NASA Contractor Report 195070

ICASE Report No. 95-27

ICASE

MULTIGRID TECHNIQUES FOR UNSTRUCTURED MESHES

D. J. Mavriplis

Lecture notes prepared for 26th Computational Fluid Dynamics Lecture Series Program of the von Karman Institute (VKI) for Fluid Dynamics, Rhode-Saint-Genese, Belgium, 13-17 March 1995.

Contract No. NAS1-19480
April 1995

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

Operated by Universities Space Research Association

MULTIGRID TECHNIQUES FOR UNSTRUCTURED MESHES

D. J. Mavriplis*

ICASE

Institute for Computer Applications in Science and Engineering

MS 132C, NASA Langley Research Center

Hampton, VA 23681-0001

United States

ABSTRACT

An overview of current multigrid techniques for unstructured meshes is given. The basic principles of the multigrid approach are first outlined. Application of these principles to unstructured mesh problems is then described, illustrating various different approaches, and giving examples of practical applications. Advanced multigrid topics, such as the use of algebraic multigrid methods, and the combination of multigrid techniques with adaptive meshing strategies are dealt with in subsequent sections. These represent current areas of research, and the unresolved issues are discussed. The presentation is organized in an educational manner, for readers familiar with computational fluid dynamics, wishing to learn more about current unstructured mesh techniques.

*This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

MULTIGRID TECHNIQUES FOR UNSTRUCTURED MESHES

D. J. Mavriplis

ICASE
Institute for Computer Applications in Science and Engineering
MS 132C, NASA Langley Research Center
Hampton, VA 23681-0001
United States

Contents

1 Introduction

This chapter is concerned with the use of multigrid methods for solving computational fluid dynamics problems on unstructured meshes. At issue is the efficient solution of the spatially discretized governing equations in time, in order to obtain the final steady-state solution, or the solution for the corresponding unsteady problem for a series of arbitrarily large time-steps. As such, the spatial discretization of the governing equations will not be considered in this chapter, only their temporal solution. The techniques in this chapter can in principle be applied to any spatial discretization, although the implementation details and their effectiveness may vary widely for higher-order discretizations. In the current discussion, we therefore assume that the equations are spatially discretized in a second-order accurate fashion, unless otherwise stated.

A standard numerical technique is to separate out the spatial and temporal discretization procedures. Thus, if the continuous set of governing partial differential equations is given by

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} + \frac{\partial h(u)}{\partial z} = 0 \tag{1}$$

then the spatially discretized equations can be written as:

$$\frac{du}{dt} + \mathbf{R}(\mathrm{u}) = 0 \tag{2}$$

This represents a large set of spatially coupled ordinary differential equations, where $\mathbf{R}(\mathrm{u})$ denotes the discretization of the spatial derivative terms in equation (1). In order to solve these equations, they must be integrated in time. For unsteady problems, the accuracy of the time discretization and integration procedures must be carefully considered. For the solution of steady-state problems, we are only concerned with the final state where all time derivatives vanish. The actual time integration procedure or transient path used to arrive at this state is thus of little consequence, and accuracy of the time integration procedure is not a concern. A particularly simple time discretization can be constructed as:

$$\frac{u^{n+1} - u^n}{\Delta t} + \mathbf{R}(\mathrm{u}^n) = 0 \tag{3}$$

thus enabling a simple procedure for advancing the solution in time:

$$u^{n+1} = u^n - \Delta t\, \mathbf{R}(\mathrm{u}^n) \tag{4}$$

This constitutes an explicit scheme, since the value of the solution at the new time-step $n+1$ can be obtained explicitly from the value at the previous time-step n. By their very construction, explicit schemes involve only local information (*i.e.*, based on the stencil of the residual $\mathbf{R}(\mathrm{u}^n)$). They are thus simple to implement, and vectorize and parallelize easily on present-day computer architectures. However, the efficiency of explicit schemes for hyperbolic equations is limited by the Courant-Freidrichs-Lewy condition, which states that the maximum permissible time-step for stability is proportional to the mesh spacing. Thus, as the mesh is refined, smaller time-steps are required to maintain stability. For unsteady problems, the allowable time-steps may be much smaller than those required by time-accuracy considerations alone, while for steady-state problems, this may result in an excessive number of time-steps to reach the steady-state converged solution. Since the number of variables which must be solved for also increases as the grid is refined, this results in an $O(N^2)$ algorithm, where N represents the total number of grid points. Another

viewpoint is that, since explicit schemes only make use of local information, as the mesh is refined, a larger number of iterations is required to transmit information across the entire domain. This type of behavior is obviously unacceptable for large problems, and more efficient solution techniques are required.

A scheme which is unconditionally stable for any size time-step can be constructed by replacing equation (3) with

$$\frac{u^{n+1} - u^n}{\Delta t} + \mathbf{R}(\mathrm{u}^{\mathrm{n+1}}) = 0 \tag{5}$$

i.e., by evaluating the discrete residual at the new time-step $n+1$ instead of at the old time-step n. Since the flow values are not explicitly known at $n+1$, the above equation is rewritten as

$$(\frac{I}{\Delta t} + \frac{\partial \mathbf{R}}{\partial u})(u^{n+1} - u^n) = -\Delta t\, \mathbf{R}(\mathrm{u}^{\mathrm{n}}) \tag{6}$$

which is obtained by linearizing the residual about the time-step n, thus introducing the Jacobian matrix $\frac{\partial \mathbf{R}}{\partial u}$. The term on the left-hand side of the equation represents a large sparse matrix which must be inverted at each time-step, in order to update the solution. In the limit of an infinitely large time-step, the first term on the left-hand side vanishes, and Newton's method is recovered. These types of implicit schemes have been employed with great success for unstructured grid problems, either as direct solvers, using a very large time-step and inverting the left-hand side matrix with a Gaussian elimination technique [1], or as iterative solvers, where finite size time-steps are taken, and the resulting linear system represented by the left-hand side matrix is solved approximately at each time-step using an iterative method [2, 3, 4, 5, 6]. In order to simplify the linear system which must be solved, it is common to use a simplified form of the Jacobian matrix, obtained by considering a first-order accurate discretization in its construction, while full second-order accuracy is maintained in the residual construction on the right-hand side of the equation, since this term determines the accuracy of the solution.

The main difficulty with implicit methods relates to the memory requirements of these techniques. Consider for example, a simple first-order accurate vertex-based discretization of the Euler equations in three dimensions. A simple discretization of this type can be shown to result in a nearest neighbor stencil, where all points involved in the stencil of a particular vertex are end-points of a mesh edge which touches the vertex. The number of non-zero entries of the Jacobian matrix is thus proportional to the number of edges in the mesh. Since the Euler equations represent a system of partial differential equations, the Jacobian matrix has a block structure, with each block consisting of a 5 by 5 sub-matrix. Thus the total number of non-zero entries in the Jacobian matrix is given by

$$\begin{aligned} & (5 \times 5) \times \mathit{Number\ of\ Edges} \times 2 \\ + \quad & (5 \times 5) \times \mathit{Number\ of\ Vertices} \end{aligned} \tag{7}$$

The factor of 2 arises due to the fact that the matrix is non symmetric, while the second term represents the diagonal contribution of the matrix. For a typical unstructured tetrahedral mesh, the number of edges is approximately six to seven times the number of vertices, thus the number of non-zero entries in the Jacobian matrix reduces to no less than 325N, where N is the total number of vertices. Since first- and second-order accurate discretizations for the Euler equations in three-dimensions which employ less than 100 storage locations per grid point are commonplace [7, 8, 9, 10], the use of an implicit method of this type entails a storage overhead at least three to four times larger than that required by a simple explicit scheme. In fact many implicit schemes incur

storage overheads equivalent to two to three times the Jacobian matrix, making them particularly undesirable for large three-dimensional problems.

On the other hand, matrix-free implicit methods are available which never explicitly require the formation of the Jacobian matrix. As an example, the application of GMRES techniques [11] to equation (6) may be achieved by only forming the required matrix vector products using finite difference techniques [12, 13] according to:

$$\frac{\partial \mathbf{R}}{\partial \mathrm{u}} \Delta u = \frac{\mathbf{R}(\mathrm{u}) - \mathbf{R}(\mathrm{u} + \epsilon \Delta \mathrm{u})}{\epsilon} \tag{8}$$

where ϵ represents the magnitude of a small perturbation to the solution vector in the direction Δu. However, to be effective, GMRES methods are usually employed in conjunction with a preconditioning technique, and the most effective preconditioning methods have so far been found to be those which rely on a Jacobian-type matrix [2], thus incurring similar storage overheads. The development of an effective matrix-free preconditioner could thus have a dramatic effect on the usability of strongly implicit methods for large problems.

Multigrid methods offer an alternative to implicit methods for efficiently solving large problems, while incurring low additional memory overheads. A notable property of a well formulated multigrid algorithm is that the number of multigrid cycles required to achieve a given level of convergence is independent of the resolution of the mesh. Thus, multigrid methods enable solutions to be obtained in O(N) operations, where N represents the number of grid points. Linear complexity of this type is considered to be optimal. Multigrid methods are also very powerful, in that they may be applied to linear as well as non-linear problems. While multigrid methods have traditionally been considered in the context of steady-state problems, they may also be utilized to solve unsteady problems. This is usually achieved by formulating the transient problem as a steady-state problem in pseudo-time at each time-step [14, 15, 16, 17]. Thus, the essential features of multigrid algorithms can be examined by restricting the discussion to steady-state problems, and the treatment of unsteady multigrid is deferred to the chapter on unsteady solution techniques.

The basic idea behind all multigrid strategies is to accelerate the solution of a set of fine grid equations by computing corrections on a coarser grid. The motivation for this approach comes from an examination of the error of the numerical solution in the frequency domain. High-frequency errors, which involve local variations in the solution, are well annihilated by simple explicit methods. Low-frequency or more global errors are much more insensitive to the application of explicit methods. This is natural, considering the local nature of the information employed in explicit schemes. In fact, the convergence rate of explicit schemes usually consists of a rather rapid initial residual reduction phase, which gradually develops into a much slower residual reduction phase, corresponding to a situation where all high-frequency errors have been eliminated and low-frequency errors dominate, as shown in Figure 1. Multigrid strategies capitalize on this rapid initial error reduction property of explicit schemes. Typically, a multigrid scheme begins by eliminating the high-frequency errors associated with an initial solution on the fine grid, using an explicit scheme. Once this has been achieved, further fine grid iterations would result in a convergence degradation. Therefore, the solution is transferred to a coarser grid. On this grid, the low-frequency errors of the fine grid manifest themselves as high-frequency errors, and are thus eliminated efficiently using the same explicit scheme. The coarse-grid corrections computed in this manner are interpolated back to the fine grid in order to update the solution. This procedure can be applied recursively on a sequence of coarser and coarser grids, where each grid-level is responsible for eliminating a particular frequency bandwidth of errors.

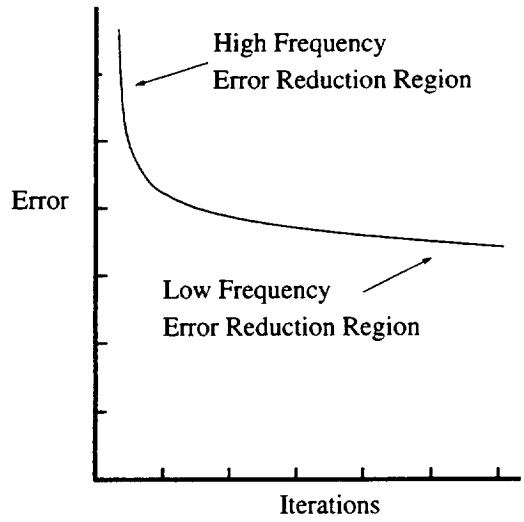

Figure 1: Typical convergence characteristics of explicit schemes.

This interpretation of multigrid is essentially an argument based on the principles of elliptic equations. While the expected performance of multigrid methods can be rigorously proved for simple elliptic problems [18], little is known for hyperbolic problems. Multigrid methods have nevertheless proven themselves for such problems. One interpretation of multigrid methods for hyperbolic problems is that the sequence of coarser grids enables the use of larger time-steps which expels disturbances out of the domain more rapidly.

Multigrid strategies are generally considered as convergence acceleration techniques, rather than solution methods themselves. In fact, they may be applied to any existing relaxation technique, explicit or implicit. The success of the overall solution strategy depends on the a close matching between the bandwidth of errors which can be efficiently smoothed on a given grid using the particular chosen relaxation strategy, with a careful construction of a sequence of coarse grids, in order to represent the entire error frequency range.

In the case of anisotropic problems, for example, there are two fundamental approaches to constructing efficient solution schemes. The difficulty with such problems is that simple explicit methods are incapable of efficiently eliminating the highest frequency errors in both the strong and weak coupling directions simultaneously (*i.e.,* for a stretched mesh in a boundary layer this corresponds to the normal and streamwise directions respectively). One approach, often employed in structured mesh cases, is to increase the error frequency bandwidth which may be handled efficiently by the base relaxation scheme, by resorting to an implicit relaxation method, such as approximate factorization (ADI) schemes [19]. The other approach is to recognize precisely which frequencies are not smoothed effectively by the simple explicit scheme (*i.e.,* in this case the high-frequency streamwise errors) and devise a sequence of coarse grids which ensures these errors are represented at some level as the dominant high-frequency contributions. For anisotropic problems, this may be achieved through semi-coarsening procedures, where coarse meshes are created by only coarsening in the direction of strong coupling [20, 21, 22]. In practice, both methods can be implemented effectively. However, the latter approach is more in the spirit of the multigrid purist, *i.e.,* to use additional coarse levels to relieve any stiffness associated with spatial effects.

Multigrid methods may also be employed to accelerate the solution of the full non-linear equation set as shown in equation (2), or they may be used to operate on the linear system which arises at each time-step in the implicit scheme of equations (6). While applying multigrid to the solution of the linear system in an implicit scheme affords certain advantages, and has been demonstrated successfully [23, 24], it forfeits one of the principle advantages of the multigrid method, which is the low memory overheads required.

One of the drawbacks of multigrid methods is that they require the construction of additional coarse-grid levels for the solution of the fine-grid equations. The generation of coarse levels reduces the automation of the solution process, and potentially decreases the robustness, as well as makes the procedure more problem dependent. This occurs as a result of the appearance of issues such as proper coarse-grid discretizations and boundary conditions. In fact, the coarse-grid levels are geometric constructions which in theory should not be required for the solution of a set of algebraic equations (as in the case of a direct solver). The development of algebraic multigrid methods [25] represents an attempt to abstract and formalize the ideas inherent in grid-based multigrid strategies to algebraic sets of equations. These techniques can be particularly attractive in the unstructured-grid context, since the construction of coarse-grid levels is not always evident, and the presence of complex geometries with widely varied boundary conditions can substantially complicate the implementation of grid-based strategies. However, the development of algebraic multigrid methods has substantially lagged that of geometric multigrid methods, a tribute to the fact that much useful information can often be acquired from the geometrical representation of the problem.

On the other hand, there is a reverse trend in the literature, known as the de-algebraization of multigrid [26]. This philosophy consists of viewing multigrid not simply as a convergence acceleration technique, but as a broad multilevel approach to simulating continuous partial differential equations, which may involve adaptive meshing, adaptive cycling strategies, and even multiple discretizations. A framework for the treatment of coupled problems such as fluid-structure interactions, and inverse design problems can be developed in this manner.

In this chapter, we are principally concerned with the use of multigrid methods to solve computational fluid dynamics problems on unstructured grids. We begin with a review of the basic multigrid principles. Next, the complexities encountered in applying these techniques to unstructured meshes are discussed. Most of the discussion centers around the construction of the coarse-grid levels. It is the conviction of the author that the most desirable multigrid methods be automated and stand-alone, *i.e.*, do not depend on a particular grid generation or solution strategy. The degree to which the various techniques fulfill these requirements is thus examined in their presentation. In a subsequent section, the similarities and discrepancies between the various methods previously described are addressed. The arguments presented are then utilized to justify the construction of a general multigrid agglomeration algorithm for the full Navier-Stokes equations, which has been employed successfully to solve large three-dimensional problems. The remainder of the chapter is devoted to the discussion of current research or speculation on techniques for improving the efficiency and robustness of multigrid methods. These include the construction of better inter-grid transfer operators and more accurate coarse-grid operators, as well as the development of more optimal coarsening strategies for coarse mesh generation for anisotropic problems, as well as for adaptive meshing problems.

2 Basic Multigrid Principles

The basic principles involved in the construction of a multigrid algorithm are discussed in this section. These principles are basic in that they do not depend on the particular set of equations being solved, the discretization and types of grids employed, or the dimensionality of the problem. These principles are then utilized to demonstrate the construction of a exemplary multigrid algorithm for solving the Euler or Navier-Stokes equations.

2.1 Multigrid Correction Scheme

Consider the solution of the discrete problem

$$L_h u_h = f_h \tag{9}$$

where the subscripts refer to the discretization of the continuous problem on a mesh of spacing h. The current estimate of the solution u_h is denoted as $\overline{u}_h$, which is obtained by approximately solving the above equation, using an iterative technique. Since $\overline{u}_h$ does not satisfy the above equation exactly, its substitution into equation (9) yields

$$L_h \overline{u}_h - f_h = r_h \tag{10}$$

where r_h is called the residual, and vanishes only when the exact solution to the discrete problem is attained. The object of the multigrid scheme is to compute a correction v_h such that the exact solution is given by:

$$u_h = \overline{u}_h + v_h \tag{11}$$

Taking the difference of equations (9) and (10), we obtain

$$L_h u_h - L_h \overline{u}_h = -r_h \tag{12}$$

If the operator L_h is linear, the above equation may be reduced to an equation for the correction v_h by substituting equation (11) into equation (12), which yields

$$L_h v_h = -r_h \tag{13}$$

Assuming that the high-frequency errors in the solution have been eliminated by sufficient fine grid smoothing cycles, the remaining correction v_h which we seek must be smooth, and can therefore be computed more efficiently on a coarser grid by solving the equation

$$L_H v_H = -I_h^H r_h \tag{14}$$

where the subscript H denotes a coarser grid, and I_h^H is an operator which interpolates residuals from the fine grid h to the coarse grid H. I_h^H is usually called the restriction operator. If the grid H is coarse enough, equation (14) may be solved exactly (either directly or iteratively). In the event this is not feasible, the present procedure may be performed recursively on coarser grids, thus yielding an approximate solution to the above equation. The exact or approximate solution of v_H must then be employed to correct the original fine grid solution. This is achieved as

$$\overline{u}_h^{new} = \overline{u}_h + I_H^h v_H \tag{15}$$

where I_H^h, which represents the interpolation of the coarse grid corrections v_H to the fine grid, is often called the prolongation operator. Once these fine grid values have been updated, they may be smoothed again by additional fine grid iterations, and the entire procedure, which constitutes a single multigrid cycle, may be repeated until overall convergence is attained. This particular multigrid strategy is called the Correction Scheme, since the coarse grid equations operate directly on the correction variables v_h. It is informative to note that, in the case where the fine grid solution has been attained, the fine grid residuals vanish, and the right-hand side of the coarse grid correction equation also vanishes. Equation (14) has a trivial solution in this case, which is $v_H = 0$, and no additional corrections are produced by the coarse grid, thus ensuring convergence of the entire multigrid procedure to the appropriate fine grid solution.

2.2 Full Approximation Storage Scheme

The multigrid correction scheme described above is only valid for the case where the operator L_h is linear. For non-linear operators, the difference $L_h u_h - L_h \overline{u}_h$ in equation (12) can no longer be replaced by $L_h v_h$, and thus the above scheme must be modified. This is achieved by introducing a new coarse grid variable $\overline{u}_H$ defined as

$$\overline{u}_H = \overline{I}_h^H \overline{u}_h + v_H \tag{16}$$

where $\overline{I}_h^H$ represents an operator which interpolates fine grid solution variables to the coarse grid. The coarse grid equation equivalent to equation (14) can now be written as

$$L_H \overline{u}_H - L_H \overline{I}_h^H \overline{u}_h = -I_h^H r_h \tag{17}$$

As previously, I_h^H represents the restriction operator which transfers residuals from fine to coarse grids. The operators $\overline{I}_h^H$ and I_h^H may in principle be different from one another.

It is useful to rewrite the above equation as

$$L_H \overline{u}_H = S_H \tag{18}$$

where

$$S_H = L_H \overline{I}_h^H \overline{u}_h - I_h^H r_h \tag{19}$$

In the above form, the coarse grid equation is seen to take on a similar structure to the original fine grid equation, with a modified source term. This enables the use of similar techniques for solving the coarse and fine grid problems. Once the coarse grid equations have been solved, either exactly or approximately, the fine grid variables are updated as

$$\overline{u}_h^{new} = \overline{u}_h^{old} + I_H^h(\overline{u}_H^{new} - \overline{I}_h^H \overline{u}_h^{old}) \tag{20}$$

which can also be written as

$$\overline{u}_h^{new} = \overline{u}_h^{old} + I_H^h v_H \tag{21}$$

Note that it is the difference between the initial and final coarse grid variables which is used, since this constitutes the definition of the correction, as per equation (16).

The source term S_H may also be rewritten as

$$S_H = f_H + \tau_H \tag{22}$$

where

$$f_H = I_h^H f_h \tag{23}$$

and

$$\tau_H = L_H \overline{I}_h^H \overline{u}_h - I_h^H (L_h u_h) \tag{24}$$

τ_H is sometimes called the defect correction [26, 18]. It may be loosely described as the difference between the coarse grid discretization and the interpolation of the fine grid discretization onto the coarse grid. The presence of the defect-correction term on the right-hand side of equation (18), ensures that the fine grid problem is represented by the coarse grid discretization, and that both coarse and fine grid equations converge to the same solution. This can be seen by considering the case where fine grid equations have been solved exactly. In this situation, the fine grid residuals all vanish, as does their interpolated result on the coarse grid. Since the right-hand side of equation (17) vanishes, the solution interpolated from the fine grid onto the coarse grid (*i.e.*, $u_H = \overline{I}_h^H u_h$) satisfies the coarse grid equation exactly, and therefore no further corrections are generated from the coarse grid.

The ability to directly handle non-linear problems is one of the great advantages of multigrid algorithms. This obviates the need to linearize the problem, with the possible complications and overheads which such operations often entail.

2.3 Coarse Grid Operators

The CS (correction scheme) and FAS (full approximation storage) multigrid schemes both result in the formulation of coarse grid equations which are similar in form to the originating fine grid equations. While the formulation of these equations has been discussed, the precise manner in which these equations are discretized on each grid level has not been addressed. Since the coarse and fine grid equation constructions are similar, the most obvious approach is to discretize them in the same manner. Thus, for example, the operator L_H can be constructed using the same discretization procedure as the operator L_h, applied to the coarser grid. This technique is called rediscretization. It is the most prevalent technique for constructing the discrete coarse grid equations in computational fluid dynamics problems. Rediscretization techniques generally ensure a consistent representation of the fine grid problem on all grid levels, and enable the use of identical solution strategies for coarse and fine grids. This greatly simplifies implementation by permitting the reuse of many subroutines for the various grid levels.

There are situations where rediscretization techniques are not practical or feasible, however, such as in the case of algebraic multigrid, where the coarse-grid levels are never explicitly formed. Even for geometric multigrid methods (as opposed to algebraic methods), rediscretization techniques rely on the assumed property that the coarse mesh discretization is valid, stable, and approximates the fine grid problem to a certain degree. These assumptions are not always guaranteed, and may break down particularly when dealing with complex geometries. An alternative procedure for discretizing the coarse grid equations is to construct the coarse grid operator as the sequence of operators

$$L_H = I_h^H L_h I_H^h \tag{25}$$

where the first term on the right-hand side represents the restriction operator, and the last term the prolongation operator. L_h refers to the original fine grid discrete operator. This construction is called Galerkin coarse grid operator, since it is a simple matter to show that, if the solution of $L_h u = f_h$ minimizes a functional over all functions spanned by the set of fine grid functions u_h, then the solution of $I_h^H L_h I_H^h u = I_h^H f_h$ minimizes the same functional over all functions spanned by the smaller set of coarse grid functions u_H [18]. The advantage of this construction is that it is entirely algebraic, *i.e.*, once the fine grid discretization and inter-grid transfer operators are known, the coarse grid operator is implicitly defined, and the geometry of the coarse grid problems does not enter explicitly into the construction of the solution technique. On the other hand, depending

on the precise forms of the fine grid and inter-grid operators, the resulting coarse grid operator can be considerably more complex to construct and evaluate than the original fine grid operator, and than the equivalent coarse grid operator obtained through rediscretization. This, in particular, may severely limit the possibility of utilizing this approach recursively in applications involving a large number of grid levels. Both techniques have been employed for unstructured mesh problems and will be demonstrated subsequently in this chapter.

2.4 Intergrid Transfer Operators

Particular constructions for the restriction and prolongation operators, as well as for the interpolation of the solution variables in the FAS scheme, remain to be defined. For computational fluid dynamics problems, the most common choices are either injection or some variant of linear interpolation. Injection corresponds to the interpolation operator which preserves a constant function exactly. As an example, the value of a coarse grid cell would be assigned to all constituent fine grid cells which are contained inside the coarse grid cell by the injection operator. Structured grid multigrid methods often employ bilinear (in two dimensions) and trilinear (in three dimensions) inter-grid transfer operators. For unstructured mesh multigrid methods based on triangular elements in two dimensions, and tetrahedral elements in three dimensions, simple piecewise linear interpolation is easily implemented, using the linear finite-element shape functions associated with these elements. Piecewise linear interpolation operators preserve linear functions exactly.

The accuracy of the restriction and prolongation operators must be sufficient to avoid introducing excessive errors into the solutions, which can in turn have a detrimental effect on convergence efficiency. A fundamental rule for the accuracy of the inter-grid transfer operators is given by [18, 27]:

$$m_r + m_p > m \tag{26}$$

where m_r and m_p represent the highest degree polynomial plus 1, which the restriction and prolongation operators interpolate exactly, respectively. Thus, m_r and m_p are equal to 1 for injection operators, while they would be equal to 2 for piecewise linear interpolation operators. m represents the order of the partial differential equation being solved. For example, for an advection equation, $m = 1$, whereas for a Poisson equation, $m = 2$. In practice, equation (26) is seldom violated since the use of linear interpolation in at least one of the inter-grid transfer operators is enough to satisfy the strict inequality for convection-diffusion equations. However, there are schemes which are in violation of this rule, as will be shown in this chapter.

2.5 Cycling Strategies

Cycling strategies refer to techniques employed to determine when to switch from one grid to the next, rather than to how to win a race on two wheels. These can be divided into two basic approaches: adaptive and fixed cycling strategies. Adaptive cycling methods involve the monitoring of the numerical convergence process. When it is determined that the high-frequency errors on the current grid have been effectively eliminated, usually by observing a sharp slowdown in the convergence rate, the jump to a coarser grid is triggered. Although adaptive cycling strategies may appear more desirable, practical considerations such as simplicity and robustness usually result in the use of fixed cycling strategies, where a fixed pattern of coarse and fine grid iterations is prescribed. The two most common cycling patterns are the V-cycle, and the W-cycle, which are depicted in Figures 2 and 3. The multigrid V-cycle begins on the finest grid of the sequence (*i.e.*, grid level 4 in Figure 2), where one relaxation or time-step is performed. The solution and residuals are then interpolated to the next coarser grid, where another time-step is performed. This

procedure is repeated on each coarser grid until the coarsest grid of the sequence is reached. At this point, the coarse grid corrections are prolongated back to each successively finer grid until the finest grid is reached.

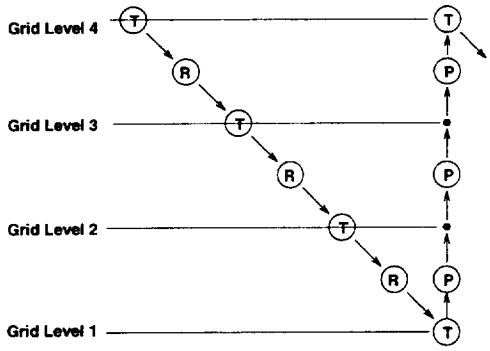

Figure 2: Multigrid V-Cycle.
T = Time-Step, **R** = Restriction,
P = Prolongation

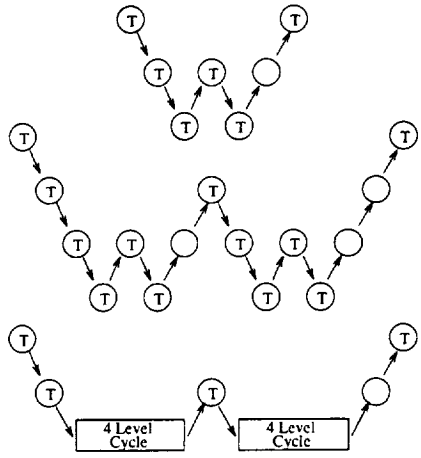

Figure 3: Multigrid W-Cycle.

This particular variant of the V-cycle, which has been employed extensively for computational fluid dynamics problems, is sometimes known as a saw-tooth cycle, since no time-stepping is performed on the coarse-to-fine phase of the cycle. In practice, it is possible to perform single or multiple time-steps on each grid level in the coarsening phase of the cycle, as well as in the refinement phase. The approximate complexity of a saw-tooth V-cycle is given by the sum of the complexities of the various grid levels. For a two-dimensional problem, where the coarsening process involves the reduction of grid complexity by a factor of 4 (as is typically the case for structured meshes), the V-cycle complexity is bounded by

$$1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{32} + ... < \frac{4}{3} \tag{27}$$

or 4/3 times the complexity of a single fine grid time-step, while in three dimensions this figure becomes 8/7 (assuming a coarsening ratio of 8:1). The W-cycle is a recursive strategy which weights coarse grids more heavily, as shown in Figure 3. Use of the W-cycle is often found to be more efficient overall, and more robust than V-cycles. The complexity of a saw tooth W-cycle in two dimensions (assuming a coarsening ratio of 4:1) is given by

$$1 + 2 \times \frac{1}{4} + 4 \times \frac{1}{16} + 8 \times \frac{1}{32} + ... < 2 \tag{28}$$

Coarsening ratios may vary widely for unstructured mesh multigrid methods, particularly when used in conjunction with optimal coarsening strategies, or adaptive meshing techniques. For coarsening ratios smaller than 4:1, the (asymptotic) W-cycle complexity may become unbounded. Thus, the choice of a particular cycling strategy must necessarily consider the complexity of the various grid levels.

The combination of mesh sequencing with a multigrid method (where the solution on the current grid is initiated from a previously computed solution on a coarser grid) results in a strategy known as the full multigrid procedure. The basic cycling strategy is depicted in Figure 4, using a saw-tooth

multigrid V-cycle. Beginning with an initial sequence of grids, the solution on the finest grid of the sequence is obtained by a multigrid procedure operating on this sequence. A new finer grid is then added to the sequence, the solution is interpolated onto this new mesh, and multigrid cycling is resumed on this new larger sequence of meshes. The procedure can be repeated, each time adding a new finer grid to the sequence, until the desired solution accuracy has been achieved, or the finest available mesh has been reached. This strategy may be employed with a fixed set of meshes, but is especially attractive for adaptive meshing problems, where new meshes are generated on the fly, so to speak.

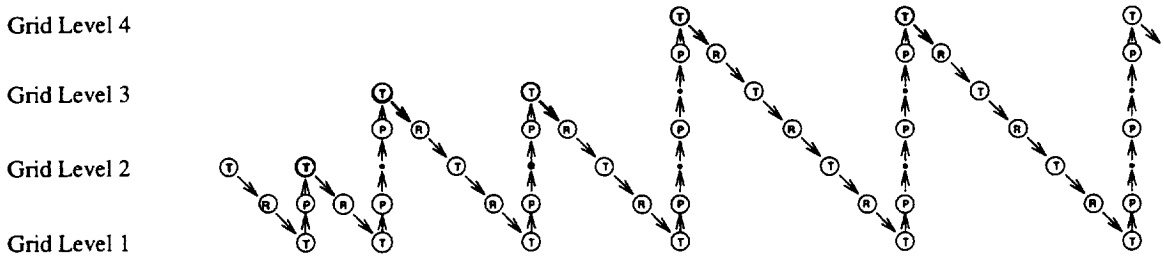

Figure 4: Full Multigrid (FMG) Strategy.
T = Time-Step, **R** = Restriction, **P** = Prolongation

2.6 Multigrid Algorithm for the Navier-Stokes Equations

The remaining procedure which must be specified in the construction of a multigrid algorithm is the single grid relaxation or time-stepping procedure. The main requirements of this procedure are that it be capable of rapidly damping out high-frequency errors. While the convergence of almost any relaxation scheme may be accelerated through a multigrid procedure, the most effective multigrid strategy is to rely on a simple and inexpensive relaxation scheme, which is specifically tailored for annihilating high-frequency errors. All lower frequency errors can then be handled effectively by the appropriate coarse-grid level. The use of multi-stage time-stepping schemes as multigrid drivers is one of the most common approaches for both structured and unstructured mesh problems. In addition to being simple and inexpensive, multi-stage time-stepping schemes can be optimized to damp high-frequency errors [28, 29, 30]. A typical multi-stage time-stepping scheme can be written as:

$$\begin{aligned} u^{(0)} &= u^{(n)} \\ u^{(1)} &= u^{(0)} - \alpha_1 \Delta t \mathbf{R}(\mathrm{u}^{(0)}) \\ u^{(2)} &= u^{(0)} - \alpha_2 \Delta t \mathbf{R}(\mathrm{u}^{(1)}) \\ &\cdot \\ &\cdot \\ &\cdot \\ u^{(q)} &= u^{(0)} - \alpha_q \Delta t \mathbf{R}(\mathrm{u}^{(q-1)}) \\ u^{(n+1)} &= u^{(q)} \end{aligned} \tag{29}$$

where q represents the number of stages, $u^{(n)}$ the initial flow values, and $u^{(n+1)}$ the updated flow values at the end of the new time-step. α_q represents the multi-stage coefficients, which may be chosen to optimize the high-frequency damping properties of the scheme. For the fine mesh, $\mathbf{R}(\mathrm{u})$ represents the discrete residual, as given in equation (2), which can also be represented as $L_h u_h - f_h$,

using the notation of equation (10). On the coarse grids, the forcing function S_H of equation (18) must be included in the $\mathbf{R}(u)$ term above.

Having described all essential elements of the multigrid strategy, a particular multigrid algorithm for the Euler and Navier-Stokes equations can now be described. The algorithmic steps are as follows:

Step 1: Identify the first grid as the finest grid level and initialize the solution on this grid.

Coarsening Phase:
Step 2: Perform a single time-step on the current grid using the scheme of equations (29).
Step 3: If the current grid is the coarsest grid go to step 7.
Step 4: Compute the residuals on the fine grid using the latest available solution values.
Step 5: Restrict these residuals to the next coarser grid and interpolate the solution variables onto the next coarser grid.
Step 6: Identify this next coarser grid as the current grid. Initialize the solution on this grid with the variables interpolated from the previous fine grid, and construct the defect correction which constitutes the right-hand side of the discrete coarse grid equations (*i.e.*, equation (18)) using these values and the restricted residuals. Go to step 2.

Refinement Phase:
Step 7: If the current grid is the finest grid, go to Step 2, otherwise identify the next finer grid as the current grid.
Step 8: Compute the corrections on the previous coarse grid and prolongate them to the current fine grid.
Step 9: Go to Step 7.

The above algorithm describes a typical implementation of an FAS multigrid scheme applied to fluid flow problems. This description constitutes a single multigrid cycle, and the entire process is applied repetitively until convergence of the *fine grid problem* is attained, which is determined by monitoring the magnitude of the *fine grid* residuals. This particular scheme corresponds to the use of a fixed saw-tooth V-cycle, which is the simplest to describe algorithmically. Other cycle types such as W-cycles and cycles with time-stepping in the refinement stages may also be employed. Note that in the case where only a single grid is specified, the coarsest and finest grids are identical, and the algorithm bounces back and forth between steps 2,3 and 7, thus reproducing the single grid algorithm. As mentioned previously, the restriction and prolongation operators are usually implemented as piecewise linear interpolation operators or injection operators for computational fluid dynamics multigrid methods.

3 Application to Unstructured Grids

The previous section described the construction of generic multigrid methods without regards to the types of grids on which these methods are to be applied. This section deals with the specifics of implementing such methods on unstructured meshes. The main difficulty with unstructured multigrid methods concerns the construction of the coarse mesh levels. For structured mesh multigrid methods, a coarse mesh can be derived from a given fine mesh by omitting every 2nd point in each coordinate direction. Recursive application of this procedure results in a sequence of coarse meshes where the complexity of the meshes decreases by a factor of 4:1 in two dimensions and 8:1 in three dimensions, for each successively coarser level.

For unstructured meshes, such techniques are no longer feasible. Due to the lack of mesh structure, simple coarsening strategies do not result in consistent coarse grid triangulations. A

variety of techniques have been proposed for unstructured multigrid coarse mesh constructions. These vary from methods which attempt to reproduce the nested property of structured mesh multigrid methods [31, 32, 33], to techniques which permit the use of arbitrary (triangular or non-triangular) coarse meshes [8, 9, 24, 34, 35, 36, 37, 38, 39, 40, 41], to algebraic methods which never consider the construction of coarse meshes altogether [25]. In general, all methods are capable of delivering similar efficiencies, and the issues involved in choosing a particular method include ease of implementation, degree of automation, and robustness for highly complex geometries. These issues will be addressed throughout the presentation of the various methods.

3.1 Nested-Mesh Subdivision

One of the simplest unstructured mesh multigrid strategies is to generate a sequence of finer meshes from an initial coarse mesh by recursively subdividing the cells of the mesh [31, 32, 33], either globally, or adaptively, using the subdivision rules described in the chapter on grid generation. This results in a fully nested sequence of grids, as shown in Figure 5, and enables a particularly simple construction of the inter-grid transfer operators.

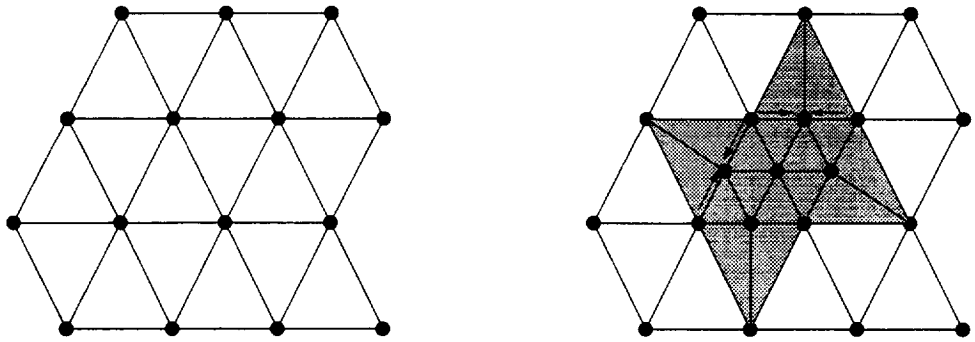

Figure 5: Original coarse mesh and adaptively subdivided fine mesh illustrating the nested construction of the fine and coarse levels, and the simple interpolation strategy for newly added vertices.

For example, in the context of a vertex scheme, the values at the vertices which are common to coarse and fine grids are simply transferred by injection. Similarly, the newly introduced fine grid points always lie midway along a coarse grid edge, and thus the values at these points may be transferred by averaging the two values at the end points of the coarse grid containing edge, which corresponds to linear interpolation. For a cell-centered scheme, volume weighted restriction is easily achieved by identifying the fine grid constituent cells of each coarse grid cell, and summing their weighted values. Another advantage of this approach is that it is easily automated.

This method has a somewhat inverted nature, *i.e.*, it begins with a coarse mesh and subsequently generates finer meshes, whereas most multigrid methods begin with the finest mesh and construct coarser levels. There are several disadvantages associated with such a strategy. The most obvious is the lack of flexibility in handling problems on a specified fine grid of unknown origin. In fact, this approach requires a tight coupling between the grid generation and the multigrid solution strategies, and has thus often been implemented in the context of adaptive meshing problems. The other difficulties are somewhat more subtle, but are interrelated. They concern the ability of the coarsest initial grid to provide efficient convergence properties for the multigrid algorithm, and the quality of the resulting fine grid. In a multigrid process, the coarsest grid of the sequence determines the convergence rate of the algorithm, while the finest grid determines the accuracy of the solution. The present multigrid strategy places conflicting demands on the coarse mesh construction. On the one hand, a very coarse mesh is desired, since this enables a rapid multigrid

convergence. However, the use of very coarse initial meshes may result in poor quality fine meshes, particularly when using simple subdivision refinement techniques. This, in turn, has a detrimental effect on solution accuracy.

One technique to alleviate these problems is to employ a relatively fine initial mesh, and make use of a direct or implicit solution procedure on this coarsest mesh alone, in order to maintain favorable convergence rates, with minimal storage overheads.

3.2 Overset Meshes

An alternate approach to unstructured multigrid methods is to generate a sequence of completely independent coarse and fine meshes, and use linear interpolation to transfer variables back and forth between the various meshes of the sequence, within a multigrid cycle [8, 9, 35, 36, 37, 38, 39]. The meshes may be generated using any grid generation technique, and will generally be non-nested, and may even not contain any common points. The only requirement is that they conform to the same domain boundaries. This technique is more flexible than the nested subdivision approach, since the fine and coarse meshes are not constrained, and may be optimized independently for accuracy and speed of convergence respectively. Furthermore, this approach can be applied to a problem with a prespecified fine mesh.

On the other hand, the construction of the inter-grid transfer operators becomes more involved. For static grid problems (*i.e.*, usually the case for steady-state problems), these operators may be precomputed and stored for use in the multigrid cycle. In the context of vertex-based schemes, it is most natural to use piecewise linear interpolation for the prolongation operator. For a given fine grid vertex to which we seek the interpolated value, the enclosing coarse grid triangle (or tetrahedron in three dimensions) must first be determined. (An efficient algorithm for achieving this will be given shortly.) Once this is known, the linearly interpolated value at the fine grid vertex can be obtained as a weighted average of the values of the three forming vertices of the coarse enclosing triangle, as shown in Figure 6. The weights may be determined using the linear finite-element shape functions associated with triangular elements. The final expression may be written as:

$$w_p = \frac{a_1}{A_C} W_1 + \frac{a_2}{A_C} W_2 + \frac{a_3}{A_C} W_3 \tag{30}$$

where w_p represents the fine grid interpolated value at point p, and W_1, W_2, W_3 represent the coarse grid variables at the three vertices of the enclosing triangle. The coefficients a_1, a_2, a_3, represent the triangle areas as depicted in Figure 6, and A_C denotes the area of the coarse grid enclosing triangle. Thus, identification of the enclosing triangle determines the addresses of the interpolation stencil, while the above formula determines the weights of the stencil. Note that in the above formula, when the fine grid vertex p coincides with a coarse grid vertex, the coefficient associated with the coinciding vertex becomes unity, the other coefficients vanish, and the formula reduces to injection. The entire prolongation operator can be stored as a set of three addresses and three weights for each fine grid vertex (four quantities in three dimensions), which are used to compute the interpolated values at each multigrid cycle as per equation (30).

A common form for the restriction operator is to construct it as the transpose of the prolongation operator [18, 25]:

$$I_h^H = \left(I_H^h\right)^T \tag{31}$$

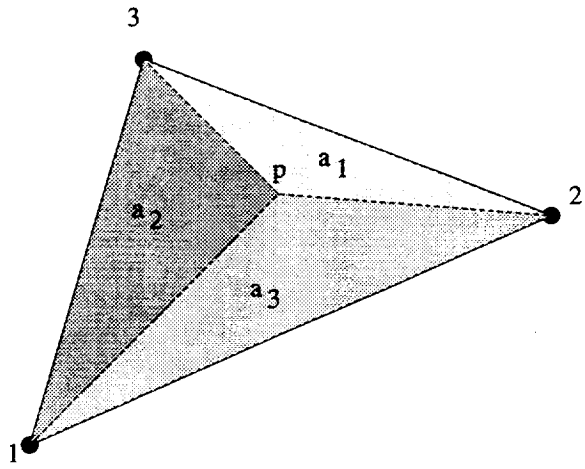

Figure 6: Definition of linear interpolation coefficients for fine grid vertex p contained in coarse grid triangle 123.

Geometrically, this corresponds to distributing the residual of a fine grid vertex p to the three vertices of the enclosing coarse grid triangle, with the same weights given by the linear prolongation operator defined in equation (30). This is an accumulation process, and each coarse grid vertex P receives residual contributions from all fine grid points which fall in any coarse grid triangles which contain P, as shown in Figure 7.

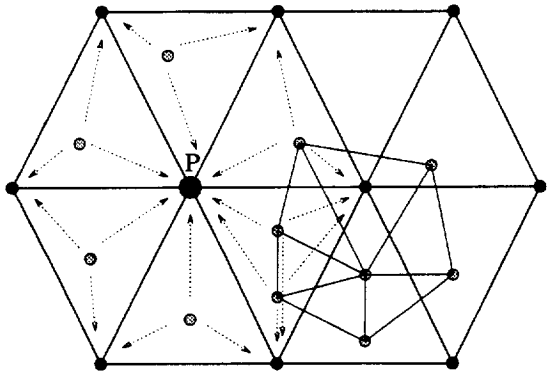

Figure 7: Illustration of conservative residual restriction for overset-mesh vertex scheme. Each coarse grid vertex receives contributions from multiple fine grid vertices.

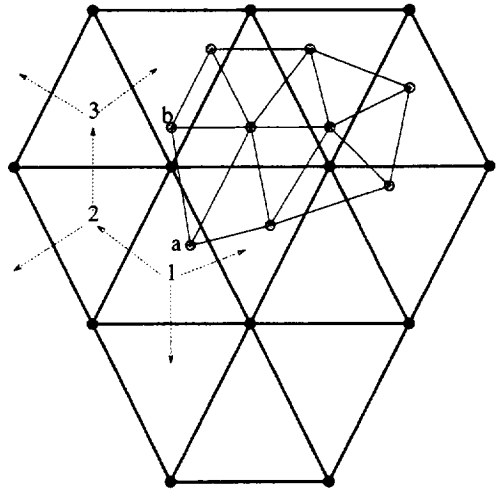

Figure 8: Illustration of a graph-traversal search algorithm for locating the enclosing triangle of vertex **b** knowing the enclosing triangle for vertex **a** .

Since the restriction and prolongation operators are transposes of one another, a single set of addresses and coefficients are required to define both operators. This particular form of the restriction operator results in a conservative transfer of residuals from fine to coarse grids, *i.e.*, the sum of the fine grid residuals is equal to the sum of the transferred residuals on the coarse grid. Thus, non-zero fine grid residuals will result in non-zero restricted residual sets, in the absence of local cancellations. However, such local cancellations correspond to high-frequency residual variations, which should be easily handled by the fine grid time-stepping procedure.

The third transfer operator which must be defined for the FAS scheme, $\overline{I}_h^H$, represents the transfer of fine-grid solution variables to coarser grids. In this case, a conservative transfer of variables is no longer of concern, since the coarse grid equations are only used to compute corrections. Accuracy of the transfer operator is of more importance. Thus, piecewise linear interpolation is employed. While the construction of the operator is similar to that described previously for the prolongation operator, the roles of the coarse and fine grids are reversed, *i.e.*, we wish to determine the value at a coarse grid point from fine grid values. This requires the determination of the fine grid triangle which encloses each coarse grid vertex, and the computation of the interpolation weights in an analogous manner. Thus, an extra set of weights and addresses must be computed and stored for this operator.

The inter-grid transfer operators are constructed in a pre-processing operation, prior to initiation of the multigrid solution procedure. This must be performed for each successive pair of grids in the multigrid sequence. If a subroutine is constructed which takes as input two arbitrary meshes, and outputs the weights and coefficients of the prolongation/restriction operator, then a simple technique for obtaining the weights and coefficients of the $\overline{I}_h^H$ operator is to switch the input order of the two meshes.

An essential step in the construction of the inter-grid transfer operators is the determination of the enclosing triangle on one grid for each vertex of the other grid. A naive implementation of this operation consists of checking every triangle on the first grid for each vertex of the second grid. This results in an $O(N^2)$ algorithm, where N represents the number of grid points of either one of the grids involved, and would be more expensive than the flow solution procedure itself. The complexity can be reduced to O(N) by making use of graph-traversal type algorithms, as illustrated in Figure 8. Assuming the enclosing triangle for a given vertex **a** on Grid 1 has been determined, we seek the enclosing triangle for a neighboring Grid 1 vertex **b**. Since this new vertex is in the neighborhood of the previous vertex, a good starting guess for the enclosing triangle would be the triangle which was found to enclose the previous vertex. This triangle is first tested, and if the test fails, we search all three neighbors of this triangle. If these tests fail, we search the neighbors of these neighbors, and continue in this fashion until the enclosing triangle is located. In order to implement this type of algorithm, it is useful to reorder the vertices of Grid 1 using in a wavefront-type pattern which guarantees that, for each successive vertex in the list, there exists a neighbor of this vertex which precedes this vertex in the list. Once this reordering has been executed, the neighboring vertex which precedes each current vertex in the ordered list is stored. A neighbor list for the triangles of Grid 2 must also be constructed (*i.e.*, a list of the three neighboring triangles for each triangle of Grid 2), as well as a list of previously tested triangles. The search algorithm can then be implemented as follows:

Step 1: Choose the first Grid 1 vertex in the list and locate its enclosing triangle on Grid 2. (In the worst case this may involve searching all coarse grid triangles).
Step 2: Choose the next Grid 1 vertex in the list and (re)initialize the list of tested triangles. Set the current position to the beginning of the tested triangle list.
Step 3: Identify the neighbor of the current vertex which precedes it in the list, and get the address of its enclosing triangle.
Step 4: Test this triangle to see if it encloses the current vertex. If the test is positive, go to Step 2, otherwise put the triangle address in the list of tested triangles.
Step 5: Choose the next triangle in tested triangle list, and test all three neighbors of this triangle (if they have not already been tested). For each triangle, if the test is positive, go to Step 2, otherwise put the triangle address in the list of tested triangles.
Step 6: Go to Step 5

The process terminates when the end of the ordered vertex list is reached. In practice, the search for the first vertex may be bypassed or shortened by placing a boundary vertex at the start of the list, and making use of known boundary information. For subsequent vertices the enclosing triangle is usually located in a single or a small number of searches. The overall efficiency of the procedure depends mostly on the relative densities of the two grids, rather than their individual size. This procedure can be implemented equally well in three dimensions, replacing triangles with tetrahedra.

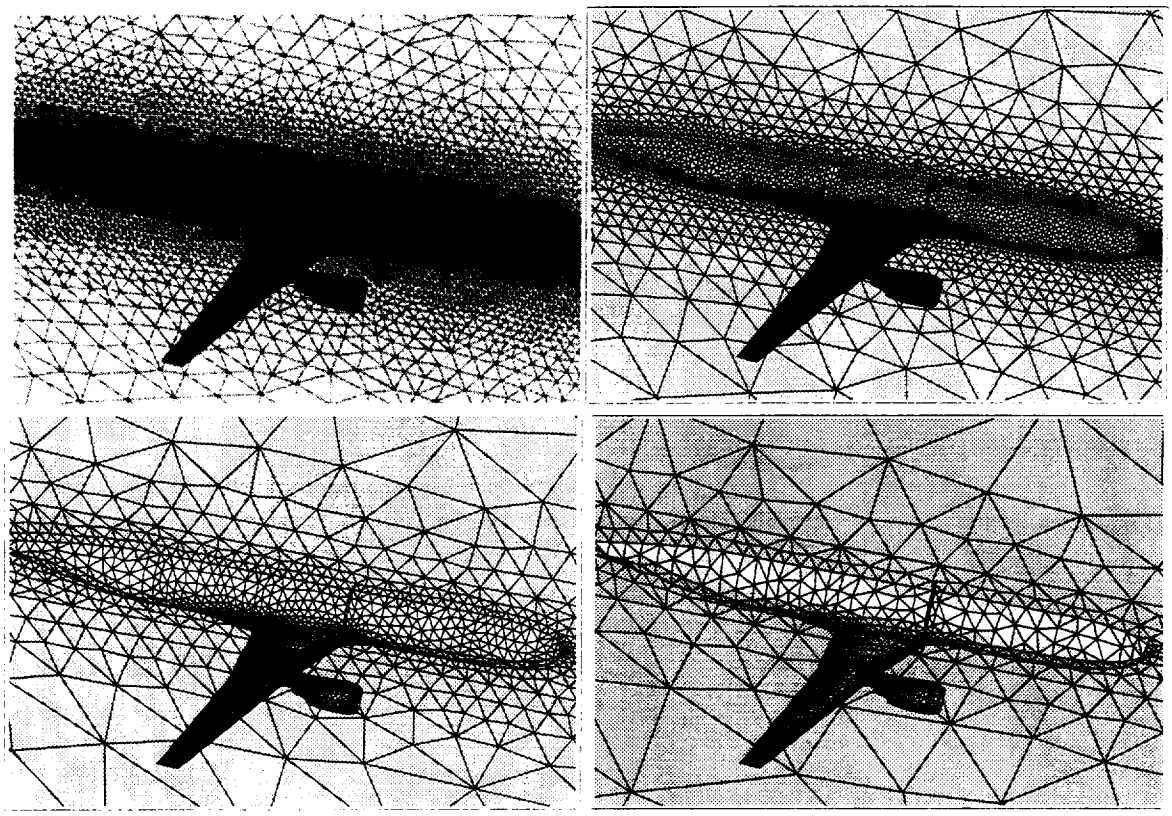

Figure 9: Illustration of sequence of grids employed to compute transonic flow over an aircraft configuration. The finest mesh contains 804,000 vertices and 4.5 million tetrahedra.

Figures 9 through 11 illustrate, as an example, the application of the overset-mesh multigrid technique to the solution of the Euler equations on three-dimensional tetrahedral meshes for a generic transport aircraft configuration using a vertex-based scheme. The finest mesh contains 804,000 vertices and approximately 4.5 million tetrahedra. A total of four mesh levels were employed in the multigrid sequence. The first, second, third, and fourth levels are depicted in Figure 9. The construction of all inter-grid transfer operators was achieved in the equivalent time required for two multigrid cycles, and thus represents an insignificant fraction of the overall solution time. The solution is depicted as a set of Mach contours on the aircraft surface in Figure 10. The freestream Mach number for this case is 0.768 and the incidence is 1.16 degrees. In Figure 11, the multigrid convergence rates using a V-cycle and a W-cycle, and the single-grid convergence rate are compared by plotting the history of the average flowfield density residuals versus the number of cycles. The multigrid method realizes a six order reduction over 100 W-cycles. Even though a multigrid W-cycle requires approximately twice the CPU time of an equivalent coarse grid cycle, the overall efficiency of the multigrid process for this case can be seen to be an order of magnitude larger than that of the single-grid (explicit) scheme. The entire multigrid run requires 96 Mwords of memory and a total of 45 minutes of CPU time on a single processor of the CRAY-C90. This observed multigrid

convergence rate is comparable to those obtained on structured meshes for similar problems using structured mesh multigrid algorithms [42, 43].

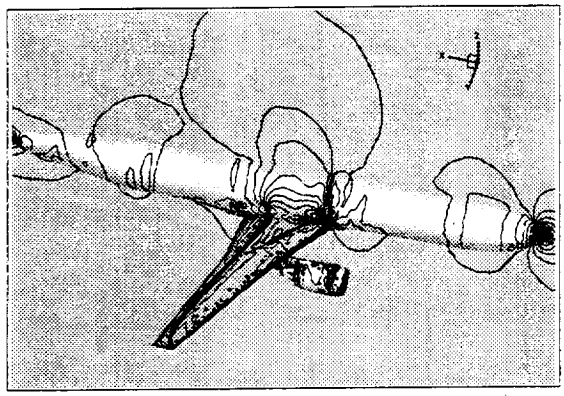

Figure 10: Computed Mach contours for transonic flow over an aircraft configuration. Mach = 0.768, Incidence=1.16 degrees.

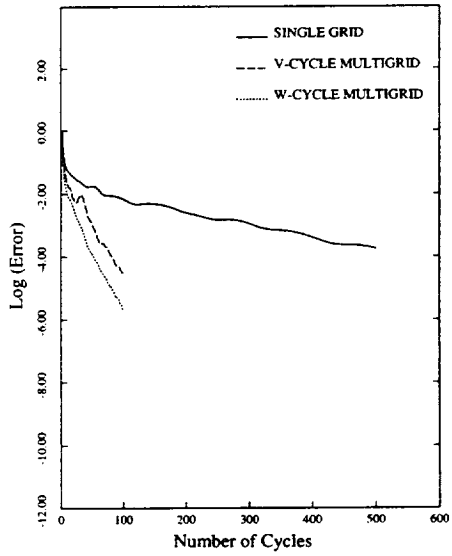

Figure 11: Convergence rates of V- and W-cycle overset-mesh multigrid schemes compared with single-grid explicit scheme.

The above description and examples concern multigrid implementations for vertex-based schemes. For cell-centered schemes, inter-grid transfers are typically based on volume weighting. For example, in the case where the coarse and fine grid cells are nested, the fine grid variables are transferred to the coarse grid by summing all variables of the nested fine grid cells for each coarse grid cell, weighted by the fraction of the fine grid cell volume to the enclosing coarse grid cell volume. In the case of non-nested overset meshes, a fine grid variable is transferred to the coarse grid according to the relation

$$W_c = \sum_{fine\ grid\ cells} w_f \frac{A_{fc}}{A_c} \tag{32}$$

where the c and f subscripts denote coarse and fine variables respectively, and A_{fc} represents the intersection area between the particular fine grid cell and the coarse grid cell of area A_c. In general, most intersections between a given pair of coarse and fine grid cells will be empty, and the sum need only be performed over the cells or non-zero intersections, as depicted in Figure 12. Equivalent prolongation operators can be constructed in a similar manner by reversing the role of the coarse and fine grids.

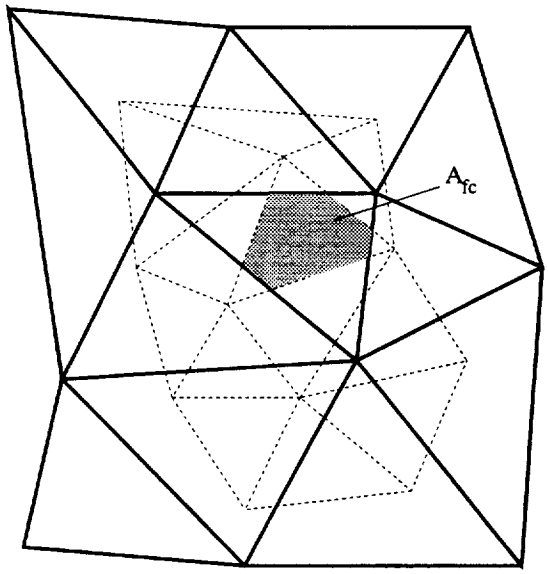

Figure 12: Illustration of intersected area between coarse and fine grid triangle pair for cell-centered multigrid scheme.

In order to construct such transfer operators, all intersecting cell areas must be determined, as well as the addresses of their originating coarse and fine grid cells. As is evident from Figure 12, these intersecting areas may take the form of complex polygons in two dimensions or polyhedra in three dimensions. These intersecting areas may be determined efficiently by first computing all intersections between the fine and coarse grid edges. The intersected edges of both grids thus constitutes a set of segments, and each segment is tagged with the address of its coarse and fine forming cells. In a single loop over all edge segments it is now possible to calculate the areas of all intersected cells using the Green-contour integration rule. This technique was implemented in [34] and is also referred to as Ramshaw's algorithm [44]. The determination of all intersected edges can be performed in O(N) time by first determining the enclosing coarse grid triangle for each fine grid vertex, using the graph-traversal algorithm described above.

3.3 Automated Coarse Mesh Construction

One of the main disadvantages of the overset-mesh multigrid algorithm is the non-automatic nature of the coarse grid construction. The user is required to manually generate the coarse-grid levels using an appropriate grid generator. A variety of techniques have been proposed to automate this procedure [45, 46, 47]. Most of them are based on producing a sequence of coarse mesh levels from a given fine mesh. This usually involves the removal of selected fine grid vertices and the retriangulation of the remaining grid points. The retriangulation procedure may be accomplished as a global operation, by regenerating the triangulation of the remaining coarse grid points, or incrementally, by removing each selected point sequentially and locally reconfiguring the mesh connectivity. For example, a reverse Delaunay point-insertion may be utilized in two-dimensions to remove mesh points. These techniques result in vertex-nested meshes, where the coarse grid vertices form a subset of the fine grid vertices, as shown in Figure 13. The triangulations themselves are not necessarily nested, since the connectivity of the coarse mesh need not be related to that of the fine mesh. Although the vertex-nested property may be employed to simplify the construction of the inter-grid transfer operators (*i.e.*, for example the $\overline{I}_h^H$ operator reduces to simple injection), the construction techniques discussed in the previous section for overset-mesh multigrid methods are equally applicable in this case.

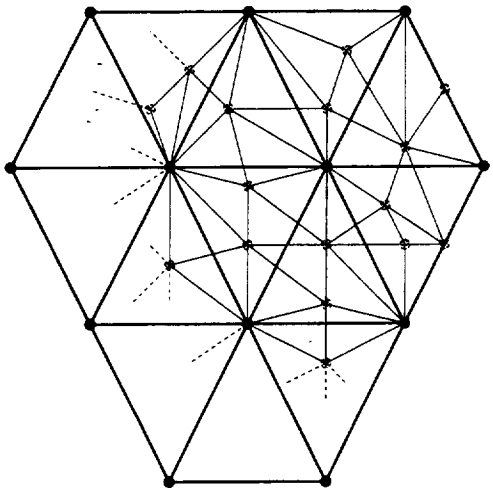

Figure 13: Illustration of a vertex-nested coarse and fine mesh pair.

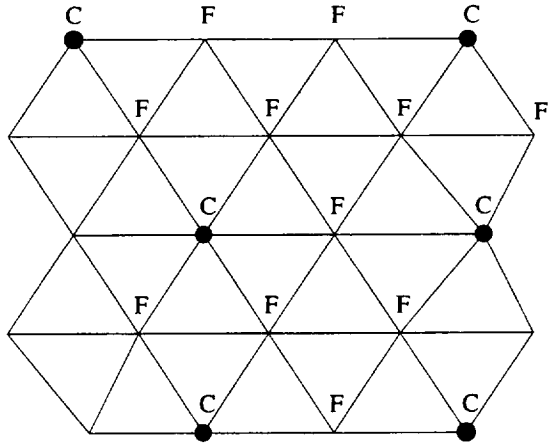

Figure 14: Determination of a subset of points **C** for the construction of a coarse level grid as a maximal independent set of the fine grid graph. Every **F** point is dominated (adjacent) to a coarse grid **C** point.

The point-removal procedure of automated coarsening strategies can be configured to generate "optimal" or near-optimal coarse meshes. This, of course, assumes some definition of optimal coarsening. A common strategy is to attempt to reproduce the coarsening characteristics encountered in structured mesh multigrid methods. Thus, coarse meshes which contain approximately half the resolution of the originating fine mesh in each coordinate direction throughout the entire domain are generally sought. If the fine grid is considered as a graph (*i.e.*, a collection of vertices and edges), then the problem may be stated in graph-theoretical terms as the construction of the maximal independent set of minimal cardinality (size) of the original graph. A subset of the vertices of a graph is termed an independent set if no two vertices of this subset are adjacent in the original graph. An independent set is maximal if any vertex not in the set is dominated (or adjacent) to at least one vertex of the set, as shown in Figure 14. The problem of generating maximal independent sets of minimum cardinality is nP complete (*i.e.*, cannot be achieved in polynomial time), and thus heuristic algorithms which generate near-optimal maximal independent sets are often employed. These are generally formulated as wavefront greedy-type algorithms, were the points being chosen form a front which propagates throughout the domain. Each chosen point corresponds to a common coarse and fine grid vertex. Each time a new point in the wavefront is chosen, all neighbors of this point which have not already been deleted are removed. The front is then advanced by placing the chosen point and all its deleted neighbors behind the front. The algorithm is greedy in that it never undoes what is done, *i.e.*, once a vertex is removed, it can never be reintroduced to resolve a conflicting situation. Such techniques are thus suboptimal, but are extremely fast, and result in coarse mesh levels which deliver competitive convergence rates. A more detailed description of an algorithm for constructing maximal independent sets will be given in the following section on agglomeration multigrid methods.

For certain problems, the uniform coarsening characteristics of maximal independent sets which mimic structured mesh multigrid methods may be far from optimal. This is particularly true for problems with large disparities in length scales and anisotropic problems. This topic is treated in more detail in section 4.4.

3.4 Agglomeration Methods

While automated coarsening strategies may relieve some of the practical difficulties in constructing coarse mesh levels for unstructured mesh multigrid algorithms, they do not address the issue of the robustness of the coarse grid constructions. For example, it may often be found that an automated coarsening procedure has removed one or several boundary mesh points which critically define the geometry, and the resulting changes in geometry between grid levels produces a slowdown or failure of the multigrid algorithm. In fact, the triangulation of a coarse point-set about a complex geometry can prove to be a difficult task. This may occur as a consequence of geometry features which are finer than the prescribed grid resolution, resulting in step changes in the geometry or even of the topology in going from a given mesh to the next coarser mesh.

These difficulties can be handled effectively by the agglomeration multigrid method [24, 40, 41]. Agglomeration methods are control-volume-based methods, and can thus be applied to either cell-centered or vertex-based schemes. For cell-centered schemes, the control-volumes are taken as the triangles themselves, whereas for a vertex-based scheme the control volumes are taken as the cells defined by the dual mesh formed by drawing the triangle median segments, as shown in Figure 15.

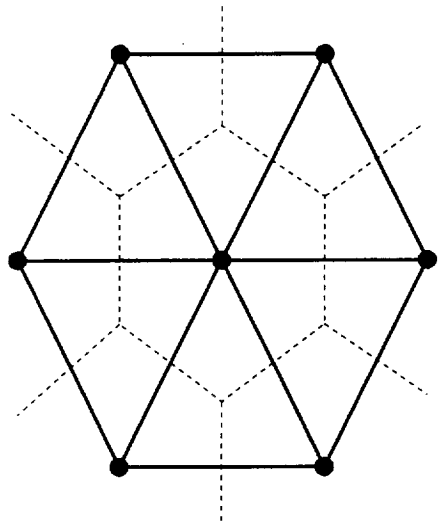

Figure 15: Median Dual Control-Volume for a Triangular Mesh.

The idea of the agglomeration method is to fuse together or agglomerate neighboring fine grid control volumes, creating a smaller set of larger polygonal (or polyhedral in three dimensions) control volumes. This process can be performed recursively, as shown in Figure 16, thus generating an entire sequence of coarse agglomerated meshes. The degree of the coarse agglomerated polygons increases on each coarser mesh level, but they always conform exactly to the original fine grid boundaries. The discretization, however, must be modified to enable the use of polyhedral cells on the coarse meshes.

The techniques employed for creating the coarse agglomerated grids are similar to the automated coarsening strategies described in the previous section. In fact there is a duality between agglomeration of control-volumes and point-removal. If each agglomerated control-volume is thought of as consisting of its seed point, *i.e.*, the point corresponding to the control volume from which the agglomeration process was initiated, and its agglomerated control-volumes (or corresponding points), as shown in Figure 17, then the seed point corresponds to a point which is retained for the coarse grid in the point-removal procedure, and the agglomerated points correspond to deleted points.

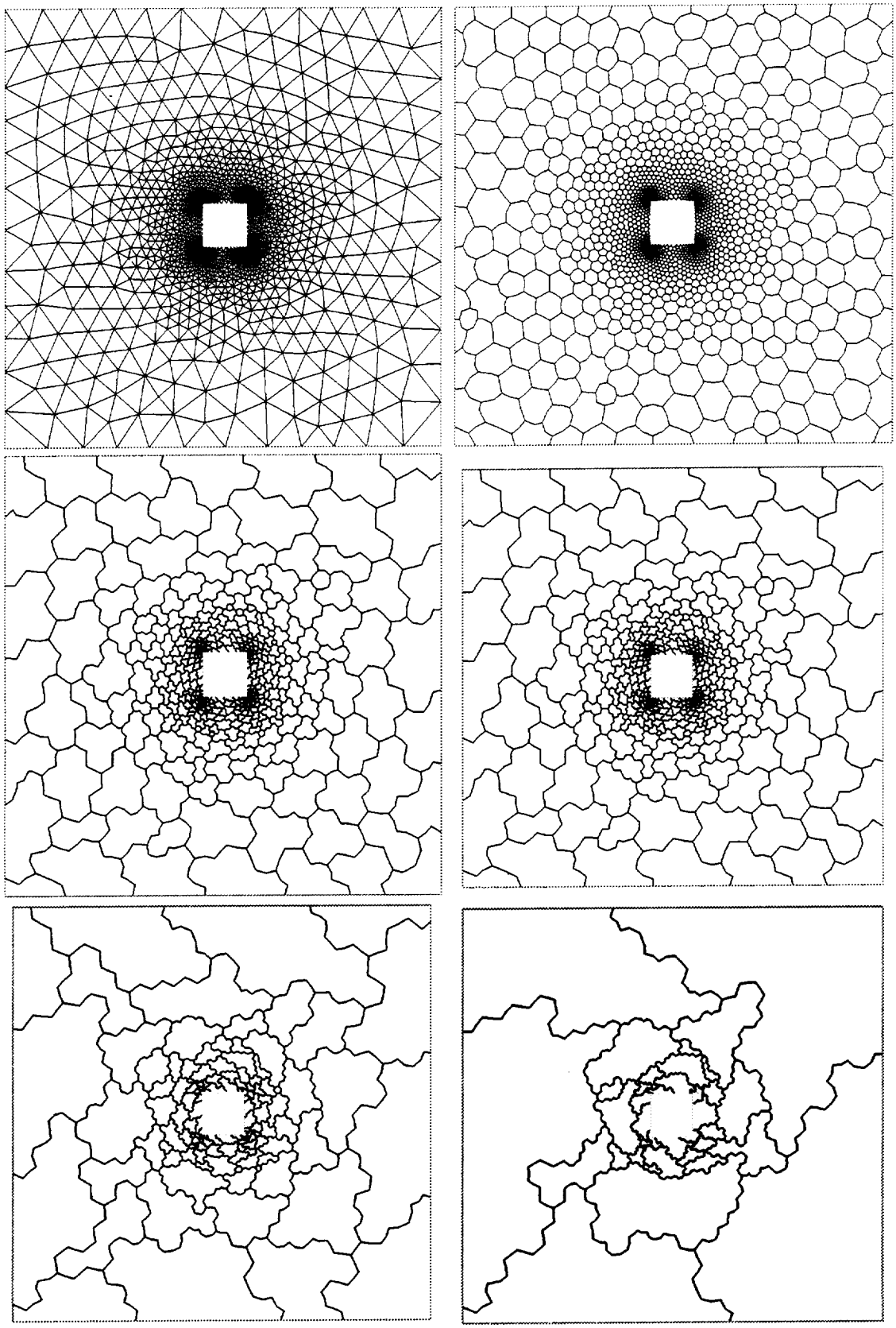

Figure 16: Original fine triangular mesh, its dual mesh, and coarse agglomerated mesh levels.

A wavefront greedy-type algorithm has been developed by the author and his co-workers for constructing coarse agglomerated grids which are maximal independent sets of the fine grid. The algorithm is detailed below:

Step 1: Pick a starting vertex on a surface element. Agglomerate control volumes associated with its neighboring vertices which are not already agglomerated.
Step 2: Define a front as comprised of the exterior faces of the agglomerated control volumes. Place the exposed edges in a queue.
Step 3: Pick the new starting vertex as the unprocessed vertex incident to a new starting edge which is chosen from the following choices given by order of priority:

- An edge on the front that is on the solid wall.
- An edge on the solid wall.
- An edge on the front that is on the far-field boundary.
- An edge on the far field boundary.
- The first edge in the queue.

Step 4: Agglomerate all neighboring control volumes of the current point which have not already been agglomerated to another vertex.
Step 5: Update the front and go to step 2 until the control volumes for all vertices have been agglomerated.

The queue of active vertices forms a wavefront which travels through the domain much like the advancing-front grid generation procedure. Once all the boundary vertices have been processed, the front edges form a simple queue, and they are extracted in the same order which they were inserted into the list. We have experimented with the use of more sophisticated priority queues to guide the advancement of the front. This has been implemented as a heap-list of the front edges, ordered either by the smallest edge length or the vertex closest to the interior boundary. In these implementations, a more regular propagation of the front is achieved by always choosing the smallest edge in the front, as in advancing-front grid generation strategies, or by always choosing the vertex closest to the inner boundary, thus advancing out from the inner boundary in equal distance level sets. In practice, little difference in overall multigrid performance has been noticed with these variants.

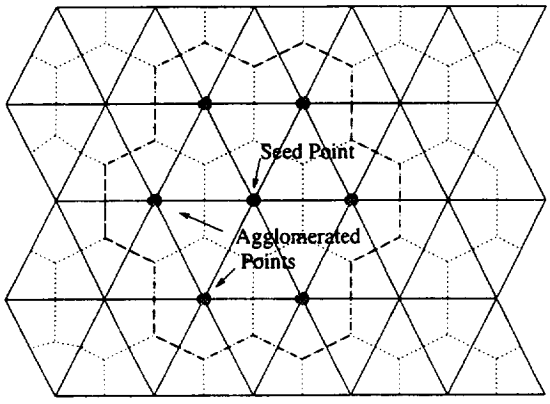

Figure 17: Illustration of seed point and agglomerated points in the agglomeration coarse grid construction strategy.

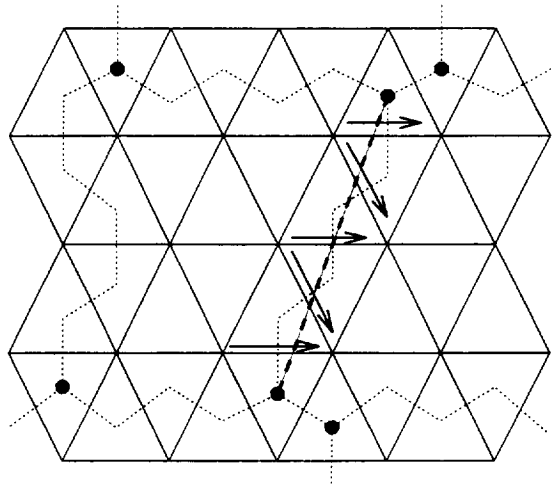

Figure 18: Replacement of segmented agglomerated edge by straight-line edge for flux integrations on coarse level agglomerated grid.

The algorithm executes in a time proportional to the number of vertices, and requires an insignificant amount of CPU time compared to the flow solution procedure. Coarse agglomerated meshes are produced which contain approximately four times fewer control volumes in two dimensions and eight times fewer control volumes in three dimensions than the previous finer grid.

The coarse level discretization is achieved by applying a finite-volume formulation to the complex polygonal control-volumes on these levels. A flux is computed for each edge of the polygonal control volume, using the average of the flow variables of each control-volume, on either side of the edge. A flux balance for each control volume is computed by integrating the fluxes around all boundary edges of the control volume. While the number of coarse level control volumes decreases approximately by a factor of four (eight in three dimensions) when going from a fine grid to the next coarser level, the number of edges only decreases by a factor of two (both in two and three dimensions) due to the increasing complexity of the control volumes on the coarser meshes. Since the complexity of updating the discrete coarse grid equations is closely related to the number of edges on these grids, this results in a substantial increase in the complexity of the overall multigrid cycle compared to using similarly coarsened triangular meshes. However, this complexity may be reduced by replacing each set of segmented edges which border on the same two agglomerated cells by a single straight edge, as shown in Figure 18. This step involves no approximation, since the flux across each segmented portion of the original edge involves the same flow variables, and invokes the edge normal in a linear fashion. Thus, the use of a single straight edge corresponds to the summation of all fluxes of the segmented edge. With this improvement, the effective number of edges decreases proportionally to the number of control-volumes when going from fine to coarse agglomerated grid levels, in both two and three dimensions, and the standard multigrid cycle complexity is recovered. The use of an edge-based data-structure, which stores the addresses of the cells on either side of each edge, and the x and y (and z in three dimensions) components of the edge normal area, is particularly attractive, since it enables the fine and coarse levels to be treated analogously, in spite of the non-triangular nature of the coarse grid cells.

The construction of the inter-grid transfer operators for agglomeration multigrid methods is particularly simple, since the coarse and fine level control volumes are fully nested. Fine to coarse residual restrictions are performed using volume weighting. Thus, the coarse restricted residual in a given control-volume is simply the sum of the residuals of the constituent fine grid control volumes, weighted by the ratio of the fine grid cell volume to the agglomerated coarse grid cell volume. The same operator is employed to transfer the flow variables from fine to coarse levels for the FAS scheme (*i.e.*, the $\overline{I}_h^H$ operator). Simple injection is employed for the prolongation operator. The correction computed on a coarse level agglomerated cell is applied directly and equally to all fine-level control volumes which are contained within the coarse level cell. This simple minded prolongation scheme is clearly less accurate than the linear interpolation strategy employed in the overset-mesh multigrid algorithm. However, because of the complex shapes of the coarse level cells, and since a piecewise constant coarse level solution is assumed, linear interpolation prolongation operators are not easily constructed. This is an unfortunate consequence of the use of coarse agglomerated levels, and will be addressed in more detail in section 4.2.

As an example of the efficiency of the agglomeration multigrid approach, the same example shown in Figures 9 through 11, for the overset-mesh multigrid method has been recomputed with the agglomeration approach [41]. Figure 19 depicts four of a total of six coarse agglomerated mesh levels, based on the finest mesh of Figure 9, which were employed for this calculation.

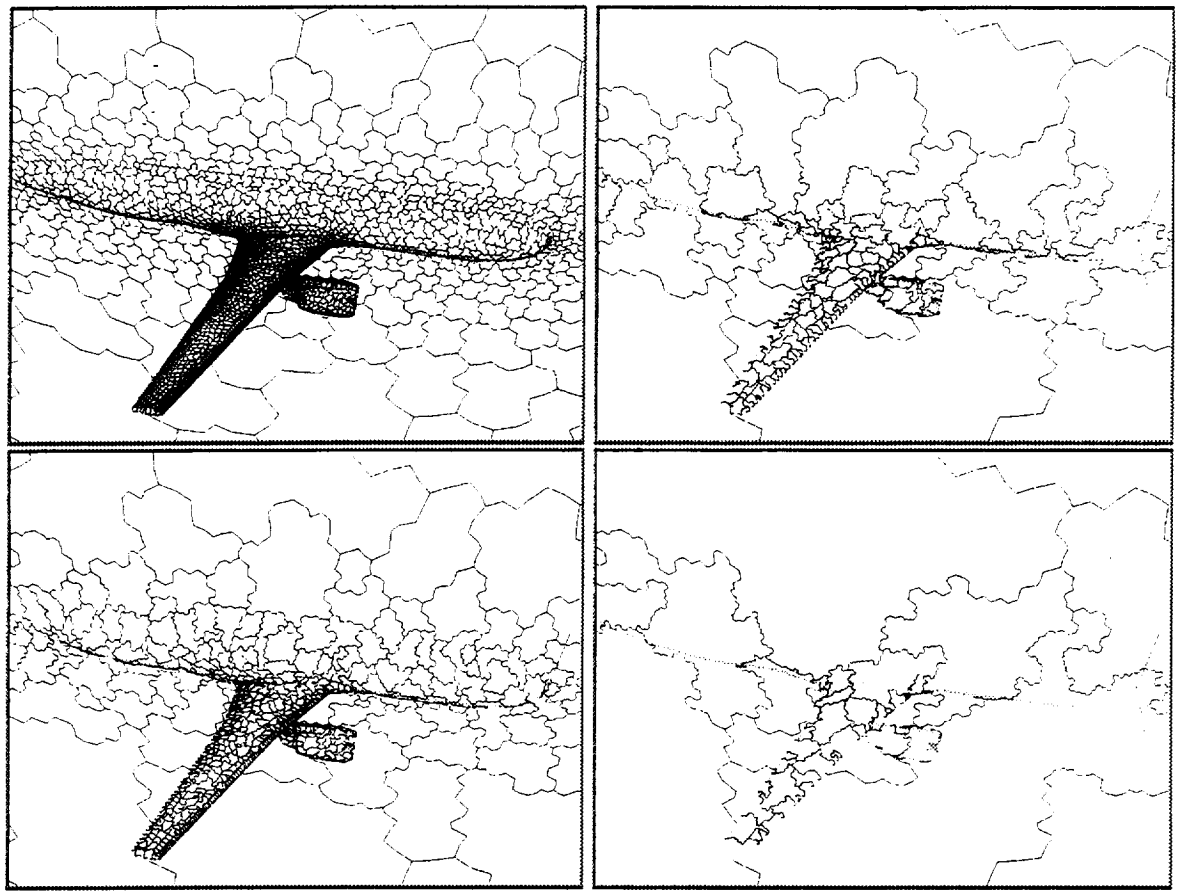

Figure 19: Several agglomerated levels employed in the calculation of flow over aircraft configuration.

As previously, the finest mesh contains 804,000 vertices and 4.5 million cells. However, in this case, a total of seven mesh levels were employed in the multigrid procedure. The coarsest mesh contains only 99 agglomerated cells. Recall that in the overset-mesh case, only four levels were possible, due to the difficulties involved in constructing consistent coarser meshes. The agglomeration procedure for generating the six coarser levels required approximately the equivalent amount of CPU time of two multigrid cycles. The flowfield conditions and the final solution are identical to those described in section 3.2.

The convergence rate given in Figure 20 illustrates how the efficiency of the agglomeration procedure parallels that of the overset-mesh approach, delivering a residual reduction of 6 orders of magnitude in 100 W-cycles. The amount of CPU time required per multigrid cycle is approximately the same for the agglomeration and overset-mesh multigrid algorithms, which corresponds to approximately double that of a single-grid explicit time-step. Therefore, as in the overset-mesh case, the agglomeration multigrid procedure provides an order of magnitude increase in solution efficiency over the single-grid approach. In addition, it affords a completely automatic coarse level construction, and enables the use of additional coarse levels over the overset-mesh procedure. In this particular case, the additional mesh levels provided an increased convergence rate over a four level agglomerated approach, but were not able to outperform the four level convergence rate of the overset-mesh multigrid approach. This case required approximately 50 minutes of CPU time on a single CRAY-C90 processor, and 97 Mwords of memory.

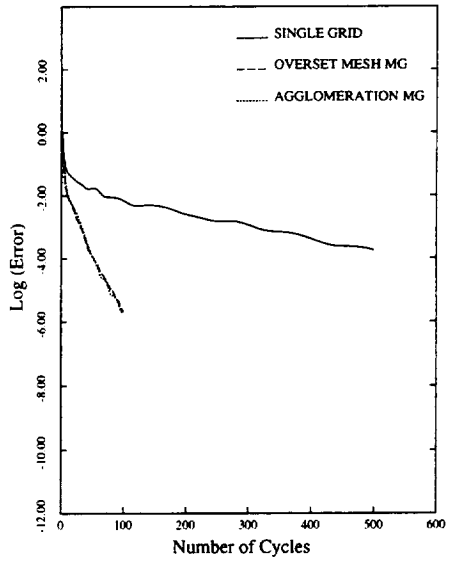

Figure 20: Convergence rate of agglomeration multigrid algorithm compared with overset-mesh multigrid algorithm and single-grid algorithm for computation of flow over aircraft configuration.

3.5 Algebraic Multigrid Methods

Algebraic multigrid methods are methods that enable the efficient solution of systems of algebraic equations, which are not necessarily derived from the spatial discretization of a partial differential equation [25]. In fact, the notion of a grid, of linear interpolation in space, and spatial smoothness are not always possible in this context. Thus, algebraic multigrid methods require the redefinition of such concepts in the context of algebraic rather than geometric quantities, in order to make use of traditional multigrid principles. In some sense, algebraic multigrid represents an abstraction or generalization of the fundamental multigrid procedures, and may lead to more fundamental understanding of the multigrid solution principles, and perhaps more general strategies. However, these methods are not nearly as well developed as their geometric counterparts, and are usually more cumbersome and restrictive. For example, present algebraic multigrid methods are only applicable to linear problems.

Consider the problem defined by the set of algebraic equations

$$Ax = b \tag{33}$$

or, for an N by N matrix,

$$\sum_{j=1}^{N} a_{ij} x_j = b_i \qquad for\ i = 1, 2, ..., N \tag{34}$$

The multigrid solution process consists of deriving a smaller set of equations from the above set, denoted as

$$A_c x_c = b_c \tag{35}$$

or

$$\sum_{j=1}^{N_c} a_{cij} x_{cj} = b_{ci} \quad for \;\; i = 1, 2, ..., N_c \tag{36}$$

where N_c represents the number of coarse level variables, and the matrix A_c with elements a_{cij} remains to be defined. The variables x_{ci} are usually taken as a subset of the fine level set of variables x_i. Restriction and prolongation operators which map fine to coarse (I_h^H) and coarse to fine (I_H^h) variables must also be defined. Once these steps have been accomplished, a multigrid cycle may be written as

$$x^{new} = G\, x^{old} + I_H^h[(A_c)^{-1} b_c - x_c^{old}] \tag{37}$$

where G represents the fine grid smoother, and it is assumed that the coarse grid matrix A_c may be easily inverted. The above sequence of operators represents a two-grid multigrid cycle. This is described here for simplicity, and in practice, a multiple level cycle may be defined by recursive application of the above two-grid procedure.

A standard algebraic multigrid construction is to take the restriction operator as the transpose of the prolongation operator:

$$I_h^H = (I_H^h)^T \tag{38}$$

as was done in the overset-mesh multigrid algorithm, and to use the Galerkin coarse grid operator construction to define the coarse level matrix A_c:

$$A_c = (I_H^h)^T A\, I_H^h \tag{39}$$

as described in section 2.3. Once these steps are taken, the complete algebraic multigrid algorithm is determined solely by the definition of the prolongation operator and the set of coarse level variables.

Since geometric information is not available, the coarse level variable sets must be determined from the algebraic information contained in the matrix A. To do this, we make use of the graph of the matrix A. The graph of a sparse matrix is defined as the graph which is obtained by drawing an edge between the two vertices which correspond to the row and column number of each non-zero entry in the matrix. An algorithm which generates a maximal independent set of this graph may be utilized to construct a coarse level subset of variables, just as in the agglomeration or automated coarsening approaches for geometric multigrid. Algebraic multigrid, however, adds an extra degree of sophistication to the process, by considering the magnitude of the non-zero matrix entries. Coarsening is performed preferentially along edges associated with large matrix entries, since this represents neighboring equations which are strongly coupled, and which will thus have similar errors (*i.e.*, the error distribution will be smooth in that direction). For problems which result from stretched mesh discretizations, this reproduces the effect of semi-coarsening.

In order to construct a suitable prolongation operator, we require that smooth error be in the range of interpolation. This assumes a definition of the concept of smoothness in an algebraic sense. An algebraically smooth error may be defined as an error which is slow to converge using a simple relaxation scheme, *i.e.*,

$$\|Ge\| \sim \|e\| \tag{40}$$

This definition means nothing more than the error must be handled by the coarser grid. However, it provides a basis for constructing a smooth operator. Since simple relaxation schemes generally refer to explicit schemes, which compute updates based on the residual, equation (40) implies a small residual for smooth errors:

$$Ae = r \sim 0 \tag{41}$$

Prolongation to fine level variables which are not contained on the coarse level may be computed by setting the residuals at those points to zero:

$$\sum a_{ij} v_j = 0 \tag{42}$$

where v_j represents the prolongated correction at vertex j. If v_{cj} represents the coarse grid correction, an algebraically smooth prolongation operator can therefore be defined as follows:

$$\begin{aligned} v_i &= v_{ci} && for\, i \in coarse\, grid \\ v_i &= \frac{1}{-a_{ii}} \sum_{j \neq i}^{N} a_{ij} v_j && for\, i \ni coarse\, grid \end{aligned} \tag{43}$$

Thus, for fine grid points which are contained in the coarse grid, corrections are obtained by injection of the coarse grid values, whereas for fine grid points which are not contained in the coarse grid, corrections are obtained by application of equation (42). The summation need only be performed over the non-zero elements a_{ij} of row i of the matrix A, which corresponds to the graph neighbors of vertex i. In the event all such neighbors are coarse grid points, an explicit formula for the prolongation operator is obtained from equation (43). In practice, all such neighboring values are seldom contained in the coarse grid. Equation (43) may then be solved approximately by applying the same process to each neighboring point which is not a coarse grid point. This process may be applied recursively, each time enlarging the stencil for the prolongation of the value v_i. An approximate stencil may then be constructed by deleting all points of the stencil which are not coarse grid points. In [25], for example, a construction of this type using up to two levels of neighbors is described in detail. This results in an approximate but explicit form of the prolongation operator, which can be constructed as a preprocessing operation and then employed throughout the multigrid solution process.

One of the drawbacks of algebraic multigrid methods is the complexity of their construction. The prolongation operator described above is not only used to transfer corrections from coarse to fine grids, but also enters into the construction of the coarse grid operator through equation (39). Thus, a prolongation operator with large or widely varying stencils may result in considerably complex coarse grid operators. In fact, the coarse grid operator obtained through equation (39) is usually much denser (contains relatively more non-zero elements) than the original fine grid operator, which results in increased coarse grid complexities for the multigrid cycle. Thus, the construction of algebraic multigrid methods necessarily involves a trade-off between accuracy of the operators and complexity of the coarse grids.

3.6 Similarities Between Agglomeration and Automated Node Nested Methods

The unstructured multigrid approaches described above have been presented in order of decreasing geometric and increasing automatic and algebraic character. Many of these methods involve similar techniques, and some can even be shown to be completely equivalent under certain conditions.

For example, it has been shown how the agglomeration process can be interpreted as a point-removal technique. There is complete duality between these two techniques: the agglomeration procedure operates on control volumes of the dual mesh, while the point-removal technique operates on vertices of the original mesh. However, a notable difference between these two methods is that the end result of the agglomeration procedure not only consists of a coarse set of control volumes, but also an implied graph upon which the coarse grid operator is based, *i.e,* the graph obtained by drawing an edge between every pair of neighboring agglomerated control volumes. On the other hand, the point-removal technique only determines a coarse level set of vertices, without any implied graph. The graph of the coarse level operator is determined by retriangulating these vertices, a step which in principle may be completely independent of the coarsening process.

The question then arises as to whether the implied graph of an agglomerated coarse grid is representative of some particular triangulation. If this is so, then the agglomeration procedure becomes completely equivalent to a combined point-removal and retriangulation technique for constructing coarse triangular meshes, with the consequence that agglomeration multigrid becomes nothing more than an overset triangular mesh multigrid technique where the coarse meshes are generated automatically.

By examining a coarse agglomerated grid, one can identify points on the grid where three agglomerated cells meet, as shown in Figure 21.

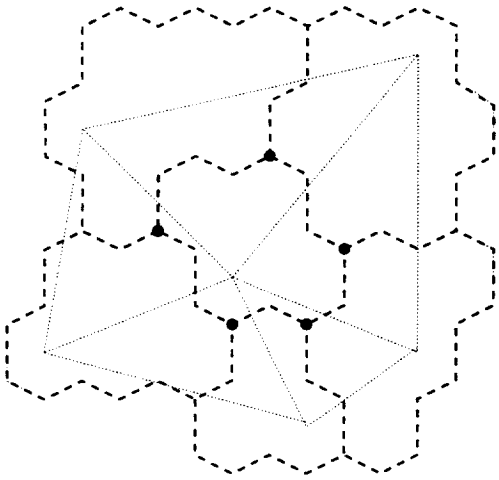

Figure 21: Illustration of dual triangulation implied by agglomerated grid. Any point within the agglomerated cells may be employed as the triangle vertices.

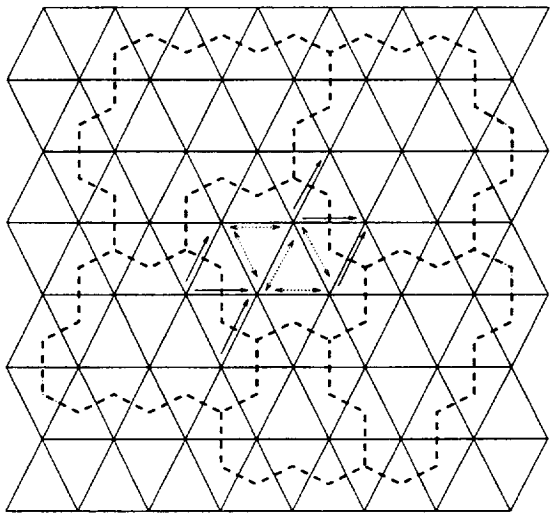

Figure 22: Contributions of edges interior to agglomerated cell cancel out for convection equation, while those along common boundaries may be summed.

These points define triangular cells, which can be constructed by joining the centroids of the three neighboring agglomerated cells together with three straight-line edges. The union of these edges constitutes a dual of the agglomerated grid, which is seen to be a triangular mesh of the agglomerated cell centroids. In fact, the centroids were chosen here for convenience, and any point within the agglomerated cells may be used to define the vertices of the triangulation. Another choice would be to draw the edges joining the centroids of the original control volume which initiated the agglomeration of each coarse cell, *i.e.*, the seed points of the cells. Since these seed points form a subset of the original triangular mesh vertices, an agglomeration procedure of this type becomes completely equivalent to a point-removal and retriangulation process, and the agglomeration multigrid approach becomes identical to the overset-mesh approach. This also permits the use of linear

interpolation for the restriction and prolongation operators as implemented in the latter approach. (Recall that this was not possible in the agglomeration formulation).

One of the difficulties with this interpretation is that the triangular meshes derived from the agglomeration process in this manner may not always be valid, *i.e.*, cross-overs may occur which result in negative area triangles. While such situations can often be remedied by displacing the vertices of the triangulation within the agglomerated cells, since there is an added degree of freedom in choosing the precise location of these points, uncorrectable triangulations inevitably occur. This highlights the main advantage of the agglomeration process, that is, its ability to construct valid coarse grids, regardless of the geometry. The fact that the coarse agglomerated grids conform exactly to the boundaries of the domain on every level is another manifestation of this principle. The robustness of the coarse level agglomeration construction is largely due to the fact that the agglomerated control volumes are formed by directly summing fine grid control volumes. Thus, the coarse level discrete equations may be thought of as formed by the sum of the subset of fine level discrete equations which are contained in each coarse agglomerated cell. Many desirable properties which may have been built into the fine level discretization, (*i.e.*, positivity, diagonal dominance, etc.) can therefore be assured on the coarse level equations. This algebraic character of the agglomeration procedure is one of its greatest assets.

3.7 Similarities Between Agglomeration and Algebraic Methods: Construction of an Agglomeration Method for the Navier-Stokes Equations

As we have hinted above, agglomeration techniques can also be interpreted as algebraic multigrid methods. In fact, our current implementation of agglomeration multigrid follows the algebraic interpretation as closely as possible [41, 48, 49]. The basic premise is that convergence acceleration techniques should not be bound by geometry-based constraints, and the removal of the influence of geometry from the multigrid process should lead to a more robust strategy. Consider, for example, the application of boundary conditions on the coarse agglomerated grids. For very large cells which overlap regions of the boundary which correspond to multiple different boundary conditions, the application of coarse level boundary conditions is ambiguous. Rather than physically applying boundary conditions on the coarse grid, an algebraic approach consists of inferring a possibly composite coarse level boundary condition from the boundary conditions and their associated equations on all constituent fine grid cells of the coarse agglomerated cell.

The use of an edge-based data-structure to represent discretizations in the agglomeration multigrid method not only permits a similar treatment of all coarse and fine-grid levels, but also provides an algebraic interpretation of the procedure, since the set of edges of the discretization on a given level may be viewed as the graph of the sparse matrix defined by the discrete operator. In fact, the algebraic point of view of agglomeration multigrid has not only resulted in a more robust algorithm, but has also enabled the extension of agglomeration multigrid techniques to other types of equations, such as the Navier-Stokes equations, and turbulence transport equations [48, 49, 50].

For a discrete operator which relies on a nearest neighbor stencil on a triangular or tetrahedral grid, the graph of its sparse matrix is equivalent to the graph of the grid. In such cases, the agglomeration procedure defined above becomes equivalent to an algebraic procedure for constructing coarse-level equations sets. In fact, the methods described in the algebraic multigrid literature may be utilized in agglomeration multigrid methods, while the maximal independent-set techniques devised for agglomeration methods may also be applied to algebraic strategies. A notable difference between the two approaches is that, while the agglomeration procedure itself defines the stencil of the coarse level operator, *i.e.*, the stencil of nearest agglomerated cell neighbors, in the algebraic multigrid case, the coarse grid stencil is defined by the Galerkin construction of the coarse grid operator (*c.f.* equation (39)). In fact, algebraic multigrid does not even contain the notion of a nearest neighbor, since the coarsening process does not generate its own graph. The stencils of

the Galerkin coarse grid operator are in general much more dense than those of an agglomerated nearest-neighbor graph, and are much more expensive to evaluate. This becomes particularly difficult in multi-level applications (as opposed to two-level applications), where the process must be invoked recursively.

Under certain conditions, the stencil of the Galerkin coarse grid operator becomes equivalent to a nearest neighbor stencil. For example, the agglomeration algorithm described previously for an advection equation or the Euler equations employs an injection operator for the prolongation operator, and volume weighted summation for the restriction operator. If A represents the fine grid discrete operator, then the Galerkin coarse grid operator

$$A_c = I_h^H A I_H^h \tag{44}$$

using injection and volume-weighting for prolongation and restriction, respectively, can be viewed as an equation summation technique, where each coarse grid discrete equation associated with an agglomerated cell is obtained by summing all the discrete equations of the fine level within the given agglomerated cell, and replacing the fine grid variables by the corresponding coarse grid variables. When the discrete equations for an advection equation are expressed in edge format, the contributions along all edges interior to the agglomerated cell cancel out, while all edge contributions which border on two neighboring agglomerated cells may simply be summed, as shown in Figure 22. This procedure naturally results in a nearest neighbor stencil, and in fact reproduces the same coarse level discrete equations for an advection equation obtained by the finite-volume analysis of the coarse agglomerated cell. For the Euler equations, complete equivalence can also be demonstrated provided the non-linearities of the equations are handled appropriately.

Therefore, in this case, the agglomeration and Galerkin coarse grid operator constructions are equivalent, providing a further demonstration of the algebraic character of the agglomeration procedure. This is particularly convenient, since it provides a mechanism for extending agglomeration procedures to equations of higher degree, such the Poisson equation, or the diffusion terms of the Navier-Stokes equations. The discretization of such equations on a mesh of complex polygonal control-volumes is not obvious, since this usually requires the computation of gradients as an intermediate step. However, using the above Galerkin construction, with the injection prolongation and volume-weighted summation restriction operator, a simple nearest neighbor stencil form of the coarse grid equations is obtained. In fact, if we employ the same edge-based representation in the discretization of the diffusion terms [36, 51], a similar equation summing technique can be invoked to construct the coarse grid equations.

Coarse Grid	Coarse Grid Op.	Restriction	Prolongation	Convergence
Independent	Rediscretization	Linear	Linear	0.100
Triangulated Seed Pts	Rediscretization	Linear	Linear	0.125
Agglomerated	Galerkin	Injection	Injection	0.512
Agglomerated	Scaled Galerkin	Injection	Injection	0.254

Table 1: Effect of coarse grid operator for agglomeration multigrid.

Table 1 compares the convergence rate obtained by this method for the solution of Laplace's equation on a two-grid agglomerated system, using the fine grid and first coarse-level agglomerated grid depicted in Figure 16, with that of a two grid overset mesh multigrid approach, using a coarse triangular grid generated independently, and a coarse triangular grid based on the seed points of the coarse agglomerated grid. For all tabulated results, a multigrid V-cycle is employed, with 3 Jacobi pre- and post-smoothing sweeps on the fine grid, and 200 sweeps on the coarse grid (in order to fully converge the coarse grid equations of the two-grid system). In both cases, the convergence rates are much faster for the overset-mesh multigrid approach, using a geometric discretization for the coarse grid operator, than that achieved with the Galerkin coarse grid operator within the agglomeration framework. This degradation of convergence may be due to the different coarse grid operator, or the prolongation and restriction operators (which are taken as linear interpolation in the overset grid method, and injection in the agglomeration approach). The effect of the relative "quality" of the coarse grid can be assessed by the difference in convergence rates between the overset grid method using an independent coarse grid, and using the triangulated agglomeration grid. These differences are rather small thus demonstrating the suitability of the agglomerated grid.

The main problem with the above formulation is that the accuracy of the transfer operators is insufficient to guarantee efficient convergence rates. A necessary relation for ensuring multigrid efficiency is given by [18, 27]:

$$m_r + m_p > m \tag{45}$$

where m_r and m_p are defined as the highest degree plus one of the polynomials that are interpolated exactly by the restriction operator I_h^H and prolongation operator I_H^h respectively, and m is the order of the partial differential equation to be solved. In this case, injection is used for the transfer operators, thus m_r and m_p are both equal to 1. Since the order of Laplace's equation is 2, the strict inequality is not satisfied. It is interesting to note that in the case of the convection equation (or the Euler equations) the strict inequality is satisfied since the order of the equations is 1, rather than 2. This explains the success of the control-volume formulation of the coarse grid equations for inviscid problems in the agglomeration multigrid approach.

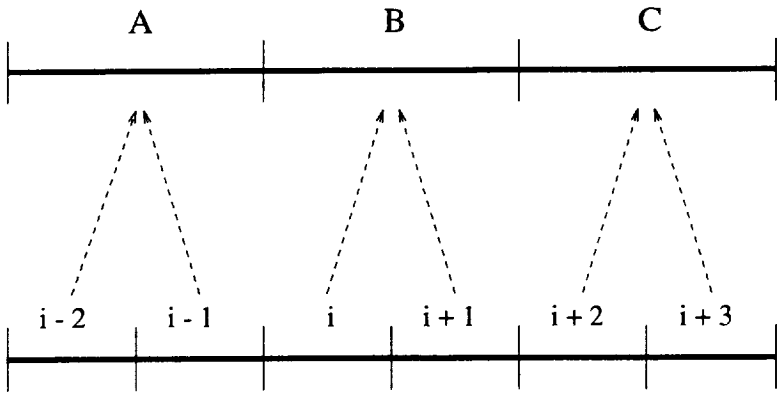

Figure 23: Simple one-dimensional two-grid multigrid example.

The accuracy of the restriction and/or prolongation operators must therefore be increased for diffusion-type problems. This will affect both the transfer operators themselves, and the coarse grid operator. Rather than strictly adhering to the construction given by equation (4), which can become considerably involved for more complex interpolation operators, we seek a simplified coarse

grid operator by examining a one dimensional example. The discretization of a Poisson equation on a one dimensional grid yields the discrete equation:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f \tag{46}$$

If a coarse grid is constructed by agglomerating neighboring pairs of cells, as shown in Figure 23, the restriction operator based on injection corresponds to simple summation of the i and $i-1$ residuals.
The prolongation operator based on injection reads

$$\begin{aligned} u_{i-2} = u_{i-1} &= \overline{u}_A \\ u_i = u_{i+1} &= \overline{u}_B \\ u_{i+1} = u_{i+2} &= \overline{u}_C \end{aligned} \tag{47}$$

where the overbar indicates coarse grid values. The discrete coarse grid equation at B is obtained by the Galerkin construction:

$$I_h^H \, A_h \, I_H^h u \; = \; I_h^H f_h \tag{48}$$

which yields:

$$\frac{\overline{u}_A - 2\overline{u}_B + \overline{u}_C}{2h^2} = f \tag{49}$$

This obviously results in an inconsistency with the fine grid discretization, for if we were to directly discretize the Poisson equation on the coarse grid we obtain

$$\frac{\overline{u}_A - 2\overline{u}_B + \overline{u}_C}{4h^2} = f \tag{50}$$

The left-hand sides of equations (49) and (50) differ by a factor of 2. This inconsistency is entirely due to the use of an inadequate prolongation operator. If we use linear interpolation for the prolongation operator, *i.e.*,

$$\begin{aligned} u_{i-1} &= \frac{3}{4}\overline{u}_A + \frac{1}{4}\overline{u}_B \\ u_i &= \frac{1}{4}\overline{u}_A + \frac{3}{4}\overline{u}_B \\ u_{i+1} &= \frac{3}{4}\overline{u}_A + \frac{1}{4}\overline{u}_C \end{aligned} \tag{51}$$

but retain injection for the restriction operator, it can be verified that equation (50) is recovered as the resulting Galerkin coarse grid operator. Note also that the inequality of equation (45) is satisfied for this case. This one-dimensional example suggests a simple fix for the multi-dimensional Galerkin coarse grid operator using injection [48, 50]. We replace the operator $I_h^H \, A \, I_H^h$ with

$$A_c \; = \; \frac{I_h^H \, A \, I_H^h}{2^{n-1}} \tag{52}$$

where $n = 2, 3, .., k$ represents the coarse-grid levels. In Table 1, this modified or scaled coarse grid operator can be seen to show a significant improvement in convergence rate over the original coarse grid operator for the two grid system. The advantage of this construction is that the coarse grid operator still relies on the nearest neighbor stencil, and maintains low complexity.

Armed with this result, we may now proceed to construct an agglomeration multigrid strategy for the Reynolds-averaged Navier-Stokes equations [49]. The graph-based agglomeration procedure described in section 3.4 is employed to construct coarse levels using a slightly modified version of the maximal independent set algorithm. Volume weighted summation is employed for restricting residuals and flow variables to coarser grid levels, while injection is employed for transferring coarse level corrections back to the fine grid. The coarse level operator is obtained using the Galerkin construction for the convective terms, while the scaled Galerkin result of equation (52) is employed for the viscous terms. The coarse grid operator is thus given by

$$\begin{aligned} A_c &= A_{cinviscid} + A_{cviscous} \\ A_c &= I_h^H \, A_{inviscid} \, I_H^h + \frac{I_h^H \, A_{viscous} \, I_H^h}{2^{n-1}} \end{aligned} \tag{53}$$

where n represents the grid level. The turbulence model of Spalart and Allmaras [52] is employed to account for turbulence effects. This model consists of a single field-equation which contains convective, diffusive and source terms. The turbulence equation is solved decoupled from the flow equations, but in a similar multigrid fashion. The convergence of the flow and turbulence equations is accelerated by the use of local time-stepping and residual averaging on all mesh levels.

As an example, this procedure has been employed to compute the flow over a rectangular wing with a partial-span flap mounted in a wind-tunnel. The mesh for this case was generated using the advancing layers method [53], and is highly stretched in the regions near the airfoil surfaces. The solution was obtained on two different meshes, a coarse mesh of 300,00 vertices (or 1.7 million tetrahedra) with a normal wall spacing of 10^{-5} at the airfoil surfaces, and a finer mesh of 2.3 million vertices (or 13.6 million tetrahedra) with a wall spacing of 5×10^{-6}. This fine mesh represents a uniform refinement of the coarser mesh. The coarser mesh is depicted in Figure 24, along with the set of coarse level agglomerated grids used in the calculation, while the fine grid is shown in Figure 25. A total of five mesh levels were employed for the coarse mesh calculation, and a total of six mesh levels were used for the fine mesh calculation. The fine grid solution is depicted qualitatively in Figure 26, as a set of Mach contours on the wind-tunnel wall, and density contours on the airfoil surface. The freestream Mach number is 0.2 for this case, the Reynolds number is 2 million, and the incidence is 10 degrees. The convergence rates of the multigrid scheme for these two cases are depicted in Figure 27, as a history of the RMS average of the density residuals throughout the flowfield and the lift coefficient versus the number of multigrid cycles. In both cases, convergence of four to five orders of magnitude is obtained in three to four hundred multigrid W-cycles. The most notable feature of this comparison is the fact that the convergence rates for the coarse and fine grids are almost identical. Considering the fact that the fine grid contains eight times the resolution of the coarse grid, this provides a strong indication of the grid independent convergence rate property of multigrid for very large three-dimensional problems. This type of convergence rate is equivalent to that obtained with similar schemes on structured meshes [43, 54]. The fine grid calculation required a total of 425 Mwords of memory, and 18 hours of total CPU time for 300 multigrid cycles on a CRAY C-90. Using all 16 processors of the machine in parallel, this calculation was achieved in 1.25 hours of wall clock time.

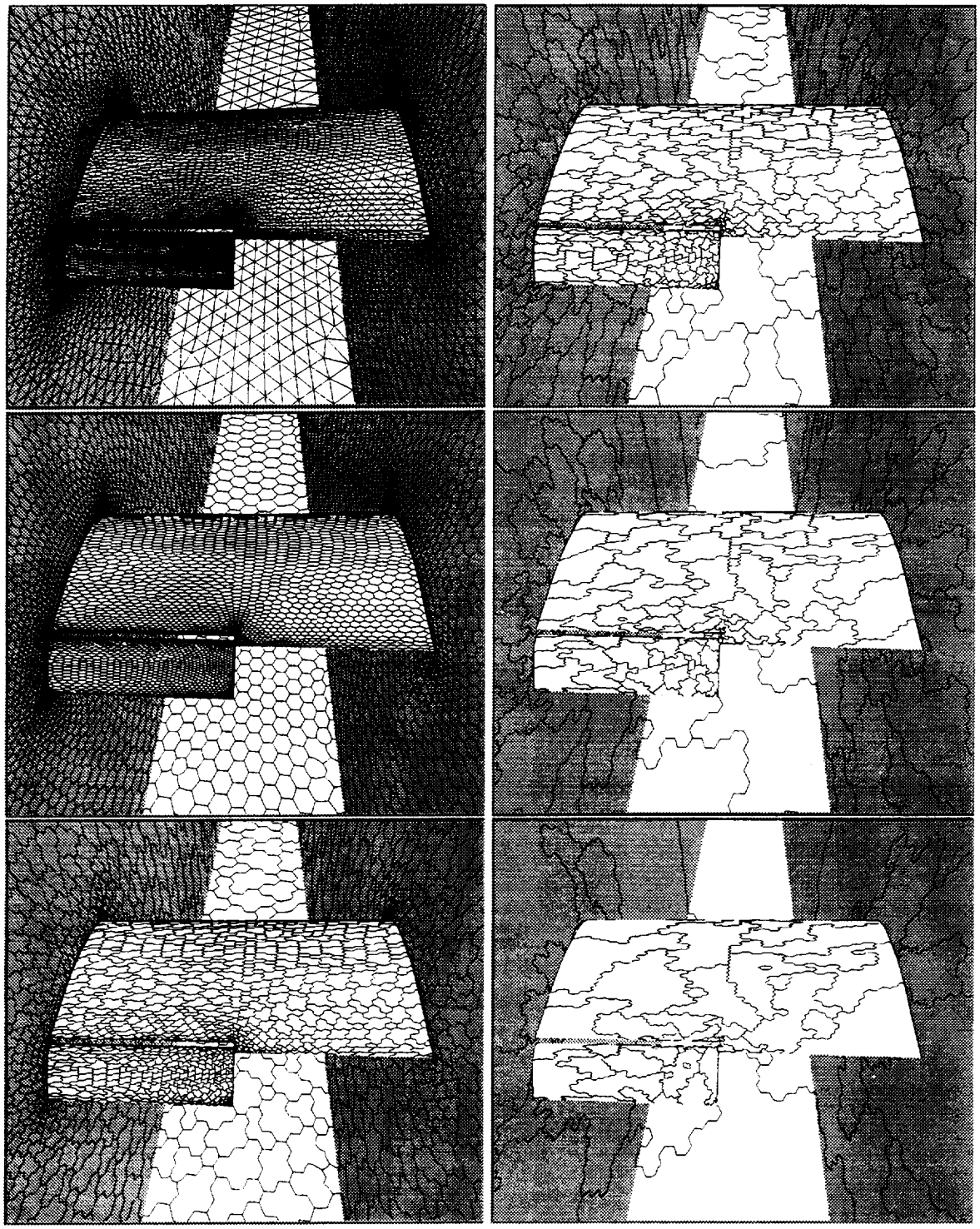

Figure 24: Unstructured grid for partial-span flap wing geometry, its dual, and four coarse agglomerated grid levels.

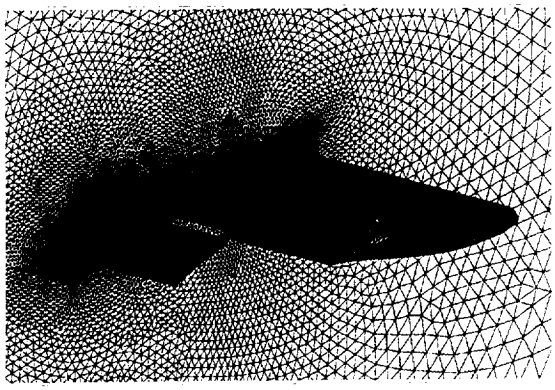

Figure 25: Fine unstructured mesh employed for computation of flow over partial-span flap wing. (Number of points = 2.4 million, Number of tetrahedra = 13.6 million).

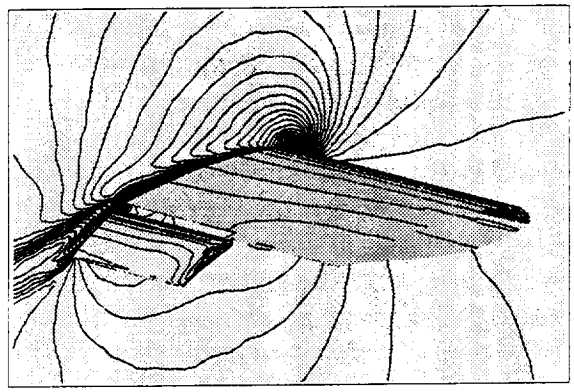

Figure 26: Computed solution for flow over partial-span flap wing. Mach contours are displayed on symmetry plane, density contours on wing surface.

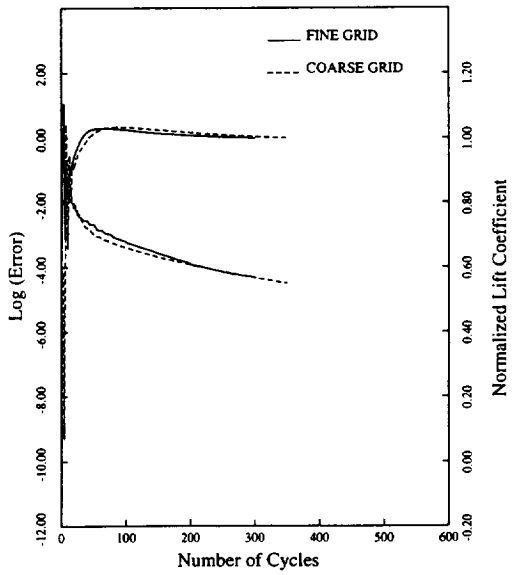

Figure 27: Convergence rates on coarse (300,000 points) and fine (2.4 million points) grid for flow over partial-span flap wing.

4 Additional Multigrid Topics

4.1 Additive Correction Schemes

Many multigrid-like methods have been developed for both geometric and algebraic problems. These have been used in a straight multigrid-like fashion, involving the creation and solution of multiple coarse level problems [55, 56, 57], or in domain decomposition strategies, as techniques for providing a level of global coupling between the various domains, which are usually solved implicitly within each domain [46, 58, 59]. Multigrid is an intuitively appealing procedure, and

many of the proposed multi-level methods have been developed purely based on intuition. This approach, however, runs the risk of resulting in an algorithm which may violate certain basic multigrid principles, thus producing a less than optimal scheme.

The so-called additive correction schemes have been employed to construct efficient solution procedures for anisotropic convection problems [55]. The derivation of these schemes is particularly simple, and can be performed entirely in an algebraic setting. The idea is to group together neighboring subsets of fine-grid points, and to construct coarse level equations by directly summing these fine level equations. The fine grid variables are then rewritten as their current value plus a coarse grid correction, and the resulting equations are solved. As an example, consider the discretization of a Poisson equation on the one-dimensional fine grid of Figure 23, with a coarse grid defined by pairwise addition of neighboring cells. The fine grid equations at cells i and $i+1$ are given by

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i \tag{54}$$

$$\frac{u_{i+2} - 2u_{i+1} + u_i}{h^2} = f_{i+1} \tag{55}$$

respectively. By summing these two equations, we obtain the coarse grid equation for cell B:

$$\frac{u_{i+2} - u_{i+1} - u_i + u_{i-1}}{h^2} = f_i + f_{i+1} \tag{56}$$

We seek corrections to the fine grid variables of the form

$$\begin{aligned} u_{i+2} &= u_{i+2}^{old} + c_C \\ u_{i+1} &= u_{i+1}^{old} + c_B \\ u_i &= u_i^{old} + c_B \\ u_{i-1} &= u_{i-1}^{old} + c_A \end{aligned} \tag{57}$$

where c_A, c_B, c_C represent coarse grid corrections at cells A, B, and C respectively. Substituting equations (57) into equation (56) we obtain:

$$\begin{aligned} \frac{c_A - 2c_B + c_C}{h^2} = & \quad f_i - \frac{u_{i+1}^{old} - 2u_i^{old} + u_{i-1}^{old}}{h^2} \\ & + f_{i+1} - \frac{u_{i+2}^{old} - 2u_{i+1}^{old} + u_i^{old}}{h^2} \end{aligned} \tag{58}$$

which is easily seen to be similar to the coarse grid equation defined by the multigrid correction scheme, since the right-hand side of equation (58) represents a restriction of fine grid residuals. This represents a simple intuitive and algebraic approach for deriving a multigrid correction scheme. However, the conventional multigrid correction scheme using rediscretization for the coarse grid operator yields

$$\frac{c_A - 2c_B + c_C}{4h^2} = \frac{r_i + r_{i+1}}{2} \tag{59}$$

where r_i is defined as the residual

$$r_i = f_i - \frac{u_{i+1}^{old} - 2u_i^{old} + u_{i-1}^{old}}{h^2} \tag{60}$$

Since these residuals are defined in a point-wise manner, the volume weighted residual restriction results in a simple averaging of the residuals r_i and r_{i+1}. The coarse grid equation generated by the additive correction scheme is thus inconsistent with equation (59). This is the same problem encountered with the agglomeration multigrid scheme, and is due to the implied use of injection for the prolongation operator in the additive correction scheme. While these schemes have been successful for certain classes of problems, they cannot be expected to perform optimally even for a simple Poisson equation.

4.2 Algebraically Smooth Prolongation Operators

One of the failings of the ACS schemes is their implied use of simple injection as a prolongation operator. In this section we briefly discuss a strategy for constructing algebraically smooth prolongation operators, and examine their effect on overall convergence. As stated in section 3.5, an algebraically smooth error may be defined as one which can no longer be reduced effectively on the fine grid using an explicit operator. This usually implies that the residual of the error is small in some sense compared to the error itself, since explicit smoothers involve the use of the residual to compute updates to the solution. Thus, the relation

$$Ae = r \sim 0 \tag{61}$$

provides a definition of a smooth error e, where A represents the operator on the current grid, and r the residual. Since the operator A usually involves a local stencil, equation (61) describes an error for which the value at the point in question and at neighboring points is distributed in such a manner as to approximately satisfy the discrete equation locally. For example, for a Galerkin discretization of a Laplacian, the above equation implies that the error has a linear distribution in the vicinity of the point in question, without specifying the slope of this linear distribution. Any highly oscillatory quantities in this region would lead to a large residual. In this case, the error is also geometrically smooth. In the general case, however, the use of equation (61) represents an algebraic extension of the concept of smoothness.

In the context of an algebraic multigrid method, or a node-nested multigrid strategy, where the coarse-grid points comprise a subset of the fine-grid points, or an agglomeration strategy, where the seed points of the coarse agglomerated cells represent a subset of the fine-grid vertices, a smooth prolongation operator may be defined as

$$\begin{aligned} v_i &= v_{ci} && for\, i \in coarse\, grid \\ v_i &= \frac{1}{-a_{ii}} \sum_{j \neq i}^{N} a_{ij} v_j && for\, i \ni coarse\, grid \end{aligned} \tag{62}$$

At fine-grid points which are common to the coarse grid (or seed points in the agglomeration algorithm), values are prolongated by injection, whereas at remaining fine-grid points, the prolongated values are obtained by setting the fine grid residual to zero. Here the coefficients a_{ij} represent the non-zero elements of the operator matrix A. This prolongation construction has been described in section 3.5. As previously mentioned, if all neighboring vertices of i are coarse-grid vertices, *i.e.*, all the v_j values, then equation (62) yields an explicit formula for smooth prolongation at point i. However, since this is seldom the case, additional approximations are required. By applying equation (62) recursively to the neighboring points of i which are not coarse-grid points, and truncating the resulting large stencil at some level, such that it involves only coarse-grid points, an approximate explicit formula for the prolongated correction at point i is obtained, as described in section 3.5. An alternate approach [48] is to consider the exact prolongation operator defined by

setting all residuals equal to zero at points such as i, in equation (62), which represent all fine-grid points not contained in the coarse grid, and to solve these equations approximately. This can be achieved by iterative means (*i.e.*, multiple Jacobi iterations). These iterations are similar to those of the base fine grid solver, since the same operator is involved. However, the appropriate boundary condition in this case is:

$$v_i = v_{ci} \qquad for\, i \in coarse\, grid \tag{63}$$

where v_{ci} represents the corrections at the coarse-grid points which are injected to the equivalent fine-grid points. The fact that the same iterative solver as the base fine grid solver can be employed makes this implicit prolongation operator simple to construct. The application of equation (63) as a boundary condition ensures rapid convergence of any simple iterative scheme, since in general each fine-grid point which is not a coarse-grid point will be surrounded by coarse-grid points, which are only one or two neighbor distances away. In particular, for the construction described in sections 3.3 and 3.4, where the coarse-grid points constitute a maximal independent set of the fine-grid points, each fine-grid point is either a coarse-grid point, or a neighbor of a coarse-grid point. For Laplace's equation, this implicit prolongation operator preserves a linear distribution exactly, and closely approximates the prolongation obtained by triangulating the coarse-grid points and using piecewise linear interpolation, as described previously for the overset-mesh multigrid approach. (Since different triangulations of the coarse-grid points lead to different piecewise linear interpolation operators, the two approaches will not be identical in general).

This approach is demonstrated for Laplace's equation using the same two grid system of section 3.7, *i.e.* the fine and first coarse-level agglomerated grid of Figure 16, in section 3.7. The agglomeration multigrid approach is employed in this case. Recalling the dual nature of the agglomeration procedure, the seed points of the coarse agglomerated grid are taken as the common coarse and fine-grid points. Three pre- and post-smoothing Jacobi cycles are executed on the fine grid, and the coarse grid is solved "exactly" (using 200 Jacobi iterations).

Coarse Grid	Coarse Grid Op.	Restriction	Prolongation	Convergence Rate
Triangulated Seed Pts	Rediscretization	Linear	Linear	0.125
Agglomerated	Scaled Galerkin	Injection	Injection	0.254
Agglomerated	Scaled Galerkin	Linear	Linear	0.159
Agglomerated	Scaled Galerkin	Injection	Linear	0.171
Agglomerated	Scaled Galerkin	Injection	Implicit (50 cycles)	0.177
Agglomerated	Scaled Galerkin	Injection	Implicit (5 cycles)	0.178
Agglomerated	Scaled Galerkin	Injection	Implicit (2 cycles)	0.195

Table 2: Effect inter-grid transfer operators for agglomeration multigrid.

In Table 2, the convergence rate of the overset-grid method using the triangulated seed points as a coarse grid and the standard agglomeration algorithm are compared. The former algorithm

uses linear interpolation for both restriction and prolongation with a geometric discretization of the coarse grid equations, while the latter employs injection for both inter-grid transfer operators (volume weighted averaging for the residual restriction is considered as an injection operation), and the Galerkin coarse-grid operator rescaled for consistency as given by equation (52). The slower convergence of the agglomeration method is accelerated when the same linear interpolation operators used in the overset-mesh method are explicitly used in the agglomeration method. In fact, even if linear interpolation is employed only in the prolongation operator, most of the benefit is achieved, as can be seen from the fourth entry in the table. In the remaining three entries in the table, the use of the implicit prolongation operator is clearly seen to produce nearly identical results to the same scheme using the linear interpolation prolongation operator. Furthermore, the overall multigrid convergence rate degrades only slightly when decreasing the number of iterations used to solve the implicit prolongation operator from 50 to 2, supporting the claim that these (implicit prolongation) equations converge rapidly.

The implicit prolongation operator construction can be particularly advantageous for complex non-linear problems. For example, in the event that a maximum principle or a positivity condition is built into the single-grid relaxation scheme, injection or even linear interpolation prolongations cannot guarantee the preservation of such properties throughout the multigrid process, particularly for non-linear equations. However, such properties are preserved in a natural manner by the implicit prolongation operator. This is easily seen since the seed points inherit the property from the coarse grid solution, while the other fine-grid points receive corrections generated using fine grid iterations. It is also interesting to note that for hyperbolic problems, many modifications to the simple linear interpolation prolongation operator have been suggested, in order to account for the hyperbolic nature of the problem [21, 37]. The present implicit formulation should presumably take such effects into account automatically.

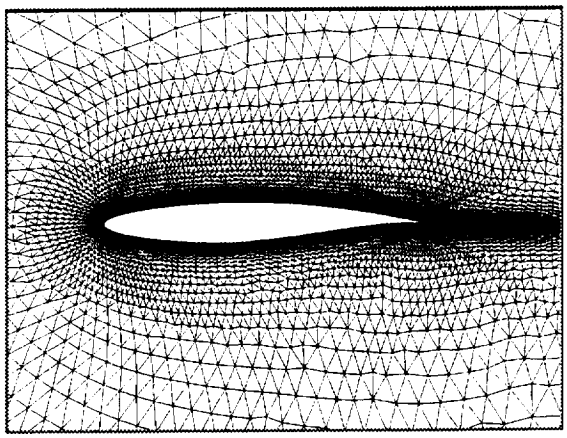

Figure 28: Fine unstructured mesh for computation of turbulent flow over RAE 2822 airfoil. (Number of vertices = 18840).

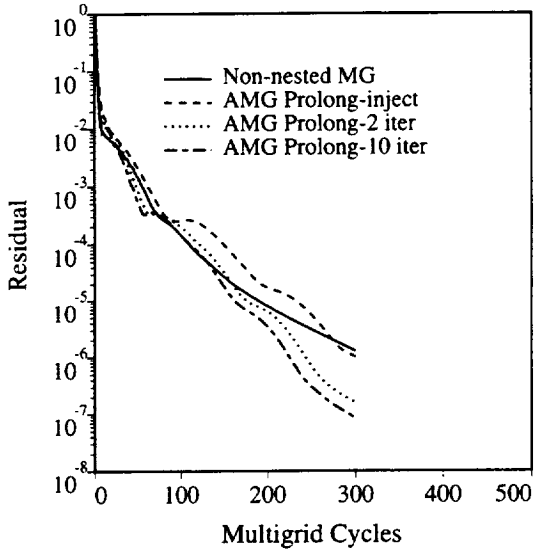

Figure 29: Multigrid Convergence rates obtained with injection and implicit prolongation operators for turbulent flow over RAE 2822 airfoil.

In Figure 29, the convergence rates of the overset-mesh multigrid approach and the agglomeration multigrid approach using the implicit prolongation operator are compared for the computation of viscous turbulent flow over an RAE 2822 airfoil on the grid of Figure 28. For this case, the

freestream Mach number is 0.73, the Reynolds number is 6.5 million, and the incidence is 2.31 degrees. Turbulence effects are accounted for by the single-equation model of Spalart-Allmaras [52], which is also solved using the multigrid procedure. As previously, the implicit prolongation operator is computed by injecting corrections to the fine-grid points which correspond to coarse agglomerated-grid seed-points, and performing multi-stage time-stepping on the remaining fine-grid points, while holding the seed-point values constant. The agglomeration algorithm using an injection prolongation operator is noticeably slower than the rate obtained with the implicit prolongation operator, which even outperforms the overset-mesh method using linear interpolation. In this case, where the equations are non-linear, the implicit prolongation operator presumably provides smoother fine grid corrections than a linear-interpolation-based prolongation operator.

When many iterations are required to solve the implicit prolongation equations, the procedure is obviously not practical, since these iterations are very expensive, *i.e.*, they correspond to fine grid iterations. Unfortunately, even though most of the convergence acceleration benefit in Figure 29 can be obtained with only 2 sub-iterations for the implicit prolongation equations, the procedure is still not worth the gain in convergence speed it affords for this case. Thus, the usefulness of the implicit prolongation operator depends on the ability to inexpensively approximate or solve the equations associated with this operator. This represents a topic of current research. The above examples, however, serve to demonstrate the potential of an algebraically-based smooth prolongation operator.

4.3 Better Coarse Grid Operators

The construction of the scaled Galerkin coarse-grid operator for diffusion problems defined in equation (52) is somewhat heuristic in nature, and is apparently not as effective as the operator obtained through rediscretization, even when identical inter-grid transfer operators are employed, as can be inferred from Table 2. The possibility of constructing more efficient coarse grid operators must therefore be considered. One possibility would be to investigate the use of a Galerkin coarse grid operator $I_h^H \, A \, I_H^h$ where I_H^h is the implicit prolongation operator described above. However, due to the implicit form of I_H^h, this construction is not straight-forward. For linear problems, an explicit form of the prolongation operator could be constructed in a preprocessing phase by (approximately) inverting the matrix which corresponds to the system of equations (62).

In order to investigate the effect of more accurate prolongation operators in the Galerkin coarse grid operator construction, the solution of Laplace's equation on the two-grid system composed of the finest and first level agglomerated grid of Figure 16 is chosen as a test problem. The convergence of the rediscretized and scaled coarse grid operator constructed using injection, and the coarse grid operator constructed using piecewise linear interpolation are compared in Table 3.
The piecewise linear interpolation operator itself is that obtained by triangulating the coarse grid seed points, and is precisely the operator used for prolongation by the overset-mesh multigrid procedure operating on an equivalent coarse grid of triangulated seed points. While this construction for the prolongation operator may not be possible in the general case, it is readily available in this case (explicitly constructed in the overset-mesh code), and affords a direct comparison with the overset-mesh multigrid method, where the coarse grid operator is constructed by rediscretization.

Clearly, the Galerkin coarse grid operator based on linear interpolation is much more effective than even the geometric coarse grid operator used in the overset-mesh approach. This demonstrates that Galerkin coarse grid operators can be even more effective than rediscretization techniques. However, even though this scheme contains the same number of coarse-grid points (or coarse level variables), the matrix of the coarse grid operator is no longer based on the implied agglomeration graph, and is much denser. This is the result of the operator no longer relying exclusively on nearest neighbor stencils. Coarse grid evaluations are over three times more expensive than in the other cases. This method is therefore not practical, particularly for multi-level applications.

Coarse Grid	Coarse Grid Op.	Restriction	Prolongation	Convergence
Independent	Rediscretization	Linear	Linear	0.100
Triangulated Seed Pts	Rediscretization	Linear	Linear	0.125
Agglomerated	Scaled Galerkin	Linear	Linear	0.159
Agglomerated	Full Galerkin (Using Linear Int. for I_h^H and I_H^h)	Linear	Linear	0.060

Table 3: Effect of coarse grid operator in agglomeration multigrid.

There are various possibilities for alleviating this difficulty, although some degree of compromise is usually required. One approach is to choose the coarse-grid points in the agglomeration or point-removal process such that the resulting stencil size of the Galerkin coarse grid operator is minimized. Simultaneously, the explicit stencils of the restriction and prolongation operators may be truncated, preserving only the strongest coefficients in the stencil. The combination of these two effects has been shown to produce coarse grid stencils of near constant complexity when applied recursively on sequences of coarse meshes [25]. Another approach is to attempt to approximate the coarse grid stencil, or to decompose it into the sum of several stencils, and to delete or approximate one or various of these individual stencils [60]. Finally, the additional cost of evaluating more complex coarse grid operators may be offset partially by reducing the complexity of the coarse grids (*i.e.*, the number of coarse-grid points), although this approach runs the risk of under-representing particular error frequencies. Clearly, additional research is required before more sophisticated coarse grid operators can be utilized effectively.

4.4 Optimal Coarsening Strategies for Anisotropic Problems

While the use of unstructured meshes was initially viewed as a complicating factor for the implementation of multigrid methods, the added flexibility of unstructured meshes turns out to be a great advantage for devising more efficient multigrid strategies. For structured meshes, the usual coarsening strategy consists of deleting every second point in each coordinate direction. For strongly anisotropic problems, involving highly-stretched meshes, local smoothers are only effective at smoothing the error in the direction of strongest coupling (small mesh spacing), leaving the high-frequency error components in the direction of high stretching relatively unsmoothed. A common technique to remedy this situation is to resort to semi-coarsened meshes, where the next coarser mesh is constructed by removing every second point in one coordinate direction only, *i.e.*, the direction of strong coupling. The difficulty of this approach is that different regions of a structured mesh may exhibit stretchings in different coordinate directions. Thus, a uniform semi-coarsening in a single direction is not sufficient, and generally, two semi-coarsened meshes are required, one for each coordinate direction in two dimensions. For recursive coarsening, this results in a diamond shaped pattern of coarse meshes, as shown in Figure 30. In three dimensions, the situation becomes prohibitively complicated and expensive.

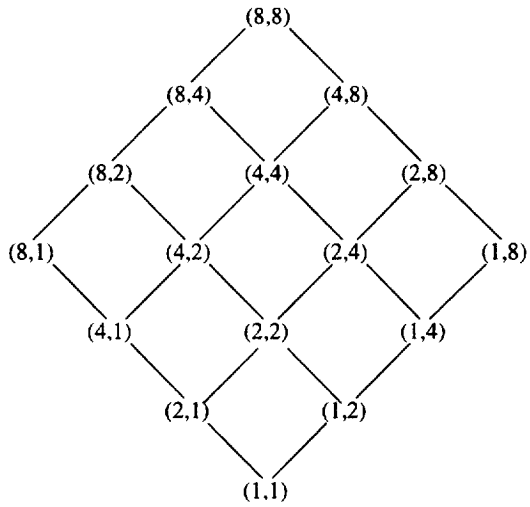

Figure 30: Sequence of coarse meshes required for two-dimensional semi-coarsening of an 8×8 fine structured mesh.

Unstructured multigrid methods, on the other hand, offer the possibility of generating optimally coarsened meshes, where each coarse mesh is tailored to represent precisely the error frequencies which could not be smoothed efficiently on the previous finer mesh. This can be accomplished either algebraically, by agglomeration, or by point removal techniques using the overset-mesh method. In fact, the coarsening strategy and the multigrid approach can be treated independently from one another. Such optimal coarsening strategies offer the possibility of resolving stiffness due to anisotropy and disparate length scales in an efficient manner. Although these techniques are only beginning to be appreciated for unstructured mesh multigrid methods, they have been exploited in certain implementations of the Additive Correction Scheme [55, 57], and in algebraic multigrid methods [25].

One of the central ideas in optimal coarsening strategies is to coarsen in the direction of strong coupling, *i.e.*, to remove or agglomerate fine-grid points which are strongly coupled to a current coarse-grid point. In the geometric context, strong coupling occurs for closely spaced points, whereas in the algebraic context strong coupling refers to a non-zero entry in the sparse matrix which has a large magnitude, *i.e.*, $a_{ij} \gg 1$ in equation (34). Since these matrix entries correspond to edges in the graph of the matrix, the weight of the edge determines the amount of coupling between the two end-points associated with the edge. Therefore, in order to achieve more optimal coarsenings, the simple graph-based techniques discussed in sections 3.3 and 3.4, such as those concerned with maximal independent sets, must be replaced with weighted-graph techniques, where weights are associated with each edge of the graph, which are then taken into consideration by the algorithm in deciding how to coarsen.

The operator stencil coefficients usually represent a good choice for defining the graph weights. This assumes that the operator may be expressed in an edge-based format with a single coefficient for each edge. This may not always be possible. For example, non-symmetric operators result in the association of two weights per edge, one for each direction of the edge, while the edge coefficients generated by non-linear operators are non-constant and may vary at each time-step. While these cases may be handled via certain simplifying assumptions, the case of a system of equations, such as the Euler or Navier Stokes equations is even more difficult, since the edge-weight becomes a matrix.

For these reasons, we will concern ourselves uniquely with the discrete Laplacian in the remainder of this section. For this operator, a unique stencil coefficient may be associated with each edge of the graph. By defining the edge weights in this manner, the coarsening algorithm produces the

same behavior for a uniform Laplacian of the form

$$u_{xx} + u_{yy} = 0 \tag{64}$$

discretized on a mesh with a stretching ratio of ϵ, as for the anisotropic Laplacian of the form

$$\epsilon u_{xx} + u_{yy} = 0 \tag{65}$$

discretized on a uniform mesh.

The coarsening algorithm should be able to handle arbitrarily anisotropic meshes, but should also be able to reproduce structured-mesh semi-coarsening patterns for uniformly anisotropic problems on structured meshes. A simple-minded technique for graph-weighted coarsening would be to modify step 4 in the algorithm described in section 3.4, by only agglomerating or removing the neighboring point which has the strongest connection to the current seed point or coarse-grid point. In the event this point has already been removed by some other neighboring coarse-grid point, the next strongest connecting point may be chosen. The difficulty with this approach is that in regions where the anisotropic character is weak, the algorithm does not revert to the multi-directional coarsening strategy of the original algorithm. A more sophisticated approach is to precompute the average connection strength at each vertex, and then to coarsen the neighbor j of a coarse grid vertex i only if

$$|a_{ij}| > \beta \frac{1}{N_{neigh}} \sum_{k \neq i}^{N_{neigh}} |a_{ik}| \tag{66}$$

where N_{neigh} represents the number of neighbors of i, or number of non-zero entries in the i^{th} row of matrix A. β is a parameter which determines the degree of anisotropic coarsening. For $\beta = 0$ unweighted coarsening is recovered, while when $\beta = 1$ coarsening occurs only along connections stronger than the average. In [25] a value of $\beta = 0.5$ is suggested, which reproduces directional coarsening in regions of strong anisotropies, and full or unweighted coarsening elsewhere.

Mesh	Method of Coarsening	MG Rate
Cartesian	Structured, Full-Coarsening	0.15
Tchebychev	Structured, Full-Coarsening	0.52
Tchebychev	Structured, Semi-Coarsening	0.50
Tchebychev	Unstructured, Directional	0.14

Table 4: Convergence rate of directional coarsening multigrid on Tchebychev mesh of various stretchings.

In Table 4 and Figure 31, the solution of a Laplacian on a Tchebychev mesh is illustrated, using a similar coarsening strategy. The triangular Tchebychev mesh is initially structured, but contains

conflicting stretching directions for simple structured semi-coarsening methods. As shown in the table, when structured full-coarsening or semi-coarsening in one direction is employed, a dramatic reduction in multigrid efficiency results. On the other hand, when the directional unstructured coarsening strategy is employed, using the meshes depicted in Figure 31, the optimal convergence rate achieved on an equally spaced cartesian mesh is recovered. In all cases, a multigrid W-cycle is employed, using two Jacobi iterations on the refinement as well as coarsening phases of the cycle.

In this case, the coarsening strategy consists of removing the neighboring vertex with the strongest available connection as the front is advanced, and retriangulating the coarse-grid point-set using a min-max triangulation [61].

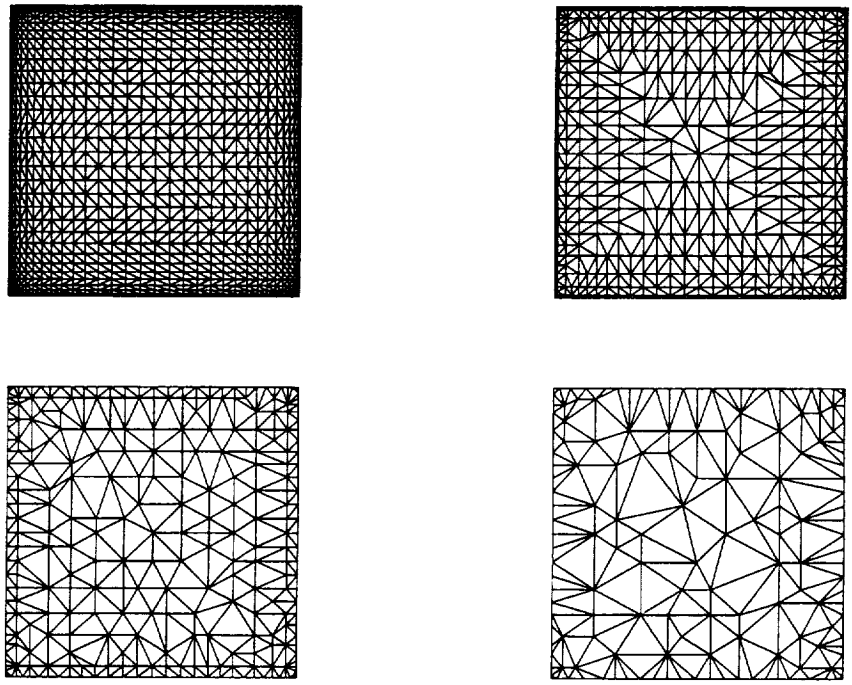

Figure 31: Triangulated Tchebychev mesh and three coarse level meshes obtained by directional coarsening.

The three-dimensional agglomerated grids displayed in Figure 24, of section 3.7, were actually generated using the coarsening criterion of equation (66), with the value $\beta = 0.0001$. In this case, since the Navier-Stokes equations represent a system of equations, the edge coefficient was taken as the inverse of the edge length, which represents a geometric definition. The low value of β produces directional coarsening only in the highly stretched regions of the mesh, near the airfoil surface, in the initial phases of the coarsening process. Figure 32 depicts the coarse agglomerated meshes which result when a value of $\beta = 0.1$ is employed. As expected, semi-coarsening is much more prevalent. For example, the span-wise resolution of the mesh is essentially constant throughout all levels. However, the quality of the coarser meshes appears to degrade. The problem stems from the complexity of the coarse grids. In highly anisotropic regions, a 2:1 coarsening is generally produced by the weighted-graph techniques, while in the isotropic regions, an 8:1 coarsening results. When the process is applied recursively, the isotropic regions of the mesh are coarsened much faster than the non-isotropic regions, thus resulting in large disparities in neighboring cell sizes, as can be seen in the coarse grids of Figure 32. This, in turn, has a negative impact on the effectiveness of

the coarse grid operator. A secondary effect is to increase the complexity of the coarse grids (as compared to an unweighted graph algorithm), thus increasing the cost of a multigrid cycle, and obviating the possibility of using W-cycles.

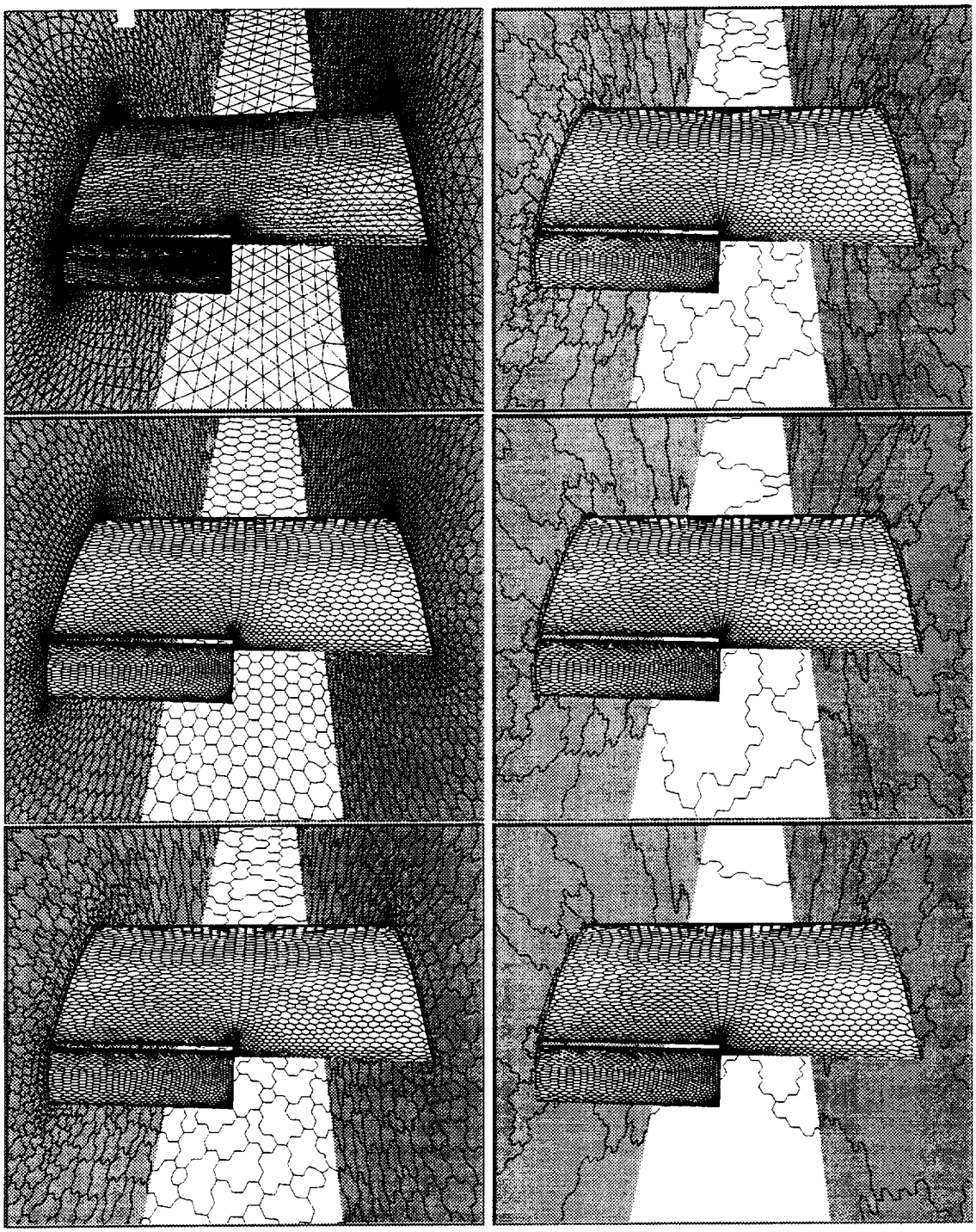

Figure 32: Coarse agglomerated grids obtained using directional coarsening strategy.

These are some of the reasons why directional coarsening techniques have not been exploited more fully to date. Similar issues arise in the context of coarsening strategies for adaptive meshing problems, where highly disparate length scales are encountered. They are dealt with in more detail in the following section.

5 Multigrid Techniques for Adaptive Meshing Problems

In addition to their flexibility for discretizing complex geometries, another great advantage of unstructured meshes is the ease with which adaptive meshing techniques may be incorporated. Although most work on adaptive meshing methods has concentrated on the logistics of refining the mesh and the formulation of suitable refinement criteria, efficient solution techniques for the resulting discrete equations are also required in order to enable both fast and accurate solutions. Adaptive meshing, in particular, provides a natural setting for the use of multigrid solvers. The various refined meshes generated from the adaptive process can be used to form the set of coarse and fine meshes of the multigrid sequence. The multigrid algorithm can then be used to accelerate the convergence to steady-state of the discrete equations on the finest adaptive mesh. In fact, the synergy between the two techniques is greater than may be initially apparent, and has roots in the ideas of multi-resolution (see Figure 33).

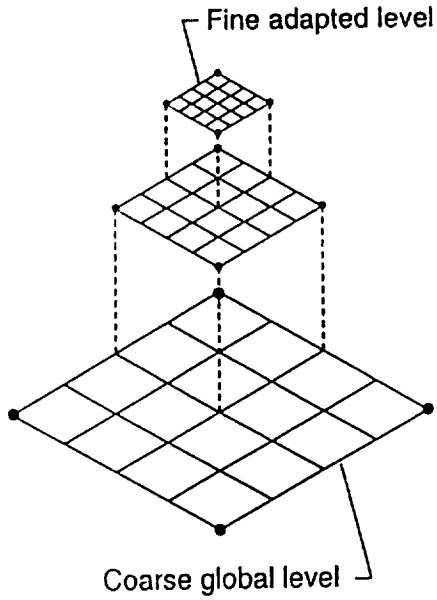

Figure 33: Illustration of the ideal multi-resolution principle of adaptive meshing combined with multigrid where each mesh level of the multigrid sequence represents a unique resolution scale.

The role of the adaptation process is to identify regions of the domain where the resolution of smaller scales is required and to generate these required new mesh levels, while the role of the multigrid solver is to eliminate the various high- and low-frequency errors of the solution on the grid level which best represents them. This has led to the development of methods such as the FAC (Full Adaptive Composite) method [62], and to the notion of the de-algebraization of multigrid, as described by Brandt [26], where the multigrid procedure is no longer viewed simply as a fast solver for discrete equation sets, but rather as part of a complete strategy for approximating the solution to the continuous partial differential equation. Spatial convergence is achieved by the adaptation process, while temporal or numerical convergence is achieved by the multigrid procedure.

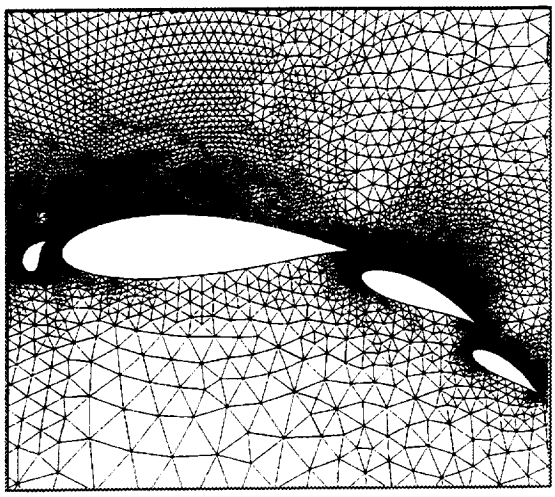

Figure 34: Final adapted mesh employed for computation of inviscid flow over four-element airfoil. (Mach=0.2, Incidence = 0 degrees, Number of points = 22,792).

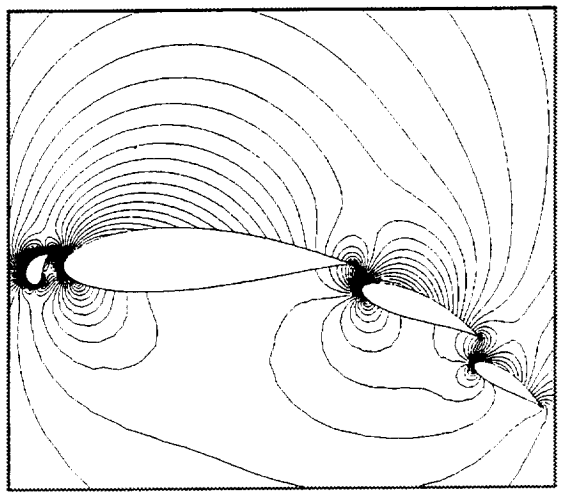

Figure 35: Computed Mach contours of flow over four-element airfoil on adapted mesh. (Mach=0.2, Incidence = 0 degrees).

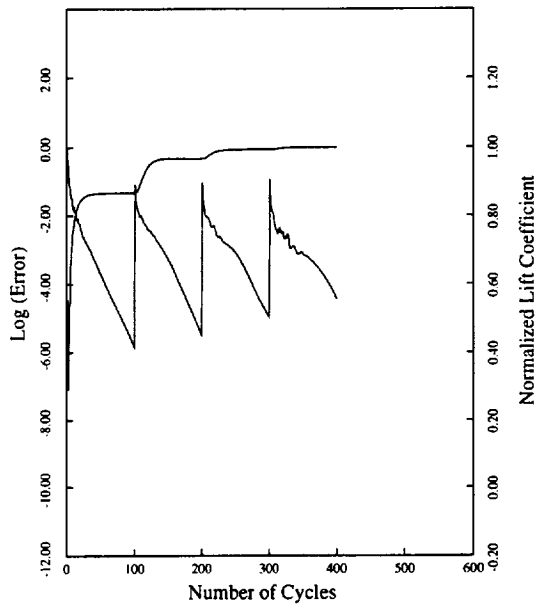

Figure 36: Multigrid and adaptive mesh convergence history for flow over four-element airfoil.

As an example, consider the solution process depicted in Figures 34 through 36. All of the techniques employed in this solution strategy have been described in the preceding sections. The overset-mesh multigrid approach is used to solve the steady-state Euler equations about an idealized four-element airfoil geometry on the grid of Figure 34. A total of seven meshes are employed in the multigrid sequence, of which the four coarsest meshes were generated using a global Delaunay triangulation procedure, and the remaining three meshes were generated adaptively, throughout the solution process, using local point-insertion followed by Delaunay reconnection. The initial

fine mesh contains a total of 6,466 points, while the final adapted mesh depicted in Figure 34 contains 22,792 vertices. The full multigrid strategy described in section 2.5 (*c.f.* Figure 4) is employed. The process begins by multigrid cycling on the four coarsest meshes of the sequence. Once a satisfactory solution on the initial fine mesh is obtained, a new adaptively generated mesh is constructed, the patterns for interpolation between this new grid and the previous grid are computed, the solution is interpolated to the new grid, and the multigrid process is resumed on the new sequence of meshes which contains the new adaptively generated grid as the finest mesh. Figure 36 depicts the convergence rate for this case, where 100 multigrid W-cycles are performed at each level of adaptivity. The slopes of the various multigrid convergence histories are nearly identical on the four different mesh levels, demonstrating the mesh independent convergence property of the multigrid algorithm. Convergence on the final mesh is only slightly slower than that on the initial levels, resulting in an average reduction rate of 0.925. The convergence history of the computed lift coefficient is also plotted. The effect of spatial grid convergence can be seen by the diminishing differences between the final lift values on consecutively finer meshes. This figure provides an illustration of the concept of using adaptive-multigrid as a method for solving the continuous set of partial differential equations, with the lift coefficient converging to the infinite resolution value, and the multigrid procedure driving the numerical solution on each level. The solution is depicted in Figure 35, as a set of computed Mach contours. The freestream Mach number is 0.2, and the incidence is 0 degrees for this case. This entire run, including all mesh adaptivity, was achieved in approximately 2 hours on an SGI Indigo R4000 workstation.

Although the previous example demonstrates the effectiveness of multigrid as an efficient solution strategy for adaptive meshing problems, certain characteristics of adaptive problems can degrade the overall efficiency of the above multigrid approach. These manifest themselves, not as degradations of the observed convergence rates, but rather as unwanted increases in complexity (number of operations) of the multigrid cycle. For example, in the non-adaptive two dimensional case, the complexity of a V-cycle is bounded by 4/3 work units, and that of a W-cycle by 2 work units, where a work unit is defined as the equivalent work of one fine grid iteration (see Figures 2 and 3 for the definition of these cycles). The above bounds are computed assuming each coarser mesh level contains 1/4 the number of points of the previous level. In the case of adaptively generated meshes, where such relations between the complexities of the various mesh levels no longer hold, the V-cycle complexity becomes equal to the sum of the complexities of all meshes in the sequence, while the W-cycle complexity can become so high as to make it impractical.

Even the V-cycle complexity is much higher than it need be. For adaptively refined meshes, refinement only occurs in localized regions of the mesh, and there are large regions of the domain where the mesh resolution is essentially unaltered between mesh levels. Repeatedly time-stepping in these regions of the mesh on various levels represents a waste of computational effort. In this section, two strategies which overcome this increase in complexity for V-cycles are described. A third approach, which results in optimum complexity, thus enabling the use of V or W cycles, is finally discussed.

5.1 The Zonal Fine Grid Scheme

The basic idea behind this scheme [63, 64] is to omit time-stepping in regions of the mesh which have not been refined with regards to the previous level. A crude implementation consists of making use of the same multigrid strategy as described previously, but blanking out the appropriate vertices on each mesh level. In actual fact, the fine mesh consists only of the regions which have been refined, with possibly some extra buffer layers. The method can be implemented by storing only these regions at each level in order to save memory (although this has not been done in the following example).

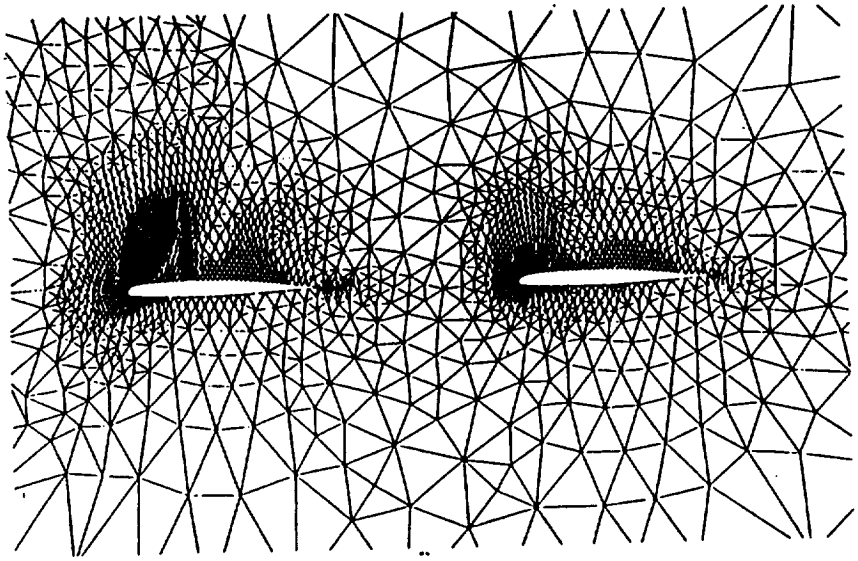

Figure 37: Adapted mesh employed to compute inviscid flow over tandem airfoil configuration.

Figure 38: Third and fourth adaptive levels in the zonal fine-grid scheme for computation of flow over tandem airfoil configuration.

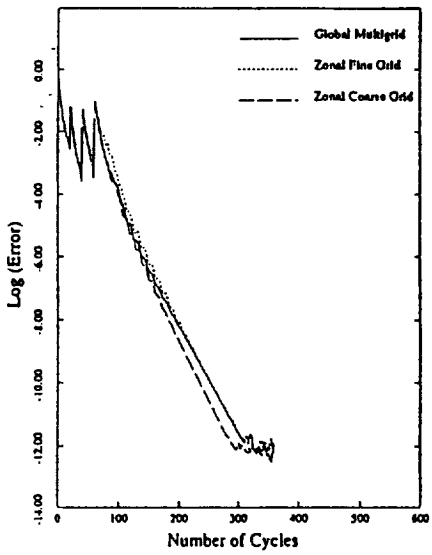

Figure 39: Convergence rates of various multigrid schemes in terms of cycles for flow over tandem airfoil configuration.

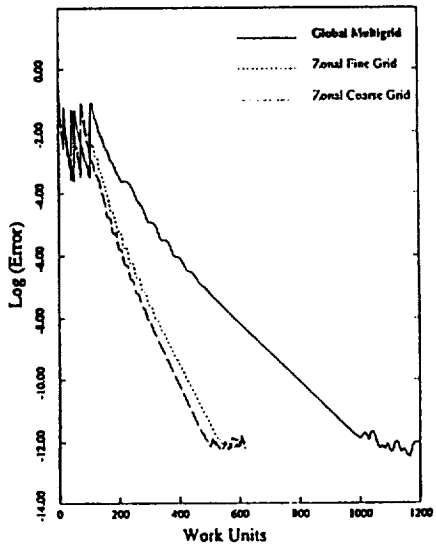

Figure 40: Convergence rates of various multigrid schemes in terms of work units for flow over tandem airfoil configuration.

As an example, consider the adaptive mesh used to compute the inviscid flow over a tandem airfoil configuration, shown in Figure 37. This mesh is the result of 6 levels of adaptivity. For the zonal fine grid scheme, the 3rd and 4th adaptive levels are depicted in Figure 38. Figure 39 compares the convergence rates of the zonal-fine grid scheme with that of the global multigrid scheme described previously for this case. There are in fact 8 mesh levels in both multigrid cases, 2 initial global levels, and 6 adaptively generated levels. (The global levels are identical for both schemes). The freestream Mach number is 0.7, and the incidence is 3 degrees. Both multigrid schemes converge at nearly identical rates, in terms of residual reduction per cycle. This result verifies the fact that multigrid time-stepping in regions where no change in resolution occurs is unnecessary. The advantage of the zonal fine grid scheme is the result of the reduction in complexity of the multigrid cycle, as shown in Figure 40, where the convergence rates are compared in terms of work units. For this case, the zonal fine grid scheme is seen to be roughly twice as efficient as the global multigrid approach.

This so-called zonal fine grid scheme is the unstructured-mesh equivalent of the fast-adaptive-composite scheme (FAC) [62], and as such embodies the principles of multi-resolution as depicted in Figure 33. Each mesh level is responsible for resolving a particular range of scales, and highly disparate length scales are not found on any common mesh, as would be the case in a global mesh with localized regions of adaptive refinement.

One of the drawbacks of this method is that the final solution lies on a composite mesh which is spread over various multigrid levels. Aside from practical difficulties involved in postprocessing the solution, this complicates other issues, such as the requirement of constructing a conservative discretization in the final solution, as well as the use of different schemes on fine and coarse mesh levels.

5.2 The Zonal Coarse Grid Scheme

The idea of the zonal coarse grid scheme is to overcome the difficulties encountered in the zonal fine grid scheme due to the composite nature of the final solution, by maintaining a global fine grid upon which the final solution is based. In order to maintain favorable complexity, time-stepping is omitted on the coarser meshes in regions of the domain where no mesh refinement takes place between two consecutive levels. This strategy is illustrated in Figure 41,

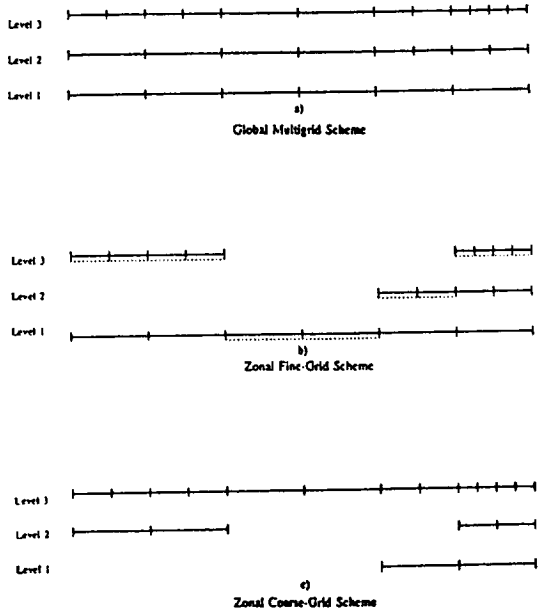

Figure 41: Illustration of the relationship between the zonal fine-grid scheme, the zonal coarse-grid scheme, and the global multigrid scheme using linear one-dimensional meshes.

using one-dimensional linear meshes, and compared to the zonal fine-grid and global multigrid strategies. The overall complexity of the zonal fine-grid and coarse-grid schemes are necessarily identical. As can be inferred from the figure, the zonal fine grid and coarse grid schemes are equivalent, except that in the former case the non refined mesh regions are represented on the coarse level meshes, whereas in the latter, these are assigned to the finest possible mesh level. Hence, the zonal coarse grid scheme simply corresponds to a reordering of the local unrefined and refined mesh regions.

The convergence rate of the zonal coarse grid scheme is compared with that of the zonal fine grid scheme and the global multigrid scheme for the transonic tandem-airfoil case on the mesh of Figure 37. As expected, all three methods yield similar convergence rates on a per cycle basis, while the zonal fine and coarse grid schemes achieve a factor two gain in efficiency over the global multigrid scheme in this case, due to the reduction in complexity, as shown in Figure 40. Thus, the zonal coarse grid scheme is equivalent to the zonal fine grid scheme in terms of efficiency, but enables the final solution to be computed on a global fine grid. The disadvantage of this approach is that each time a new adaptively refined mesh is generated, the zonal coarse meshes must be reassigned to the appropriate levels.

5.3 Aggressive Coarsening Strategies

While the zonal fine and coarse grid schemes achieve substantial reduction in the complexity of a multigrid cycle for adaptively generated meshes, the use of a W-cycle with such schemes is still impractical, due to the relative complexities of the various mesh levels. Since the W-cycle performs frequent visits to the coarse level meshes within a single cycle, the mesh complexity must be reduced

by at least a factor of four when going to the next coarser level, in order to guarantee a bound on the overall W-cycle complexity, as the number of mesh levels increases. Another characteristic of the zonal multigrid schemes described above is that they rely on the adaptive refinement history in order to identify the coarse and fine mesh levels. Such methods cannot be used effectively in cases where this information is not available, or in the case of a mesh of arbitrary construction, which may contain large variations in cell sizes.

The point removal, agglomeration, or algebraic coarsening strategies discussed in the previous section can be employed to overcome these difficulties. These represent automated coarsening strategies, which are used to coarsen the given fine grid, without consideration of the adaptive construction history of the grid. The use of such techniques for adaptively generated meshes represents a philosophy in which multigrid is decoupled form the adaptive process, and employed simply as a fast solver for a discrete fine-grid problem, much in the same manner as an implicit or direct solver would be employed.

Aggressive coarsening relates to the attempt in an automated coarsening process to optimize the complexity of the generated coarse mesh levels. For a multigrid smoother which is designed to damp high-frequency errors (as is usually the case), the optimal reduction in coarse grid complexity between two successive levels is 4:1 in two dimensions, and 8:1 in three dimensions. For isotropic problems, the unweighted graph techniques described in the previous section, (*i.e.*, which rely on maximal independent sets) generally achieve near optimal coarse grid complexities, thus enabling the use of V or W-cycles. However, although the complexity of the multigrid cycle may be optimal, the overall solution efficiency can only be competitive provided the multigrid convergence rate does not degrade substantially.

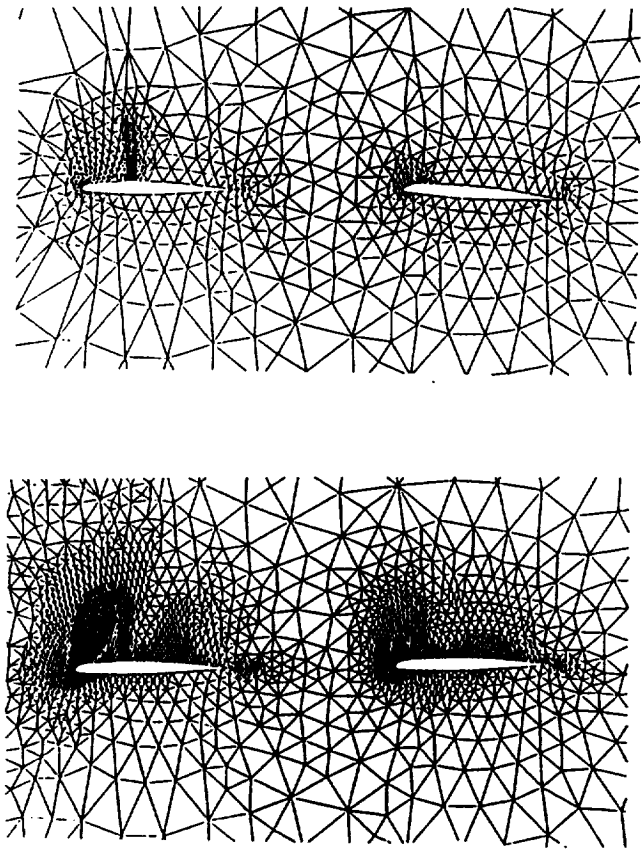

Figure 42: Resulting coarse mesh using two passes of aggressive coarsening and equivalent mesh level used in global multigrid sequence.

Figure 42 provides a comparison between the coarse mesh level obtained by two passes of aggressive coarsening on the fine mesh of Figure 37, and the equivalent mesh from the global multigrid sequence (6th level out of 8). Because each cell of the original grid is forced to "grow" at the same rate, the large outer boundary cells are seen to grow much more rapidly throughout the coarsening process than the small refined cells in the shock region of the fine mesh. This results in large discontinuities in cell size which become even more pronounced on the coarser levels. This, in turn, may degrade the observed convergence rate of a multigrid scheme based on these mesh levels. This is the same problem encountered with directional coarsening strategies for anisotropic problems (*c.f.* Figure 32 of section 4.4). These methods are evidently in complete violation of the multi-resolution principle associated with (adaptive) multigrid methods, where each mesh level is responsible for a given range of scales. Not only does each mesh level contain a wide range of scales in these approaches, but the bandwidth of this range increases on the coarser mesh levels.

Nevertheless, for many adaptive problems, aggressive coarsening strategies are highly desirable, both due to their fully automatic nature, and their low complexity. Such methods could obviously be improved by trading off complexity for more regularity in the coarse mesh levels, and thus better multigrid efficiency. However, this task generally requires global information about the current fine mesh construction (*i.e.*, in the adaptive mesh case, the history of refinement). This has important implications for the future design of optimal automated coarsening techniques, both for adaptive problems, and for anisotropic problems, since at present, most of these methods rely exclusively on local information, such as that given by equation (66) for constructing coarser levels.

6 Conclusion

As was pointed out eleven years ago, in a similar VKI lecture series [26], multigrid methods are very powerful techniques which in principle can be utilized to solve almost any type of numerical problem in a near optimal fashion. At that time, unstructured mesh techniques for computational fluid dynamics were in their infancy, and multigrid methods had not yet been considered for such problems. Today, unstructured multigrid methods are among the most efficient techniques for solving large unstructured mesh problems, and have seen widespread application. While incurring additional complications, the application of multigrid methods to unstructured meshes opens up new possibilities for developing more efficient algorithms. One of the themes of the above presentation has been to stress the abstraction or algebraization of the process, while pointing out the similarities with more traditional approaches. This emphasis stems from the belief that convergence acceleration techniques should not be bound by geometry-based constraints, and the removal of the influence of the geometry from the multigrid process should lead to a more robust strategy. Of course, completely algebraic multigrid methods for complex non-linear problems are not available, and much research is still required in this area.

Other treatises on multigrid methods have stressed the de-algebraization of multigrid, invoking multi-resolution principles, and advocating the use of multigrid methods as part of a comprehensive grid-based strategy for approximating solutions to the continuous partial differential equations. These two views of multigrid methods need not necessarily be in conflict with one another. The determination of the relevant length scales of a discrete problem and their assignment to appropriate levels, as well as the integration of multigrid principles with adaptive meshing techniques can be pursued either as a grid-based approach, or in a more abstract or algebraic setting. The extraction of the relevant information from the problem is the main issue.

Although current multigrid techniques are relatively efficient, there are many unresolved issues which require further investigation. These include the construction of more efficient coarse grid operators, the use of more accurate inter-grid transfer operators, and the development of optimal coarsening strategies for anisotropic as well as adaptive meshing problems. Finally, at a more fundamental level, additional insight is required to resolve the discrepancies between existing multigrid

efficiencies for elliptic and hyperbolic problems [65].

References

[1] V. Venkatakrishnan and T. J. Barth. Application of direct solvers to unstructured meshes for the Euler and Navier-Stokes equations using upwind schemes. AIAA Paper 89-0364, January 1989.

[2] V. Venkatakrishnan and D. J. Mavriplis. Implicit solvers for unstructured meshes. *Journal of Computational Physics*, 105(1):83–91, June 1993.

[3] J. T. Batina. Implicit upwind solution algorithms for three-dimensional unstructured meshes. *AIAA J.*, 31(5):801–805, May 1993.

[4] D. C. Slack, D. L. Whitaker, and R. W. Walters. Time integration algorithms for the two-dimensional Euler equations on unstructured meshes. *AIAA J.*, 32(6):1158–1166, 1994.

[5] W. K. Anderson. A grid generation and flow solution method for the Euler equations on unstructured grids. *Journal of Computational Physics*, 110(1):23–38, January 1994.

[6] L. Fezoui and B. Stoufflet. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J. Comp. Phys.*, 84:174–206, 1989.

[7] A. Jameson, T. J. Baker, and N. P. Weatherill. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86-0103, January 1986.

[8] D. J. Mavriplis. Three-dimensional multigrid for the Euler equations. *AIAA Journal*, 30(7):1753–1761, July 1992.

[9] J. Peraire, J. Peirö, and K. Morgan. A 3D finite-element multigrid solver for the Euler equations. AIAA Paper 92-0449, January 1992.

[10] H. Luo, J. D. Baum, R. Löhner, and J. Cabello. Adaptive edge-based finite element schemes for the Euler and Navier-Stokes equations on unstructured grids. AIAA Paper 93-0336, January 1993.

[11] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[12] Z. Johan, T. J. R. Hughes, K. K. Mathur, and S. L. Johnsson. A data parallel finite element method for computational fluid dynamics on the Connection Machine System. *Comput. Methods Appl. Mech. Engrg.*, 99:113–124, 1992.

[13] X. C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. Newton-Krylov-Schwarz methods in CFD. In *Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations*, volume 47 of *Notes in Numerical Fluid Mechanics*, Braunschweig/Wiesbaden, 1994. Vieweg Verlag.

[14] A. Jameson. Time-dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. AIAA Paper 91-1596, July 1991.

[15] J. C. Vassberg. A fast, implicit unstructured-mesh Euler method. AIAA Paper 92-2693, 1992.

[16] N. D. Melson, M. D. Sanetrik, and H. L. Atkins. Time-accurate Navier-Stokes calculations with multigrid acceleration. In *6th Copper Mountain Conf. on Multigrid Methods*, pages 423–439, 1993. NASA Conference Publication 3224.

[17] V. Venkatakrishnan and D. J. Mavriplis. Implicit method for the computation of unsteady flows on unstructured grids. Paper submitted to the 12th AIAA CFD Conf., San Diego, CA., June 1995.

[18] P. Wesseling. *An Introduction to Multigrid Methods.* John Wiley and Sons, 1991.

[19] W. K. Anderson J. L. Thomas and D. L. Whitfield. Multigrid acceleration of the flux-split Euler equations. *AIAA Journal*, 26(6):649–654, 1988.

[20] W. A. Mulder. A new multigrid approach to convection problems. *Journal of Computational Physics*, 83(2):303–323, August 1989.

[21] B. Koren and P. W. Hemker. Damped direction dependent multigrid for hypersonic flow computations. *Applied Numerical Mathematics*, 7:309–328, 1991.

[22] R. Radespiel and R. C. Swanson. Progress with multigrid schemes for hypersonic flow problems. ICASE Report No. 91-89, NASA CR-189579, December 1991.

[23] W. K. Anderson and D. L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers Fluids*, 23(1):1–21, 1994.

[24] M. Lallemand, H. Steve, and A. Dervieux. Unstructured multigridding by volume agglomeration: Current status. *Computers and Fluids*, 21(3):397–433, 1992.

[25] J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods*, SIAM Frontiers in Applied Mathematics, pages 73–131, Philadelphia, 1987. SIAM.

[26] A. Brandt. Multigrid techniques with applications to fluid dynamics:1984 guide. In *VKI Lecture Series*, pages 1–176, March 1984.

[27] W. Hackbush. *Multigrid Methods and Applications.* Springer-Verlag, Berlin, Germany, 1985.

[28] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. AIAA Paper 81-1259, 1981.

[29] A. Jameson. Solution of the Euler equations by a multigrid method. *Applied Mathematics and Computation*, 13:327–356, 1983.

[30] B. van Leer, C. H. Tai, and K. G. Powell. Design of optimally-smoothing multi-stage schemes for the Euler equations. AIAA Paper 89-1933, June 1989.

[31] E. Perez. A 3D finite-element multigrid solver for the Euler equations. INRIA Report No. 442, September 1985.

[32] V. Parthasarathy and Y. Kallinderis. New multigrid approach for three-dimensional unstructured, adaptive grids. *AIAA J.*, 32(5):956–963, 1994.

[33] S. D. Connell and D. G. Holmes. A 3D unstructured adaptive multigrid scheme for the Euler equations. *AIAA J.*, 32(8):1626–1632, 1994.

[34] D. J. Mavriplis. *Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes.* PhD thesis, Princeton University, MAE Department, 1987.

[35] D. J. Mavriplis. Multigrid solution of the two-dimensional euler equations on unstructured triangular meshes. *AIAA Journal*, 26(7):824–831, July 1988.

[36] D. J. Mavriplis. A three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. AIAA Paper 94-1878, June 1994; to appear in AIAA J.

[37] M. P. Leclercq. *Resolution des Equations d'Euler par des Methods Multigrilles Conditions aux Limites en Regime Hypersonique.* PhD thesis, Univerite de Saint-Etienne, Applied Math, April 1990.

[38] E. Morano and A. Dervieux. Looking for O(N) Navier-Stokes solutions on non-structured meshes. In *6th Copper Mountain Conf. on Multigrid Methods*, pages 449–464, 1993. NASA Conference Publication 3224.

[39] K. Riemslagh and E. Dick. A multigrid method for steady Euler equations on unstructured adaptive grids. In *6th Copper Mountain Conf. on Multigrid Methods, NASA conference publication 3224*, pages 527–542, 1993.

[40] W. A. Smith. Multigrid solution of transonic flow on unstructured grids. In *Recent Advances and Applications in Computational Fluid Dynamics*, November 1990. Proceedings of the ASME Winter Annual Meeting, Ed. O. Baysal.

[41] V. Venkatakrishnan and D. J. Mavriplis. Agglomeration multigrid for the three-dimensional Euler equations. AIAA Paper 94-0069, June 1994; to appear in AIAA J.

[42] A. Jameson. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flow. AIAA paper 93-3359, Proceedings of the AIAA 11th Computational Fluid Dynamics Conference, Orlando, Florida, July 1993.

[43] V. Vatsa and B. W. Wedan. Development of a multigrid code for 3D Navier-Stokes equations and its application to a grid refinement study. *Computers and Fluids*, 18(4):391–403, August 1990.

[44] Ramshaw J, D. Conservative rezoning algorithm for generalized two-dimensional meshes. *Journal of Computational Physics*, 59(2):193–199, 1985.

[45] H. Guillard. Node nested multigrid with Delaunay coarsening. INRIA Report No. 1898, 1993.

[46] T. F. Chan and B. Smith. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. UCLA CAM Report 93-42, Dept. of Mathematics, University of California, Los Angeles, December 1993.

[47] L. Matheson and R. Tarjan. Unstructured multigrid strategies on massively parallel computers: A case for integrated design. In *27th Hawaiian International Conference on Systems Science, Vol 2, IEEE*, pages 169–178, 1994.

[48] D. J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for viscous turbulent flows. AIAA Paper 94-2332, June 1994; to appear in Computers and Fluids.

[49] D. J. Mavriplis and V. Venkatakrishnan. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. AIAA Paper 95-0345, January 1995.

[50] B. Koobus, M. H. Lallemand, and A. Dervieux. Unstructured volume-agglomeration MG: Solution of the poisson equation. INRIA Report 1946, June 1993.

[51] T. J. Barth. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. AIAA Paper 91-0721, January 1991.

[52] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. AIAA Paper 92-0439, January 1992.

[53] S. Pirzadeh. Viscous unstructured three-dimensional grids by the Advancing-Layers Method. AIAA Paper 94-0417, January 1994.

[54] L. Martinelli and A. Jameson. Validation of a multigrid method for the Reynolds-averaged Navier-Stokes Equations. AIAA Paper 88-0414, January 1988.

[55] B. R. Hutchinson and G. D. Raithby. A multigrid method based on the additive correction strategy. *Numerical Heat Transfer*, 9:511–537, 1986.

[56] Y. Takahashi. A lumping method for numerical calculations of stationary distributions of markov chains. Research Rep. No. B-18, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, June 1975.

[57] G. Horton and S. T. Leutenegger. A multi-level solution algorithm for steady-state Markov chains. ICASE report no. 93-81, NASA CR 191558, November 1993.

[58] V. Venkatakrishnan. Parallel implicit unstructured grid Euler solvers. *AIAA J.*, 32(10):1985–1991, 1994.

[59] C. B. Jenssen and P. A. Weinerfelt. A coarse grid correction scheme for implicit multi-block euler calculations. AIAA Paper 95-0225, January 1995.

[60] N. Decker and J. Van Rosendale. Operator induced multigrid algorithm using semi-refinement. In *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods, J. Mandel et. al. eds., SIAM*, pages 87–105, April 1989.

[61] E. Morano D. J. Mavriplis and V. Venkatakrishnan. Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems. ICASE report to appear, April 1995.

[62] S. McCormick and J. Thomas. The fast adaptive composite grid (FAC) method for elliptic equations. *Mathematics of Computations*, 46:439–456, 1986.

[63] D. J. Mavriplis. Zonal multigrid solution of compressible flow problems on unstructured and adaptive meshes. ICASE Rep. No. 89-35, NASA CR-181848, May 1989.

[64] P. Vankeirsbilck. *Algorithmic Developments for the Solution of Hyperbolic Conservation Laws on Adaptive Unstructured Grids.* PhD thesis, Katholieke Universiteit Leuven, March 1993.

[65] S. Tassan. Canonical forms of multidimensional steady inviscid flows. ICASE report no. 93-34, NASA CR 191488, June 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1995	Contractor Report

4. TITLE AND SUBTITLE
MULTIGRID TECHNIQUES FOR UNSTRUCTURED MESHES

5. FUNDING NUMBERS
C NAS1-19480
WU 505-90-52-01

6. AUTHOR(S)
D. J. Mavriplis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science
and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 95-27

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-195070
ICASE Report No. 95-27

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Dennis M. Bushnell
Final Report
Lecture Notes for the 26th CFD Lecture Series of von Karman Institute; AGARD publication

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 64

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An overview of current multigrid techniques for unstructured meshes is given. The basic principles of the multigrid approach are first outlined. Application of these principles to unstructured mesh problems is then described, illustrating various different approaches, and giving examples of practical applications. Advanced multigrid topics, such as the use of algebraic multigrid methods, and the combination of multigrid techniques with adaptive meshing strategies are dealt with in subsequent sections. These represent current areas of research, and the unresolved issues are discussed. The presentation is organized in an educational manner, for readers familiar with computational fluid dynamics, wishing to learn more about current unstructured mesh techniques.

14. SUBJECT TERMS
Unstructured; Multigrid

15. NUMBER OF PAGES
61

16. PRICE CODE
A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

NSN 7540-01-280-5500

Standard Form 298(Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102