# A UNIFIED MULTIGRID SOLVER FOR THE NAVIER-STOKES EQUATIONS ON MIXED ELEMENT MESHES

*D. J. Mavriplis*
*and*
*V. Venkatakrishnan*

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

A unified multigrid solution technique is presented for solving the Euler and Reynolds-averaged Navier-Stokes equations on unstructured meshes using mixed elements consisting of triangles and quadrilaterals in two dimensions, and of hexahedra, pyramids, prisms and tetrahedra in three dimensions. While the use of mixed elements is by no means a novel idea, the contribution of the paper lies in the formulation of a complete solution technique which can handle structured grids, block structured grids, and unstructured grids of tetrahedra or mixed elements without any modification. This is achieved by discretizing the full Navier-Stokes equations on tetrahedral elements, and the thin layer version of these equations on other types of elements, while using a single edge-based data-structure to construct the discretization over all element types. An agglomeration multigrid algorithm, which naturally handles meshes of any types of elements, is employed to accelerate convergence. An automatic algorithm which reduces the complexity of a given triangular or tetrahedral mesh by merging candidate triangular or tetrahedral elements into quadrilateral or prismatic elements is also described. The gains in computational efficiency afforded by the use of non-simplicial meshes over fully tetrahedral meshes are demonstrated through several examples.

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

## 1 Introduction

Over the last decade, unstructured mesh techniques have gained popularity due to the flexibility they afford in dealing with complex geometries. However, a major drawback of such techniques remains their lower efficiency and resulting increased computational overheads as compared to structured mesh techniques.

This lower computational efficiency is due to several factors. The most obvious of these is the requirement of explicitly storing the connectivity of the mesh and accessing memory locations through indirect addressing. The difficulties associated with constructing sophisticated implicit or multigrid solution procedures for unstructured meshes have resulted in widespread use of simple algorithms such as explicit time-stepping, which generally results in inefficient solution procedures. Finally, the use of simplicial meshes (triangles in two dimensions and tetrahedra in three dimensions) often results in discretizations which are considerably more expensive to evaluate than what may be achieved using quadrilateral/hexahedral structured or block-structured meshes, both in terms of the required number of mesh points and the connectivity between the mesh points.

While the indirect addressing overheads cannot be avoided in an unstructured mesh technique, the overheads associated with most other factors can be overcome, or at least minimized. Indeed, multigrid techniques for unstructured meshes have been demonstrated both in two dimensions and in three dimensions for the Euler and the Navier-Stokes equations by several authors [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Unstructured multigrid methods are capable of delivering convergence rates for steady-state problems which are competitive with the best structured-grid multigrid solvers available, on a per cycle basis.

Thus, the most promising area for further increasing the overall efficiency of unstructured mesh techniques is to decrease the cost of the discretization itself. The additional cost incurred by the use of simplicial meshes can be demonstrated by considering a structured hexahedral mesh of N vertices. For a vertex-based finite-volume scheme, which results in a nearest neighbor stencil, there are N unknowns and 3N fluxes which must be evaluated. (For nearest neighbor stencils, the number of fluxes to be evaluated is given by the number of edges in the mesh, which asymptotically tends to 3N for a hexahedral mesh, neglecting boundary effects). If this hexahedral mesh is now subdivided into a tetrahedral mesh (by dividing each hexahedron into 6 tetrahedra), the equivalent finite-volume scheme consists of N unknowns, as previously, but the evaluation of 7N fluxes is now required to construct the discretization (since the number of edges in a tetrahedral mesh tends to the number of vertices plus the number of tetrahedra, neglecting boundary effects). Thus, for a given distribution of vertices, a tetrahedral mesh discretization is roughly twice as expensive to evaluate as a hexahedral mesh discretization.

Another difficulty associated with simplicial meshes is their isotropic nature. The very fact that triangular and tetrahedral elements do not exhibit any preferred direction is what makes them ideal for discretizing arbitrarily complex configurations. In fact, most unstructured mesh generation techniques rely on this property. However, when a configuration with a preferred direction is to be discretized, and different resolutions in the various directions are desired, unstructured mesh generation techniques are generally known to experience great difficulty in robustly delivering the different desired directional resolutions. Consider, for example, a high aspect-ratio wing geometry. Wing geometries require high resolution near their leading and trailing edges. When using an isotropic unstructured mesh, the high leading and trailing edge resolution requirements also result in high spanwise resolutions in these areas, which greatly increase the number of grid points. This degree of spanwise resolution is unnecessary, since the spanwise gradients are known by design to be relatively small. Similar difficulties occur in the boundary layer regions near a wall for high-Reynolds number viscous flows, where the normal gradients are several orders of magnitude greater than the streamwise gradients.

The generation of stretched unstructured meshes has been pursued to alleviate this problem, both in two and three dimensions. However, in hindsight, it is apparent that all of these methods introduce some degree of structure into the stretched mesh regions, either simply as a result of the overall algorithm [11, 12], or by design [13, 14, 15, 16, 17]. For example, it is well known that for a stretched two-dimensional triangulation, obtuse triangles must be avoided due to the poor approximation accuracy which they afford [18]. Since the sum of the three angles in a triangle must be equal to 180 degrees, the only permissible stretched triangles are those with one small angle, and two angles which are close to 90 degrees. Hence, nearly right angle triangles are required in the highly stretched regions of the mesh. It is then a simple matter to see that any set of right angle triangles which tessellate a domain can be grouped into a set of quadrilaterals, simply by identifying and removing the appropriate diagonal edges. A local mesh structure can therefore always be identified in such regions. In fact, three dimensional stretched unstructured meshes are often derived from local structured or semi-structured meshes of hexahedral or prismatic elements, which are then subdivided into tetrahedral elements.

In addition to the increased complexity of simplicial meshes, recent two-dimensional experiments have suggested that upwind finite-volume discretizations may suffer an accuracy degradation on stretched right-angle triangular meshes [19], due to the irregular shape of the resulting median-dual control volume. In fact, the use of a containment-circle based dual control volume, which de-emphasizes the contributions of the diagonal edges in such meshes has been proposed [20].

Rather than simply de-emphasizing such edges, we may choose to remove them altogether, thus forming quadrilaterals out of pairs of triangular elements in two-dimensions, or prisms, pyramids and hexahedra out of groups of tetrahedra in three dimensions. The goal is to reduce the cost of the discretization as much as possible by switching to such types of elements in regions of high grid stretching. The idea of using mixed elements in an unstructured mesh technique is by no means novel, and has been previously advocated by several authors [13, 14, 21, 22, 23, 24]. In fact, many have recognized the benefits of mixed elements, but have nevertheless advocated the use of simplicial meshes (using tetrahedral elements only), due to the homogeneity in data-structures and added simplicity such meshes provide. The main contribution of this paper is therefore to demonstrate how a solution procedure for mixed element meshes may be devised which retains the properties of homogeneity and simplicity coupled with rapid convergence and simple parallelization.

In the following section, it will be shown how a single edge-based data-structure can be employed to construct the discretization of the Euler equations on meshes of mixed hexahedra, prisms, pyramids and tetrahedra. The use of an agglomeration multigrid approach for efficiently solving the discrete equations is then outlined in section 3, and several inviscid flow calculations which illustrate the efficiency of this approach are given in section 4.

The extension of this methodology to the Navier-Stokes equations is described in section 5. An automatic algorithm for reducing the complexity of unstructured meshes by merging simplicial elements into other types of elements is then described in section 6. Finally, two and three dimensional examples of viscous flow computations on hybrid meshes are given in section 7.

## 2 Inviscid Discretization

The basic discretization for the inviscid flow terms are formed using a central difference finite-volume scheme with added artificial dissipation. For tetrahedral elements, this can also be formulated as a Galerkin finite-element scheme. Both finite-volume and finite-element schemes on tetrahedral meshes result in the same nearest neighbor stencil. Since all nearest neighbors are joined to the vertex under consideration by an edge, the stencil can be expressed using an edge-based data-structure. For hexahedral elements, the finite-volume and finite-element discretizations are no longer equivalent. The finite-element formulation results in a 27 point stencil, involving all corner points of the 8 hexahedra which touch the considered vertex. This type of stencil cannot be represented by an edge based data-structure, since all points in the stencil are not linked to the considered vertex by an edge. The finite-volume discretization, where the control-volume is constructed by joining the centroids of all 8 hexahedra which touch the considered vertex, results in a 7 point stencil, where all the points of the stencil are located on edges emanating from the considered vertex, as shown in Figure 1. This type of discretization can be represented by the same edge-based data-structure employed for the tetrahedral element discretization.

We therefore employ a finite-volume/element discretization on tetrahedral elements and a finite-volume discretization on hexahedral and other types of elements (prisms and pyramids). The discretization across the entire mesh is represented by a single edge-based data-structure. This type of construction has previously been proposed by several authors (see [21, 22] for example).

## 3 Multigrid Technique

Once the governing flow equations have been discretized in space, they form a set of coupled ordinary differential equations which must be integrated in time to obtain the steady-state solution. An explicit multi-stage Runge-Kutta time-stepping scheme is employed to advance these equations in time. For high resolution meshes, simple explicit schemes of this type yield very slow convergence, and result in inefficient solution schemes.

An agglomeration multigrid technique is therefore employed to accelerate convergence to steady-state [7, 8, 9]. The idea behind agglomeration multigrid is to construct coarse grid levels automatically from the fine grid by fusing or agglomerating fine grid control volumes together. The agglomerated coarse grid control volumes become large polygons (polyhedra in 3D) and require a rediscretization of the governing equations. However, since the coarse grid control volumes may have arbitrary polygonal shapes, the type of elements constituting the fine grid is irrelevant. Agglomeration multigrid can therefore be applied as easily to structured and block structured grids, as to tetrahedral unstructured and mixed element meshes.

Agglomeration multigrid can also be interpreted as an algebraic multigrid technique for sparse matrices. In fact, the algorithm only requires as input a sparse matrix, which is usually given in the form of a weighted graph, i.e. vertices and edge-based coefficients, where the edge coefficients represent the non-zeros of a sparse matrix. The agglomeration procedure can then be viewed as a technique for eliminating certain vertices, and obtain a new smaller or coarser problem on the remaining subset of vertices. The equations to be solved on this new coarser level can be entirely derived by a projection of the fine grid matrix operator, (i.e. by summation of the constituent fine grid equations within an agglomerated cell), and thus rediscretization of the governing equations on the coarse levels is not required. The structure or character of the fine grid which generates the equations to be solved is therefore irrelevant in the agglomeration multigrid approach. This may not necessarily be true for more traditional multigrid strategies, where coarse mesh levels are explicitly constructed. For example, for mixed-element meshes it remains unclear whether regions of the domain must contain similar type elements on the coarse levels as on the fine mesh, when employing an overset-mesh multigrid strategy.

The agglomeration procedure is accomplished by using a greedy-type frontal algorithm. A priority queue is maintained, which constitutes the front, and new vertices incident to the front are picked to initiate a local agglomeration of their neighbors. Each set of agglomerated control volumes is placed behind the front, and the process is repeated until the entire domain is covered.

In order to maintain favorable overall complexity of the multigrid cycle, each coarser level should contain approximately 8 times fewer (4 times fewer in 2D) vertices and edges than the previous level. The specific algorithm described in [7] roughly achieves such results on tetrahedral meshes by deriving maximal independent sets, but fails to produce adequate reductions in coarse level complexity for non-simplicial meshes. This is a result of the sparser graphs associated with such meshes; a hexahedral mesh, for example, contains less than half the number of edges associated with the tetrahedralization of the same point set. Therefore, the algorithm is modified as described below.

At every step of the agglomeration procedure, having picked a seed vertex (vertex that will be retained), the modified algorithm for agglomeration in 2D/3D consists of the following steps:

1. Form a list of neighbors not already agglomerated.

2. Agglomerate up to a maximum of 2 (3 in 3D) available neighbors from the list.

3. Augment the list by adding the neighboring vertices of the agglomerated vertex.

4. Determine a vertex from the list that is incident to at least two vertices already agglomerated into the present coarse grid control volume. Agglomerate the control volume associated with this vertex.

5. If the number of agglomerated vertices is fewer than 3 (7 in 3D) go to Step 3.

The central idea is to generate groups of agglomerated vertices which are as highly interconnected as possible. The resulting algorithm is applicable to simplicial and nonsimplicial grids and yields favorable coarse grid complexities.

## 4 Inviscid Results

In order to illustrate the flexibility and efficiency of the current discretization and multigrid approach, and to demonstrate the reduction in complexity afforded by non-simplicial meshes, three dimensional inviscid flow over a NACA 0012 wing has been computed using a fully tetrahedral mesh, a fully prismatic mesh, and a fully hexahedral mesh. All three meshes are constructed by stacking a two-dimensional triangular or quadrilateral O-mesh in the spanwise direction. A straightforward stacking of quadrilaterals yields a hexahedral mesh, while stacking of triangles yields a prismatic mesh. The tetrahedral mesh is then obtained by subdividing each prism into three tetrahedra. The three meshes are based on the same set of vertices, but differ in the connectivity of the vertices. The three meshes are depicted in Figure 2. They each contain 40,000 points. While the tetrahedral mesh contains 267,328 edges, the prismatic mesh contains 155,200 edges, and the hexahedral mesh contains only 116,800 edges.

The computed solution on each of the three meshes is depicted in Figure 3, as a set of Mach contours on the surface. The freestream Mach number is 0.8 and the incidence is 1.25 degrees for this case. The familiar strong upper shock and weak lower shock pattern is obtained. Figure 4 illustrates the multigrid convergence rate achieved for this case on each of the three meshes. The convergence rate in terms of multigrid cycles is almost identical for each of the three meshes, indicating the agglomeration algorithm performs equally well on meshes with elements other than tetrahedra. When these convergence rates are plotted in terms of cpu seconds in Figure 5, the hexahedral mesh result is seen to incur less than half the cost of the tetrahedral mesh result. Furthermore, the prismatic mesh result is only slightly more expensive than the hexahedral mesh result. These gains in efficiency for the non-simplicial meshes are a direct result of the lower number of edges on these meshes (and their coarse mesh multigrid counterparts) when compared to the tetrahedral mesh. A total of 4.4 Mwords of memory was required for the tetrahedral mesh case, while the prismatic and hexahedral mesh cases required 3.6 Mwords and 3.2 Mwords respectively. Since there are a substantial number of vertex-based arrays, these memory decreases are less than proportional to the decrease in the number of edges. (However, in viscous flow cases, the relative memory gains are higher since more edge-based arrays are required).

An additional advantage of the current approach is the ability to handle traditional block-structured meshes. For any $C_o$ continuous block structured mesh, given a list of points in each block, and the inter-block connectivity, it is a simple matter to convert this information into a list of unstructured hexahedra. The flow equations can then be discretized and solved on the hexahedra as described previously. Since all mesh levels are treated as unstructured data-sets, the agglomeration procedure is unconstrained by the macro-structure of the original block-structured mesh, and the familiar problem of requiring grid point distributions which are powers of 2 in each block in each direction is avoided. This is especially appealing for block-structured grids which contain a large number of blocks, such as those generated by current automatic blocking techniques. As an example, the inviscid flow over an ONERA M6 wing has been computed on a block-structured grid. The grid was provided by the P. Eiseman of the Program Development Corporation [25], and contains a total of 129,187 points and 141 blocks.

The mesh is illustrated in Figure 6. The freestream Mach number and the flow incidence for this case are 0.84 and 3.06 degrees respectively. The computed solution, in terms of Mach contours on the surface, is shown in Figure 7. The convergence rate of the agglomeration multigrid procedure for this case is depicted in Figure 8. In 100 cycles, the residuals are reduced by 5.5 orders of magnitude. This type of convergence is consistent with that obtained in the previous examples, as well as with that obtained by the same algorithm on fully tetrahedral meshes [7], and by structured grid multigrid solvers [26]. This calculation requires a total of 5 minutes of CPU time and 10 Mwords of memory on the CRAY C90 machine.

## 5 Extension to the Navier Stokes Equations

### 5.1 Discretization of Viscous Terms

On tetrahedral meshes, the viscous terms for the full Navier-Stokes equations are discretized using a finite-element Galerkin approximation. This discretization results in a nearest neighbor stencil, and can be represented using an edge-based data-structure [2, 27]. Furthermore, the resulting edge coefficients are symmetric and thus only 6 additional coefficients per edge are required for the viscous terms.

For hexahedral meshes, the discretization of the full Navier-Stokes terms invariably leads to a 27 point stencil involving points which are not directly connected by a mesh edge to the vertex under consideration. Our approach in this case is to resort to the thin-layer form of the Navier-Stokes equations. By neglecting the cross-diffusion terms in the full Navier-Stokes equations, and resorting to a thin-layer description in each coordinate direction of the hexahedra, a nearest neighbor stencil is recovered, and the edge-based data-structure may be used to represent this discretization. A similar strategy can be employed for prismatic and pyramidal elements. This strategy amounts to replacing the diffusion terms by a Laplacian operator. This can be achieved using one additional coefficient per edge and does not require any information about the direction and forming element type of each edge. It is important, however, to realize that this technique amounts to more than just neglecting the cross-diffusion terms, and is only strictly valid in the presence of the thin-layer assumption. For example, the full diffusion terms in two-dimensions for the x-momentum equation read:

$$\frac{4}{3}u_{xx} + u_{yy} + \frac{1}{3}u_{xy} \tag{1}$$

while present approach yields

$$u_{xx} + u_{yy} \tag{2}$$

Only under the condition :

$$u_{xx} \ll u_{yy} \qquad u_{xy} \ll u_{yy} \tag{3}$$

which amounts to the thin-layer assumption, is the present approach justified.

The use of the full viscous terms in certain regions of the domain and the thin-layer terms in other regions of the domain results in the solution of different governing equations in various regions of the domain, depending on the types of elements employed in the mesh. The justification for such an approach is based on employing elements other than tetrahedra only in regions of high mesh stretching, or in regions where viscous effects are negligible. When high mesh stretching is present in one or more directions, the flow gradients in the direction of low resolution (e.g. the streamwise direction of a boundary layer) are usually known to be small and thus a thin-layer approximation is justified. In the event these gradients are larger than anticipated, they cannot be accurately resolved due to the low mesh resolution in this direction, thus the omission of the cross-diffusion terms is still justified.

Another possibility is to form the full viscous terms by first computing the flow gradients at the vertices of the mesh by integrating around the boundary of the inviscid control volume associated with each vertex, and then repeating this operation, integrating the gradients themselves about each control volume, in order to construct a second difference. The difficulty with such an approach is that it corresponds to the use of a stencil 2h for the viscous terms, where h is a measure of the cell size, which results in lower accuracy. In regions where the thin-layer assumption is valid, the present approach can be expected to be more accurate.

One can devise schemes which combine both approaches, where the main thin-layer terms are computed using the present approach, and additional cross-terms and streamwise corrections are added using the repeated gradient integration approach. However, in regions where the thin-layer assumption is not valid, it may be simpler to subdivide the mesh elements into tetrahedra.

### 5.2 Turbulence Model

The single field-equation turbulence model of Spalart-Allmaras [28] is employed for modeling turbulence effects. This model consists of a single transport equation with convective, diffusive and source terms. The convective terms are discretized using a first-order upwind method, which results in a nearest neighbor stencil on meshes of mixed elements and may be represented by the edge-based data-structure. The particular formulation of the diffusion terms in the model contains no cross-terms, and is thus easily represented as a nearest neighbor stencil by the edge-based data-structure for mixed element meshes. The source terms require only point-wise information and are also easily constructed on meshes of arbitrary elements. The turbulence model is solved decoupled from the flow equations using a Jacobi iteration. Convergence to steady-state of the turbulence equation is accelerated using the multigrid algorithm in an analogous manner to the flow equations.

## 6 Mesh Merging Algorithm

While the present solution strategy enables the use of various types of meshes, such as block structured, hybrid and fully unstructured meshes, the main motivation for this approach is to reduce the complexity associated with unstructured grid solutions. Since the majority of unstructured meshes are fully tetrahedral, and many tasks such as mesh adaptation and solution visualization are most easily performed on fully tetrahedral meshes, a preprocessing algorithm which merges simplicial elements into more complex element types prior to the flow solution phase enables a reduction in the associated solution overheads, without complicating the implementation of other such tasks. In two dimensions, pairs of neighboring triangles may be merged into quadrilaterals by removing their common diagonal edge. Since a perfect quadrilateral (*i.e.* a rectangle) contains four co-circular points, neighboring pairs of triangles which are formed by four nearly co-circular points may be flagged as candidates for merging. The resulting diagonal edges to be removed can be identified by constructing the Voronoi dual of the mesh. This dual mesh is obtained by drawing straight-line segments between the circumcenters of neighboring triangles, as shown in Figure 9.

Each mesh edge is associated with a perpendicular dual edge. When two triangles are co-circular, the dual edge of their common diagonal becomes vanishingly small. Candidate edges for removal may therefore be identified by comparing the length of their Voronoi dual edge with the perimeter of the Voronoi control volume about each of the edge end-points. Such edges are then removed if they represent the smallest dual edge of one of their forming triangles, and if neither triangle has already been merged into a quadrilateral.

This procedure is closely related to the edge removal technique described in [29], as well as to the containment dual control-volume approach advocated in [20] in order to de-emphasize the contributions of diagonal edges in a finite-volume scheme. (In fact, the containment dual may be substituted for the Voronoi dual in the present algorithm.)

As an example, the above algorithm is applied to a triangular unstructured mesh generated by the advancing layers method [16]. The original mesh, a partially merged version, and a fully merged version of this mesh are depicted in Figure 10. The partially merged mesh is obtained by employing a stringent criterion on the relative size of the dual edge, which results in merging only in regions of high grid-stretching where the elements are almost right angle triangles. On the other hand, for the fully merged mesh, a more liberal criterion results in a 30 % reduction in the number of edges, which is very close to the maximum possible for the two dimensional case.

In three dimensions, tetrahedra may be combined into prisms, pyramids or hexahedra. For a mesh of N vertices, a tetrahedral mesh contains approximately 7N edges, a hexahedral mesh 3N edges, and a prismatic mesh 4N edges. Thus, the majority of the reduction in complexity is obtained in going from tetrahedra to prisms. For simplicity, the three dimensional algorithm only seeks to construct prismatic elements in regions of high grid stretching. This is achieved by searching through neighboring element lists, and identifying groups of three nearly co-circular high aspect ratio tetrahedra which form a well shaped prism. After the prismatic elements have been formed, a subset of these may be re-subdivided in order to ensure compatible quadrilateral and triangular face patterns between neighboring tetrahedra and prisms.

## 7 Viscous Flow Results

As a two-dimensional example, the viscous turbulent flow over a three-element airfoil has been computed on the three meshes of Figure 10. All three solutions are in good agreement with each other, as well as with the corresponding experimental data taken from [30], as illustrated by the surface pressure plots in Figure 11. The convergence rates for these three cases are compared in Figure 12. The agglomeration multigrid performs equally well for the triangular mesh as for the mixed quadrilateral/triangular meshes, achieving a residual reduction of 5 orders of magnitude over 400 cycles. While this is somewhat slower than the convergence rates previously displayed for the inviscid cases, it is nevertheless representative of the performance of multigrid methods for high-Reynolds number viscous flows, where considerable grid stretching is present. The reduction of complexity in two dimensions is, however, not as impressive as in three dimensions, since a maximum of 33 % of the edges may be removed in merging triangles to quadrilaterals.

As a three dimensional example, a hybrid mesh has been constructed over a partial-span flap geometry. In a first step, a fully tetrahedral unstructured mesh was constructed over the center portion of the wing, which contains the principal geometric complexities such as the junction between the flapped and unflapped span of the wing. This portion of the mesh contains 553,401 points, and 3.1 million tetrahedra, and was generated by S. Pirzadeh using the advancing layers technique [16]. The normal spacing at the airfoil surfaces is $10^{-5}$ chords.

The three-dimensional mesh merging algorithm was then employed to transform the unstructured center span mesh into a mixed tetrahedral and prismatic mesh. Almost half of the tetrahedra were merged into prisms, resulting in a total of 489,000 prisms, and 1.7 million remaining tetrahedra in the center span mesh. The merging operation was confined to the high-aspect ratio tetrahedra in the near wall region, as illustrated in Figure 13. Each end of this mesh was then extruded in the spanwise direction, thus resulting in a mixture of prisms and hexahedra in these regions.

The final mesh contains a total of 927,000 vertices, with 1.7 million tetrahedra, 1 million prisms, and 112,500 hexahedra. A coarser but topologically similar mesh of 160,000 points is depicted in Figure 13, illustrating the structure of the mesh away from the center span, and near the airfoil surfaces. (The fine mesh cannot be displayed effectively due to printing resolution limitations). The fine mesh contains a total of 4.4 million edges, whereas a tetrahedralization of the same point set would contain approximately 6.5 million edges. Figure 14 illustrates one of the coarse level agglomeration graphs produced by the multigrid algorithm. The solution on the fine mesh is depicted in Figure 15, in terms of Mach contours on the wind-tunnel wall, and density contours on the wing surface. For this case, the freestream Mach number is 0.2, the incidence is 10 degrees, and the Reynolds number is 3.7 million. The convergence of the agglomeration multigrid algorithm is plotted in Figure 16, where a residual reduction of 3.5 orders of magnitude over 300 cycles is observed. This convergence rate is similar to that achieved for viscous flows in two dimensions, as well as in three dimensions on a fully tetrahedral mesh for this same geometry [9]. This computation required a total 135 MWords of memory and 8 CPU hours, which could be executed in about 50 wall clock minutes using all 16 processors of the CRAY C-90, but in a time sharing mode where 54% of the machine was allocated to this specific job. This represents roughly a 30 % reduction in CPU time and 20 % reduction in memory over what would be required on a fully tetrahedral mesh of the same point set. Furthermore, the extrusion process enables a low spanwise resolution at the end-walls of the wing, while maintaining the high chordwise resolution present in the center span, and therefore reduces the required number of grid points for a given solution accuracy. In Figure 17, the surface pressure distributions at several spanwise locations are compared with experimental data taken from [31], and with a calculation performed on a fully tetrahedral mesh of 2.34 million points [9]. The two computational results are in close agreement with the experimental data. Although the precise relative accuracy of these two computations remains to be determined, it is noteworthy that the mixed element solution was obtained at almost 1/3 the cost of the tetrahedral mesh solution of [9].

## 8 Conclusion

The use of nearest-neighbor stencils, an edge-based data-structure, and an agglomeration multigrid technique, results in an efficient solver which can operate on meshes of arbitrary polyhedral cells. In the present work, we have restricted ourselves to meshes of tetrahedra, prisms, pyramids, and hexahedra for simplicity. This approach enables the use of tetrahedral unstructured meshes, hybrid meshes, and even block-structured meshes. When employed in conjunction with a mesh-element merging algorithm, substantial savings in computational memory and time can be achieved for a given unstructured tetrahedral mesh calculation.

A drawback of this approach is the need to resort to the thin-layer assumption in regions where non-simplicial elements are employed, or to incur an accuracy degradation in the viscous terms by resorting to a larger (2h) stencil.

An avenue for future work is to combine mesh adaptation and mesh merging operations to obtain a more optimal distribution of mesh points, and to ensure a reduction in the mesh complexity only in regions where the thin-layer assumption is justified.

## 9 Acknowledgments

This work was made possible in large part due to the CRAY-C90 computational facility made available by the NAS aerodynamic simulation facility. The authors would also like to thank S. Pirzadeh for supplying the viscous unstructured tetrahedral meshes, and J. Ross for permitting the use of recently acquired experimental data for the partial span-flap test case.

## References

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

[2] D. J. Mavriplis. A three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. *AIAA Journal*, 33(3):445–4531, March 1995.

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

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

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

[6] 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.

[7] V. Venkatakrishnan and D. J. Mavriplis. Agglomeration multigrid for the three-dimensional Euler equations. *AIAA Journal*, 33(4):633–640, April 1995.

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

[9] 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.

[10] 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.

[11] D. J. Mavriplis. Unstructured and adaptive mesh generation for high-Reynolds number viscous flows. In *Proc. of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Barcelona, Spain*, pages 79–92, New York, 1991. North-Holland. eds. A. S. Arcilla, J. Hauser, P. R. Eisman, and J. F. Thompson.

[12] M. G. Vallet F. Hecht and B. Mantel. Anisotropic control of mesh generation based upon a Voronoi type method. In *Proc. of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Barcelona, Spain*, pages 93–103, New York, 1991. North-Holland. eds. A. S. Arcilla, J. Hauser, P. R. Eisman, and J. F. Thompson.

[13] K. Nakahashi. A finite-element method on prismatic elements for the three-dimensional Navier-Stokes equations. In *Lecture Notes in Physics*, volume 323. Springer Verlag, 1989.

[14] Y. Kallinderis and S. Ward. Hybrid prismatic/tetrahedral grid generation for complex geometries. AIAA Paper 93-0669, January 1993.

[15] R. Lohner. Matching semi-structured and unstructured grids for Navier-Stokes calculations. AIAA paper 93-3348, July 1993.

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

[17] D. L. Marcum. Generation of unstructured grids for viscous flow applications. AIAA paper 95-0212, January 1995.

[18] I. Babushka and A. K. Aziz. On the angle condition in the finite-element method. *SIAM Journal of Numerical Analysis*, 13(6), 1976.

[19] M. Aftosmis, D. Gaitonde, and T. S. Tavares. On the accuracy, stability and monotonicity of various reconstruction algorithms for unstructured meshes. AIAA Paper 94-0415, January 1994.

[20] T. J. Barth and S. W. Linton. An unstructured mesh newton solver for compressible fluid flow and its parallel implementation. AIAA paper 95-0221, January 1995.

[21] R. Struijs, P. Vankeirsbilck, and D. Deconinck. An adaptive grid polygonal finite-element method for the compressible flow equations. AIAA paper 89-1957-CP, June 1989.

[22] T. J. Barth. On unstructured grids and solvers. In *VKI Lecture Series VKI-LS 1990-03*, March 1990.

[23] P. N. Childs, J. A Shaw, A. J. Peace, and J. M. Georgala. SAUNA: A system for grid generation and flow simulation using hybrid structured/unstructured grids. In *Computational Fluid Dynamics '92*, pages 875–882, 1992. Ed Hirsch, Ch, Periaux, J., Kordulla, W., Elsevier.

[24] S. D. Connell, J. S. Sober, and S. H. Lamson. Grid generation and surface modeling for CFD. In *Surface Modeling, Grid Generation and Related Issues in CFD Solutions*, pages 29–43, May 1995. NASA Conference Publication 3291.

[25] N. Lu and P. R. Eiseman. Interactive high-quality grid generation. In *Surface Modeling, Grid Generation and Related Issues in CFD Solutions*, pages 833–844, May 1995. NASA Conference Publication 3291.

[26] V. N. Vatsa, M. D. Sanetrik, and E. B. Parlette. Development of a flexible and efficient multigrid-based multi-block flow solver. AIAA Paper 93-0677, 1993.

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

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

[29] M. L. Merriam. An efficient advancing-front algorithm for Delaunay triangulation. AIAA paper 91-0792, January 1991.

[30] W. O Valarezo, C. J. Dominik, R. J. McGhee, and W. L. Goodman. High-Reynolds number configuration development of a high-lift airfoil. In *Proceedings of AGARD Conference on High-Lift Aerodynamics*, October 1992. Paper 10, Banff, Canada.

[31] B. L. Storms, T. T. Takahashi, and J. C. Ross. Aerodynamic influence of a finite-span flap on a simple wing. Paper to be presented at the SAE Aerotech Conference, Los Angeles, CA September 9-12, 1995.

-

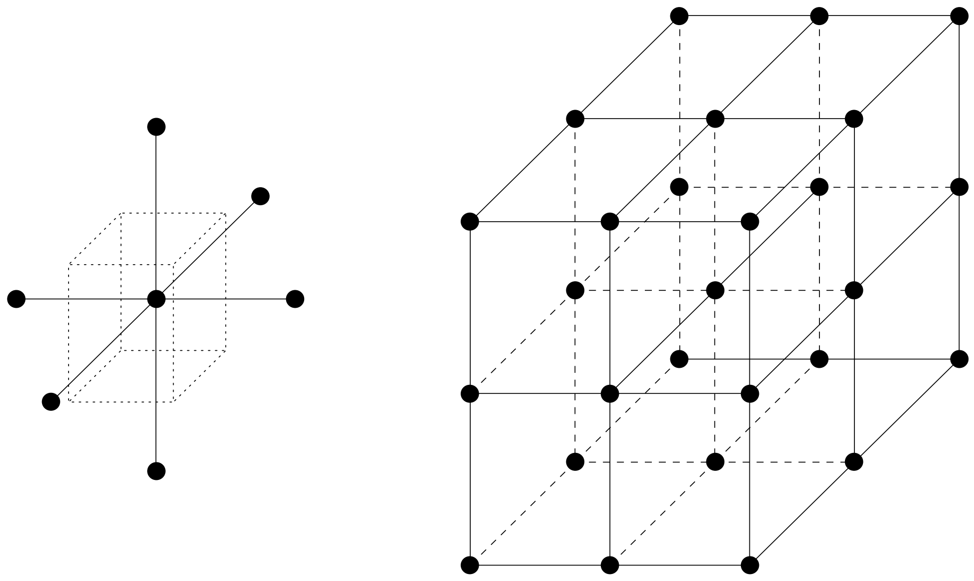


Figure 1: Illustration of 7 point finite-volume and 27 point finite-element stencil and associated control-volumes for hexahedral mesh

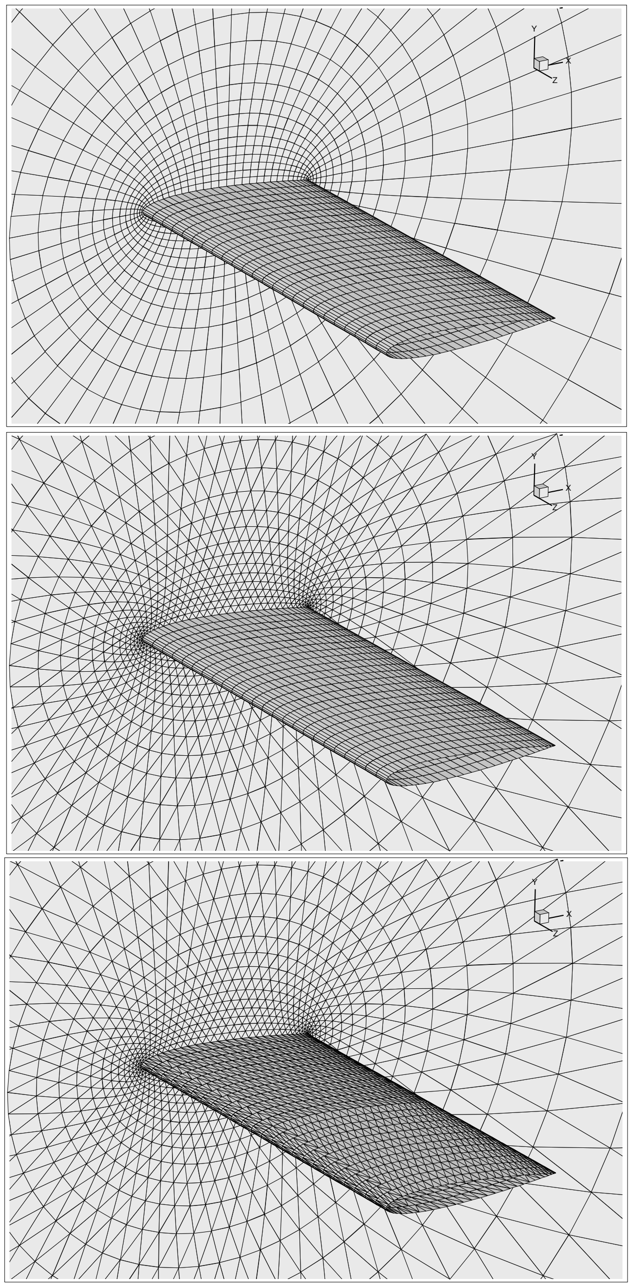


Figure 2: Hexahedral, Prismatic, and Tetrahedral Meshes about NACA 0012 Wing

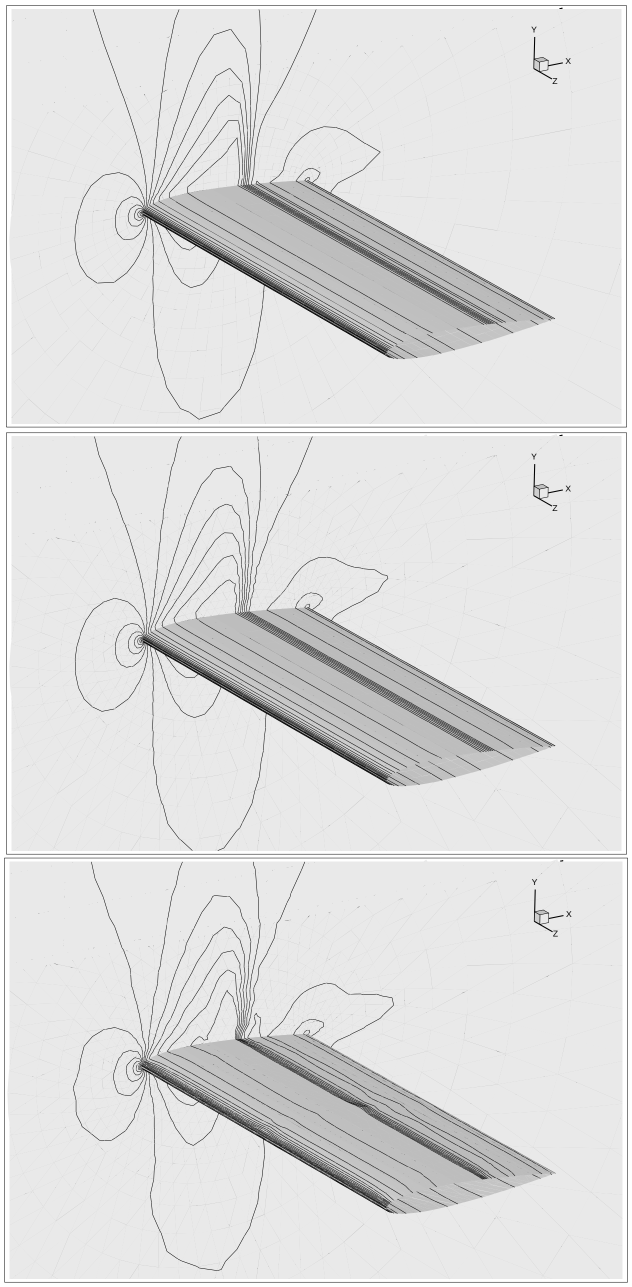


Figure 3: Computed Mach Contours for Flow over NACA0012 Wing on Hexahedral, Prismatic, and Tetrahedral Meshes

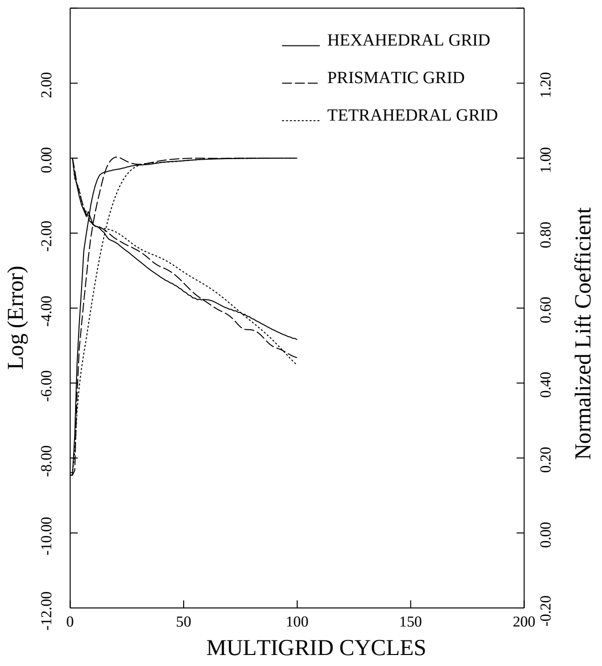


Figure 4: Convergence Rate of Various Meshes in Terms of Multigrid Cycles

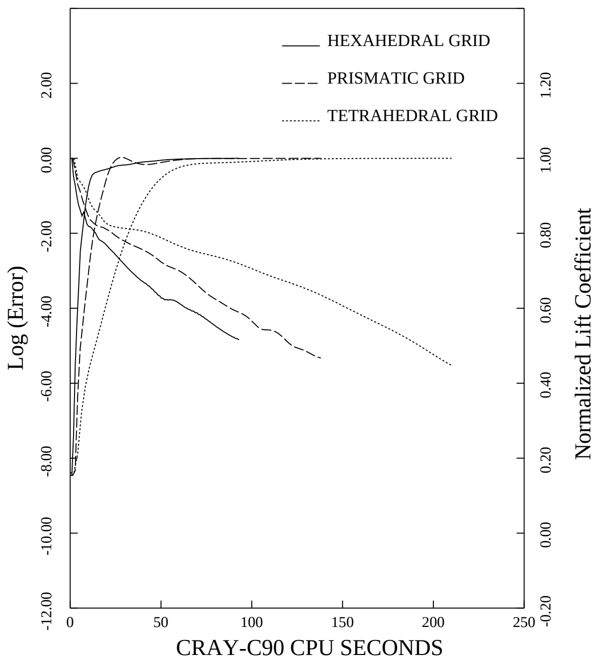


Figure 5: Convergence Rate of Various Meshes in Terms of CPU Time

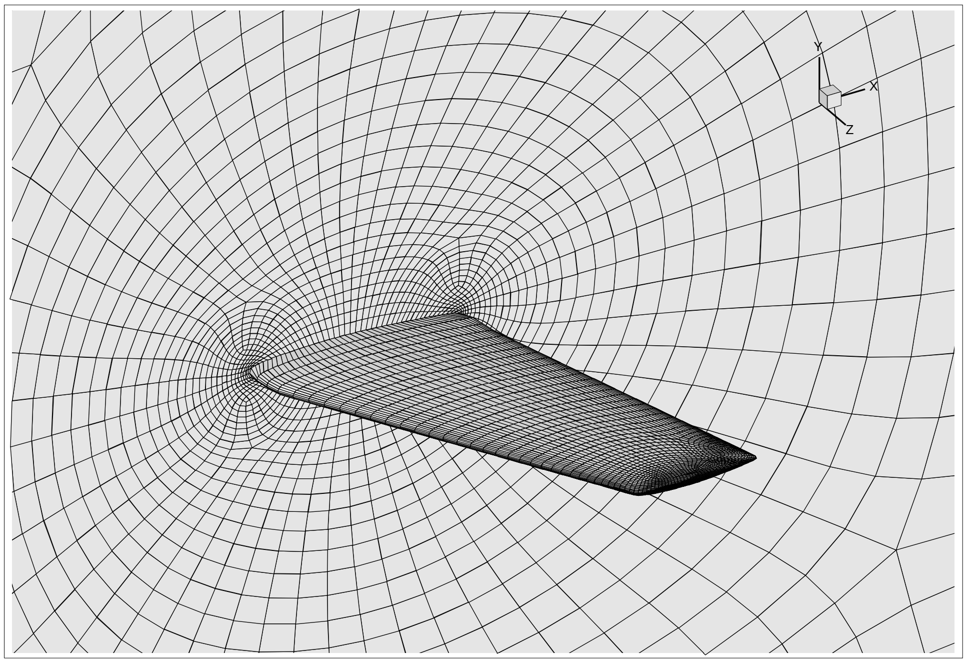


Figure 6: Block Structured Mesh about ONERA M6 Wing

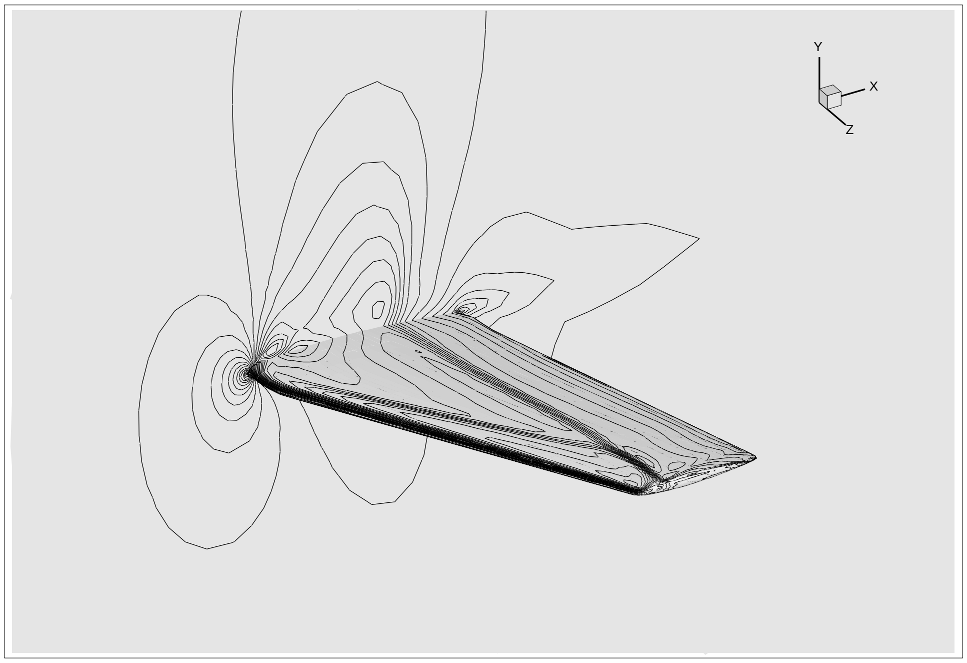


Figure 7: Computed Mach Contours on ONERA M6 Wing (Mach = 0.84, Incidence = 3.06 degrees)

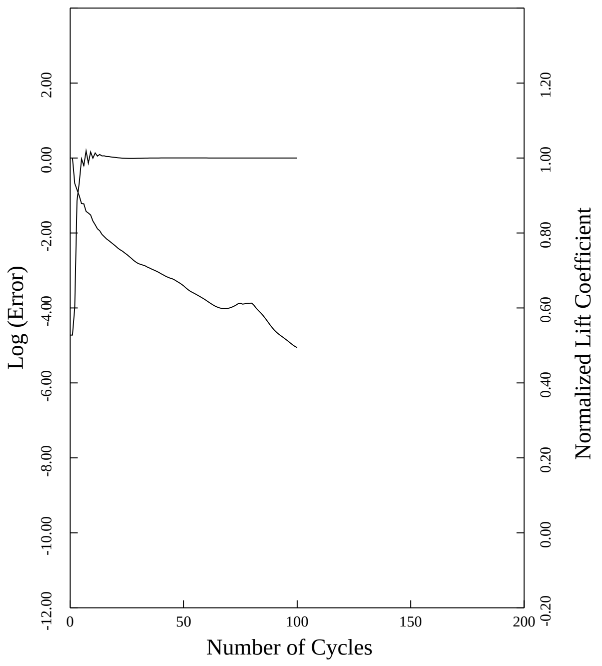


Figure 8: Convergence Rate of Agglomeration Multigrid on Multiblock Mesh for ONERA M6 Wing

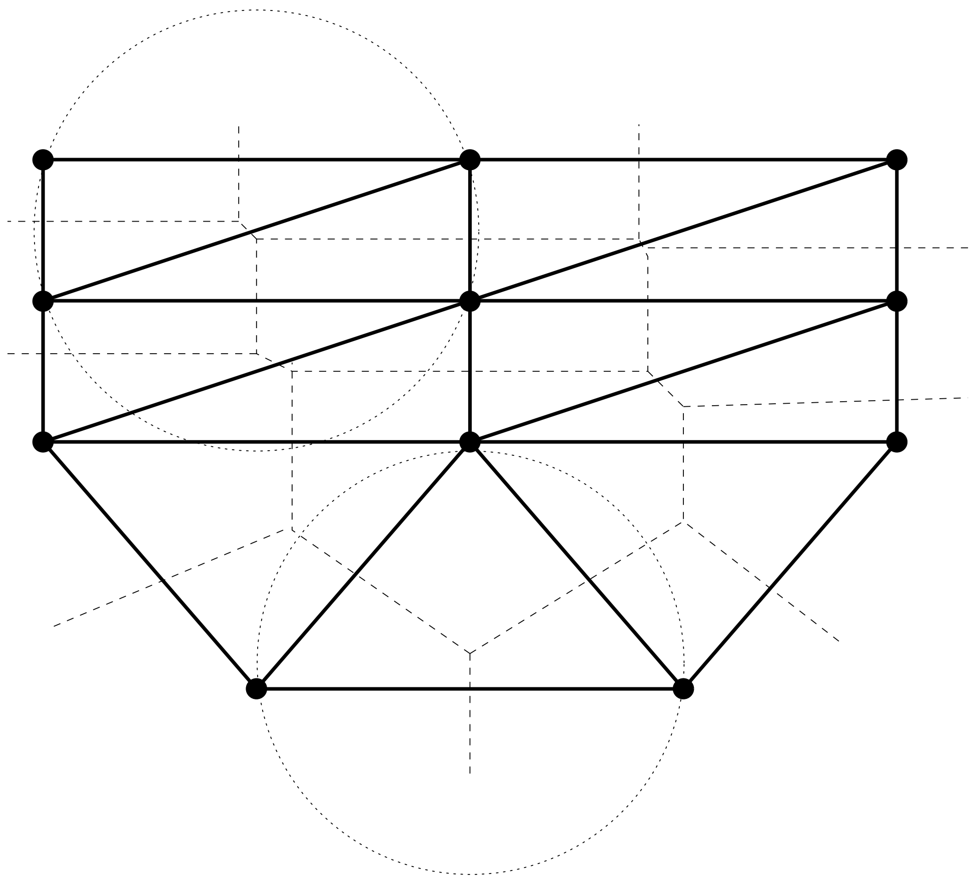


Figure 9: Voronoi dual associated with triangular mesh

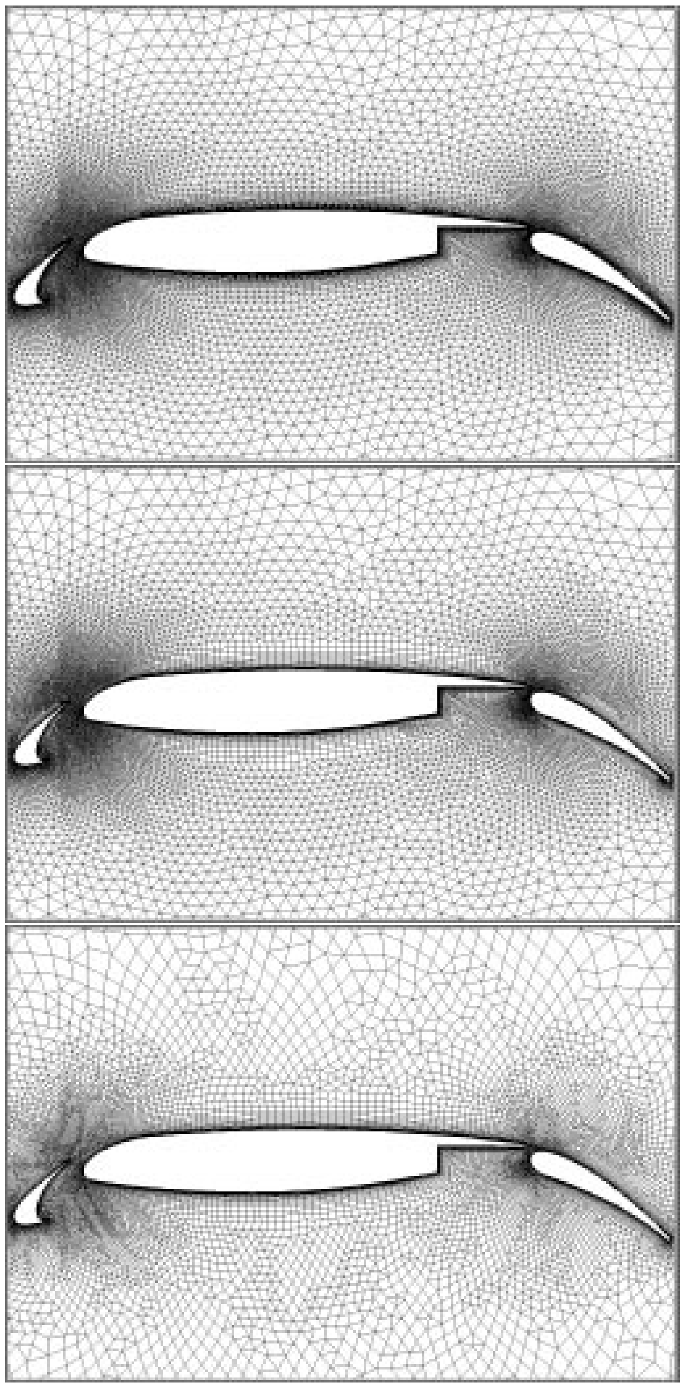


Figure 10: Fully triangular, partially, and fully merged meshes using two-dimensional mesh merging algorithm for three-element airfoil geometry

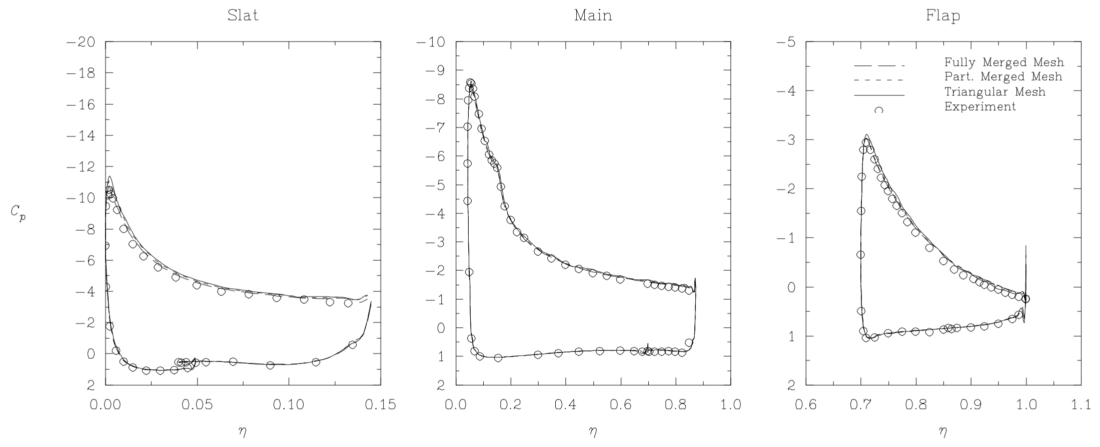


Figure 11: Comparison of Computed and Experimental Surface Pressures for Three-Element Airfoil Geometry

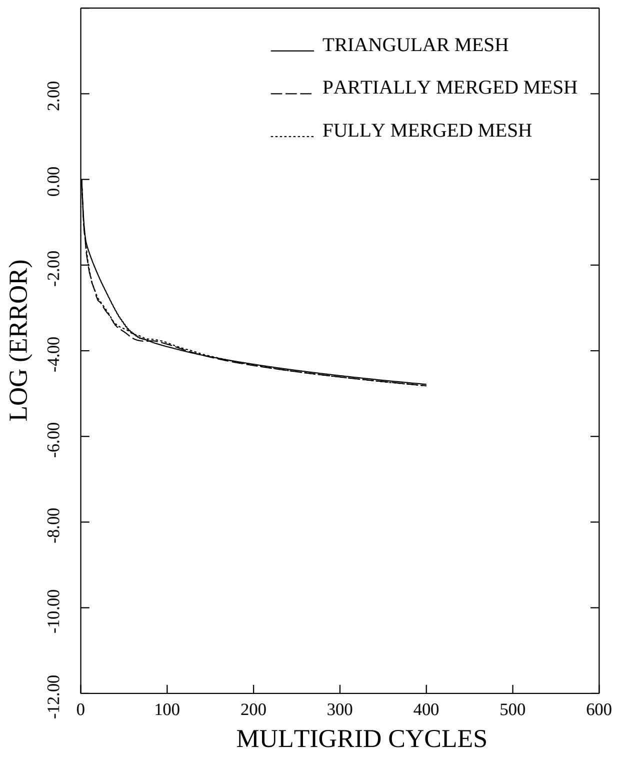
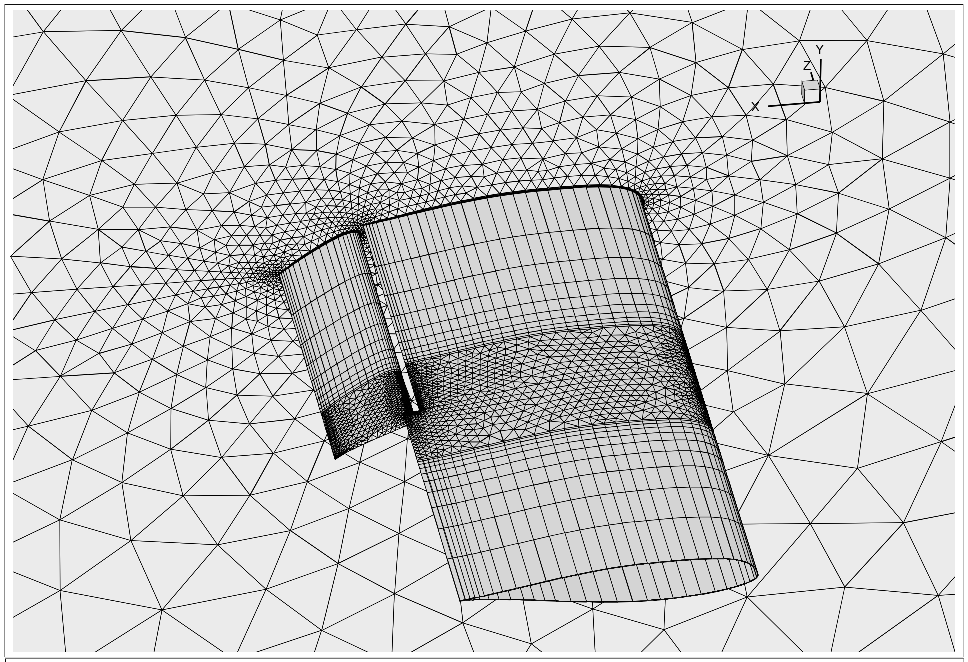


Figure 12: Convergence Rates in Terms of Multigrid Cycles for Triangular Mesh and Merged Meshes about Three-Element Airfoil Geometry



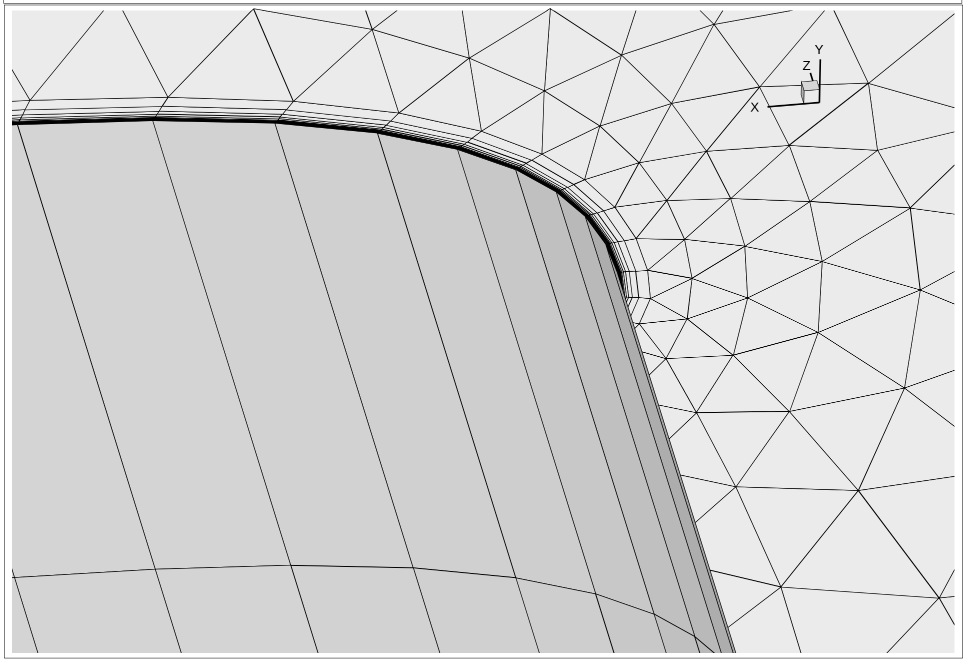


Figure 13: Mixed element mesh showing detail in near wall region for partial-span flap geometry

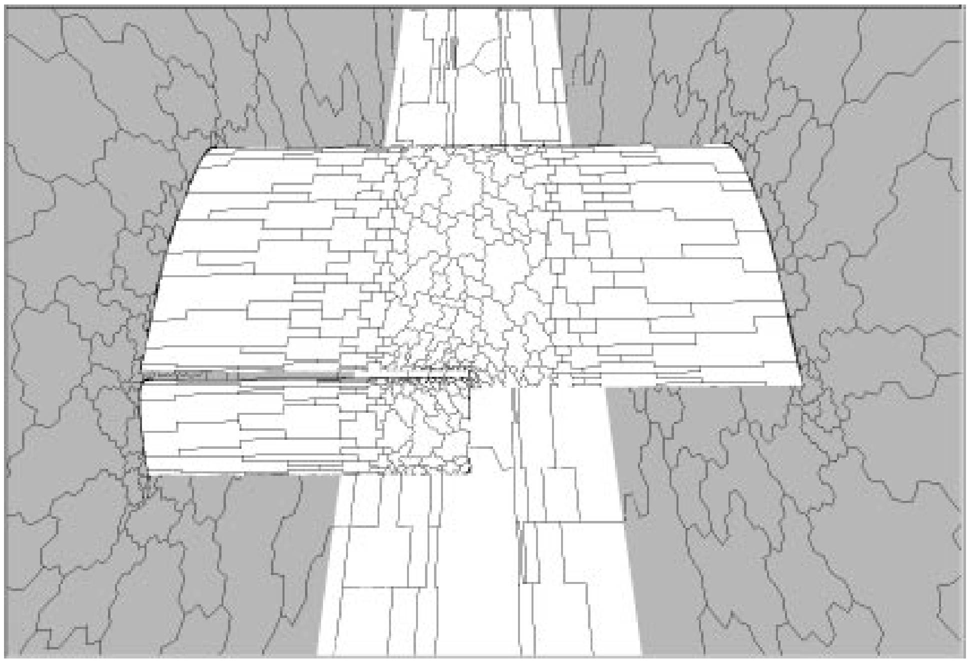


Figure 14: Second level agglomeration graph for above mesh

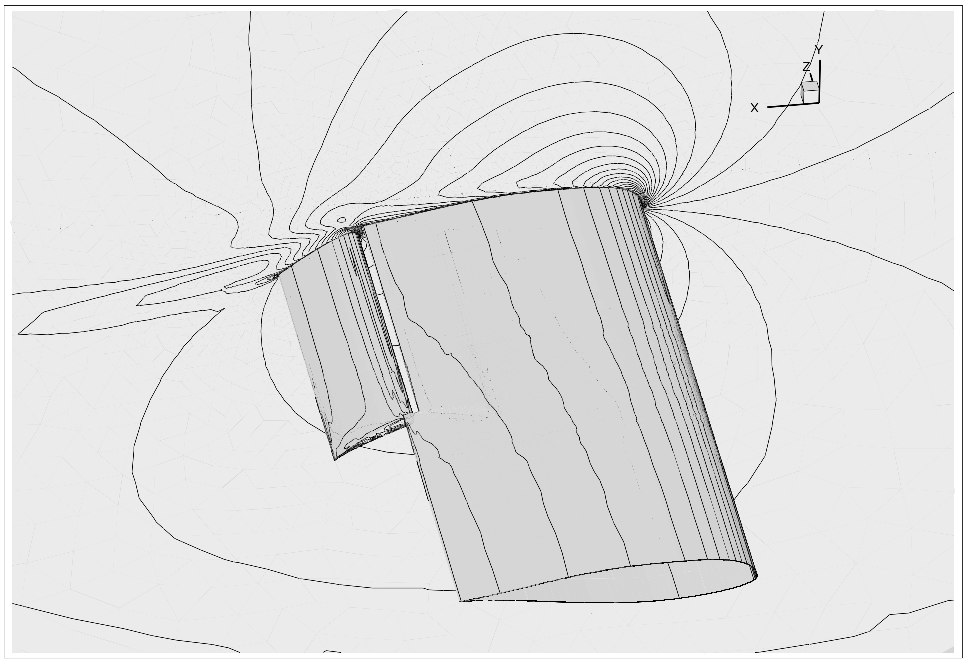


Figure 15: Computed solution displayed as Mach contours on wind-tunnel wall and pressure contours on airfoil surfaces for partial-span flap geometry

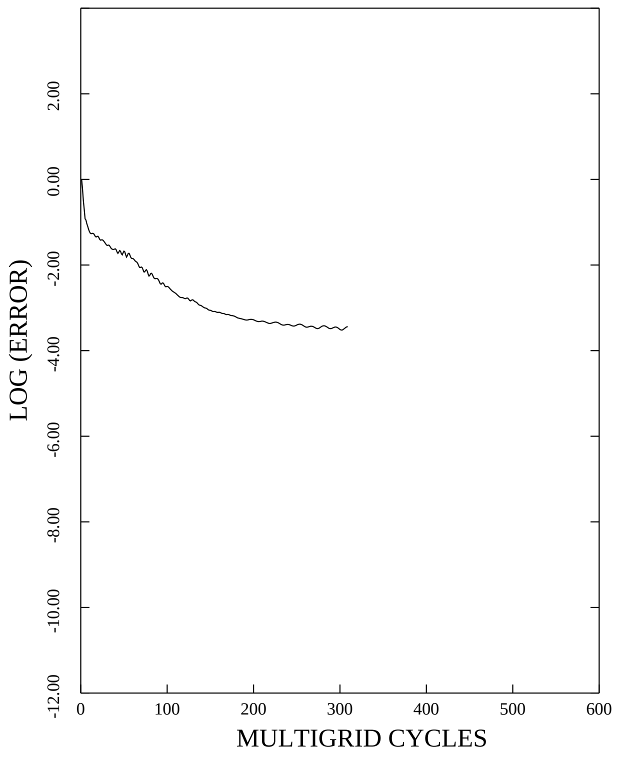


Figure 16: Convergence Rate of for Viscous Flow over Partial Span Flap on Fine Mixed Element Mesh

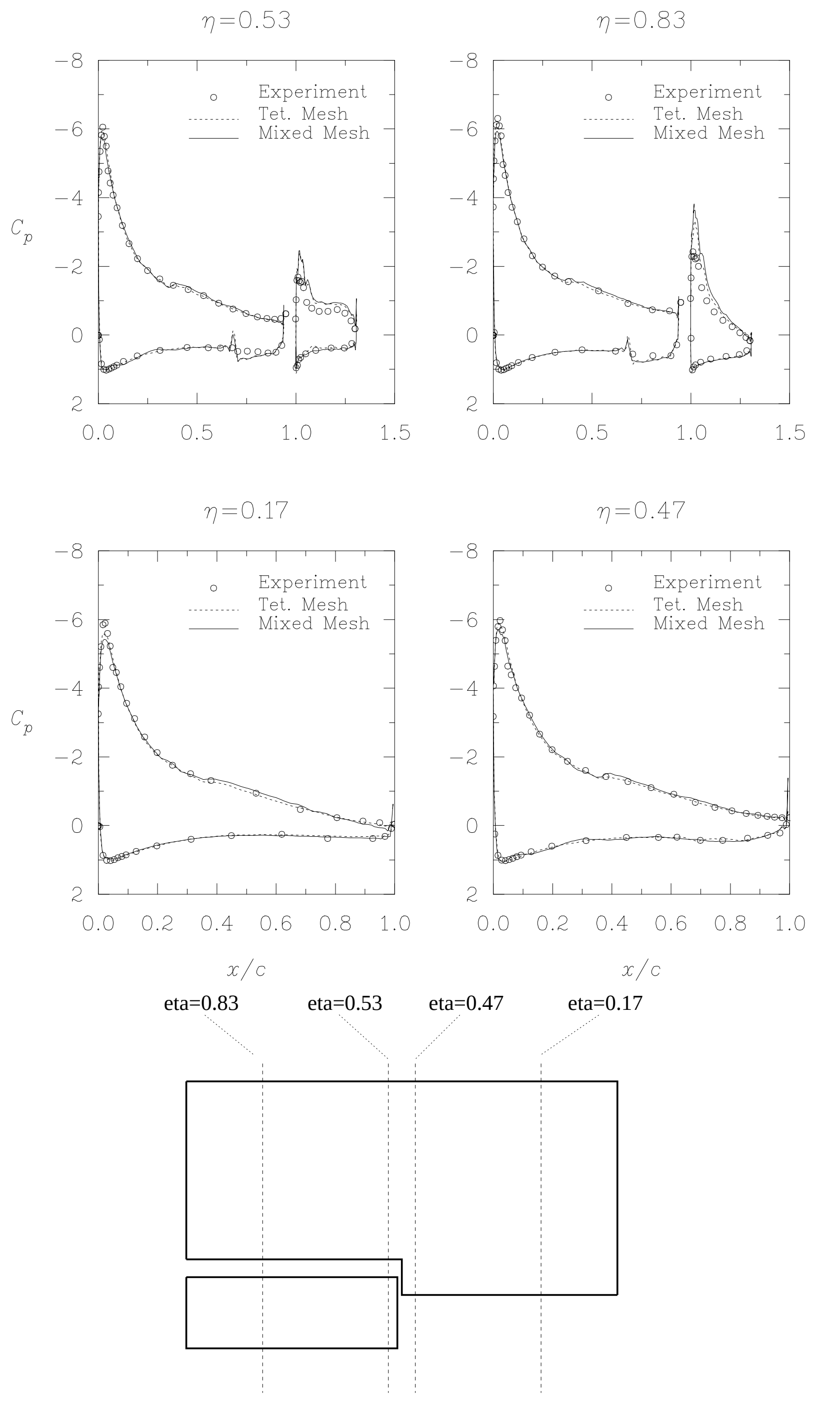


Figure 17: Comparison of Calculated and Experimental Surface Pressures for Partial Span Flap Geometry