accuracy, than V-cycle multigrid.

2.2. Concurrent multigrid algorithms

The desire to overcome the inherent sequential nature of the coarse grid correction step in the standard multigrid algorithm has led several researchers to propose algorithmic changes that allow computation on different grids to proceed in parallel. We will briefly discuss a few of these algorithms and refer the reader to references for details. Our focus is on two dimensions for simplicity, although all of the comments extend easily to higher dimensions.

In (Frederickson and McBryan, 1988), the authors sought to develop a parallel superconvergent multigrid (PSMG) method that creates useful work for the processors that would be idle when standard multigrid reached the coarser grids. To accomplish this, in addition to the standard coarse grid (obtained from the fine grid by taking every other grid line in each coordinate direction), they use three other coarse grids that are simple translates of the standard coarse grid (the translation is by the fine grid mesh size in either or both of the coordinate directions). They can then perform calculations on these four coarse grids in parallel and correct the fine grid approximation by a combination of these coarse grid error approximations. Because the total number of coarse grid points (summing over the four coarse grids) is equal to the number of fine grid points, the idle processor problem is avoided. The convergence factors for PSMG are better than for the standard V-cycle algorithm, but the question of efficiency is not yet settled. Independent researchers (see Decker, 1991 and Matheson, 1994) have examined the performance of the algorithm applied to Poisson's equation using different computational models. When the number of processors is of the same order as the number of unknowns, their conclusions about PSMG's performance differ. Both agree that it is less efficient than standard parallel multigrid when there are relatively fewer processors than unknowns, but such a regime is not for what PSMG was intended.

The multigrid algorithm first proposed in (Douglas and Miranker, 1988) and further developed in (Douglas, 1991) is in some ways similar to PSMG. In this algorithm, the error equation on the fine grid is split into a large number of problems posed on much coarser subspaces. These problems can be solved in parallel and the solutions combined in such a way as to yield, under certain conditions, a direct solver or a very fast iterative method. The performance depends critically on choice of coarse subspaces and the way in which the solutions are combined. The references contain the details of the algorithm and some computational work estimates for model problems, showing that the method has the potential to be more efficient than standard multigrid. Whether this potential can be realized for practical applications is, to our knowledge, an open question.

Recent work (see Dendy, 1995, Dendy et al., 1992, and Naik and Van Rosendale, 1993) has focused on variants of the frequency decomposition multigrid method (Hackbusch, 1989) and the multiple semi-coarsened method (Mulder, 1989). Like PSMG, these methods use multiple coarse grids, but the motivation is somewhat different: they use multiple coarse grids to represent error frequencies that cannot be represented on a single coarse grid. Their chief advantage is the potential to solve problems with discontinuous and anisotropic coefficients without special relaxation. For these types of problems, to obtain proper smoothing of the error, standard multigrid methods need line relaxation, which can be costly on parallel computers. For acceptable multigrid convergence, the error left after a few relaxation steps must be accurately represented on the coarse grid. For point relaxation, the single standard coarse grid cannot accomplish this for these problems, but multiple coarse grids can often combine effectively to properly approximate such error. These methods can be implemented to allow parallel processing of grids; however, it is not yet clear if these multiple coarse grid algorithms are more efficient than standard multigrid for realistic problems on the current generation of parallel computers.

While the previously discussed multigrid methods address the idle processor problem by creating additional coarse grids, there are several other methods designed to allow computation on different grids of the standard multigrid algorithm to proceed in parallel. Four such algorithms are: concurrent iteration (Gannon and Van Rosendale, 1986) a Fourier transform multigrid (Matheson, 1994), the Chan-Tuminaro algorithm (Chan and Tuminaro, 1987), and the Greenbaum algorithm (Greenbaum, 1986). While these algorithms differ in their approach they all attempt in effect to decompose the problem space into a set of essentially non-interfering subproblems that can then be solved in parallel. Typically, the residual on the fine grid is redistributed among all the grids in an attempt to match error frequencies with mesh size. The aim is to allow each grid to resolve only those error frequencies for which relaxation is effective. These algorithms are necessarily more complicated than the standard V-cycle multigrid algorithm, but they no longer require the sequential processing of grid levels.

In (Matheson and Tarjan, 1994), the performance of these decomposition-based concurrent multigrid algorithms is compared to a parallel implementation of standard multigrid. This study used a variety of computational models, designed to reflect the fine-grain structure of the previous generation of parallel computers (for example, the CM-200) and the medium-grain structure of the present generation (for example, the Intel Paragon). The conclusion was that, unless there are an unreasonable number of processors (again, something on the order of the number of fine grid unknowns), these concurrent multigrid algorithms are not as efficient as standard multigrid algorithms. The authors point out that this conclusion remains true regardless of the characterization of inter-processor communication costs.

There are several areas where concurrent multigrid methods may be more efficient than standard multigrid. One is in time-dependent calculations. As we will say more about in Section 3, applying Brandt's "golden rule" to time-dependent parabolic PDEs (see Brandt and Greenwald, 1991) leads to an algorithm where fine spatial grids are activated only rarely in the time-stepping process. At many time levels, the spatial grid may be coarse enough that one would have a sufficient number of processors to make a concurrent multigrid algorithm a practical alternative to standard multigrid. A second area is in process control applications where a solution may be needed in fractions of a second, so the problem may have a relatively small number of unknowns. A third area where simultaneous processing of levels may be more efficient is in applications with local refinement. This is the topic of the next section.

2.3. Adaptive refinement

In applications, the need to model complex small-scale physics (boundary layers, turbulence,...) accuracy leads to the use of either very fine global grids or adaptive local refinement. The use of very fine global grids violates Brandt's "golden rule" if, in many regions, the solution is relatively smooth. A global mesh size tailored to resolve, say, a boundary layer results in far too much computational work in these regions. Although it might appear at first glance that the increased power afforded by parallel computers will allow the use of fine global grids and thus relieve the need for adaptive methods, just the opposite is probably true. Adaptive methods are designed to achieve more capabilities from a given computer than are possible with conventional methods. Expanding computer power will likely bring with it greater problem sophistication that will quickly tax new capacities. Efficient local resolution will again be required only now the enhancements in the physical model will significantly accentuate disparities in scale. This should make adaptive methods imperative for parallel computation.

The idea of combining adaptive local refinement with multigrid methods is certainly not new (Brandt, 1977), and because both deal with grids of varying mesh size, the two ideas fit naturally together. Much of what has been said about parallelizing multigrid methods in Section 2.1 can be directly applied to multilevel local refinement methods. However, we believe that allowing simultaneous processing of levels may be much more important here than in the global grid case. Our reasoning is that, in the global grid case, as stated before, the finer grids will typically have more unknowns than available processors. Because most of the computation is on these grids, where there is no idle processor problem, there is little to be gained (and, in fact, some efficiency is lost) by concurrent multigrid methods. This may not be the case in local refinement applications, where it may well be that there are no levels that have more unknowns than available processors. In fact, the number of points on most of the levels might be much smaller than the number of available processors, and there may be many such levels (many more than in the global grid case). Therefore, the idle processor problem can occur on all levels unless the algorithm allows concurrent level processing. It may be possible to apply some of the methods of the last section to local refinement problems, but there should be more to gain from two multigrid methods, AFAC (Hart and McCormick, 1989, and McCormick, 1989) and BPX (Bramble, 1990), that allow simultaneous processing of all refinement levels. (These methods are generally thought of in the context of adaptive refinement, although they apply to the global grid case as well.)

To briefly explain these two algorithms, consider a model two-level problem, where Ω^{2h} is a global uniform grid and Ω^h is a uniform refinement patch with half the mesh size of the global grid. We are interested in solving the PDE $Lu = f$ discretized on the composite grid $\Omega^c = \Omega^{2h} \cup \Omega^h$. We refer to the resulting discrete problem as $L^c u^c = f^c$. In both methods, the residual r_c is calculated on the composite grid and then transferred, using appropriate restriction operators, to both Ω^h and Ω^{2h} to form the right-hand side for two discrete problems $L^h u^h = R_c^h r_c$ and $L^{2h} u^{2h} = R_c^{2h} r_c$. These two problems can be approximately solved by relaxation in parallel.

In BPX, the solutions u^h and u^{2h} are then interpolated to the composite grid and used to correct the current approximation to the solution u^c. Used as a stand-alone iteration, BPX will generally not converge. However, it yields an effective preconditioner when combined with a conjugate gradient-like method.

To understand why BPX alone will not generally converge, we need to consider the roles of Ω^h and Ω^{2h} in the solution process. The aim of the coarse global grid Ω^{2h} is to resolve global error components varying on the coarse scale, and the aim of the local fine grid Ω^h is to resolve local error components varying on the fine scale. Consider an error component that that is zero outside of the refinement region but nonzero and smooth within. This component will be resolved on *both* Ω^h and Ω^{2h}, so the correction to the composite grid solution will overshoot this error component by a factor of two when the grid h and grid $2h$ corrections are computed exactly. What saves BPX from catastrophe (i.e., dependence of its preconditioning property on the number of levels) is its restriction to simple relaxation, which is progressively weaker at resolving increasingly smooth error.

The AFAC solution to this overshoot problem, which allows for any type of solver and leads to convergence as a stand-alone iteration, is to insert the grid $\Omega^h \cap \Omega^{2h}$ between Ω^h and Ω^{2h}. This new grid, which has the resolution of the global coarse grid and the domain of the local fine grid, is used to resolve and eliminate the error components that are duplicated on the original pair of grids. The cost of this additional processing is almost negligible.

We have only briefly explained the basics of these two methods. We refer the interested reader to (McCormick and Quinlan, 1992), where a convergence rate comparison between BPX and AFACx (an AFAC algorithm using only relaxation on the subgrids) is made for an idealized application. For timing results of applications on parallel computers see (McCormick and Quinlan, 1989) for AFAC and (Bastian, 1993) for BPX.

3. Multigrid Methods for Time-Dependent Problems

In most time-dependent applications, multigrid methods are only used (if at all) to solve the discrete set of equations arising at each time step of some implicit time-stepping algorithm. Unfortunately, true integration of multigrid concepts into time-dependent applications is uncommon. There are several possible reasons for this. First, it is easy enough to be content with using multigrid as an implicit equation solver because it is relatively easy to do and many of the multigrid benefits already accrue. Second, many consider multigrid methods to be restricted to essentially elliptic PDE's, although they have long been applied successfully to nonelliptic problems. In applications, the inclusion of the time variable is almost always in a nonelliptic way, so there is some inclination to think that multigrid methods cannot be directly applied. A third possible reason is more philosophical than mathematical: there is an innate tendency to think of the time variable as inherently different from the space variables. A fourth possibility is that the storage requirements of a three-dimensional spatial application may well tax computer capacity, so that there is resistance to a space-time grid that may be seen as simply too large. A fifth reason, possibly the only truly legitimate one, is that there has been less research done on temporal than on spatial multigrid.

We now briefly look at some of the research that has been done on multigrid methods for time-dependent applications, most of which has focused on parabolic initial value problems.

3.1. FMG methods for time-stepping algorithms

Multigrid methods can be very effective even when they are restricted to use within an implicit time-stepping algorithm. Consider the parabolic initial value problem

$$u_t(\mathbf{x},t) - Lu(\mathbf{x},t) = f(\mathbf{x},t),\ \mathbf{x} \in D \subset \mathcal{R}^2,\ t \in [0,T],$$

where L is a second-order elliptic operator and the appropriate initial and boundary value conditions are supplied. The multigrid algorithm for solving this equation described in (Brandt and Greenwald, 1991) involves a standard sequential time-stepping procedure, but contains some important features that differentiate it from other methods. At each time step, the algorithm uses an FMG-like approach that uses coarser spatial grids to solve the increment equation, which is derived from the original PDE and describes the change in the solution from one time level to another. The previous discussion of parallelizing multigrid algorithms, particularly those dealing with the efficiency of FMG algorithms, would apply to the problem solved at each time step. In order to maintain proper balance between computational work and change in the solution, an adaptive procedure is advocated for deciding which grids must be used to represent the increment function accurately. The required accuracy corresponds to the level of discretization error of the increment equation on the finest grid. The point is that the tendency for parabolic problems to introduce only smooth changes over many time steps means that this accuracy can often be obtained using only coarser grids at most time levels. The resulting algorithm obeys Brandt's "golden rule" in that when there is little change in the solution (more precisely, little information in the change), then only coarse grids are used, so that little computational work is performed.

Other researchers have considered multigrid algorithms for time-dependent parabolic equations (for early work, see (Brandt, 1977, and Hackbusch, 1984). In addition to describing some earlier multigrid algorithms for parabolic problems, the book (Vandewalle, 1989) contains a description of an alternative algorithm, multigrid waveform relaxation, which we will say more about in the next section. This reference is particularly useful in that it emphasizes parallel computation.

3.2. Parabolic multigrid methods

In the multigrid algorithm for parabolic time-dependent problems discussed above, the FMG solver at each time step can be parallelized, but the time stepping procedure remains sequential. The basic idea of using multiple grids in time as well as space and processing these grids in parallel is not new (see Brandt, 1977, Brandt, 1989, and Hackbusch, 1984); however, only fairly recently have parallel multigrid algorithms based on these concepts appeared in the literature We use the term parabolic multigrid (coined in Hackbusch, 1984) to refer to those algorithms that use multigrid principles on grids in time as well as space.

One approach is a time-parallel multigrid method based on Hackbusch's parabolic multigrid. An algorithm using this approach is presented in (Horton, 1992) that allows computations on different time levels to proceed in parallel. However, this algorithm applies to a fairly limited class of parabolic problems; the article (Horton et al., 1995) discusses these limitations and presents an alternative, and more widely applicable, algorithm based on a combination of multigrid waveform relaxation and parallel cyclic reduction. Although the method is not presented in this way, because of the close connection between cyclic reduction and multigrid (see Schaffer, to appear), it is possible to view this method as a temporal multigrid algorithm. An algorithm perhaps more in the true multigrid spirit is presented in (Horton and Vandewalle, 1995), which treats the entire space-time domain in a multigrid way. The algorithm is characterized by an adaptive coarsening strategy (semi-coarsening in time or space). The paper contains the details of the algorithm, including comments about its potential parallel performance and some promising early results for simple model problems. Because this algorithm has yet to be implemented on a parallel computer, much less tested on more realistic problems, assertions about its practicality are premature. However, we see this as a step in the direction toward implementation of a full time-space parallel multigrid method.

4. Conclusions

Easy access to powerful parallel computers will soon bring about significant changes in the field of computational mathematics. The large increase in computing power afforded by these machines will lead many scientists to model larger and more complicated physical systems, and in some cases bring new disciplines, such as computational chemistry, into the large-scale computational arena. While it is impossible to predict the number of processors that future machines may have, scientists will likely model systems that will continue to tax machine capacity. This suggests that the most efficient algorithms for such machines will be those derived from algorithms that are efficient in serial computation, and it leads us to believe that multigrid methods will remain an important computational tool even as serial computing gives way to parallel computing.

While we are optimistic about the future of parallel multigrid methods, there is clearly much work to be done. Real parallel multigrid applications are uncommon in the literature. This is partially because developing parallel codes is currently a difficult task that can be compounded by the incorporation of multigrid methods. One active, and potentially very useful, direction of research is the development of software that would make the coding of parallel adaptive multigrid methods (and some other grid based algorithms) much simpler for the developer. See (Lemke et al., 1993, and Ritzdorf, et al., 1994) for some significant progress in this direction. In a perfect world, software like this would handle all of the questions of domain partitioning and processor assignment for a particular parallel computer. The code developer would be left with the task of designing an efficient standard multigrid algorithm for the particular application.

For very large-scale applications on the current generation of parallel computers, an efficient parallel implementation of the standard parallel multigrid algorithm without local refinement will outperform the more complicated concurrent multigrid algorithms. However, the use of multiple coarse grids in parallel may improve the robustness of a multigrid algorithm and thus possibly result in a more efficient solver. In practice, multiple coarse grids are useful to represent error frequencies that cannot be represented on a single coarse grid.

Of course, to achieve the most from a particular computer, adaptive local refinement is needed. In most applications, using a uniform global grid is inefficient because the mesh is either too fine in regions where the solution is smooth, or too coarse in regions where it is not. In applications with local refinement, the sequential treatment of levels can severely limit efficiency: there may well be no single level with enough grid points to keep all of the processors busy. Here, concurrent processing of levels will likely result in a more efficient algorithm. To press a computer's capabilities to the limit, one should consider adaptive refinement of a space-time grid, and some of preliminary work that has been done on multigrid for space-time grids is promising. It is our belief that a coupling of multigrid ideas and parallel computing is perhaps one way that will allow realistic three-dimensional time-dependent simulations of complicated physical systems. We are obviously not there yet, but it is our hope that this paper has shown some, though certainly not all, of the significant steps that have been made toward this goal.

Acknowledgements

The authors would like to thank those who assisted with this project, especially those who attended the informal discussion group at the Seventh Copper Mountain Conference on Multigrid Methods and Paul Frederickson for helping organize it. We wish to personally thank Paul and John Van Rosendale for their helpful comments. Finally, the authors would like to acknowledge the unavoidable de-

ficiency of this paper: any brief overview of such a broad field will no doubt omit a lot of important work. With this in mind, the interested reader is encouraged to consider further study of this field, which might well begin with the two previous survey articles (Chan and Tuminaro, 1987, and McBryan et al., 1991).

References

Bastian, P., 1993. "Parallel adaptive multigrid methods," Technical Report IWR 93-60, Interdisciplinary Center for Scientific Computing, Universität Heidelberg.

Bramble, J.H., 1993. *Multigrid Methods,* Pitman Research Notes in Mathematical Sciences, **294**, Longman Scientific & Technical, Essex, England.

Bramble, J.H., Pasciak, J.E., and Xu, J., 1990. "Parallel multilevel preconditioners," *Math. Comp.*, **55**, pp. 1–22.

Brandt, A., 1977. "Multi-level adaptive solutions to boundary-value problems," *Math. Comp.*, **31**, pp. 333–390.

Brandt, A., 1981. "Multigrid solvers on parallel computers," *Elliptic Problem Solvers*, M.H. Schultz, ed., Academic Press, pp. 39–83.

Brandt, A., 1984. *Multigrid techniques: 1984 guide with applications to fluid dynamics*, GMD-Studien Nr.85. Gessellschaft für Mathematik und Datenverarbeitung, St. Augustin.

Brandt, A., 1989. "The Weizmann Institute research in multilevel computation: 1988 report," *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods,* J. Mandel et al., ed., SIAM, Philadelphia, pp. 13–53.

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

Brandt, A., and Greenwald, J., 1991. "Parabolic multigrid revisited," *Multigrid Methods III,* W. Hackbusch and U. Trottenbergb, eds., Birkhäuser Verlag, pp. 143–154 (of ix + 394).

Briggs, W.L., 1987. *A Multigrid Tutorial,* SIAM, Philadelphia.

Briggs, W.L., Hart, L., McCormick, S.F., and Quinlan, D., 1988. "Multigrid methods on a hypercube," *Multigrid Methods: Theory, Applications, and Supercomputing,* S.F. McCormick, ed., in *Lecture Notes in Pure and Applied Mathematics,* Marcel Dekker, New York, pp. 63–83.

Chan, T.F., and Tuminaro, R.S., 1987. "Design and implementation of parallel multigrid algorithms," *Proceedings of the Third Copper Mountain Conference on Multigrid Methods,* S.F. McCormick, ed., Marcel Dekker, New York, pp. 101–115.

Chan, T.F., and Tuminaro, R.S., 1987. "A survey of parallel multigrid algorithms," *Proceedings of the Symposium on Parallel Computations and their Impact on Mechanics,* ASME, New York.

Decker, N.H., 1991. "Note on the parallel efficiency of the Frederickson-McBryan multigrid algorithm," *SIAM J. Sci. Stat. Comput.,* **12**, pp. 208–220.

Degani, A.T., and Fox, G.C., 1995. "Parallel multigrid computation of the unsteady incompressible Navier-Stokes equations," Technical Report SCCS 739, Northeast Parallel Architectures Center at Syracuse University.

Dendy, J.E., 1995. "Revenge of the semicoarsening frequency decomposition multigrid method," *Proceedings of the Seventh Copper Mountain Conference on Multigrid Methods.*

Dendy, J.E., Ida, M.P., and Rutledge, J.M., 1992. "A semicoarsening multigrid algorithm for SIMD machines," *SIAM J. Sci. Stat. Comput.,* **13**, pp. 1460–1469.

Douglas, C.C., 1991. "A tupleware approach to domain decomposition methods," *Appl. Numer. Math.,* pp. 353–373.

Douglas, C.C., and Douglas, M.B., 1995. "MGNet Bibliography," In `mgnet/bib/mgnet.bib`, on anonymous ftp server `casper.cs.yale.edu`, Yale University, Department of Computer Science, New Haven, CT.

Douglas, C.C., and Miranker, W.L., 1988. "Constructive interference in parallel algorithms," *SIAM J. Numer. Anal.,* **25**, pp. 376–398.

Frederickson, P.O., and McBryan, O.A., 1988. "Parallel superconvergent multigrid," *Multigrid Methods: Theory, Applications, and Supercomputing,* S.F. McCormick, ed., in **10** of *Lecture Notes in Pure and Applied Mathematics,* Marcel Dekker, New York, pp. 195–210.

Gannon, D.B., and Van Rosendale, J., 1986. "On the structure of parallelism in a highly concurrent PDE solver," *J. Parallel Distrib. Comput.,* **3**, pp. 106–135.

Greenbaum, A., 1986. "A multigrid method for multiprocessors," *Appl. Math. Comput.,* **19**, pp. 75–88.

Hackbusch, W., 1984. "Parabolic multi-grid methods," *Computing Methods in Applied Sciences and Engineering VI,* R. Glowinski and J.-L. Lions, eds., Amsterdam, North Holland.

Hackbusch, W., 1989. "The frequency decomposition multigrid method, part I: Application to anisotropic equations," *Numer. Math.,* **56**, pp. 229–245.

Hart, L., and McCormick, S.F., 1989. "Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Basic ideas," *Parallel Comput.,* **12**, pp. 131–144.

Horton, G., 1992. "The time-parallel multigrid method," *Comm. Appl. Num. methods,* **8**, pp. 585–595.

Horton, G., and Vandewalle, S., July 1995. "A space-time multigrid method for parabolic PDEs," *SIAM J. Sci. Stat. Comput.,* **16**(4).

Horton, G., and Vandewalle, S., and Worley, P., May 1995. "An algorithm with polylog parallel complexity for solving parabolic partial differential equations," *SIAM J. Sci. Stat. Comput.,* **16**(3).

Lemke, M., Witsch, K., and Quinlan, D., 1993. "An object-oriented approach for parallel self adaptive mesh refinement on block structured grids," *Sixth Copper Mountain Conference on Multigrid Methods,* N.D. Melson, T.A. Manteuffel, and S.F. McCormick, eds., **CP 3224**, NASA, Hampton, VA, pp. 345–359.

Matheson, L.R., 1994. *Multigrid Algorithms on Massively Parallel Computers,* PhD thesis, Department of Computer Science, Princeton University, Princeton, NJ.

Matheson, L.R., and Tarjan, R.E., 1994. "A critical analysis of multigrid methods on massively parallel computers," Technical Report TR-448-94, Department of Computer Science, Princeton University.

McBryan, O.A.,Frederickson, P.O., Linden, J., Schüller, A., Solchenbach, K., Stüben, K., Thole, C.A., and Trottenberg, U., 1991. "Multigrid methods on parallel computers, a survey of recent developments," *Impact Comput. Sci. Eng.,* **3**, pp. 1–75.

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

McCormick, S.F., 1992. *Multilevel Projection Methods for Partial Differential Equations,* **62**, of *CBMS-NSF,* SIAM, Philadelphia.

McCormick, S.F., and Quinlan, D., 1989. "Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Performance results," *Parallel Comput.,* **12**, pp. 145–156.

McCormick, S.F., and Quinlan, D., 1992. "Idealized analysis of asynchronous multilevel methods," *Proceedings of the Symposium on Adaptive Multilevel and Hierarchical Computational Strategies,* ASME, New York, pp. 1–8.

Mulder, W.A., 1989. "A new multigrid approach to convection problems," *J. Comput. Phys.,* **83**, pp. 303–323.

Naik, N.H., and Van Rosendale, J., 1993. "The improved robustness of multigrid elliptic solvers based on multiple semicoarsened grids," *SIAM J. Numer. Anal.,* **30**, pp. 215–229.

Ritzdorf, H., Schüller, A., Steckel, A.B., and Stüben, K., 1994. "L_iSS – An environment for the parallel multigrid solution of partial differential equations on general 2D domains," *Parallel Comput.,* **20**, pp. 1559–1570.

Schaffer, S. "A semi-coarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients," To appear in *SIAM J. Sci. Stat. Comput.*

Tuminaro, R.S., and Womble, D.E., 1993. "Analysis of the multigrid FMV cycle on large scale parallel machines," *SIAM J. Sci. Stat. Comput.,* **14**, pp. 1159–1173.

Vandewalle, S., 1993. *Parallel Multigrid Waveform Relaxation for Parabolic Problems,* B.G. Teubner Verlag, Stuttgart.

Xu, J., 1992. "Iterative methods byi space decomposition and subspace correction," *SIAM Review,* **34**, pp. 581–613.

Yserentant, H., 1993. "Old and new convergence proofs for multigrid methods," *Act Numerica,* pp. 285–326.