# Data Structures and Mesh Modification Tools for Unstructured Multigrid Adaptive Techniques*

Carlo L. Bottasso† Ottmar Klaas‡ Mark S. Shephard§

Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

**Key Words:** Multigrid, adaptivity, data structures.

## Abstract

A complete set of data structures and mesh modification tools for effectively defining unstructured three–dimensional multigrids on general curved domains is presented. The mesh adaptive procedures can be used for generating hierarchies of unstructured grids by means of uniform or local refinement and coarsening, while a local retriangulation algorithm is used for controlling the degradation of the quality of the mesh during adaptation. Intergrid transfer operators are efficiently realized "on the fly" during adaptation. The data structure allows the efficient storage and handling of multiple grids, where mesh entities belonging to multiple levels can be stored just once. The capabilities and performance of the proposed procedures are exemplified by means of examples.

*Submitted to *International Journal for Numerical Methods in Engineering.*

† Currently Ricercatore, Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Milano, Italy, formerly Visiting Assistant Professor, Dept. of Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy, NY, USA.

‡ Currently Research Assistant, Institut für Baumechanik und Numerische Mechanik, Universität Hannover, Hannover, Germany, formerly Visiting Scholar, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY, USA.

§ Director, Scientific Computation Research Center, Johnson Chair of Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA.

## List of Symbols

| | |
|---|---|
| $g$ ..... : | Geometric model |
| $m_r$ ..... : | Mesh model at level $r$ |
| $\Omega_v$ ..... : | Domain associated with model $v$ $(v = g, m)$ |
| $\partial(\Omega_v)$ ..... : | Boundary of model $v$ |
| $\bar{\Omega}_v$ ..... : | Closure of domain of model $v$ $(\Omega_v \cup \partial(\Omega_v))$ |
| $G_i^{d_j}$ ..... : | Topological entity $i$ of dimension $d_j$ in the geometric model |
| $M_{i\,(p,q)}^{d_j}$ ..... : | Topological entity $i$ of dimension $d_j$ in the mesh model, appearing at mesh levels $p$ through $q$ |
| $M_{i\,(r)}^{d_j}$ ..... : | Topological entity $i$ in the mesh model of dimension $d_j$, considered at mesh level $r$ |
| $\partial(M_i^{d_j})$ ..... : | Boundary of topological entity |
| $[\,\cdot\,]$ ..... : | Ordered list of entities |
| $\{\,\cdot\,\}$ ..... : | Unordered list of entities |
| $M_{i\,(r)}^{d_j}\{M^{d_j}\}$ ..... : | Unordered group of topological entities of dimension $d_j$ that are adjacent to $M_{i\,(r)}^{d_j}$ at mesh level $r$ |
| $\sqsubset$ ..... : | Classification, i.e. association of a topological entity with a model entity |

## Introduction

Multigrid methods are at the heart of many powerful algorithms for the efficient solution of complex problems of engineering interest. In fact, multigrid methods offer the appealing advantage of solving problems in $N$ unknowns in $\mathcal{O}(N)$ operations. The spectrum of applicability of multigrid methods is impressive, as survey papers [3] clearly indicate, with elliptic and hyperbolic partial differential equations ranking among the most widely researched areas. However, the efficient analysis of PDE's cannot only address the solution of the discrete problem, but must also consider the critical issue of the adaptation of the discretization to the features of the solution. In fact, only through adaptation one can try to optimize the computational effort for a given requested accuracy in the solution.

The interaction of multigrid procedures with adaptivity represents an

efficient and flexible way to the solution of a vast class of boundary value problems. In fact, adaptivity allows the exploitation of the local smoothness of the solution and to efficiently resolve boundary layers and singularities. At the same time, the multigrid approach takes advantage of the relations among different discretizations of the governing PDE's, for delivering a fast numerical solution.

Historically, the multigrid method has been mainly applied in the context of structured discretizations, the reason being that the inherent order of such grids makes the generation of nested hierarchies of successively coarser meshes a very fast and simple operation. On the other hand, structured discretizations suffer from serious limitations, in particular they are difficult to apply to the automatic meshing of geometrically complex domains and they are limited in their ability to perform $h$–adaptivity. The alternative offered by unstructured grids has become increasingly popular in recent years due to the ability to perform fully automatic mesh generation of non–manifold objects of arbitrary geometric complexity. Moreover, unstructured grids can be modified locally by deleting or inserting grid points, allowing to adapt and optimize the discretization to the features of the solution.

From the standpoint of unstructured multigrid methods, a popular approach to the generation of the multiple mesh levels has been that of using independent grids obtained with a mesh generator [7][8]. Another approach uses adaptive refinement or coarsening of an initial mesh [9][10]. The approach of obtaining the different mesh levels through local mesh modification tools presents critical advantages over the independent meshes approach:

- The mesh modification tools can be used for creating a hierarchy of finer or coarser meshes starting from a base mesh through uniform refinement or coarsening, and for adapting the discretization to the local features of the solution based on error estimates. This opens the way to fully automated adaptive multi–level techniques.

- The intergrid mappings needed for developing the restriction/prolongation operators can be computed "on the fly" as the mesh levels are generated through refinement and/or coarsening, and involve only local searches that do not depend on the mesh size. Moreover, they do not rely on complicated data structures for avoiding expensive global searches.

- Uniform coarsening and especially uniform refinement produce tetrahedrons at rates which are significantly higher than meshing procedures.

- Since in general mesh entities of all dimensionality can be shared by more than one grid level, more compact data structures can be used that optimize the memory occupation by storing these multiple mesh entities just once. This is clearly impossible with independently generated meshes, since in general they do not share any entity, then requiring higher memory storage space.

In this paper, we present a set of automated procedures that can be used for the three–dimensional unstructured adaptive multigrid solution of PDE's in domains of arbitrary complexity. The application of this methodology to the multigrid adaptive solution of the compressible Euler equations is the subject of a paper in preparation. We give procedures that allow adaptive refinement and coarsening of an initial discretization, while a local retriangulation tool is available for controlling the degradation of the quality of the triangulation during its modification. An initial set of grid levels can be obtained by uniform selective edge–based refinement and/or coarsening of the starting grid. As the solution proceeds, additional mesh levels which complement, or replace, the initial ones can be obtained by local adaptation as guided by an error indicator. We consider edge based mesh modification tools that lead to multiple grids for which vertices at a given mesh level are also vertices at the finer mesh levels. This greatly simplifies the intergrid mappings.

A unique characteristic of the present method is that the mesh modification tools are built on top of a geometry–based mesh database. The database stores complete knowledge of the relations (termed "classification" in the following) of each mesh entity with the underlying geometric model, and it is directly interfaced with several general purpose geometric modelers that provide access to the geometric description of the domain. The understanding of these relationships allows the procedures to guarantee the validity of the generated discretizations [11], to determine when local modifications of the mesh would create invalid elements, and to guarantee that refinement improves the geometric approximation that a mesh gives of the true geometry through the positioning of newly generated vertices on the model boundaries.

The paper is organized as follows: the first section reviews the geometry–based mesh data structure that was designed to support evolving unstructured discretizations. This is followed by a discussion of the extensions to this data structure for supporting multi–level discretizations. The second section presents the mesh modification tools that are used for generating the multiple mesh levels. The refinement, coarsening and local triangulation

procedures are briefly discussed, and the procedures for the construction of the intergrid transfer operators are given. The third section presents some example problems which demonstrate the advantages and performance of the proposed procedure.

## Mesh Data Structures for Geometry–Based Mesh Modification and Adaptation

Although conventional mesh data structures have been successfully employed for many years in the context of unstructured finite element and finite volume approaches on fixed meshes, richer structures are required, and have been developed [1], for supporting mesh generation and adaptive mesh modification procedures. The structure used here [1] is characterized by several distinguishing features:

First, it makes use of the concept of classification of a derived model from its parent model. In this context, classification is defined as the unique association of a topological mesh entity of dimension $d_j$ (region, face, edge or vertex), $M_i^{d_j}$, to a topological geometric domain entity of dimension $d_k$, $G_j^{d_k}$, being $d_j \leq d_k$. This is usually written $M_i^{d_j} \sqsubset G_j^{d_k}$. In practice, classification is the mechanism through which the relationship between the mesh and the geometric model is represented. This is critical for ensuring the validity of a mesh [11] and for supporting automated mesh modification procedures. However, it is also extremely convenient in that it allows to specify the analysis attributes (such as for example boundary conditions, material properties, loads, etc.) with respect to topological entities of the geometric model, rather than the mesh. This is particularly important in adaptive environments [12].

Second, this richer data structure contains the information relative to the true geometry of the problem, i.e. the geometric domain $\bar{\Omega}_g$ over which a set of partial differential equations are defined and must be solved. This is done by means of a boundary–based representation [6] of the topology of the domain and by some form of parametric representation of the shape of the entities of the domain itself. In practice, this is achieved interfacing the database with a computer aided design (CAD) system, that provides the required geometric and/or topological interrogations through a limited set of functional operations [13]. Access to the true geometry of the problem is clearly fundamental during the mesh generation phase, but it is also critical during adaptive refinement, where newly generated mesh vertices must be properly "snapped" to the model boundary in order to improve the geomet-

ric approximation that the mesh makes of the model. Moreover, it can be useful at the finite element or finite volume level, for example in a fluid dynamic application when one has to prescribe essential boundary conditions through the evaluation of a normal at a certain location on a model face.

Third, a boundary representation is used also for the mesh. Since this data structure is designed to support operations that are highly demanding from the topology understanding point of view, it efficiently provides all the possible adjacencies, i.e. the relationships among topological entities that bound each other, of the entities that define both the geometric domain and the mesh.

Automated adaptive multi–level procedures for unstructured meshes must be able to perform efficiently refinement, coarsening and optimization of the mesh. Moreover, they must efficiently support the restriction and prolongation operators as well as the analysis process on evolving meshes. Given the difficulty of these tasks when dealing with real geometries and complex topologies, we believe that these objectives are best accomplished within the framework of a geometry–based data structure. In this work, we extend the data structure given in [1] for single–level meshes, to support multi–level representations. In this approach, all the adjacencies between entities one dimension apart are stored (Figure (1)). This choice effectively compromises the contrasting requirements of quick determination of adjacencies of each order, and control of the amount of data storage required. This adjacency set is capable of producing any adjacency through local traversals and sorts which are not function of the mesh size.

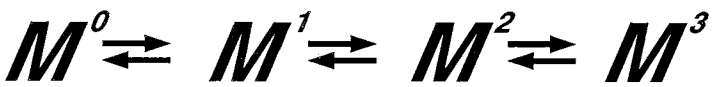


Figure 1: First order topological adjacencies.

## Data Structures for Multi–Level Meshes

In the multi–level data structure implemented in this work, all the entities of all the different mesh levels are stored with the previously explained adjacencies. To optimize memory occupation, mesh entities appearing in at least two consecutive mesh levels are stored just once.

Figure (2) exemplifies this situation: the solid line triangle is a mesh face $M^2_{1\,(1,1)}$ that appears only at level 1, while the dashed line triangle represents a mesh face $M^2_{2\,(2,2)}$ that appears only at level 2. The two mesh faces share a mesh edge $M^1_{1\,(1,2)}$ and the two mesh vertices $M^0_{1,(1,2)}$ and $M^0_{2\,(1,2)}$, that thus appear both at level 1 and level 2. The vertices $M^0_{1,(1,2)}$ and $M^0_{2\,(1,2)}$ and the edge $M^1_{1\,(1,2)}$ are represented just once in the data structure, even if they appear at two different levels.

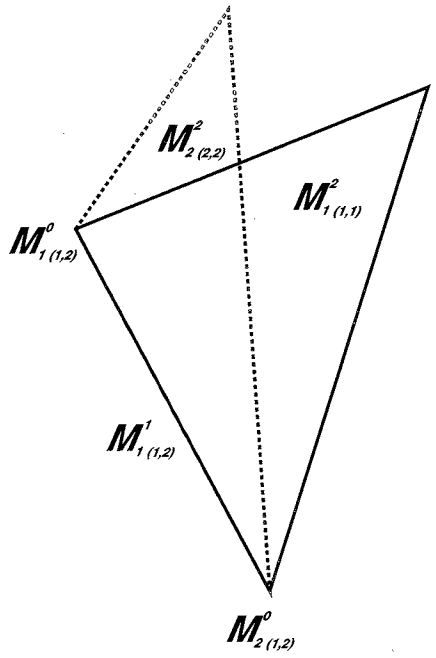


Figure 2: Compact representation of a multi–level mesh.

Adjacencies one dimension apart are stored as if all the entities belonged to the same single–level mesh. For determining at which levels one particular entity belongs, two unsigned integers are stored with each entity, corresponding respectively to the minimum and maximum level to which it appears.

It is important to note that with this approach all the first order downward adjacencies are immediately available, while the first order upward adjacencies require a local search. For example, with reference to the simple case of Figure (2), the upward adjacency $M^1_{1\,(1)}\{M^2\}$ at level 1 returns both $M^2_{1\,(1)}$ and $M^2_{2\,(2)}$, and this last entity must be discarded since it does not appear at level 1. The local traversal is inexpensive, since it is proportional to the number of mesh levels that the entity in question belongs to, and this number is bounded by the total number of mesh levels represented, which is small in multigrid applications.

It is interesting to note that, with the proposed approach, the extensions required to support a multi–level mesh are limited and simple to implement. The required modifications to the single mesh data structure previously discussed are in fact only the storage of the two unsigned integers and the provision for the fact that a mesh face can be shared by more than two mesh regions. Another implementational detail that must be considered is that a mesh face or mesh edge might have different classifications with respect to the geometric model when considered at different mesh levels. For example, a mesh edge might be classified on the model boundary when considered at a coarse level, and it might be classified on the model interior at a finer level. This situation is exemplified in Figure (3). From a given initial configuration on level 1 (Figure (3)*(a)*), a refinement procedure may split the edge $M^1_{1\,(1,1)}$ introducing a vertex $M^0_{3\,(2,2)}$ and a vertex $M^0_{4\,(2,2)}$ on the boundary $G^1_1$ (Figure (3)*(b)*). If the mesh quality can be improved by collapsing vertex $M^0_{4\,(2,2)}$ to vertex $M^0_{1\,(1,2)}$, an edge is created between the two vertices $M^0_{1\,(1,2)}$ and $M^0_{2\,(1,2)}$ (Figure (3)*(c)*). Since the edge already exists at level 1, only the maximum level of that edge has to be updated. Note that the classification of the edge $M^1_{1\,(1,2)}$ is now different for the two levels. On level 1 the edge is classified on the model edge $G^1_1$, whereas the classification on level 2 is interior. It is however simple in these occasions (which rarely appear in our experience) to store the additional classification information that is required for the entity in question.

To allow the efficient traversal of each mesh level by entity type, lists of entities of the same dimension are maintained for each mesh level. In this way, traversal at each mesh level does not require any search. The small additional storage required by these lists is compensated by the faster traversals, which otherwise would require to scan the complete pool of entities in order to select only the ones belonging to the level considered.

Generation and deletion of shared entities from a given level is a trivial operation that simply requires the update of one of the two unsigned integers

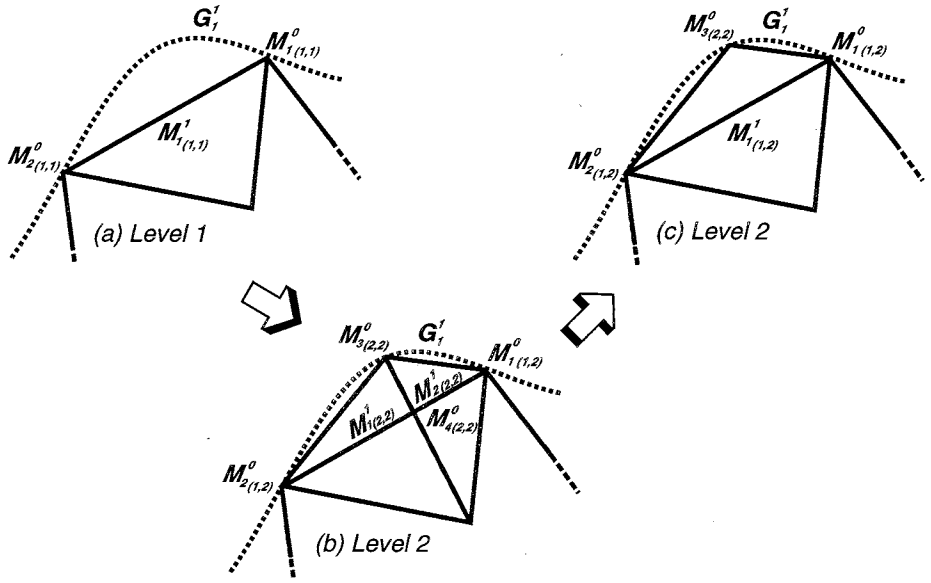


Figure 3: Multiple classification for a mesh edge appearing at two different levels.

that represent the minimum and maximum level for that entity. In order to avoid expensive searches, when an entity must be deleted from a mesh level, the corresponding list is not updated, but rather the entity is marked for deletion. Actual deletion from the list takes place the first time the entity is retrieved from the list during traversal.

## Mesh Modification Tools

In general in any adaptive multi–level technique, one would like to be able to generate a hierarchy of coarser or finer meshes starting from a given base mesh. As the solution proceeds, one then can use some form of error estimates to locally adapt the discretization to the features of the solution, for example in the presence of boundary layers, shocks, singular points, etc. or to exploit local smoothness. These adapted grids can then be incorporated in the hierarchy of meshes used by the multi–grid algorithm, leading to very efficient multi–level adaptive solution schemes [2].

In this work, the initial multiple levels and the adapted grids are obtained by means of a set of mesh adaptation tools, that include refinement,

coarsening and local retriangulation as a way to locally improve and control the mesh quality [4]. The mesh entity splitting procedure creates a vertex along each refined edge, while the edge collapsing deletes a vertex. This implies that all the vertices of each coarse level are also vertices at all the finer levels. This greatly simplifies the restriction/prolongation operators. Note that coarsening does not depend on previous refinement, so that also the initial mesh can be coarsened.

These fully automated procedures relieve the burden of generating multiple meshes from the user. Moreover, using this approach there is no need for complicated search structures for computing the intergrid transfer operators. In the following we review the mesh modification tools implemented in this work.

## Coarsening

Following [4], coarsening is performed by means of edge collapsing. The algorithm loops on all the edges marked for coarsening at a certain mesh level, and an attempt is made at collapsing it to one of its end vertices, $M_1^0$ or $M_2^0$. Before physically performing the collapsing, checks are performed to ensure that the collapsing is topologically and geometrically possible, and that the quality of the created mesh regions is above a predetermined threshold. Finally, the target vertex for the collapsing, say $M_1^0$, that produces the best triangulation with respect to a given mesh quality measure is chosen. A limit can be placed on the longest edge to collapse, in order to avoid the creation of excessively large elements.

In the case of collapsing to $M_1^0$, the algorithm proceeds by deleting all the mesh regions connected to $M_2^0$, creating a polyhedral cavity within the mesh. The edge collapsing is then completed connecting all the faces of the cavity to $M_1^0$ in order to form the new mesh regions. The procedure is illustrated in Figure (4).

Since the edge collapsing operation deletes a vertex from a mesh level, the mapping from this vertex to the corresponding mesh entity at the coarser level must be obtained, if the restriction and prolongation operator have to be computed. For example, for a deleted mesh vertex $M_2^0$ classified on model region, this is readily accomplished searching within the set of newly created mesh regions the one, $M_1^3$, that contains the vertex, i.e. $P(\mathbf{x}) \in M_1^3$, where $P(\mathbf{x})$ is the geometric location of $M_2^0$. Note that this operation involves only a local search and it is thus inexpensive. Moreover, since the mapping is computed "on the fly" at the end of the collapsing of each mesh edge, there is no need to resort to complicated data structures for performing the

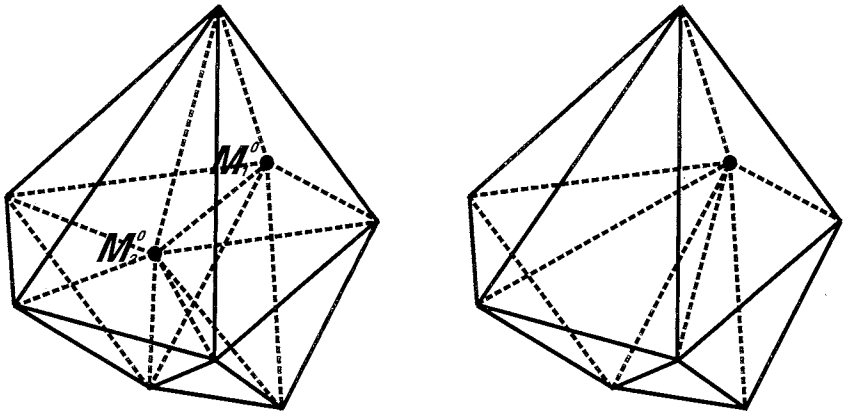

Figure 4: Derefinement by edge collapsing.

search, as it is case when independent multiple meshes are used.

Close to the model boundary, vertices which are deleted from a given mesh level might fall outside of all the regions of the mesh at the coarser level. This is due to the fact that the mesh domains defined by the two mesh levels are in general not the same, since they represent different geometric approximations of the geometric model. Once again, the geometric information and entity classification provides the natural way for resolving efficiently and accurately these domain mismatch problems.

The intergrid transfer operators are realized by storing for each region considered at a given level the list of vertices that map into it from the finer level, and for each of these vertices the corresponding region at the coarser level, together with the values of the projection coefficients for performing the intergrid transfers (for example, in a finite element context, these might be the values of the shape functions obtained through a parametric inversion process). Clearly, as regions are removed from a given mesh level by the coarsening procedure, the maps to and from them are updated in terms of the newly generated regions.

## Refinement

The refinement scheme adopted in this work is also edge based [4], in the sense that edges marked for refinement are split. The algorithm implements all possible subdivision patterns corresponding to all possible configurations of marked edges, to allow the maximum flexibility in how mesh refinement

is accomplished. A limit can be placed on the shortest edge to split, in order to avoid excessive refinement.

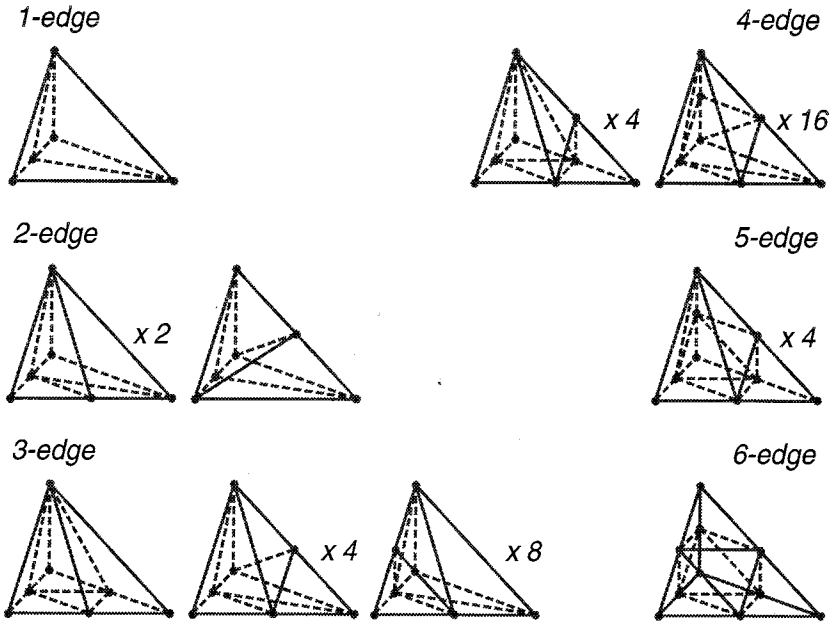


Figure 5: Subdivision patterns for refinement.

In the presence of curved geometries, newly generated vertices classified on model boundary must be properly placed on the true geometric boundary. This snapping procedure is critical, in the sense that it represents the mechanism through which refinement of a given mesh improves the geometric approximation that the mesh gives of the geometry. This operation relies on the interaction of the refinement procedure with the geometric modeler storing the geometric information, and the classification information of the mesh entities. The snapping of a refined vertex can produce invalid regions of negative volume or of poor quality. In this case, the retriangulation procedure explained in the following is applied and repeated until the vertex can be successfully snapped.

For supporting the computation of the restriction and prolongation operators, a double link is stored from the vertex to the edge and back, together with the value of the split location along the edge. This is realized "on the fly" during refinement of each marked edge. Clearly, not even a local search is needed in this case.

## Local Retriangulation

Local retriangulation algorithms are an important aspect of any automated mesh modification procedure, their goal being the control and the improvement of the quality of a mesh with respect to a given criterion. The optimization procedures implemented in this work are edge removal and multi–face removal, which do not change the number of vertices, and edge collapsing and splitting of one or more edges, faces or a region, which remove or add vertices [4][5].

Edge removal deletes an edge from a mesh by introducing one or more faces, depending on the number of regions surrounding the edge. Face or multi–face removal represents the dual operation, removing one or more faces by introducing a new edge. Figure (6) gives an example of these swaps for the simplest configuration, a 3 to 2 swap of the three elements $[M_1^o, M_2^o, M_\alpha^o, M_\beta^o]$, $[M_2^o, M_3^o, M_\alpha^o, M_\beta^o]$ and $[M_3^o, M_1^o, M_\alpha^o, M_\beta^o]$ surrounding edge $[M_\alpha^o, M_\beta^o]$. The 3 to 2 swap is performed by introducing a new face $[M_1^o, M_2^o, M_3^o]$ and by deleting the edge $[M_\alpha^o, M_\beta^o]$, yielding two new elements $[M_1^o, M_2^o, M_3^o, M_\alpha^o]$ and $[M_1^o, M_2^o, M_3^o, M_\beta^o]$. The reverse operation (2 to 3 swap) is given by deleting the face $[M_1^o, M_2^o, M_3^o]$ and introducing the edge $[M_\alpha^o, M_\beta^o]$. As previously explained, edge collapsing removes an edge by merging two vertices and removing all regions connected to that edge. The splitting procedures introduce new vertices on one or more edges of a region. The new configuration is then given by applying the corresponding refinement subdivision pattern.

The procedures are region–based, in the sense that the algorithm tries to improve all regions which violate a given mesh quality criterion. Given the impact that large dihedral angles usually have on the condition number of the discrete problem that approximates the set of PDE's to be solved, we typically use dihedral angles as optimization targets. As a first step, the dihedral angles of all elements considered for optimization are calculated. Each element violating a user–defined threshold value is put into a linear list. Depending on the configuration of each element in the list (number of dihedral angles above the threshold value, topological or geometrical constraints, etc.), a suitable subset of the above mentioned optimization procedures is applied to eliminate that element in favor of improved elements. The procedures might fail for a specific element if the resulting configuration is topologically or geometrically not valid, or if they lead to a degradation of the quality of any element involved in that local retriangulation. In this case, or if the element is improved but the largest dihedral angle is still above the threshold value, the element is considered for improvement in a second

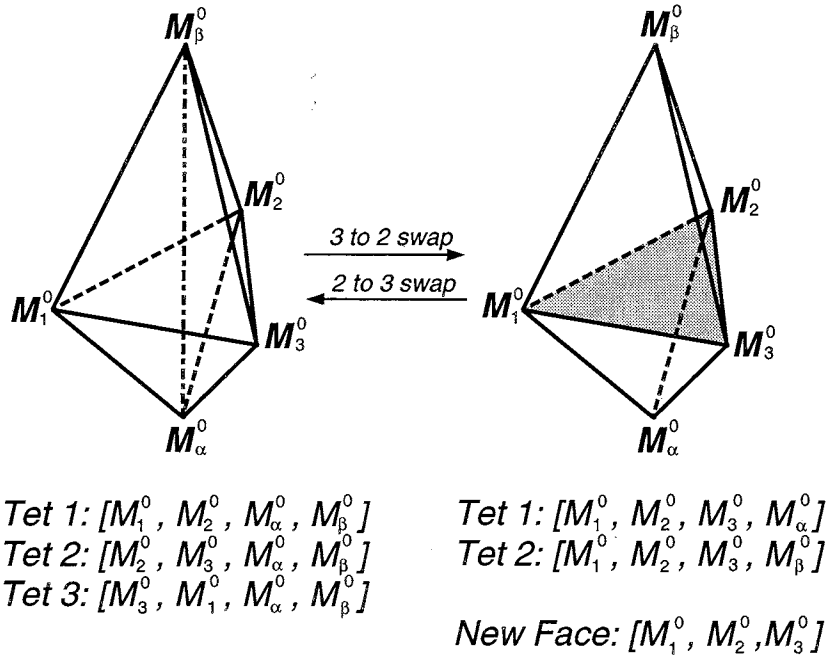


Figure 6: 3 to 2 and 2 to 3 swap

pass, after all elements have been processed. Since the neighborhood of the elements that failed in the first pass may have been modified, it is possible that they can be fixed in a second pass. The procedure is repeated until a given threshold value is reached or no further improvement can be achieved.

The local retriangulation algorithm is used to improve the meshes produced by the refinement and coarsening procedures. Since the refinement is an edge based operation that takes into account all possible subdivision patterns, refining an element is a localized procedure that does not affect the neighborhood of that element, and consequently the local retriangulation can be performed after the refinement procedure is completed. The situation is different when coarsening is considered. The coarsening procedure itself tends to give a mesh of poor quality, since collapsing of an edge has a strong — mostly negative — impact on the dihedral angles of the surrounding elements. To prevent loosing control of the mesh quality, especially when multiple coarsening steps are performed, it is necessary to introduce a threshold value to be satisfied by the largest dihedral angle in

each of the newly generated elements. However, this usually represents a strong constraint and prevents a large coarsening ratio. It is therefore advantageous to improve the neighborhood of a to–be–removed element before picking the next edge for collapsing. This can be done by sending a list of regions connected to the target vertex of the edge collapsing to the local retriangulation procedure, after each edge collapsing is performed. Such a locally improved mesh makes the next edge collapsing more likely to lead to acceptable elements.

Local procedures, such as the ones here considered can only lead to local optima with respect to the quality of the triangulation. Nonetheless, these tools have been proven to be valuable in controlling the degradation of the mesh quality during its adaptive modification. They also find application in the context of curved model boundaries, when snapping of newly created vertices can create invalid or poorly shaped regions. In this case, local retriangulation tools can be used for eliminating those regions that prevent snapping [4].

# Numerical Examples

In this section we show the capabilities of the developed automated procedures, and assess their performance characteristics. The goal is to demonstrate the flexibility of the approach in generating and handling multigrids on complex geometries.

## Turbine Stator

In this example, we analyze the performance of the mesh modification tools for realizing a hierarchy of unstructured meshes through refinement starting from a coarse base mesh.

The model considered is that of a cooled high–pressure turbine stator, and the initial mesh is depicted in Figure (7)*(a)*. The initial mesh for this structural mechanics problem was obtained with the Finite Octree mesh generator [13], and it is characterized by 2,383 tetrahedral elements. Since we are interested in showing the behavior of the algorithms when several refinement steps are performed, only the longest edge of each element was marked for refinement in order to reduce the sizes of the finer meshes, obtaining a refinement ratio approximately equal to 1:3[§]. Clearly, marking

[§]We define the refinement and coarsening ratios as the number of elements on mesh $i$ divided by the number of elements on the next finer or coarser mesh, $i-1$ or $i+1$.

| | Level 1 | | Level 2 | | Level 3 | | Level 4 | |
|---|---|---|---|---|---|---|---|---|
| | #tets | [%] | #tets | [%] | #tets | [%] | #tets | [%] |
| | Dihedral angles | | | | | | | |
| $\leq 145$ | 2298 | 96.43 | 6582 | 98.03 | 18726 | 98.94 | 56506 | 99.25 |
| 145...160 | 85 | 3.57 | 123 | 1.83 | 192 | 1.01 | 418 | 0.73 |
| 160...170 | 0 | 0.00 | 7 | 0.10 | 7 | 0.04 | 7 | 0.01 |
| $> 170$ | 0 | 0.00 | 2 | 0.03 | 2 | 0.01 | 0 | 0.00 |
| | Aspect ratio | | | | | | | |
| $\leq 2$ | 79 | 3.32 | 243 | 3.62 | 695 | 3.67 | 2293 | 4.03 |
| 2...4 | 879 | 36.89 | 2873 | 42.79 | 8766 | 46.31 | 27974 | 49.14 |
| 4...6 | 622 | 26.10 | 1960 | 29.19 | 5415 | 28.61 | 16330 | 28.68 |
| 6...10 | 552 | 23.16 | 1200 | 17.87 | 3178 | 16.79 | 8482 | 14.90 |
| 10...20 | 231 | 9.69 | 391 | 5.82 | 804 | 4.25 | 1726 | 3.03 |
| 20...40 | 19 | 0.80 | 36 | 0.54 | 51 | 0.27 | 94 | 0.17 |
| $> 40$ | 1 | 0.04 | 11 | 0.16 | 18 | 0.10 | 32 | 0.06 |

Table 1: Mesh statistics of the largest dihedral angle and the aspect ratio for the hierarchy of meshes of the turbine vane model.

all the edges would have lead to a ratio close to 1:8. Three finer mesh levels of 6,714, 18,927 and 56,931 elements were obtained in this way. After each refinement step the local retriangulation procedure was executed with a threshold value of $145^o$ for the largest dihedral angle. The mesh statistics for the largest dihedral angles and the aspect ratio of the four mesh levels are reported in Table (1).

Since the complex geometry of the problem implies strong constraints on the generation of the coarse mesh, there are 3.57% of the elements in the initial mesh which have angles above the threshold value of $145^o$. However, creating new vertices during refinement introduces more flexibility for mesh modifications without violating geometric constraints, yielding a reduced number of poorly shaped elements. In fact, the portion of elements with an angle above the threshold value of $145^o$ decreases from 3.57% to 0.74% after three refinements. The statement holds true also for the quality expressed in terms of the aspect ratio. Table (1) shows that the portion of elements with an aspect ratio larger than 10 decreases from 10.53% for the initial mesh to 3.26% for the finest mesh. It is interesting to point out that the proposed procedures are not only able to control the quality of the mesh, but they can also improve it.

| | Level 2 | | Level 3 | | Level 4 | |
|---|---|---|---|---|---|---|
| | #tets | [%] | #tets | [%] | #tets | [%] |
| | Dihedral angles | | | | | |
| $\leq 145$ | 6141 | 98.52 | 18113 | 99.60 | 57361 | 99.83 |
| 145...160 | 92 | 1.48 | 68 | 0.40 | 95 | 0.17 |
| 160...170 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| $> 170$ | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| | Aspect ratio | | | | | |
| $\leq 2$ | 725 | 11.63 | 2870 | 15.79 | 11519 | 20.05 |
| 2...4 | 3754 | 60.23 | 12727 | 70.00 | 40525 | 70.53 |
| 4...6 | 1237 | 19.84 | 2089 | 11.49 | 4676 | 8.14 |
| 6...10 | 454 | 7.28 | 466 | 2.56 | 692 | 1.20 |
| 10...20 | 58 | 9.31 | 29 | 0.16 | 43 | 0.07 |
| 20...40 | 5 | 0.08 | 0 | 0.00 | 1 | 0.00 |
| $> 40$ | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |

Table 2: Mesh statistics of the largest dihedral angle and the aspect ratio for the fine mesh of the turbine vane model created by the Finite Octree mesh generator.

For comparison, meshes denoted by a similar number of elements were created with the mesh generator. The statistics for these meshes are given in Table (2). In this case, the mesh quality is slightly superior. However, it should be remarked that Finite Octree includes a node point repositioning procedure to improve the mesh quality. Inclusion of similar algorithms in the proposed procedures would make the determination of the intergrid transfer operators significantly more complicated and expensive in terms of CPU time. In addition, it is interesting to point out that creating the levels 2, 3 and 4 with the mesh generator is twice as expensive in terms of CPU time as obtaining similar meshes using the refinement procedure with a local retriangulation after each refinement.

The impact of the optimization procedures on the final quality of the mesh can be appreciated by examining Table (3), where we give the mesh statistics for the levels of refinement obtained without any local retriangulation. 1.08% of the elements on level 2 have dihedral angles larger than $170^o$ and 0.5% have an aspect ratio of more than 40. The percent values increase to 2.00% and 1.05% respectively for the mesh on level 3 and to 2.88% and 1.70% on level 4. We point out that, when the optimization

| | Level 2 | | Level 3 | | Level 4 | |
|---|---|---|---|---|---|---|
| | #tets | [%] | #tets | [%] | #tets | [%] |
| | Dihedral angles | | | | | |
| $\leq 145$ | 6439 | 82.07 | 19439 | 77.56 | 60151 | 74.72 |
| 145...160 | 966 | 12.31 | 3610 | 14.40 | 12191 | 15.14 |
| 160...170 | 356 | 4.54 | 1513 | 6.04 | 5845 | 7.26 |
| $> 170$ | 85 | 1.08 | 502 | 2.00 | 2316 | 2.88 |
| | Aspect ratio | | | | | |
| $\leq 2$ | 209 | 2.66 | 619 | 2.47 | 2261 | 2.81 |
| 2...4 | 2584 | 32.93 | 8388 | 33.47 | 27123 | 33.69 |
| 4...6 | 2019 | 25.73 | 6099 | 24.33 | 18349 | 22.79 |
| 6...10 | 1781 | 22.70 | 5548 | 22.14 | 16678 | 20.72 |
| 10...20 | 1040 | 13.26 | 3275 | 13.07 | 11213 | 13.93 |
| 20...40 | 174 | 2.22 | 873 | 3.48 | 3507 | 4.36 |
| $> 40$ | 39 | 0.50 | 262 | 1.05 | 1372 | 1.70 |

Table 3: Mesh statistics of the largest dihedral angle and the aspect ratio for the mesh of the turbine vane model for three levels of refinement without local retriangulation.

procedures are employed, these values are substantially smaller: Table (1) shows that dihedral angles larger than $170^o$ are 0.03%, 0.01% and 0.0% at the various levels, and aspect ratios greater than 40 are 0.16%, 0.10% and 0.06%. A poor mesh quality in terms of dihedral angles and aspect ratio can increase the condition number of the discrete problem. Moreover, it gives significantly more iterations while solving the system of equations with an iterative solver and can decrease the quality of the solution, especially if derivatives of the primary variables are computed.

## Engine Nacelle

In this example we consider the model of an engine inlet with a center body. The initial CFD mesh of 119,861 tetrahedra was generated with the Finite Octree mesh generator (Figure (8)*(a)*). A 4–level multigrid was then generated by means of uniform coarsening marking all the edges for derefinement, obtaining coarser levels of 24,619, 6,477 and 1,819 tetrahedra, respectively. We set a limit of $160^o$ to the largest dihedral angle generated during coarsening, while we targeted for optimization all the regions with at least one angle above $145^o$. The coarser meshes are presented in Figure (8)*(b), (c)*

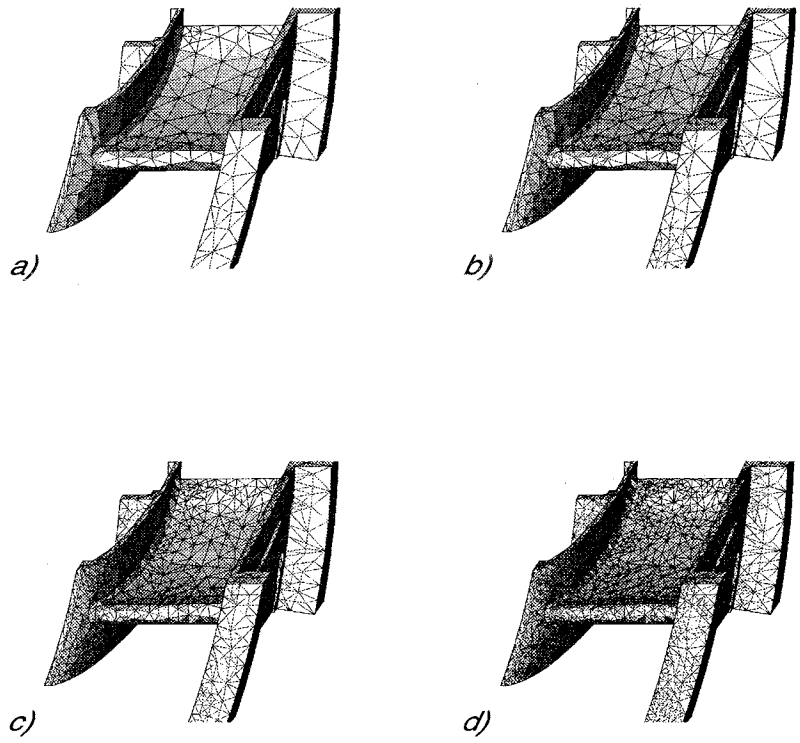


Figure 7: Hierarchy of successively finer meshes obtained by refinement for the turbine vane model. *(a)* Level 1: initial base mesh (2,383 tetrahedra); *(b)* Level 2: first refined mesh (6,714 tetrahedra); *(c)* Level 3: second refined mesh (18,927 tetrahedra); *(d)* Level 4: third refined mesh (56,931 tetrahedra).

| | Level 1 | | Level 2 | | Level 3 | | Level 4 | |
|---|---|---|---|---|---|---|---|---|
| | #tets | [%] | #tets | [%] | #tets | [%] | #tets | [%] |
| | Dihedral angles | | | | | | | |
| $\leq 145$ | 121922 | 99.99 | 30254 | 98.05 | 7201 | 97.00 | 1643 | 96.08 |
| 145...160 | 7 | 0.01 | 601 | 1.95 | 223 | 3.00 | 67 | 3.92 |
| 160...170 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| $> 170$ | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 |
| | Aspect ratio | | | | | | | |
| $\leq 2$ | 15148 | 12.42 | 1219 | 3.95 | 217 | 2.92 | 51 | 2.98 |
| 2...4 | 101891 | 83.57 | 17914 | 58.06 | 3686 | 49.65 | 733 | 42.87 |
| 4...6 | 4627 | 3.79 | 8818 | 28.58 | 2292 | 30.87 | 467 | 27.31 |
| 6...10 | 262 | 0.21 | 2758 | 8.94 | 1046 | 14.09 | 324 | 18.95 |
| 10...20 | 1 | 0.00 | 146 | 0.47 | 181 | 2.44 | 112 | 6.55 |
| 20...40 | 0 | 0.00 | 0 | 0.00 | 2 | 0.03 | 18 | 1.05 |
| $> 40$ | 0 | 0.00 | 0 | 0.00 | 0 | 0.00 | 5 | 0.29 |

Table 4: Mesh statistics of the largest dihedral angle and the aspect ratio for the hierarchy of meshes of the engine nacelle.

and *(d)*.

The mesh statistics of the largest dihedral angles and the aspect ratio for the four mesh levels are reported in Table (4). Analysis of the table clearly indicates that the proposed algorithms are able to deliver a large coarsening ratio with limited mesh quality degradation. As opposed to the refinement procedure, an improvement of the mesh quality cannot be expected since coarsening introduces constraints by deleting degrees of freedom. Nevertheless, the coarsening procedure was able to reduce the number of elements by a factor of approximately 74 in three levels. The final mesh has 1,710 elements, where 96% of the them have a largest dihedral angle below the threshold value of $145^o$. The usage of mesh (8)*(d)* for numerical computations is clearly limited, since it gives a poor discretization of the complex curved model, but the goal of this example is mainly to show that we are able to control the quality of the meshes even for a large coarsening ratio. The selection of the right coarse mesh for a specific problem depends strongly on the type of analysis to perform and it is not investigated here.

The effect of locally improving the mesh using the retriangulation procedures during coarsening was investigated. The initial mesh was derefined six times without optimization after each edge collapsing. The final coarse

mesh is denoted by 6,512 elements, and the corresponding mesh statistics are given in Table (5). In this case, for facilitating the coarsening process, the constraint on the largest dihedral angle was relaxed from $160^o$ to $175^o$. The ratio of coarsening is given in Figure (9). The diagram clearly indicates that coarsening without local retriangulation is not able to produce more than one or two coarser meshes. In fact, due to the increased number of badly shaped elements after each coarsening, constraints are introduced in the process and most of the attempted edge collapsings fail. In contrast, coarsening using local retriangulation after each edge collapsing is able to maintain a nearly constant, slightly increasing coarsening ratio. This gives the possibility to use the coarsening procedure to create a coarse mesh as coarse as it is necessary.

## Onera M6 Wing

In this section we show the ability of the proposed methodology to generate multiple mesh levels by means of local adaptation based on error estimates. The model used is that of an Onera M6 wing. The wing is characterized by an aspect ratio of 3.8, a leading edge sweep angle of $30^o$, and a taper ratio of 0.56. The airfoil section is an Onera D symmetric section with 10% maximum thickness–to–cord ratio.

We consider a steady flow problem characterized by an angle of attack $\alpha = 3.06^o$ and a value of $M = 0.8395$ for the freestream Mach number. In such conditions, the flow pattern around the wing is characterized by a complicated double–lambda shock on the upper surface of the wing with two triple points. The problem is modeled by means of the compressible Euler equations.

The initial mesh was generated with the Finite Octree mesh generator, and it is composed of 51,948 tetrahedrons. The meshing procedure made use of rectangular octants with an aspect ratio of 1:6, in order to generate stretched elements in the direction of the wing leading edge.

The multigrid procedure was then initiated. Since at the beginning no solution is available, a first coarse grid was obtained by uniform derefinement of the base mesh, obtaining a discretization denoted by 18,320 tetrahedrons. The maximum threshold on the allowable dihedral angles was set to $145^o$.

The solution was then advanced in time using an implicit linear multigrid algorithm. The procedure makes use of a constant–in–time discontinuous Galerkin time stepping scheme, which is denoted by excellent smoothing properties, while the linear system smoother is a block–diagonally preconditioned SSOR algorithm. Only 4 sweeps are performed at each grid level.

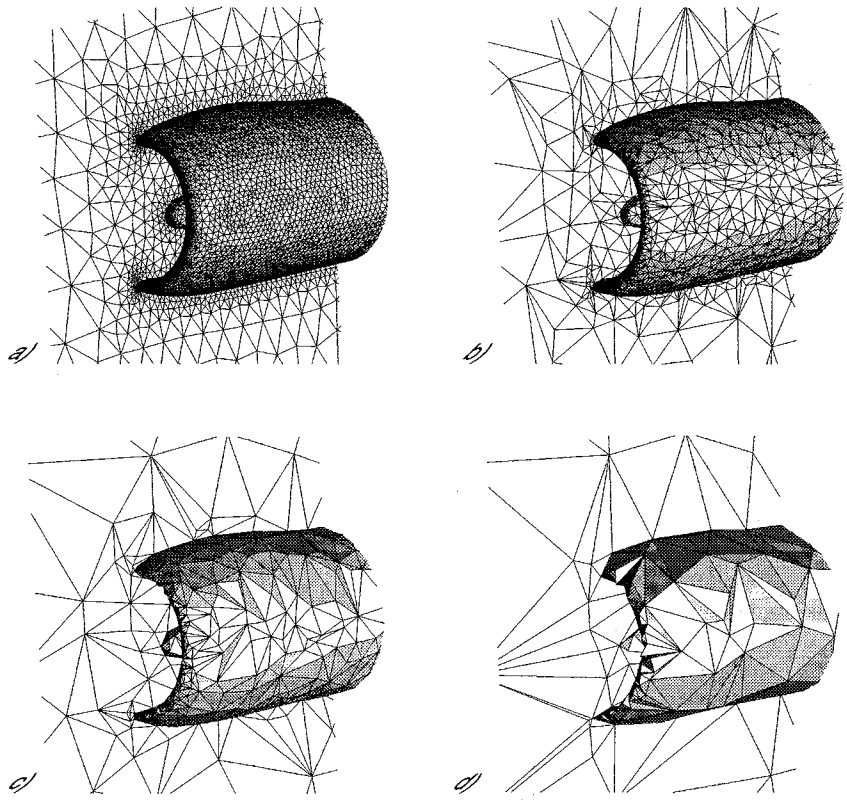


Figure 8: Hierarchy of successively coarser meshes obtained by uniform coarsening for the nacelle model. *(a)* Level 1: initial base mesh (119,861 tetrahedra); *(b)* Level 2: first derefined mesh (24,619 tetrahedra); *(c)* Level 3: second derefined mesh (6,477 tetrahedra); *(d)* Level 4: third derefined mesh (1,819 tetrahedra)

| | #tets | [%] |
|---|---|---|
| | Dihedral angles | |
| $\leq 145$ | 5717 | 87.79 |
| $145 \ldots 160$ | 743 | 11.41 |
| $160 \ldots 170$ | 52 | 0.80 |
| $> 170$ | 0 | 0.00 |
| | Aspect ratio | |
| $\leq 2$ | 170 | 2.61 |
| $2 \ldots 4$ | 2644 | 40.60 |
| $4 \ldots 6$ | 2145 | 32.94 |
| $6 \ldots 10$ | 1388 | 21.31 |
| $10 \ldots 20$ | 165 | 2.53 |
| $20 \ldots 40$ | 0 | 0.00 |
| $> 40$ | 0 | 0.00 |

Table 5: Mesh statistics of the largest dihedral angle and the aspect ratio for the mesh of the engine nacelle using coarsening without local retriangulation.

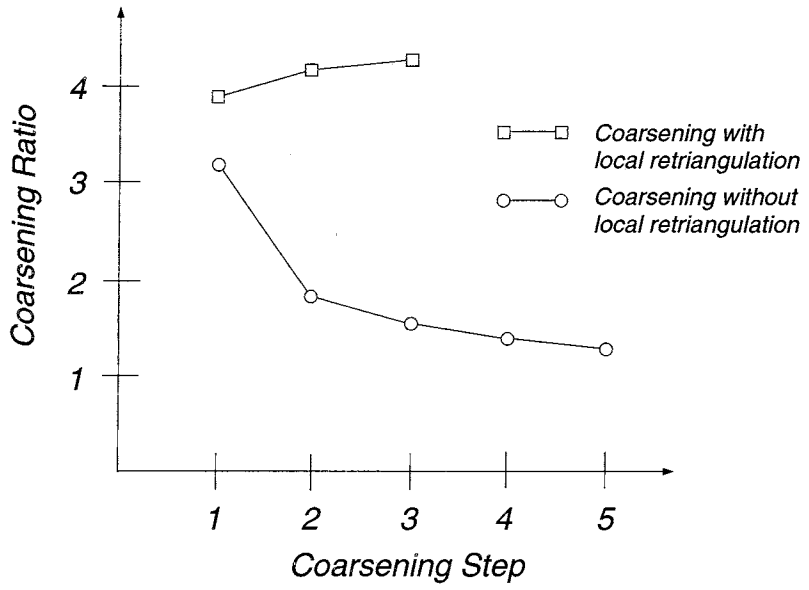


Figure 9: Coarsening ratio for coarsening with and without local retriangulation.

Local time stepping is used for speeding up convergence, using a CFL condition equal to 20 throughout the whole convergence history.

Once a converged solution was obtained, after a drop in the energy norm of the residual of 3.5 orders of magnitude, the mesh was adapted based on an error indicator computed with the scaled norm of the gradient of density. The refined mesh of 125,458 tetrahedrons is reported in Figure (10), and it was used as the fine level of a three–level multigrid, where the intermediate level is the initial mesh and the coarse grid is the one obtained by uniform coarsening. The solution was interpolated to the new fine mesh level during adaptation, and the Euler simulation was then restarted until reduction in the energy norm of the residual by four orders of magnitude. Table (6) shows the statistics for the refined mesh. It is interesting to note that, even with the stretched mesh in the direction normal to the gradients in the flow, large dihedral angles are extremely well controlled.

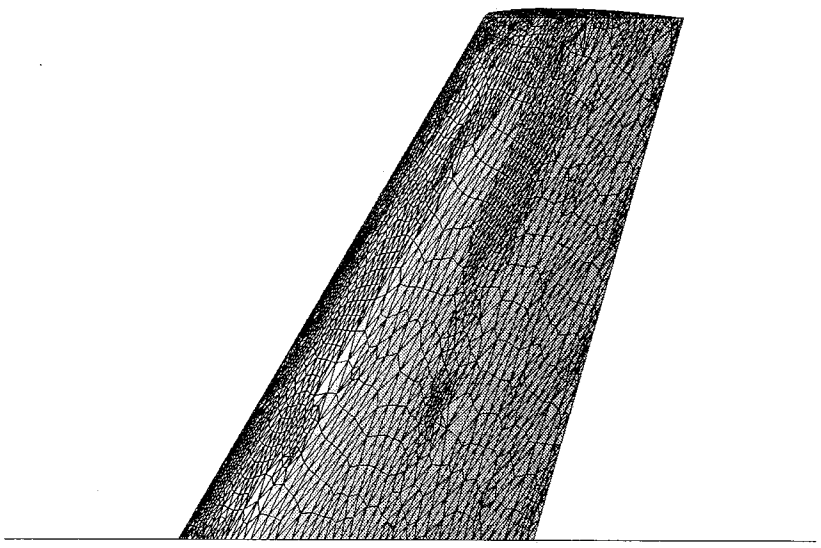

Figure 10: Adapted mesh for the transonic flow problem over the Onera M6 wing.

The three-level multigrid was then analyzed for determining the saving in memory occupation due to the unique storing of duplicated entities on multiple levels. The storing of vertices is reduced by 39%, of edges by 26%,

| | #tets | [%] |
|---|---|---|
| | Dihedral angles | |
| $\leq 145$ | 113784 | 98.56 |
| $145 \ldots 160$ | 1658 | 1.44 |
| $160 \ldots 170$ | 8 | 0.01 |
| $> 170$ | 0 | 0.00 |
| | Aspect ratio | |
| $\leq 2$ | 331 | 0.29 |
| $2 \ldots 4$ | 11858 | 10.27 |
| $4 \ldots 6$ | 33188 | 28.75 |
| $6 \ldots 10$ | 56818 | 49.21 |
| $10 \ldots 20$ | 12939 | 11.21 |
| $20 \ldots 40$ | 307 | 0.27 |
| $> 40$ | 9 | 0.01 |

Table 6: Mesh statistics of the largest dihedral angle and the aspect ratio for the adaptively refined mesh of the Onera M6 wing.

of faces by 22% and of regions by 20%. Indicating with $N_M^3$ the number of mesh regions in the grid, then the storage of an all tetrahedral mesh of linear elements is given by $(6 + 14 + 11 + 4)N_M^3 = 35N_M^3$, where the first term represents the contribution of regions, the second of faces, the third of edges and the fourth of vertices [1]. Considering the percentage of duplicated entities previously reported, this translates in a memory saving of 25% with respect to a full representation.

## Conclusions

A set of automated procedures that can be used for developing unstructured multigrid adaptive solutions of PDE's in domains of arbitrary complexity has been presented. The methodology employs edge-based mesh modification tools for generating the multiple mesh levels, and a topological mesh data structure for storing and handling the multiple grids.

The procedures are geometry–based: they understand the relationship of each mesh entity with the corresponding geometric model entity and they are directly interfaced with commercial geometric modelers that provide the necessary shape information. This aspect is a key ingredient towards automating the entire adaptive analysis process.

The mesh modification tools can handle refinement, coarsening and local retriangulation as a way to control the element quality. Coarsening does not depend on previous refinement, so that also the initial mesh level can be coarsened. It has been shown that achieving effective mesh quality control is critical, especially when performing multiple derefinements. The intergrid mappings are realized during adaptation, and consequently they do not have to rely on complicated search algorithms for being efficient.

The procedures have been tested on structural mechanics and CFD models, and the main features of the proposed approach have been verified. This methodology has recently been incorporated in a CFD code for the solution of the compressible Euler equations, with application to rotary wing problems. Testing is in progress for making the multigrid solver robust and reliable for all the flow conditions of interest.

## Acknowledgments

The authors would like to acknowledge the contribution of Mark W. Beall in the development of the single–mesh data structures, Hugues L. De Cougny for the adaptive procedures, Pascal J. Frey and Marcel K. Georges for the mesh optimization algorithms.

The first and third authors gratefully acknowledge the Army Research Office for funding this research through the ARO Rotorcraft Technology Center at Rensselaer Polytechnic Institute (DAAH04–93–G–0003, G. Anderson project monitor). The second author gratefully acknowledges the support for his visit at the Scientific Computation Research Center (SCOREC) granted by the Deutsche Forschungsgemeinschaft (DFG) and SCOREC. He also gratefully acknowledges Professor E. Stein for the kind support of his visit. The third author also acknowledges the support of the National Science Foundation through grant DMS-93-18184.

## References

[1] M.W. Beall and M.S. Shephard, 'Mesh data structures for advanced finite element applications', Scientific Computation Research Center, RPI, Troy, NY, submitted to *Int. J. Num. Meth. Eng.*

[2] A. Brandt, 'Multi–level adaptive solutions to boundary–value problems', *Math. Comp.*, **31**, 333–390 (1977).

[3] A. Brandt, 'Multilevel computations: review and recent developments', in S.F. McCormick, editor, Multigrid Methods, Lecture Notes in Pure and Applied Mathematics 110, Marcel Dekker Inc., New York, 1988.

[4] H.L. De Cougny and M.S. Shephard, 'Local modification tools for adaptive mesh enrichment and their parallelization', Scientific Computation Research Center, RPI, Troy, NY, submitted to *Comp. Meth. Appl. Mech. Eng.*

[5] B.E. De l'Isle and P.L. George, 'Optimization of tetrahedral meshes', in I. Babuska, J.E. Flaherty, J.E. Hopcroft, W.D. Henshaw, J.E. Oliger and T. Tezduyar, editors, Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, Springer–Verlag, New York, 1995.

[6] M. Mäntylä, *Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland, 1988.

[7] D.J. Mavriplis and A. Jameson, 'Multigrid solution of the two–dimensional Euler equations on unstructured triangular meshes', AIAA Paper 87–0353 (1987).

[8] D.J. Mavriplis, 'Three dimensional unstructured multigrid for the Euler equations', AIAA Paper 91–1549 (1991).

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

[10] V. Parthasarathy and Y. Kallinderis, 'Directional viscous multigrid using adaptive prismatic meshes', *AIAA J.*, **33**, 69–78 (1995).

[11] W.J. Schroeder and M.S. Shephard, 'On rigorous conditions for automatically generated finite element meshes', in J. Turner, J. Pegna and M. Wozny, editors, Product Modeling for Computer Aided Design and Manufacturing, pp. 267–281, North–Holland, 1991.

[12] M.S. Shephard, 'The specification of physical attribute information for engineering analysis', *Eng. with Comp.*, **4**, 145–155 (1988).

[13] M.S. Shephard and M.K. Georges, 'Automatic three–dimensional mesh generation by the Finite Octree Technique', *Int. J. Num. Meth. Eng.*, **32**, 709–749 (1991).