Multigrid and Multilevel Methods in Science and Engineering

Craig C. Douglas

The fundamental idea behind all multigrid methods is to compute using different scales in order to obliterate error components of different scales. Error components can be thought of as waves. A long wave, which has low frequency, on one grid is a short wave, or high frequency, on a coarse enough related grid. A number of iterative methods (e.g., relaxation and alternating direction implicit methods) are very efficient at damping short wave error components, but not not as efficient at damping long wave error components. A particularly well written motivation for multigrid methods can be found in Brigg's book [1].

Closely related to multigrid are multilevel methods. For problems arising from partial differential equations on grids, there is no real difference between these methods. For problems in which the grid is not used in forming the coarser approximations or for problems in which there never was a grid, there is a methodology that is quite similar to multigrid. A sequence of problems of various sizes are solved (iteratively) with prolongation and projection methods used to transfer information between the solution spaces on each level.

Multigrid (and multilevel methods) are not terribly new. In fact, one variant predates electronic computers by decades if not more than a century. Further, it generalizes to a quite common method in use in many engineering problems.

Multigrid methods are well known to be difficult to make work on nontrivial problems. Yet they have flourished during the 1980's and 1990's to become quite standard iterative methods. This is due to their rapid convergence rates and low work estimates. A general realization occured that if a "unilevel" code was well designed in the first place, adding a multilevel procedure to it was not as difficult as it appeared to be.

While multigrid is not the first procedure for solving elliptic partial differential equations (PDE's) in an optimal order of time, it is the first do this and be both stable and use an optimal order of space simultaneously. This has led the multigrid community to extend these methods to many problems which are not elliptic PDE's.

It should be noted that when solving quite complicated problems, multigrid does not usually run in the linear run time commonly associated with, say, Poisson's equation on a square with a uniform mesh. For many problems, multigrid lowers the work estimate by a polylog factor. For problems with hundreds of thousands or millions of unknowns, this is quite a substantial speed up.

Multigrid and multilevel methods are now commonly used in a number of fields. These include aerospace simulation (flow over a wing, airplane, missile, or space shuttle), petroleum engineering (reservoir or pipeline simulation), environmental studies (pollution tracking, acid rain tracking, ocean modeling, weather prediction, and hurricane tracking), combustion (stable or turbulent flames), and computers (displaying video signals).

Historical perspective

There have been several interesting periods in the history of multigrid or multilevel methods. In most cases, a new collection of people entered the field simultaneously and made some new discoveries.

Before 1960

In the era of personnel computing (mostly before 1960), a common computing methodology was to collect a group of people into a room and assign part of a domain to each person. People computed (typically synchronously, but not necessarily) using relaxation methods. A common procedure for generating the initial guess on this grid was to solve the problem first on a much coarser grid with about one tenth as many grid points. This approximate solution was interpolated onto the fine grid. Then parallel computing began on the personnel computers. In his 1940 book, Southwell [2] describes such a process as common in British aeronautics companies in the 1920's.

I have been maintaining for some years that parallel multigrid is a natural progression from single processor multigrid. After all, it is just doing what people were doing for years by hand.

1961–1971

The Soviet school of numerical mathematics produced four major papers on this topic during this period. Essentially all of these papers were ignored in the West for many years after they were written until Achi Brandt popularized the method during the middle 1970's.

The 1961 and 1962 papers by Federenko described in complete detail a multigrid procedure for solving Poisson's equation on a square with uniform grid elements. A central difference discretization was employed. The convergence rate was determined using Fourier analysis that included the boundary conditions.

The 1966 paper by Bakhvalov analyzed general, second order, variable coefficient elliptic problems on a square domain with a uniform mesh. A correction scheme (see later) was analyzed analytically. For N unknowns, this method was proven to converge to the order of the truncation error in work proportional to $N \log N$. Had Bakhvalov started from the coarsest level instead of the finest one, he would probably have discovered the optimal order work estimate later associated with Brandt.

The 1971 paper by Astrakhantsev analyzed a finite element method solved using multigrid. Once again, the optimal order work estimate was just missed. There is quite a similarity between this paper and a later one by Bank and Dupont.

Back in the West, Wachspress in 1966 invented a multiplicative multigrid method for solving nuclear reactor problems. Up to this point, all multigrid methods had been additive. Wachspress did not realize the connection with multigrid until years later, however.

A similar set of algorithms, known as aggregation-disaggregation methods, were becoming common tools in the economics field. Both additive and multiplicative multilevel methods were devised. These methods eventually were applied to circuit simulation problems and electrical power networks.

The Middle to Late 1970's

Achi Brandt is known as the father of multigrid methods with good reason. He was one of the first to recognize its potential and was willing to stand on desk tops (literally) at conferences and point out its advantages in a way that got people's attention and convinced many. Brandt's 1977 paper [3] is still considered the origin of modern multigrid by many.

Brandt has not been a multigrid theoretician. While he uses local mode analysis to estimate multigrid rates of convergence, this technique does not offer a rigorous method of computing convergence rates. In fact, it underestimates the rate which can lead to predictions of convergence when the procedure actually diverges.

During this period, there were several theoreticians whose works still stand out. Hackbusch, Wesseling, Hemker, Bank, and Nicolaides all wrote or co-wrote pivotal papers. Many of these papers were written at about the same time with no knowledge of the others' works. The results were in some cases quite similar. Unlike on the World Wide Web (WWW) of today, preprints were quite difficult to acquire at this time.

The Big 80's

In my opinion, if there will ever be a golden age for multigrid, it will be the 1980's. A tremendous amount of theory was published in this era. The method was applied to a large number of application areas as a standard solution method. Finally, the multigrid practitioners gelled into a fairly cohesive community that could disseminate information rapidly.

Four sets of multigrid conferences began during the 1980's. The European multigrid conferences, the Copper Mountain conferences, the GAMM workshops (in East Germany for many years), and the Oberwolfach workshops.

Each of these has provided a wealth of papers and advances that continue to this day. This was probably the single largest advance in the field in this era.

Multigrid became useful in more areas during this period. There was a new collection of people producing papers with new theoretical tools. This extended the field into areas like hyperbolic and parabolic differential equations. In particular, several theoretical tools became common during this period.

Discussions over which of two methods for solving nonlinear elliptic PDE's were put to rest (more on this later) by theoretical analyses of the competing methods.

Douglas (in my dissertation and later published in [4]) provided a tool for analyzing the convergence rate abstractly without having to come up with a different proof for each of several discretizations. This led to a paper by Bank and Douglas [5] which produced a theoretical tool for sharply estimating convergence rates for nontrivial problems.

Papers by Braess, Mandel, McCormick, Parter, Verfürth, and Yserantant extended the known theory to numerous multigrid algorithms and model problems without regard to the machine architecture. A number of parallel multigrid algorithms were developed, analyzed, and experimented with on actual machines during this period by numerous groups of people.

Hierarchical basis multigrid [6] became an interesting tool, particularly in finite element multigrid solvers based on adaptively refined meshes. In many cases this is a slower solution method than more standard adaptive mesh multigrid methods, but has the rather distinct advantage of being provably convergent.

Bramble and Pasciak [7] developed a regularity free theory. This theory requires that the solvers used on each level commute with the differential operator. Since this eliminates the standard variant of Gauss–Seidel, many in the multigrid field do not consider this to be an advantage.

The 1990's

A significant portion of new multigrid research has been motivated in this period so far by three computational developments which have profoundly changed the applied mathematics field in general.

The first was the easy availability of extremely fast, relatively inexpensive

RISC based work stations that when first released were on the order of a Cray supercomputer of about 3 years earlier. Suddenly, people who did not have $30,000,000 computer budgets could do very high performance computing at their office or home.

The second was the appearance of reliable parallel computers. These are typically made up of a hierarchy, network, or cache of workstations (known as HOW, NOW, COW, respectively) or a network of servers (NOS). Manufacturers can produce them cheaply as a side effect of their workstation and server lines and can afford to update them quite often (once a year is typical). This is much more often than the traditional supercomputer industry. Further, these machines scale well as far as the manufacturers are concerned.

Suddenly, many more people were computing in what was once the stratified air of truly high performance computing. Many of these people were used to reading the literature and were aware of multigrid methods, but had not had any reason to use them yet. When a problem takes a CPU day or week by conventional methods, multigrid becomes an interesting method to explore. Based on the number of multigrid papers appearing per year during this period [8], probably 85% per year fit this category.

At the same time, some very interesting work has been done in both the multigrid and domain decomposition communities trying to determine when it makes sense to use multigrid inside of a domain decomposition procedure or to use domain decomposition inside of a multigrid procedure (more on this later). In general, when one works, so does the other.

The third development was the explosion of information on the Internet. MGNet began in 1991 as a monthly digest for multigrid and domain decomposition issues. It quickly became a resource for storing in one place codes, preprints, conference proceedings and announcements, and an attempt at all of the multigrid citations. The world wide web has also promoted fast information dissipation.

Theory continued forward in the areas of nonconforming finite element methods, methods and analysis for nonlinear elliptic problems, and multilevel domain decomposition methods. Multigrid methods became quite common in extremely complicated application areas.

What Do Multilevel Algorithms Look Like?

Before giving a mathematical set of definitions, let's see how multilevel algorithms actually compute on different scales. We will look at several examples in different contexts. First we will see how different grids are used in basic multigrid algorithms. Second we will see how different types of grids are used.

Computational Wanderings by Level

Two common algorithms which arise in many different areas of multigrid are the *V cycle* and the *W Cycle*. How computation moves from one level to another is pictured in Figure 6. On each level, a different scale for the problem is used.

The V cycle starts on the finest level and traverses all of the grids, one at a time, until it reaches the coarsest grid. Then it traverses all of the grids again until it reaches the finest level. From a recursive algorithm viewpoint, the V cycle uses the coarser levels once per level for corrections.

The W cycle is similar except that it does two corrections per level. For many problems, a W cycle is considered more robust and easier to analyze theoretically than a V cycle. W cycles are much harder to parallelize efficiently due to their spending more time on the coarser levels than a V cycle.

Another class of algorithms start on the coarsest level and wind their way to the finest level. These are sometimes referred to as *nested iteration* methods since they use standard correction multilevel algorithms inside their definitions. How they move from one level to another is pictured in Figure 7.

The oldest one is the one way multigrid algorithm. This never computes a correction problem on a coarser level. The coarser levels' problems are solved sequentially and used (after interpolation) as the initial guess to the next finer level's problem. While not of optimal order for work, this is surprisingly effective for hard engineering problems. (In fact, it has been the standard algorithm for combustion problems for at least 25 years.) It is the generalization of the method described by Southwell in his 1940 book [2].

The nested iteration V cycles and W cycles are also quite common. There are also a number of hybrid methods which have been defined and analyzed over the years.

One of the charming aspects of the multigrid community is that depending on which school you follow, the definitions are not necessarily identical for these algorithms. For example, some would argue that the W cycle picture in Figure 6 should have two sets of corrections from level 3. Another example is the nested iteration W cycle, which is pictured in Figure 7. This is sometimes referred to as an F cycle. However, the F cycle is a hybrid correction scheme: one V cycle followed by one F cycle, where on the finest grid only the V cycle is performed.

It all comes down to how the correction algorithms are defined on the finest level: either one correction from that level or possibly many. Since the definitions are usually recursive, it has always seemed to me that the definition had better be for one correction from the finest level if the recursive definition is going to make sense mathematically.

Different Scales

Let us consider now what is meant by computing on different scales. The most simple examples are where the grids have different mesh spacings. Consider Figure 2.

On the left side of Figure 2 is an example of a uniformly refined set of grids. The scales are quite clearly differentiated by a factor of two. Hence, what takes four grid lines to represent on Level 3 takes only two on Level 2 and one on Level 1.

On the right side of Figure 2 is an example of a set of adaptively (or nonuniformly) refined grids. This type of grid refinement usually is seen only in finite element multigrid solvers coupled with a nested iteration algorithm. The scales are different only in the region of the domain in which the solution is not adequately resolved on a coarse grid.

For adaptively chosen grids, there are several variants of multigrid. One is to have basis functions for all of the elements on a level. Another is to have basis functions for only the new elements forming a hierarchical set of basis functions.

A third approach is to use a domain decomposition approach and to have locally uniform meshes which may overlap. There are many advantages to this approach including that most of these codes run fast on RISC based work stations and vectorize well for certain classes of supercomputers.

A set of methods is being developed where only the finest grid is given (and completely unstructured) and the coarser grids must be constructed from this. Just removing vertices does not work since many elements then lose their triangular or tetrahedral shapes. A Delauney triangulation procedure is frequently applied. This procedure identifies which vertices will be in the new grid and produces triangles or tetrahedra to fit.

Implementing multigrid on unstructured grids leads to a number of interesting data structure issues. Without a great deal of care these codes run at the speed of the integer unit of a CPU since the floating point operations become less than 10% of the total operations. A good source of information on this topic is contained in [9].

There are several disjoint groups in multigrid. Two which are constantly in conflict are those who refine given coarse grids to get finer grids and those who coarsen given fine grids to get coarse grids. Each group tends to number levels in the opposite order of the other. Further, they complain loudly to people who provide free software that number the grids in the "wrong" order.

Algorithms

There are two primary algorithms for each of the following areas:

- linear problems
- nonlinear problems
- time dependent problems
- telescoping parallel algorithms
- nontelescoping parallel algorithms

Inside each primary algorithm there may well be a number of well known (and used) variants.

As can be expected in a highly competitive field, there have been numerous very heated discussions over which is best in certain areas. There was a famous "discussion" at a cocktail party before a conference dinner in 1980 over nonlinear methods which is still sometimes referred to in the multigrid folklore as the two martini argument.

Linear Problems

Multigrid was first recognized (Federenko) to be a highly efficient algorithm for the numerical solution of simple elliptic partial differential equations, e.g., Poisson's equation on a square domain.

For linear problems, we can formulate them into a sequence of linear systems of equations, e.g.,

$$A_i u_i = f_i, \quad 1 \le i \le k. \tag{1}$$

For PDE's, the matrices A_i are derived using some discretization method.

Suppose the order of each matrix A_i is N_i and that $N_i \le N_{i+1}$. In particular, many multilevel methods assume that

$$N_{i+1} \approx \sigma N_i, \quad \sigma \in \mathbb{R}, \quad \sigma \ge 1. \tag{2}$$

A common form for σ in a d-dimensional problems is $c2^d$, where c is derived from the discretization method and how the grid is refined ($c = 1$ is the most common, however).

The primary linear algorithm, a *correction* one, is defined loosely as follows:

Algorithm MGC (j, $\{A_i\}$, x_j^0, f_j)
  Approximately solve $A_j x_j = f_j$ starting from an
  initial guess of x_j^0.
  If $j > 1$, then possibly do until a condition is met:
    Calculate a residual $r = f_j - A_j x_j$.
    Set $x_{j-1}^0 = 0$ and project r onto f_{j-1}.
    Get u_j by interpolating the result of
      MGC($j-1$, $\{A_i\}$, x_{j-1}^0, f_{j-1}).
    Approximately solve $A_j x_j = f_j$ starting from
    an initial guess of $x_j + u_j$.
  Return x_j as the approximate solution.

This algorithm starts computation on the finest level and uses the coarser levels only to solve correction problems. This works well when a good initial guess on the finest level is available.

There are several pieces of this algorithm which are major computational issues in their own right. The approximate solve steps are usually an iterative method on all but possibly the coarsest level. On the coarsest level, a direct solver is used by many. This step is frequently referred to in the multigrid community as the smoothing steps. See the detour, *What Are Smoothers and Roughers*?

Two other steps are the grid transfer steps, namely, interpolation and projection. (For problems not arising from grids, the corresponding nomenclature is prolongation and restriction.) Both of these steps involve a small amount of the total running time, but are crucial to making multigrid work. See the detour, *Grid or Level Transfer Operators*.

The secondary linear algorithm, a *nested iteration* one, is defined loosely as follows:

> Algorithm NIC (j, $\{A_i\}$, x_1^0, $\{f_i\}$)
>   Get x_1 from the result of MGC(1, $\{A_i\}$, x_1^0, f_1).
>   If $j > 1$, then do $i = 2, \cdots, j$:
>   Get x_i^0 by interpolating x_{i-1}.
>   Get x_i from the result of MGC(i, $\{A_k\}$, x_i^0, f_i).
>   Return x_j as the approximate solution.

This algorithm is sometimes referred to as *Full Multigrid* or *FMG*.

Algorithm NIC starts computation on the coarsest level and proceeds to solve problems on each level in order to get an initial guess on the next finer level until the level j problem is solved. The coarser levels are also used to solve correction problems using Algorithm MGC, hence, the name of the algorithm.

One way multigrid corresponds to Algorithm NIC with the main part of Algorithm MGC not performed. Only the initial approximate solve step is executed.

As noted earlier, for problems in which you have a good initial guess, Algorithm MGC is a very good choice. For problems in which you do not have an initial guess, Algorithm MGC is not a good choice, but NIC is.

There is another fundamental difference between these two algorithms. Suppose the cost of computing the approximate solve and the grid transfers on any level i is CN_i. In order to reduce the error to of the order of the truncation error of the discretization method on the j^{th} level, Algorithm MGC would normally cost $O(N_j \log N_j)$ while Algorithm NIC would normally cost $O(N_j)$. This difference occurs because Algorithm NIC must only reduce the error by a constant factor on each level (other than the coarsest one) if each level's problem is solved to of the order of the truncation error.

Linear problems that are suitable to multilevel approaches come in many well hidden forms. A good example is how many computer display adapters and displays interact to put an image on a screen. The adapter receives data in the form of a graphics image. This is coarsened using a discrete Fourier transform (DFT) several times ($4-8$ typically). The much reduced image is transferred across the connecting wires to the display. The same number of inverse DFT's are applied before the image appears on the screen. This is just a V cycle with one smoother before the coarse grid correction (the DFT) and another smoother after the correction (the inverse DFT). A multilevel wavelet approach to this can be substituted for the DFT.

Nonlinear Problems

For nonlinear problems, multigrid can be applied in two extremely different manners. It can be applied directly using nonlinear iterative methods (e.g., nonlinear Gauss-Seidel) or indirectly as part of a linearization technique (e.g., Newton's method).

There are applications where one of these sets of methods is clearly superior to the other. What seems to make the difference is the cost of function evaluations. When this is inexpensive, the direct application is more cost effective than computing a Jacobian matrix, which is required by indirect approaches.

All nonlinear multigrid algorithms assume that the original problem has one or more solutions (normally only one is assumed, however). Other assumptions are swept under the rug, but are closely related to the choices of iterative solvers, solution spaces, and the discretization method.

Direct applications of multigrid to nonlinear problems are frequently modifications of Algorithm MGC. The most popular of these algorithms is the *Full Approximation Scheme*, or *FAS*. It is defined loosely as follows:

Algorithm FAS (j, $\{A_i\}$, x_j^0, f_j)
  Approximately solve $A_j x_j = f_j$ starting from an
  initial guess of x_j^0.
  If $j > 1$, then possibly do until a condition is met:
    Calculate a residual $r = f_j - A_j x_j$.
    Project x_j onto x_{j-1}^0.
    Set $f_{j-1} = A_{j-1}x_{j-1}^0 -$ the projected r.
    Get u_{j-1} from the result of
      FAS($j-1$, $\{A_i\}$, x_{j-1}^0, f_{j-1}).
    Get u_j by interpolating $u_{j-1} - x_{j-1}^0$.
    Approximately solve $A_j x_j = f_j$ starting from

an initial guess of $x_j + u_j$.
Return x_j as the approximate solution.

When the A_i's are linear operators, this is equivalent to Algorithm MGC. Algorithms FAS and NIC can be combined to get what is sometimes referred to as the *Full Multigrid-FAS* or FMG-FAS.

There are many components that can be shared between a linear and a FAS multigrid solver. For example the interpolation and projection routines can be the same. The major difference is in the solvers used on each level: specialized relaxation methods for nonlinear problems are usually constructed (see [10] for some examples). Here, the iterative solvers typically require a large number of function evaluations. However, the Jacobian is not formed, which can be a quite large savings.

Indirect methods are typically modifications of Algorithm NIC. The secondary nonlinear algorithm, a *Newton multigrid* one, is defined loosely as follows:

Algorithm NMG (j, $\{A_i\}$, x_1^0, $\{f_i\}$)
Do $i = 1, \cdots, j$:
If $i > 1$, then get x_i^0 by interpolating x_{i-1}.
Do $s_i = 1, \cdots$ until convergence is met:
Form the Jacobian $J_i^{(s_i)}$ using x_i^0.
Get x_i from the result of MGC(i, $\{J_k\} \cup J_i^{(s_i)}$, x_i^0, f_i).
If x_i is not adequate, set $x_i^0 \leftarrow x_i$.
Set $J_i = J_i^{(s_i)}$.
Return x_j as the approximate solution.

The hope in all of the Newton multigrid variants is that as the number of the level increases that the number of Jacobian evaluations is reduced to one. There are several variants of Algorithm NMG, including ones by Bank, Hackbusch, and Reusken.

Algorithm NMG is really just Algorithm NIC with a twist. For each level, there are s_j Newton-like iterations where $s_j \rightarrow 1$ as $j \rightarrow \infty$. The linear correction steps in Algorithm MGC just use the last Jacobian formed on a level.

Once again, there are many components that can be shared between an existing linear and a Newton multigrid solver. For example the interpolation, projection, and solver routines can be the same. Absolutely standard linear equation solvers are used. The additional routines are for the Jacobian evaluation and convergence tests.

Bank was the first to provide a convergence result for nonlinear multigrid using a damped Newton procedure. The indirect approaches have many theoretical advantages since they are relatively straight forward to analyze. FAS, on the other hand, is notoriously hard to analyze.

Even today at Copper Mountain multigrid conferences, one of the easiest ways of starting a heated and lively discussion is to declare to a group of random participants that application area X (take your choice from semiconductors, reservoir simulation, flame simulation, etc.) is *obviously* best solved by either FAS or a damped Newton multigrid method. Then just stand back and listen to the arguments from proponents of either methodology. It can be quite educational, particularly to graduate students who have only heard one side before.

Time Dependent Problems

There are two basic approaches to multigrid for time dependent problems. One is to apply it directly to well known time stepping methods. An indirect approach is to multigrid in both time and space.

Multigrid has become a standard tool for solving hyperbolic and evolutionary problems. Virtually all commercial airplane wings that have come into use since the early 1980's have been designed using multigrid based numerical simulation. Examples of what grids look like for these problems are found in Figures 8 and 9. These grids are nonnested and the finest grid is considered to be a given in this example. The coarser grids were produced by a coarsening strategy and a Delauney triangulation procedure completed each grid.

The direct approach is very simple. At each time step a stationary problem (linear or nonlinear) must be solved. The stationary problem is solved using a standard multigrid method. For parabolic problems, the stationary problem is an elliptic problem. If the convergence rate for the elliptic problem using the multigrid algorithm is γ, then the convergence rate for the parabolic problem is $c\gamma$, where $c < 1$.

For many applications, using multigrid in this manner is not useful. A number of excellent time extrapolation methods have been developed which can be coupled to conjugate gradient-like methods. Once the first few time steps are solved (slowly), the extrapolation method provides such a good initial guess to the solution on the next time step that only a few (e.g., $2-5$) iterations of the conjugate gradient-like method are necessary to move on to the next time step. Multigrid simply cannot compete in this situation.

The indirect approach was first investigated in the 1970's by Brandt and his colleagues [11] and more recently by Horton et al [12]. After the first few time steps, the multigrid algorithm does not always go to the finest level before changing time. Since the coarse grids have larger mesh spacing, the time steps can be larger, too.

For a number of application areas in which only a stationary solution is wanted, this method works and is extremely fast. However, this does not work for all stationary problems and a number of conditions are required to ensure convergence. Further, for problems in which the transient solutions are needed, this method obviously does not apply.

Parallel, Telescoping Algorithms

The multigrid methods we have considered so far have the property that the number of unknowns per level is reduced as the level number is reduced, i.e., $N_i < N_{i+1}$ as in (2). There are two quite distinct families of parallel multigrid algorithms based on maintaining this telescoping property (nontelescoping algorithms will be discussed shortly).

The first serious parallel multigrid article was by Brandt [13]. This paper covered the main issues that occur when domain decomposition is used on each level (see Figure 3). This was an invited paper at the 1980 Elliptic Problem Solvers conference in Santa Fe. While this paper does not have numerical experiments on existing machines (though the number of parallel computers in 1980 was pretty limited), it is just as relevant today as it was then.

There are two ways of combining multigrid and domain decomposition for parallel computers. One is to do multigrid with domain decomposition used to get the parallelism (see Figure 3), which we will refer to as *MG+DD*. The other is to first decompose the fine grid using a standard domain decomposition method and then do serial multigrid on each of the subdomain problems (see Figure 4), which we will refer to as *DD+MG*. Clearly multigrid people prefer the former approach and domain decomposition people prefer the latter.

For simple elliptic PDE's, MG+DD usually takes the same number of multigrid cycles as in the single processor case. The number of iterations of the iterative method may increase a bit, but not which cycle is used. The method is not usually $O(N/p)$ in time, but usually resembles $O((N/p)\log p)$.

On the other hand, DD+MG usually takes a constant number of iterations of the domain decomposition algorithm (say $C \approx 7-8$). The cost of the multigrid cycle on each subdomain problem takes about as much time as one iteration of MG+DD. Hence, DD+MG usually costs C times as much as MG+DD.

An interesting case study of MG+DD versus DD+MG can be found in [14]. Relatively simple to extremely difficult problems on complex domains are considered. In each case, MG+DD is about a factor of 10 faster than DD+MG. Of course, domain decomposition methods work on complicated domains that multigrid cannot readily be applied to.

In general, however, *when DD+MG works, so does MG+DD and vice versa.* This is not always the case, however. My favorite example is one suggested to me by Patrick Le Tallec and Jan Mandel. Consider the air system for an office building complex connected by relatively small, long hallways. DD+MG models the air flow and temperature distribution beautifully, but MG+DD misses the interesting physics completely.

The other approach to parallel, telescoping multigrid is designed specifically for massively parallel computers. By this, we assume that there is one processor per grid point on all levels. Gannon and Van Rosendale describe one such algorithm for single instruction, multiple data (SIMD) machines (see [15], but their 1982 ICASE research report 82-36 on this algorithm is infinitely more readable). The same computation occurs on all levels and points simultaneously. Hence, iterative methods like Jacobi or conjugate gradients can be used, but not Gauss-Seidel. The one drawback to this method is that twice as much communication is required as might be expected in order to maintain stability. This method is described in more detail in [16].

Parallel, Nontelescoping Algorithms

A completely different class of parallel multigrid algorithms has been developed based on keeping the number of unknowns the same on all levels. These were first developed for PDE problems that had odd properties which made it impossible to represent the error on a single coarse grid. Only by having multiple ones allowed a hybrid multigrid method to converge [17].

There are two basic approaches used in this style of algorithm. The direct approach is to coarsen grids by assigning each fine grid point to one of a set of coarse grids in a regular pattern. The indirect approach is to use the symmetry groups of the PDE to decompose the operator. Both approaches can be done in geometricly.

The direct approach is easiest to understand by considering a concrete example. Here we use a two dimension tensor product grid as in Figure 2. We can assign points in the finest grid to sets $1-4$:

$$\begin{array}{ccccccc}
3 & 4 & 3 & 4 & & 3 & 4 \\
1 & 2 & 1 & 2 & & 1 & 2 \\
3 & 4 & 3 & 4 & & 3 & 4 \\
1 & 2 & 1 & 2 & \cdots & 1 & 2 \\
 & & & \vdots & & & \\
3 & 4 & 3 & 4 & & 3 & 4 \\
1 & 2 & 1 & 2 & & 1 & 2
\end{array}$$

Hence, if the finest level is j, then there are 4 grids on level $j-1$. This can be done recursively, resulting in 4^{j-i} grids on level $j-i$. In three dimensions, 4 is replaced by 8.

There are several nontelescoping multigrid methods that have been explored. The three most common are the ones by Brandt and Ta'asan, Frederickson and McBryan, and Hackbusch. For oddly shaped domains, Dendy has spent a considerable amount of time (years, in fact) arguing that Brandt and Ta'asan's method is superior to the other two, particularly if you are trying implement one of these. From a convergence point of view, McBryan has demonstrated that his method is superior to the other two methods (assuming all three converge).

The indirect approach uses the fact that the domain can be folded in particular ways related to the symmetry groups of the PDE. This leads to two level methods with a possibly great many subproblems (8 for a square, $64-192$ for a cube). The subproblems are usually constructed so as to be mutually orthogonal under the natural inner product for the original problem. A number of papers have been written about this topic by myself and a collection of my co-authors, principally Miranker, Mandel, and Barry Smith.

In addition, these problems are themselves boundary value problems with predictable boundary conditions. Assuming all of the boundary conditions were Dirichlet originally, a combination of Dirichlet and Neumann conditions result in the subproblems. Other, simple boundary conditions can be handled automatically.

An 8 subproblem example for a square is contained in Figure 5. The figure is a representation because all of the subdomains would usually be mapped into the upper right quadrant and overlap with each other. Here it is pictured in a form that domain decomposition people are comfortable with.

An interesting fact is that many of the subproblems are similar to each other except for the boundary conditions. A sparse matrix solver was devised that solved problems of the form

$$(A_i + C_i)u_i = f_i,$$

where A_i is sparse and C_i is super sparse. This was devised to take advantage of the fact that the rows of C_i are almost entirely all 0's. Doing this allows for a parallel multilevel solver that uses much less storage than a conventional iterative or multigrid procedure. In [18] are all of the details.

An extensive description of both styles of parallel, nontelescoping multigrid is in [16] along with a list of citations.

Sources of information

There are a number of sources of information. There are all of the proceedings of the European, Copper Mountain, GAMM, and Oberwolfach conferences on multigrid methods. There are also the proceedings of the domain decomposition symposia.

On the Internet, there is a large repository available by anonymous ftp and the world wide web. There are numerous web sites which are cross linked.

MGNet and the Multigrid Digest

The multigrid community began an Internet special interest group (MGNet [19]) in 1991. A number of preprints, conference proceedings, computer packages, and a quite large BibTeX database of references are maintained on MGNet.

A monthly newsletter has been distributed electronically since 1991 (making it one of the older electronic digests on the Internet). To subscribe to the digests, send an e-mail message to

mgnet–requests@cs.yale.edu

with your preferred e-mail address in it.

MGNet can be accessed by anonymous ftp or through the world wide web. MGNet is located at Yale University (New Haven, CT, USA). The anonymous ftp site for MGNet is the following:

Machine (address)	casper.cs.yale.edu (128.36.12.1)
Directory	mgnet

The world wide web site for MGNet is the following:

http://casper.cs.yale.edu/mgnet/www/mgnet.html

University of Colorado(USA)

Two campuses of the University of Colorado are multigrid sources. The applied math group at the Denver campus has produced much theory over many years. See

http://www-math.cudenver.edu/faculty/

and look through various faculty members home pages.

The applied math program at the Boulder campus has acquired two senior people from the Denver campus. See

http://amath-www.colorado.edu/appm/appm.html.

Naval Postgraduate School (USA)

Van Henson maintains a lively home page with recent publications in it. He has connections with multigrid, Fourier transforms, image reconstruction, and wavelets. See

http://math.nps.navy.mil/ vhenson/.

ICASE (USA)

ICASE has long been investigating multigrid for use in CFD applications related to aerospace. There are papers here on virtually all aspects of multigrid. See

http://www.icase.edu.

University of California at San Diego (USA)

Scott Baden's group has been doing structured adpative mesh refinement multigrid for quantum chemistry. See

http://www-cse.ucsd.edu/users/baden/samr.html.

Various (USA)

Bramble's school of multigrid is now fairly dispersed. The principal sites to look at are the following:

http://msg.das.bnl.gov/pasciak/
http://www.math.psu.edu/xu/

Technical University of Munich (Germany)

The German Scientific Computing pages have a multigrid algorithms area. An online tutorial can be found in

http://www5.informatik.tu-muenchen.de/sci-comp/xwb/xwb.html.

A separate database that was developed at CERN is now found in

http://www5.informatik.tu-muenchen.de/MG/MG.html.

A modified version of these databases are now maintained on MGNet.

GMD St. Augustin (Germany)

The GMD, like ICASE, has long been investigating multigrid for a variety of applications. These include basic theory, how to implement it on a variety of parallel architectures, and various problems arising in physics and engineering. See http://www.gmd.de/GMD/Institutes/SCAI/scai_home.html.

Hamburg multigrid group (Germany)

Papers showing how to use multigrid to solve problems arising in fermions, neural networks, Monte Carlo lattice gauge theories are here. Also, a description of a C++ object oriented multigrid package is here. See

http://info.desy.de/pub/www/projects/MG/HMG.html.

Technical University of Chemnitz (Germany)

A number of papers on general multigrid issues are here. They are mostly authored by Jung, Rüde, and their colleagues. See

http://www.tu-chemnitz.de/~pester/sfb/spc.html.

Konrad-Zuse-Zentrum Berlin (Germany)

Some multigrid papers are here. The primary application area is modeling semiconductors. Deuflhard has also been working on applying finite element methods and multigrid to problems arising in medicine. See

http://www.zib-berlin.de/Numerik/.

CWI Numerical Mathematics Department (The Netherlands)

Hemker and Koren are noted for their work in defect correction, adaptive methods, CFD, singular perturbation problems, and robust multigrid solvers. See

http://www.cwi.nl/cwi/departments/NW.html.

References

[1] W. L. Briggs, *A Multigrid Tutorial.* Philadelphia: SIAM Books, 1987.

[2] R. V. Southwell, *Relaxation Methods in Engineering Science.* Oxford: Oxford University Press, 1940.

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

[4] C. C. Douglas, "Multi–grid algorithms with applications to elliptic boundary–value problems," *SIAM J. Numer. Anal.*, vol. 21, pp. 236–254, 1984.

[5] R. E. Bank and C. C. Douglas, "Sharp estimates for multigrid rates of convergence with general smoothing and acceleration," *SIAM J. Numer. Anal.*, vol. 22, pp. 617–633, 1985.

[6] R. E. Bank, T. Dupont, and H. Yserentant, "The hierarchical basis multigrid method," *Numer. Math.*, vol. 52, pp. 427–458, 1988.

[7] J. H. Bramble and J. E. Pasciak, "New convergence estimates for multigrid algorithms," *Math. Comp.*, vol. 49, pp. 311–329, 1987.

[8] C. C. Douglas and M. B. Douglas, "MGNet Bibliography." In mgnet/bib/mgnet.bib, on anonymous ftp server casper.cs.yale.edu, Yale University, Department of Computer Science, New Haven, CT. Last modified on January 6, 1996.

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

[10] K. Stüben and U. Trottenberg, "Multigrid methods: Fundamental algorithms, model problem analysis and applications," in *Multigrid Methods* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, (Berlin), pp. 1–176, Springer–Verlag, 1982.

[11] A. Brandt and N. Dinar, "Multigrid solutions to elliptic flow problems," in *Numerical Methods for Partial Differential Equations* (S. Parter, ed.), pp. 53–147, New York: Academic Press, 1979.

[12] G. Horton, "The time-parallel multigrid method," *Comm. Appl. Num. Methods*, vol. 8, pp. 585–595, 1992.

[13] A. Brandt, "Multigrid solvers on parallel computers," in *Elliptic Problem Solvers* (M. H. Schultz, ed.), pp. 39–83, New York: Academic Press, 1981.

[14] U. Gärtel and K. Ressel, "Parallel multigrid: grid partitioning versus domain decomposition," in *Proceedings of the 10th International Conference on Computing Methods in Applied Sciences and Engineering* (R. Glowinski, ed.), (New York), pp. 559–568, Nova Science Publishers, 1992.

[15] D. B. Gannon and J. R. Rosendale, "On the structure of parallelism in a highly concurrent pde solver," *J. Parallel Distrib. Comput.*, vol. 3, pp. 106–135, 1986.

[16] C. C. Douglas, "A review of numerous parallel multigrid methods," in *Applications on Advanced Architecture Computers* (G. Astfalk, ed.), Philadelphia: SIAM, 1996.

[17] A. Brandt and S. Ta'asan, "Multigrid methods for nearly singular and slightly indefinite problems," in *Multigrid Methods II* (W. Hackbusch and U. Trottenberg, eds.), (Berlin), pp. 99–121, Springer–Verlag, 1986.

[18] C. C. Douglas, "A tupleware approach to domain decomposition methods," *Appl. Numer. Math.*, vol. 8, pp. 353–373, 1991.

[19] C. C. Douglas, "MGNet: a multigrid and domain decomposition network," *ACM SIGNUM Newsletter*, vol. 27, pp. 2–8, 1992.

[20] C. C. Douglas and J. Douglas, "A unified convergence theory for abstract multigrid or multilevel algorithms, serial and parallel," *SIAM J. Numer. Anal.*, vol. 30, pp. 136–158, 1993.

[21] P. W. Hemker, "On the order of prolongations and restrictions in multigrid procedures," *J. Comput. Appl. Math.*, vol. 32, pp. 423–429, 1990.

[22] C. C. Douglas, "Caching in with multigrid algorithms: problems in two dimensions," Tech. Rep. 20091, IBM Research Division, Yorktown Heights, New York, 1995.

Craig Douglas is a noted promoter of multigrid methods and a member of the scientific computing community. He is an associate editor of two journals and the managing editor of MGNet. He is an organizer of the Copper Mountain Conferences on Multigrid Methods, held on odd numbered years in early April.

He can be found most often in an airplane, but has been sighted at IBM's Research Division (T. J. Watson Research Center, Yorktown Heights, NY, USA) and Yale University's Department of Computer Science (New Haven, CT, USA). He has also been a frequent visitor at CERFACS (Toulouse, France) from late 1993 through 1995.

He can be reached conveniently by e–mail at *cdouglas@na–net.ornl.gov*, which will forward e–mail to wherever he is. A number of papers, codes, and special interests of Douglas can be found at the following URL:

http://www.cs.yale.edu/users/douglas-craig/

He regularly publishes papers on multigrid methods, domain decomposition methods, parallel algorithms, linear algebra, and sparse matrix methods. Some of his high performance computing examples include problems arising from numerically simulating flames (with and without detailed, finite rate chemistry), the oil industry, and ocean modeling.

He received an A.B. in mathematics from the University of Chicago and a M.S., M.Phil., and Ph.D. in computer science from Yale University. He has been a faculty member in computer science at Duke University in addition to his current posts.

Detour: What Are Smoothers and Roughers?

The term *smoother* [3] comes from the effect an iterative method has on its error components in Fourier space (see Figure 1). In this space, the components are waves, anywhere from short to long. For a problem like the Poisson equation on a square with a uniform grid, the error analysis can be done entirely with basis functions based on products of sine functions. A relaxation method like Gauss-Seidel will reduce the amplitude of every error component by some bounded factor. Hence, the components are all "smoothed" by this process.

The ultimate smoothers are direct solvers like (sparse) Gaussian elimination, the fast Fourier transform, or cyclic reduction. However, direct solvers only make sense in multigrid when used on the coarsest grid's problem.

Relaxation methods have been observed to damp out the high frequency error components quite quickly while taking a great deal longer for low frequency components. In Figure 1, the error component is represented as a sine curve. On grid 3, it is a low frequency wave in the sense that it covers 8 grid elements. On grid 1, it is a high frequency wave covering only 2 grid elements. On a grid more coarse than grid 1, the component would not be representable. A typical relaxation method would dampen this wave out much quicker on grid 1 than grids 2 or 3.

Multigrid methods were motivated by attempts to accelerate (or precondition) relaxation methods. Using the right scale for an error component means the difference between easily damping it out or having to work very hard.

The term *rougher* [20] refers to an iterative method which does not reduce every error component at every iteration. In fact, something like SSOR or conjugate gradients will increase the amplitude of some while reducing others. While the error or residual norm is reduced, some components may be severely "roughed" up.

Many roughers have the property that after a correction step in a multilevel algorithm, the solver amplifies some of the error components that have been essentially obliterated on coarse grids. Most relaxation methods and alternating direction implicit methods do not do this. Hence, roughers sometimes must re-correct error components on coarser grids that would not reappear if a smoother had been used.

Of course, the choice of whether to use a smoother or a rougher is not entirely trivial. For many problems, roughers provide a better convergence rate for the multigrid algorithm. Hence, it comes down to optimizing the convergence rate versus the cost of the iterative procedures.

Detour: Grid or Level Transfer Operators

Transferring information between problems on different levels is of the utmost importance in making multigrid methods work well.

For simple problems like Poisson's equation on a square, how information is transferred between grids is well known. When computing the right hand side for a correction problem, a weighted average of nearby residuals on the finer grid is common. Common weightings are the discrete $\mathcal{L}_2$ projection, represented by the stencil centered about the point in common between the two grids:

$$C \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad C \in \{1, \frac{1}{16}\}.$$

The choice of constant is determined by the discretization method (it is 1 if the mesh spacing is not factored into the right hand side).

Other common projection methods are transposes of discrete interpolation operators. Common interpolation operators are based linear or cubic interpolation.

An interesting problem in its own right is what is the correct choice for the minimum orders of the projection and interpolation operators for a partial differential equation which is p^{th} order. Theory [21] states that the orders may be almost anything as long as the sum of the two is at least $2p - 1$. While violating this sum of the orders rule guarantees disaster normally, this theory does not give a hint as to what the sum of orders does for improving the solution method. For many hard engineering problems, finding a good projection-interpolation pair is the limiting factor on whether or not using multigrid makes sense.

Consider the example of a typical high temperature flame. By the flame front, large changes occur in the temperature, velocities, and information about the chemical species. Even though the computed solutions are continuous, they behave like ones with jumps across the flame front. While adaptively chosen grids reduce the size of these jumps, there is still a noticeable jump on reasonable sized grids. In fact, there are very few computers in existence with enough memory to reduce the jumps to something adequately small. As a direct result of these jumps, the nonlinear multigrid methods used in this problem do not always compute a correction on a coarser grid which is acceptable. Finding better projection-interpolation pairs alleviates this problem.

Detour: Structured Spaghetti Code Can Win Big

Multigrid for PDE's consists of four major components:

1. Approximate solve (e.g., relaxation methods)
2. Residual computation
3. Interpolation
4. Projection (e.g., weighted average of residuals)

Each of these is usually coded separately, providing a well structured code in which the pieces can easily be replaced with a different algorithm.

A different approach is to write highly structured codes which implement a part of each of these components assuming that there is a well defined loop structure over the domain. Most computer languages that are used by scientists and engineers have a construct for including information from other files into a program. Using this technique, all four of the components can be combined into a much more complicated code. However, what has to be written is not really any harder to code than the usual programs.

By slight amount of tuning of one or two parameters that describe how many cache lines and the size of each, it is possible to re-use data in cache many times. In fact, if enough effort is made, data passes through cache once per pass through a level in the multigrid algorithms. If data would normally pass through cache m times, this can lead to a speed up of not quite $m-1$ times since moving data to and from main memory tends to determine the actual speed of most algorithms on RISC based machines.

A description of this technique for two dimensional problems can be found in [22], which is also available electronically on MGNet.

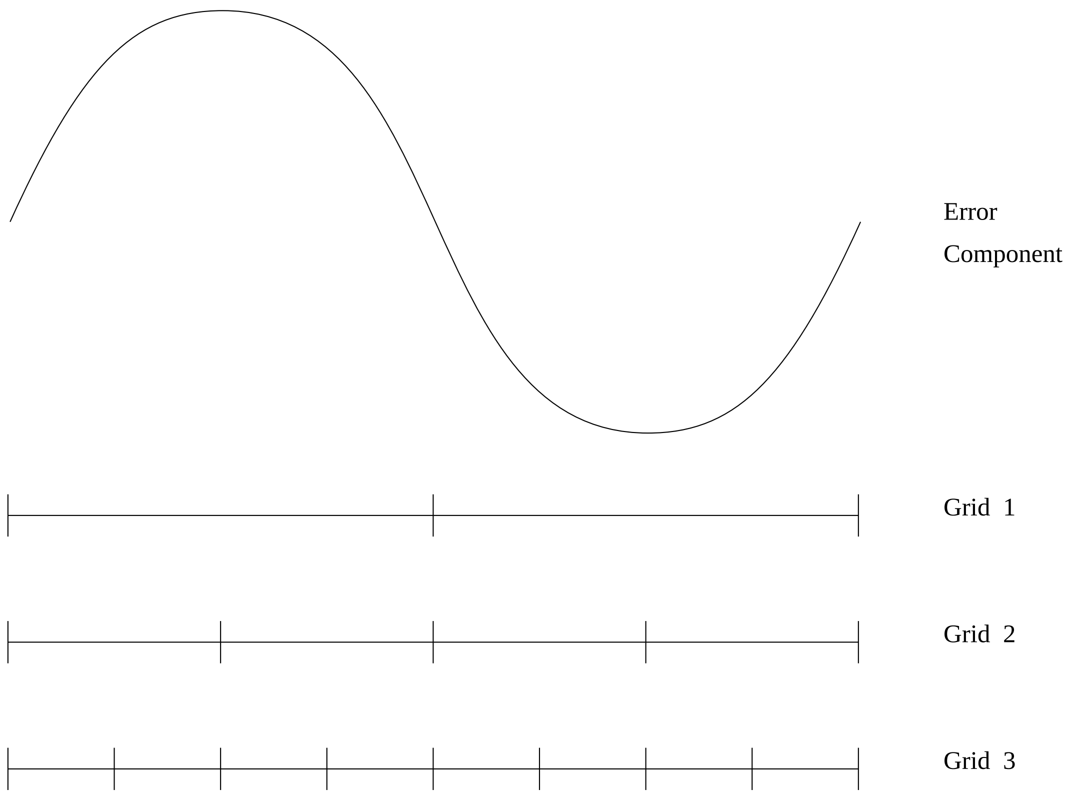

Figure 1: Wavelength scales on different grids

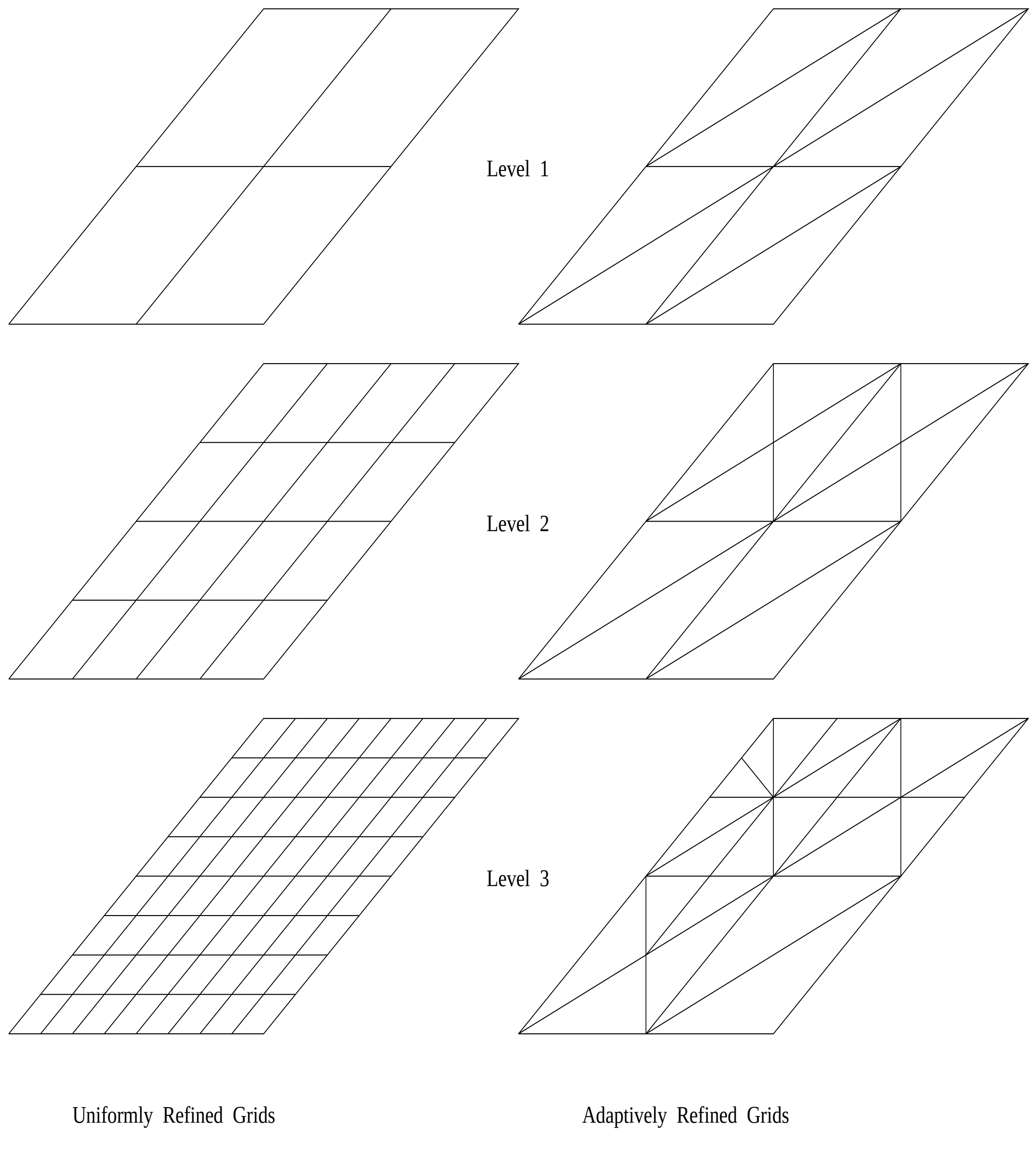

Figure 2: Three Level Example Grids

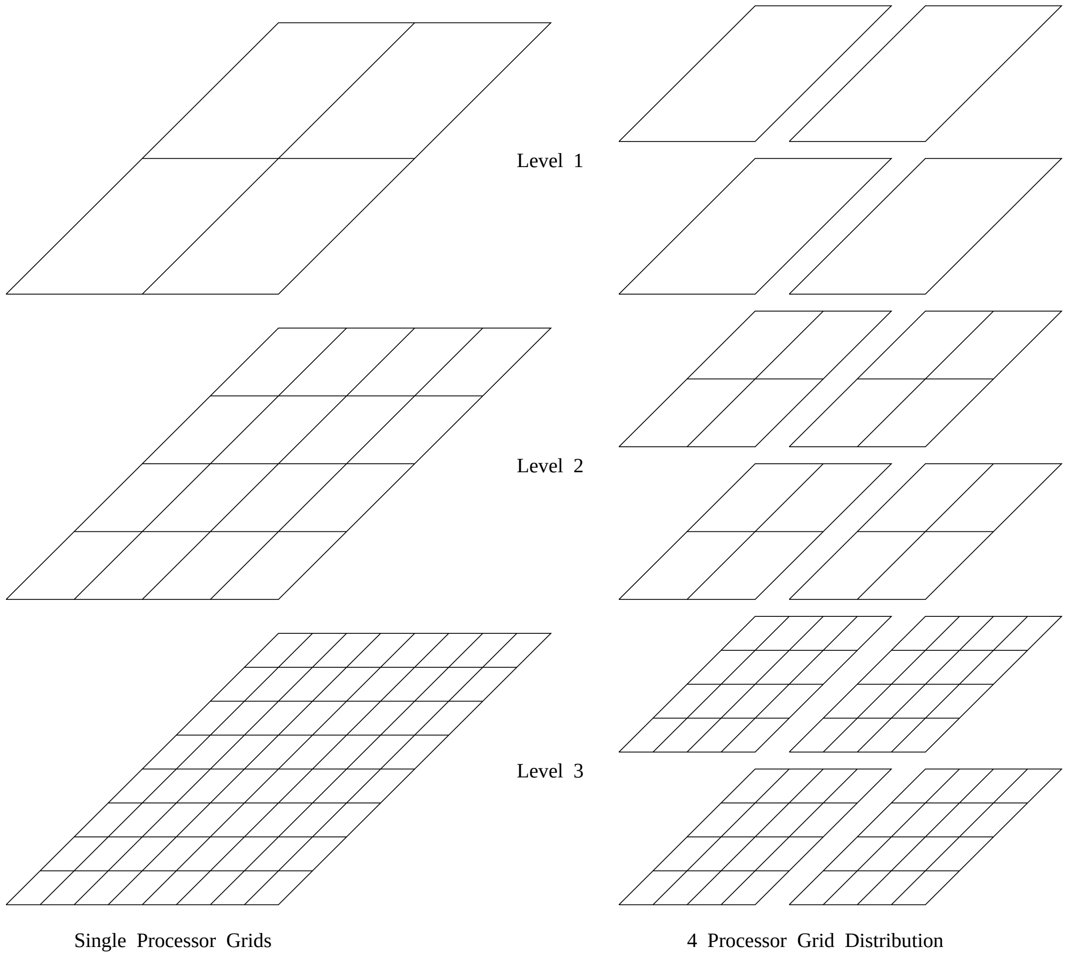

Figure 3: Multigrid with domain decomposition per level

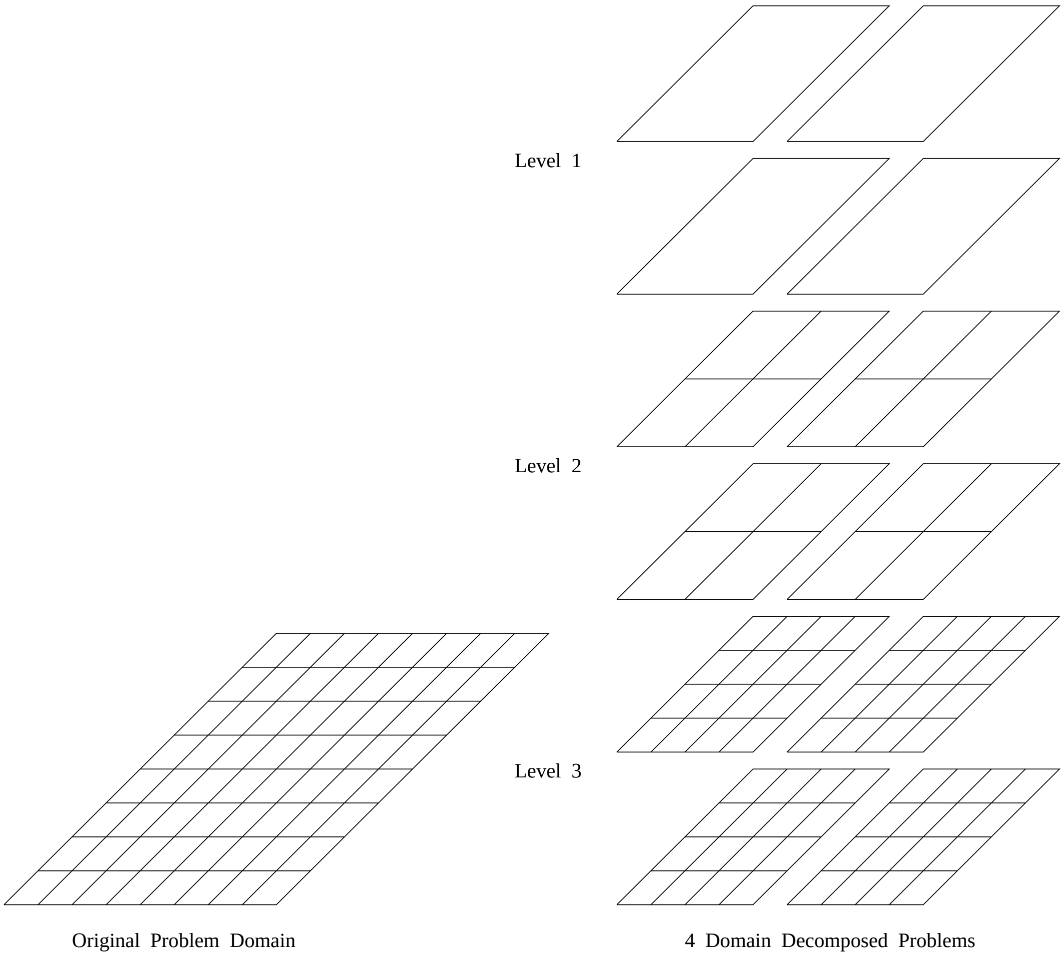

Figure 4: Domain decomposition with multigrid per subdomain problem

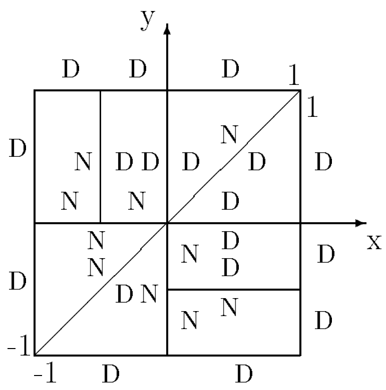

Figure 5: 8 way domain reduction on a square misrepresentation

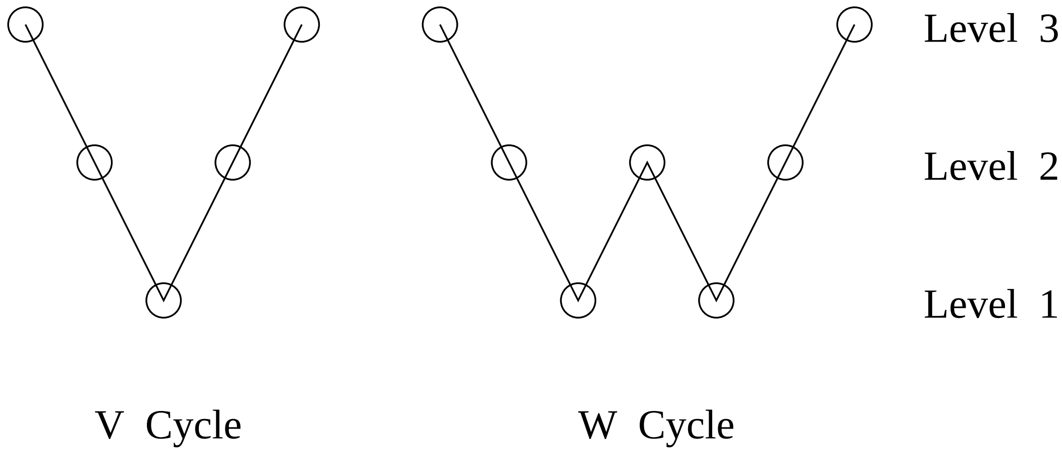

Figure 6: Multigrid algorithmic flow: correction methods

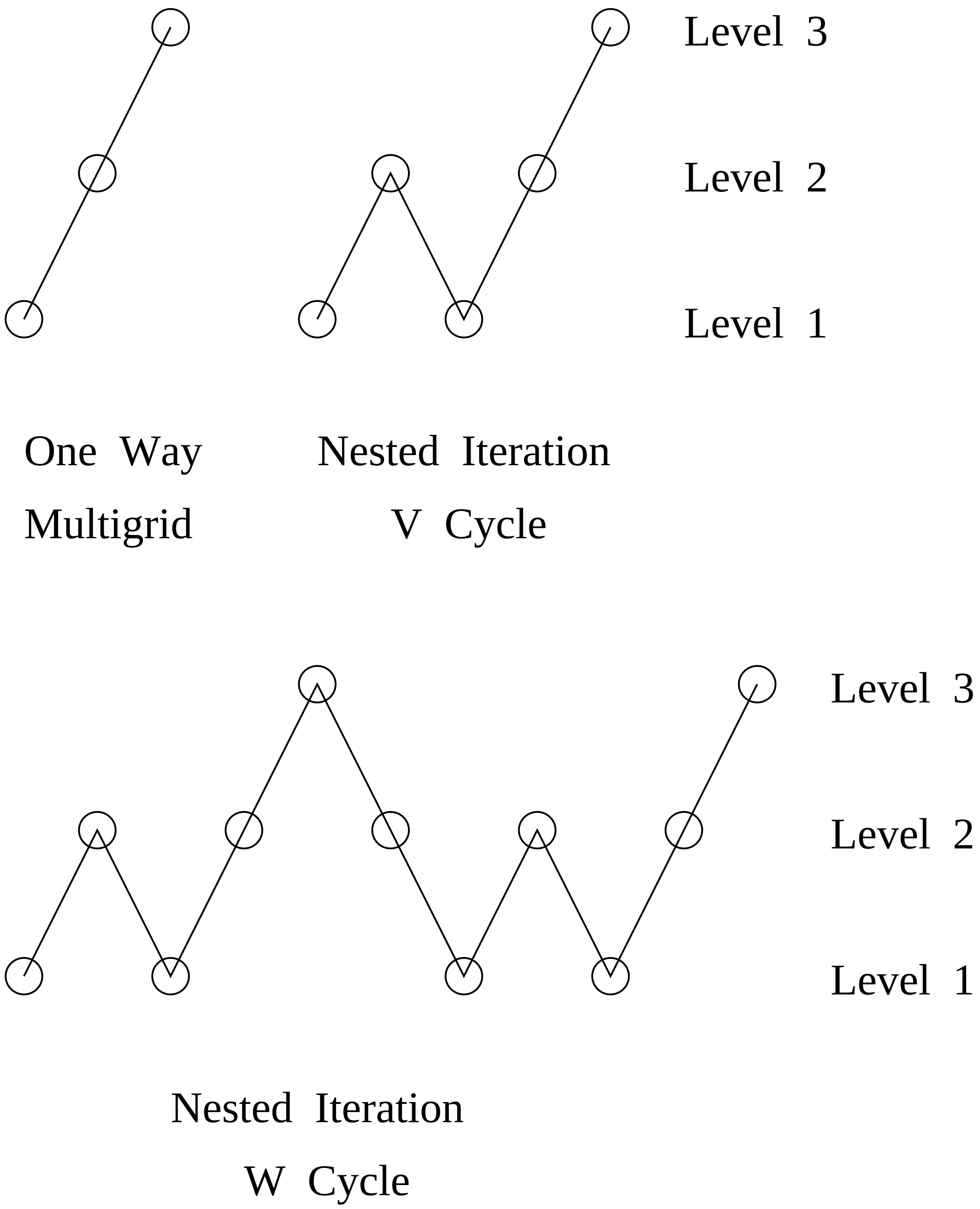

Figure 7: Multigrid algorithmic flow: nested iteration methods

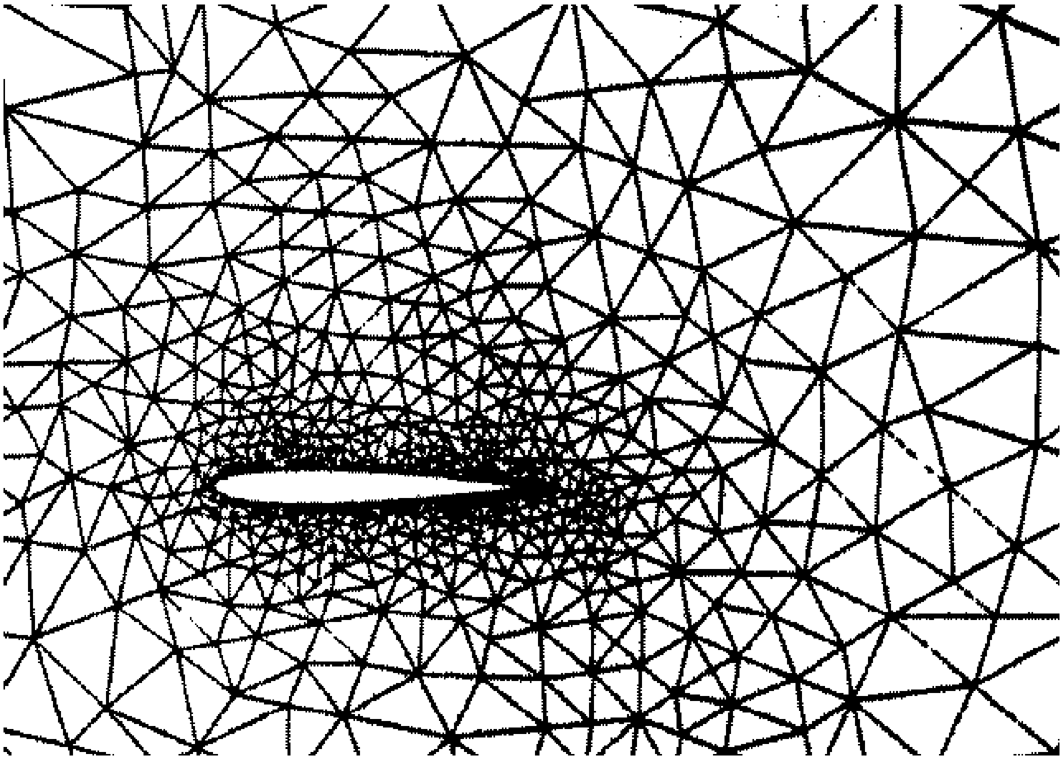

Figure 8: NACA–0012 example coarse grid

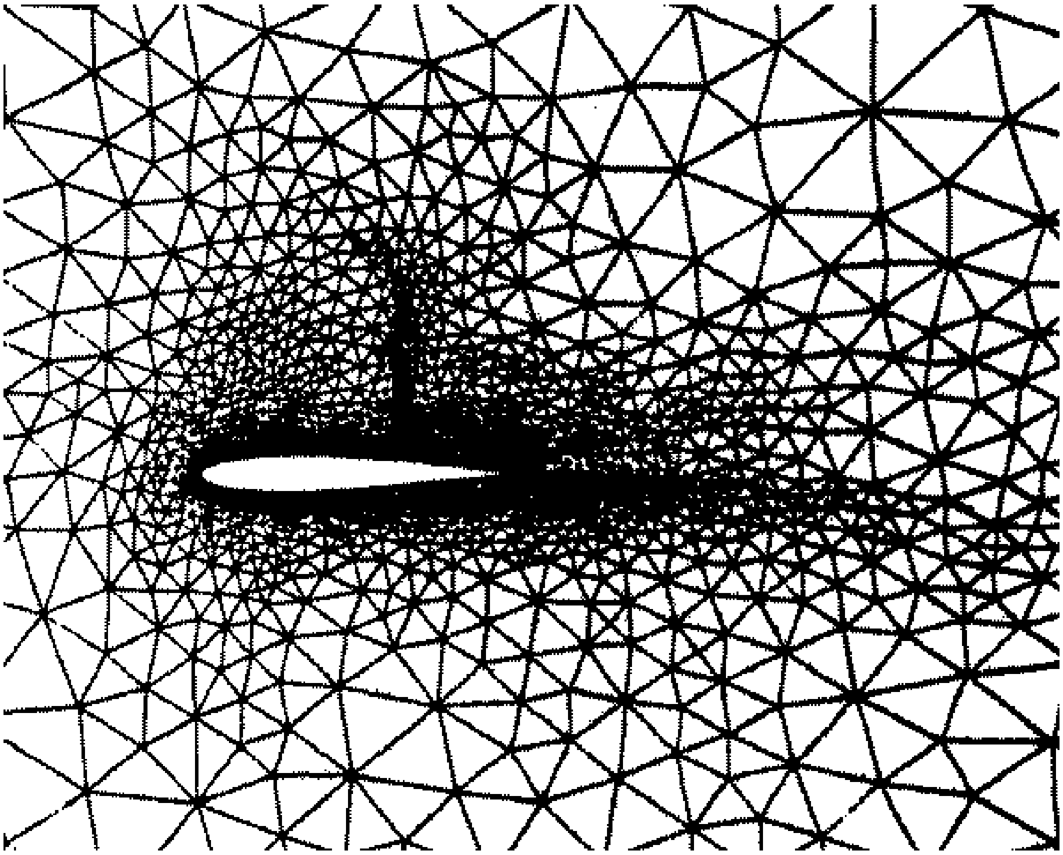

Figure 9: NACA–0012 example fine grid