DEFORMATION OF UNSTRUCTURED VISCOUS GRIDS

Denis B. Kholodar,* Scott A. Morton,§ and Russell M. Cummings†
Department of Aeronautics, United States Air Force Academy
USAF Academy, CO 80840-6400

ABSTRACT

A mesh deformation algorithm for unstructured grids is presented. It is designed for high Reynolds number flow problems. Such grids are employed in aerodynamic and aeroelastic studies of wings or complete aircraft configurations in flows where the viscous effects are important. Given a surface deformation, the method efficiently recalculates new locations of high aspect ratio cells that make up the viscous layers of the grid and then deforms the inviscid part of the grid using an established method based on a torsional spring analogy technique. Results are presented for monitoring the deterioration of the quality of the grid during subsequent deformation steps for aeroelastic studies as well as to ensure the time efficiency of the method. Results for grid deformation of a 1.4 million cell AGARD 445.6 wing grid designed for flow at high Reynolds numbers due to typical deformations are also presented. Finally, a discussion of the parallelization performance and comparison of the running time of the mesh deformation algorithm to that used by the flow solver is made.

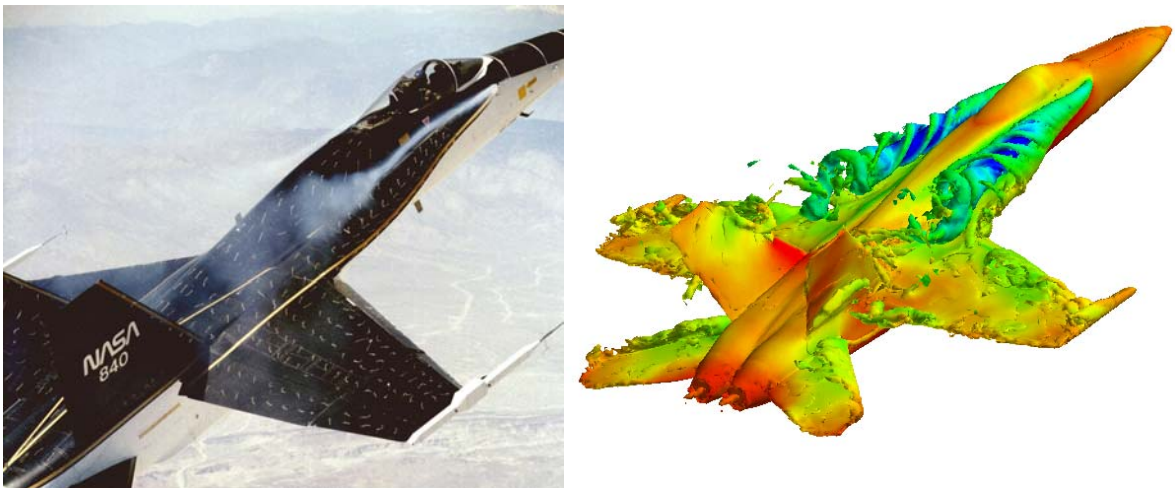
I. INTRODUCTION

ONE very challenging goal in the field of Computational Aeroelasticity (CA) is to be able to study fluid-structure interactions and predict response of flight vehicles in high Reynolds number flows. Numerical simulations of fluid flow in such flight regimes require employment of Computational Fluid Dynamics (CFD) grids that contain very thin, stretched (i.e. high aspect ratio) cells resolving the boundary layer around the body surfaces. Deformations of the surfaces require moving the CFD grid surrounding the body. Robust algorithms for moving an unstructured grid is an important and recent subject in the fields of CFD, CA, controls and others. Moving an unstructured grid that is designed for a high Reynolds number flow presents an even more challenging task. The development of an algorithm for this type of grid (referred viscous grids in the paper) is shown in the current work.

The complexity of a mesh deformation algorithm that a CA program uses depends on the complexity of the problem, grid size, computer resources, type of fluid and structural solvers that communicate to the grid, and various other factors. In particular, deformation attributes (large or small, translation, rotation or bodies in relative motion), topology of the grid (structured or unstructured) and finally presence of high aspect ratio cells suitable for viscous flows determines the difficulty of the problem. The first class of methods based on creation of a network of linear springs connecting all vertices in the grid was introduced in Ref. 1. This method worked well for many applications, but is limited to small deformations. Since then many advancements have been made including those that permit large deformations using the spring method (see, for example, Refs. 2-5). Most of the methods developed in these studies have been validated on inviscid grids. The presence of highly stretched cells in viscous grids with aspect ratios in the thousands makes the deformation problems even more challenging. That is why the past year saw a number of papers trying to address this problem. The authors of Ref. 6 considered the tetrahedral mesh as a deformable elastic solid, while moving the near-wall prism cells as a rigid body. They also designed coarsening and refinement techniques to alleviate the inevitable distortion of the grid during large degrees of motion. Ref. 7 proposes a selective solid-extension mesh moving technique governed by the equations of elasticity that allowed for limiting the distortion in thin layers and reduce the frequency of remeshing. In Ref. 8 the torsional spring method of the three-dimensional grid is derived from the two-dimensional torsional spring method differently than that in the Ref. 2 by "taking into account the dynamics of the three-dimensional object by analyzing the equations of volume and area for a tetrahedron and its faces and edges". [8]

---

* Research Engineer, Member AIAA.
§ Professor of Aeronautics, Associate Fellow AIAA.
† Professor of Aeronautics, Associate Fellow AIAA.



a) **b)**

Fig 1. F-18 Tail buffet problem: a) NASA F-18 High Angle of Attack Research Vehicle (HARV). Photo courtesy of NASA Dryden, and b) Computational Detached Eddy Simulation turbulence model.[9]

In many current CFD and CA studies viscous unstructured grids of large size are employed. For example, to correctly analyze the unsteady flowfield phenomena of Fig. 1a, vortex breakdown induced tail buffet, a grid designed for a Reynolds number of 13 million is required. This grid may have from 5 to 15 million cells for the half plane with up to half of them in the boundary layer. These boundary layer cells will have stretched cells with aspect ratios in the thousands to accurately capture the boundary layer velocity profiles. In a recent paper by the current authors on the F/A-18 tail buffet[9] problem, such a grid was necessary in order to apply various computational turbulence models. One of these models - the so called Detached Eddy Simulation approach - reproduced the experimentally observed unsteadiness of the tail loads at flight conditions. This was accomplished on a rigid grid but the goal is to perform a CA simulation with deforming meshes on the tails. The aspect ratio of near-wall cells in the grid used in Fig. 1b is close to 2000. This makes the grid deformation problem very challenging.

In this paper a deformation strategy is presented for such stretched viscous grids. The approach is a combination of two separate techniques: surface vector readjustment for moving grid points in the boundary layer, and the torsion springs technique[2] for moving the inviscid portion of the grid. A description of the method is presented below.

II. METHOD

Consider a typical three-dimensional unstructured viscous grid in the vicinity of a surface as shown in Fig 2. The viscous near-wall part of such a grid may have 8-20 layers of stretched cells. The grids used were generated by the grid generator VGRIDns developed at the NASA Langley Research Center. [10] The viscous cells are generated by the advancing layers method while the remaining volume has tetrahedra generated by the advancing front method. Designing a robust method to deform such a mesh for an arbitrary deformation of the surface is the objective of the present work. A deformation strategy should be developed with the particular kind of meshes in mind, and other research groups may have viscous unstructured meshes that are different. Therefore, we will try to emphasize the limiting factors of the current method.

The main concern is obviously to prohibit the inter-penetration of the neighboring cells as the grid is moved. The next challenge is to make sure that the grid quality does not deteriorate significantly after a number of steps, thus avoiding local regeneration of the grid during consecutive applications of the method. Among other challenges is the parallelization and time performance of the method.

The presented procedure consists of two mechanisms: one for the viscous boundary layer and the other for the inviscid part of the grid.

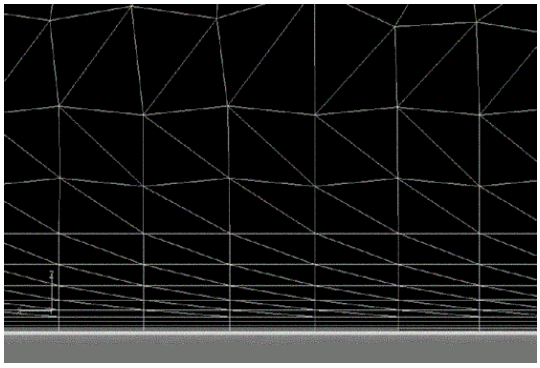


Fig 2. Example of a hybrid viscous unstructured grid.

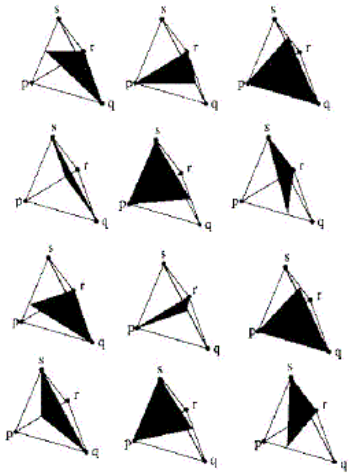


Fig 3. Insertion of triangles with torsional springs into a tetrahedron (courtesy to the authors of Ref. 2).

The inviscid part of the grid is deformed using a three-dimensional torsional spring analogy method proposed in Ref. 2. At the core of this method is the introduction of twelve triangles each having three torsional springs of varying stiffness into each computational tetrahedron to prevent it from collapsing during the deformation (see Fig. 3). The stiffness coefficients of the springs in the corners of the triangles are inversely proportional to the squared sine of the triangle angles, making it impossible for the triangle to have either 0 or 180 degree angles. This prevents the nodes that share an edge with each other from colliding (something that a linear spring method can also do), but also prevents the grid nodes from crossing any edge or faces. The method has been tested on many configurations and is shown to work in cases of dramatic deformations when traditional linear spring methods failed. For example, consider a simple nine node (18 tetrahedra) benchmark system shown in Fig. 4. The two-dimensional version of this system is considered in Ref. 11, while the three-dimensional system is used in Refs. 4 and 8. In the current paper the system in Fig. 4 is considered to test the grid deformation algorithm by bringing down the top node (number 1) by 99.5% to point A, which is just above the plane of the bottom nodes 2, 3, and 4 in Fig 4. Results follow in section III of the paper. A very good performance of the method in such cases of very large displacement was one of the reasons we considered using this method. Another reason was the performance of the method on the aeroelastic case of the AGARD 445.6 wing (see Fig. 5) as was demonstrated in the original paper.[2] One must note however that the unstructured AGARD 445.6 wing grid used in Ref. 2 is made of 128 thousand inviscid tetrahedra. For the linear spring analogy method to perform well the grid considered in Ref. 2 is too fine and the deformations are too big while the torsional analogy method worked well – this was all shown in Ref. 2. The AGARD 445.6 wing is considered in the current paper as well. Since the target problems (e.g. F-18 tail buffet) require fine viscous grids, the AGARD 445.6 wing considered here is finer than the one considered in Ref. 2. Another very important consideration is the parallel performance of the method. Grids of interest can be of many millions of cells. Thus, it becomes very important to have an efficient parallelization strategy. The torsional spring method consists of two parts - computation of the grid stiffness matrix and solution of a sparse linear system of equations. Both these tasks can be performed locally on all processors with limited communication time – this issue is addressed further in the parallel performance discussion. These were some of the reasons behind choosing a torsional spring analogy method for the grids of interest.

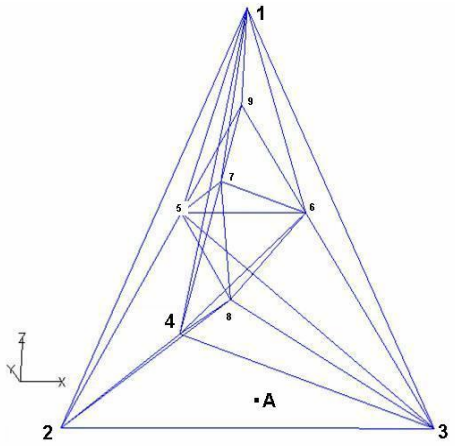


Fig 4. Benchmark nine node system.

Very often cells other than tetrahedra are generated in the volume of the grid, e.g. prisms or pyramids. In that case a division of such cells into three and two tetrahedra respectively proved to be a very simple geometric task that needs to be performed only once at the input of the grid information into the deformation module. Some unstructured grid generation codes keep the information about the tetrahedra that formed the hybrid cells, so in that case no division is necessary. It is also important to note that a flow solver will continue to use the hybrid cells as it only needs updated values of node coordinates from the mesh deformation module.

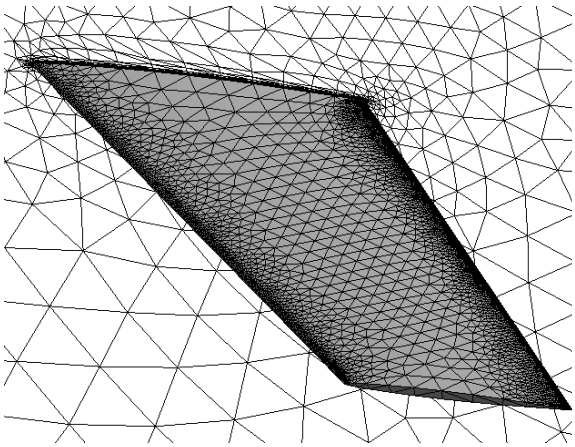


Fig 5. AGARD 445.6 wing.

There are two reasons why this (torsional spring analogy) method or similar deformation methods may fail to deform a grid designed for high Reynolds number flows, i.e. *viscous* grids. Firstly, the cells in the boundary viscous layers may be thinner than the deformation of the structure during a single deformation step, in which case the method loses its robustness in these layers. Secondly, even if the deformation does not exceed the size of the viscous cells, it is still too large. After a number of deformation steps the quality of the

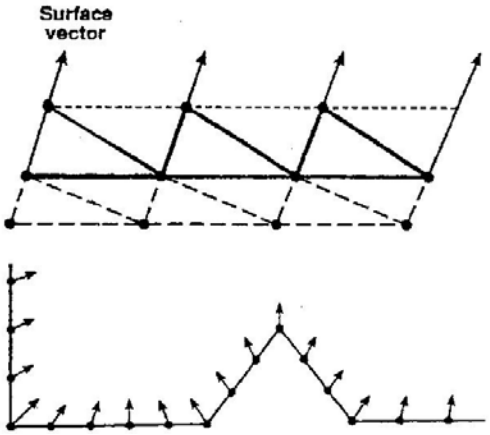


Fig 6. Surface vectors (courtesy to the author of Ref. 3).

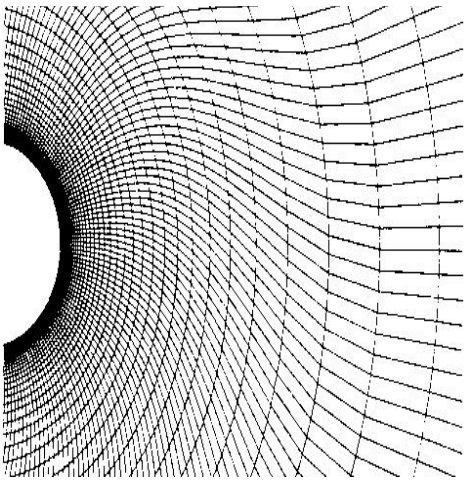


Fig 7. Deformed structured grid example. [15]

viscous cells deteriorates significantly which may result in either a creation of "negative" cells on the next mesh deformation step or flow solver problems. In the original development of the method for the two dimensional case[11] this restriction is noted. For example, citing Ref. 11: "Even though we are interested in flow problems where some boundaries may undergo a motion with large amplitude, we assume in the following derivation that the increment of mesh motion between two time-steps is sufficiently small to justify the assumption of small displacements and small rotations."

For the above reason, another deformation approach is used in the boundary layer region of the grid. This part of the grid is moved by first recomputing the surface vectors (see Fig. 6) of the deformed surface and then redistributing the grid points in the boundary layer in the direction of these vectors. This idea has proved to be a simple and very efficient method. No generation of new nodes is required during this process since the grid structure remains intact. Also, no strict requirement on the size of the surface deformation is present in this method. This approach can be viewed as very similar to some in the structured grid community[15] – the deformation of the surface is propagating along the "nodal lines" that originate on the surface (see Fig. 7). The nodal lines remain normal to the surface in the boundary layer in these methods. In the type of grids that are considered here, the cell edges near the surface boundary (Fig. 2) indeed resemble the nodal lines of a structured grid. The main difficulty then is to find the consecutive nodes of the nodal line of the unstructured volume grid, which becomes more difficult the further from the deforming surface one goes. However, it can be done using the particularities of a grid, e.g. the fact that the prisms in the viscous layers are stacked on top of each other. If no prisms are created by the grid generator, then one can use the fact that the next node in the nodal line is the closest of all nearby nodes. Once the nodal lines are known, a popular structured grid deformation approach such as algebraic method (see, for example, Ref. 15) can be applied to deform the viscous part of the grid. Also note that because these nodal lines are in layers where the aspect ratio of the cells is very high, the distance between the nodes along the nodal line is smaller than the distance between the nodal lines themselves. Therefore, the method for moving or deforming this part of the grid can have lighter restrictions. In this particular work the method to move and redistribute the nodal lines was done as follows. The viscous layers were originally produced by the process of advancing layers along normal projections from the surface triangular faces. Since each surface node has multiple faces, the normal for the advancing layer is a weighted average. To deform these viscous

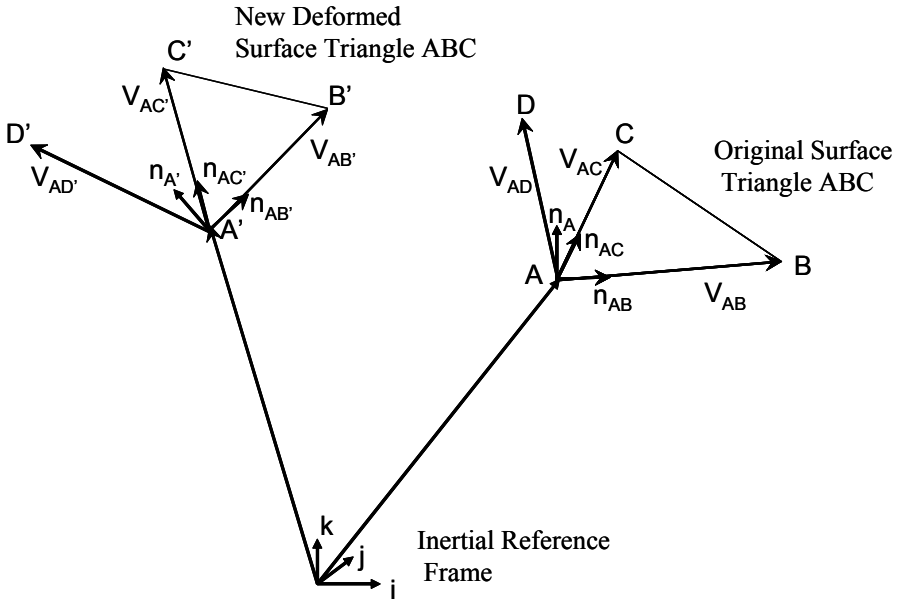


Fig 8. Viscous layer deformation strategy.

layers, one of the faces using the surface node of interest is arbitrarily chosen as the starting point for computing a set of basis vectors (see Fig. 8). Follow-on methods may modify this to include the effect the surrounding faces on the surface to smooth the normal vector. With the current method for a surface triangle **ABC**, two surface vectors are formed from points **A** to **B** ($\mathbf{V_{AB}}$) and points **A** to **C** ($\mathbf{V_{AC}}$). A normal vector is formed by normalizing the cross product of $\mathbf{V_{AB}}$ and $\mathbf{V_{AC}}$. This normal vector $\mathbf{n_A}$ along with vectors formed by normalizing $\mathbf{V_{AB}}$ and $\mathbf{V_{AC}}$ ($\mathbf{n_{AB}}$ and $\mathbf{n_{AC}}$) form a set of basis vectors. The grid generation process produced prism cells above this surface triangle with one of the nodes of each prism approximately along this normal vector $\mathbf{n_A}$. These nodes make nodal lines in the current method. Vectors from point **A** on the surface to all the nodes of the nodal line above it (e.g. node **D** in Fig. 8) are expressed in terms of the new basis vectors through the use of dot products between the vector $\mathbf{V_{AD}}$ and the basis set, $\mathbf{n_A}$, $\mathbf{n_{AB}}$ and $\mathbf{n_{AC}}$. After surface deformation occurs, the new surface triangle may have changed shape, been rotated, and translated to new coordinates **A'**, **B'**, and **C'** as shown in Fig. 8. Using the new primed coordinates and following the same procedure as above, we create a new set of basis vectors $\mathbf{n_{A'}}$, $\mathbf{n_{AB'}}$, and $\mathbf{n_{AC'}}$. The new coordinate **D'** is produced by applying the components of $\mathbf{V_{AD}}$ expressed in $\mathbf{n_A}$, $\mathbf{n_{AB}}$ and $\mathbf{n_{AC}}$ directions to the new basis vectors ($\mathbf{n_{A'}}$, $\mathbf{n_{AB'}}$ and $\mathbf{n_{AC'}}$) . This method is very fast, maintains layers suitable for viscous boundary layer calculations, and will maintain good grid quality (i.e. no nodal lines crossing to produce negative cells) as long as the aspect ratio of the cells is fairly high.

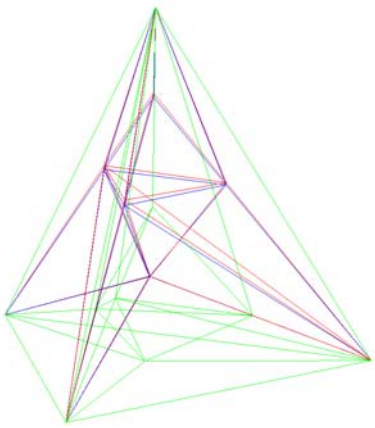


Fig 9. Original (blue) and after deformation cycle meshes: 90% (red) and 99.5% (green).

The contribution of this work is to combine these two methods (for the inviscid and viscous parts of the grid), devise an interface strategy between them, and apply them to a realistic configuration in a high Reynolds number flow problem.

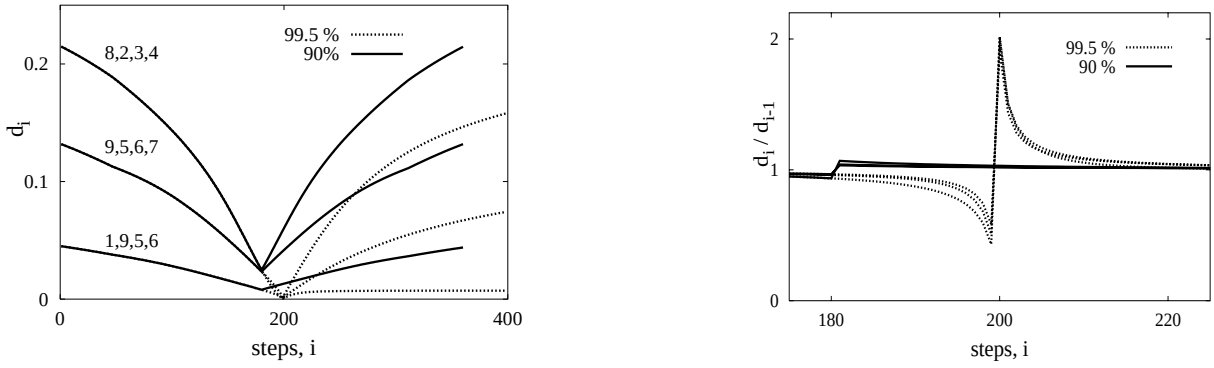
III. Results

A. Benchmark Problem

The following results describe the performance of the torsional spring grid deformation method over many steps. A good example of when this is important is an aeroelastic problem of limit cycle oscillations where a wing surface deforms back and forth in many cycles over time. Consider again a nine node mesh shown in Fig. 4. This very simple system is used to test the performance of the method.

The **effect of deformation range** is studied first. The top Node 1 is moved downward toward the point A located in the plane of the fixed nodes 2, 3, and 4 (see Fig. 4) with a step of Δz=0.5%. So if initially the Node 1 vertical coordinate was z=1, then in the first case of 180 steps it became z=0.10 (a 90% case), and in the second case in additional 19 steps (99.5% case) it becomes z=0.05. In both cases, Node 1 is then moved back to its original position in the same manner. A picture of the systems after the cycle of deformation (red and green color) along with the original system (blue) is shown in Fig. 9.

Obviously if these deformations caused no damage to a grid, its picture should coincide with the original. In this case the grid



a) **b)**

Fig 10. Effect of the size of deformation range: a) History of inscribed sphere diameter; b) History of diameter ratio.

that was deformed 90% and back is very close to its original state. The additional 19 steps of deformation that almost flattened the second grid caused some significant damage to the grid. A simple but effective way to monitor these effects is by observing the diameter of the sphere inscribed into a tetrahedron, specifically a ratio of the diameters before and after a step of deformation. A calculation of the diameter of a sphere inscribed into a tetrahedron is a very inexpensive additional calculation in the case when a volume of a tetrahedron is calculated, which is done by the finite volume CFD scheme of the flow solver and is also used for the purpose of checking if negative volumes appeared in the deformation process. Fig. 10a, shows a "time" history of such diameters for three out of eighteen tetrahedra present in this system (the four nodes of each tetrahedron are marked on Fig. 10a), while Fig. 10b shows a ratio of these diameters, d_i / d_{i-1}, before and after each step during the interval from step 180 to 220. This ratio deviates sharply from the unity during the steps when Node 1 reaches the bottom, z=0.05. Thus a high deformation rate (e.g. larger than $|d_i - d_{i-1}| / d_{i-1} = 0.1$) leads to significant deteriorating changes in the grid. This is a very important conclusion, and its consequences reappear when examining the case of the AGARD 445.6 wing.

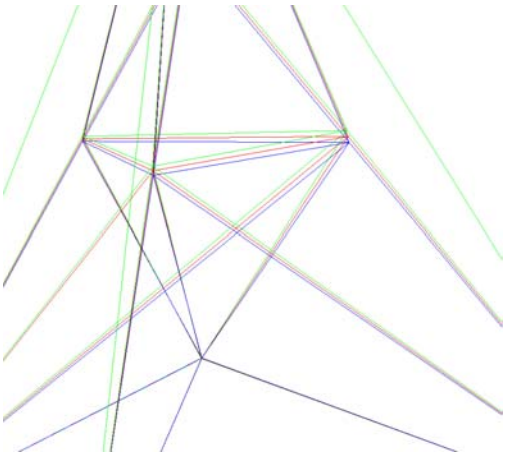
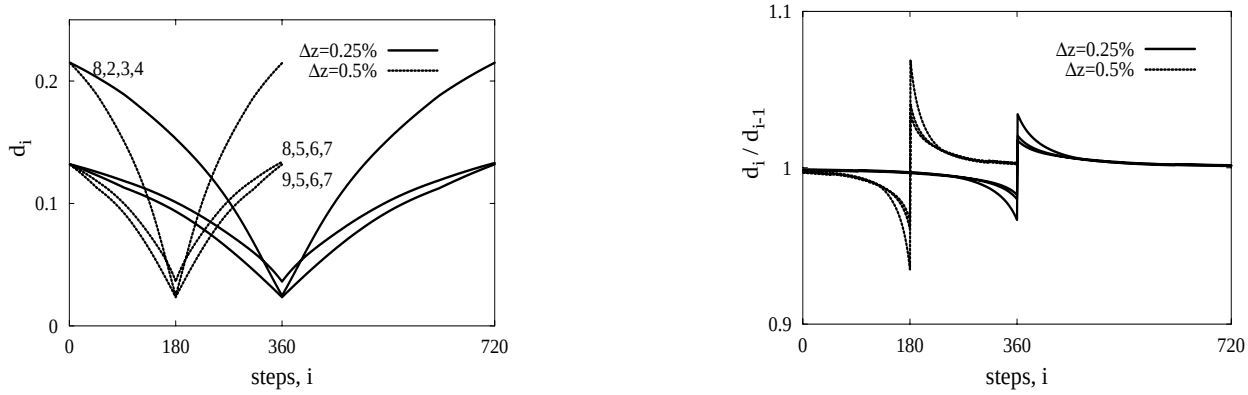


Fig 11. Original (blue) and after deformation cycle meshes: small step (red) and large step (green).

A relative **effect of deformation step size** is considered next (see Fig. 11). This time Node 1 is brought to z=0.1 and back with steps of two different sizes – a step Δz=0.005 (green color) and Δz=0.0025 (red). In this case one can see only a marginal edge shift in the deformed tetrahedra with the larger difference present in the case of larger Δz step comparing to the original one.

Fig. 12a shows time histories of the diameters for three inscribed spheres, while Fig. 12b shows a ratio of the diameters before and after each step during the process. Note that this ratio, d_i / d_{i-1}, in this case stays very close to 1 and, of course, the largest deviation from 1 is higher for the case of the larger deformation step size, Δz. When the deformation size used on every step is large, the final displacement of the grid nodes when the system returns to the original position is large as well.

Thus, it was established that d_i / d_{i-1} is an inexpensive deformation measure that allows monitoring the changes in the element quality. To keep the original grid quality during deformations it is desirable to keep this value as close to unity as possible for all cells on all time-steps. The exact value of an "acceptable" ratio depends on the problem, cell location, and many other factors and should be programmed by the user. A detailed description of a similar but more involved and strict deformation measure is available in Ref 6.



a) **b)**

Fig 12. Effect of the size of deformation step: a) History of inscribed sphere diameter; b) History of diameter ratio.

B. AGARD 445.6 Wing Problem

Next, let us return to the AGARD 445.6 wing case (see Fig. 5). This configuration is used by many researchers for validation of aeroelastic models because of the number of experimental and computational results available for the transonic flutter problem of this wing. A number of grids in the range from 1.5 million to 4.7 million tetrahedra were created using the software package VGRIDns.[10] For example, in the case of 1.5 million tetrahedra, half of the tetrahedra are in the inviscid portion, while the remaining tetrahedra are in the viscous portion and can be combined into layers of approximately 250 thousand prisms. The whole grid has 250 thousand nodes, 15 thousand of them being on the wing surface (that surface mesh is shown in Fig. 5). For the wing root chord of 1, the smallest inscribed diameter in a tetrahedron near the wing surface is 8×10^{-6}; the aspect ratio for this tetrahedron is equal to 1800. Suppose the wing is undergoing a deformation along its first bending mode of the amplitude of 1% compared to the size of its root chord. In this case, keeping the cell deformation smaller than a cell size requires more than a thousand time steps. However, the above described deformation ratio, $\left|d_i - d_{i-1}\right| / d_{i-1}$, on each time-step in that case is on the order of unity for the boundary tetrahedra. As has been already discussed, this leads to the deterioration of the grid quality within a few steps. Thus, one needs to perform many more thousand steps to deform the grid with a smaller deformation increment size. Meanwhile using the viscous deformer (i.e. redistributing the nodes along the "nodal lines" in the viscous part of the grid during deformations) leaves only remaining inviscid tetrahedra to deform, and the smallest inscribed diameter in that part is 3×10^{-4} (an improvement by a factor of almost 40). Shown in Fig 13 are five neighboring nodal lines that penetrate 15 viscous layers of this grid in the most deformed place of the wing during the first bending mode deformation, i.e. the corner of the trailing edge and tip. The initial nodal lines when the wing is not deformed are indicated by the red color. The same nodal lines at 10% deformation (based on the root chord size) are indicated by blue. Finally, the same nodal lines the 20% deformation are indicated by yellow. The surface vectors were recomputed in both cases and the nodes redistributed along the nodal lines without any intersection of nodal lines. These calculations require an insignificant amount of time compared to the inviscid cell deformations. The remaining inviscid part of the grid can now be deformed. For example, the deformation along the first bending mode of the amplitude of the 1% of the root chord size can be achieved in five hundred steps with the deformation ratio of a most deformed tetrahedron on each step never exceeding $\left|d_i - d_{i-1}\right| / d_{i-1} = 2\times10^{-2}$. The average deformation ratio in the most deformed layer of the inviscid terahedra in this case does not exceed 1×10^{-4} on each step. Admittedly, this is still rather a large number of steps for a deformation of this magnitude, yet the grid is deformed and the grid quality is preserved allowing for consecutive cycles of the grid motion without a need to regenerate invalid areas of the grid. As discussed further below, the time performance of the combined viscous-inviscid grid deformation method is good. Thus, even though a large number of steps is required to maintain the quality of the grid during deformation, the method performance is acceptable.

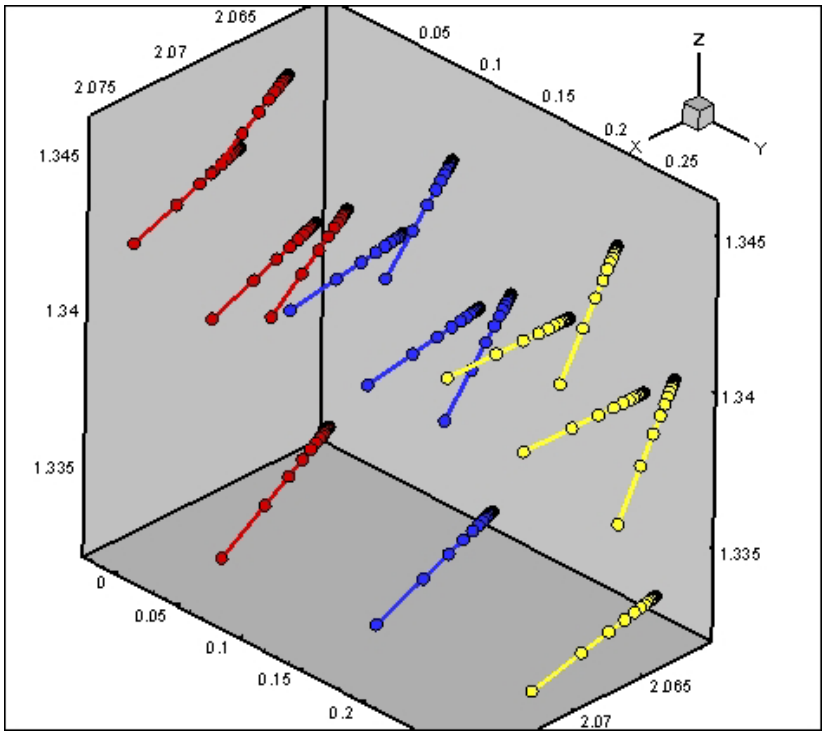
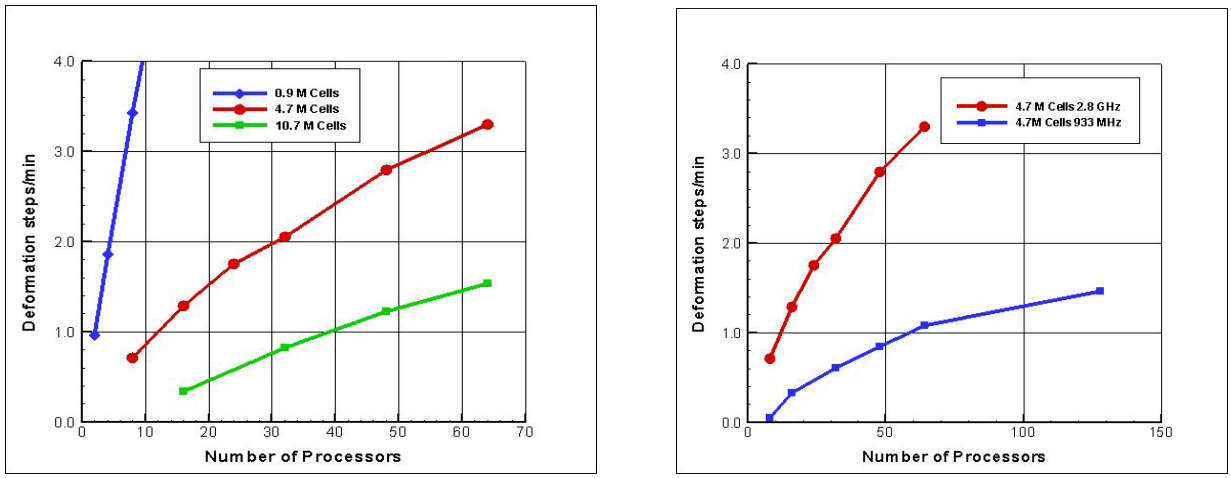


Fig 13. AGARD 445.6 Wing Viscous Nodal Line Deformation.

C. Parallel Performance

As has been already mentioned, the two main tasks of the spring methods are: the stiffness matrix computation and solving the large linear system of equations. A few details of the process are discussed here. For the grids of interest, the stiffness matrix is very large (despite being sparse, so only nonzero elements are computed and stored). Storing it as a whole on each processor is often not possible due to memory requirements. To compute the matrix, it is necessary to divide the entire grid between the available processors. So each processor has a certain number of tetrahedra to work with. On the other hand, solving the

system requires a division of nodes between processors (or to be precise, degrees-of-freedom that each node has). The optimal division of the grid by terahedra does not usually contain an optimal division by nodes. Numerous factors contribute to this fact. The node and tetrahedron numbering in the grid is not done in a very uniform fashion during the generation of the grid. A number of boundary nodes can vary between processors, thus affecting the dimension of the part of the system that each processor solves. For a given node to have its stiffness coefficients computed correctly, all tetrahedra with that node must be considered. For many nodes, this requires either passing some data between processors that have the participating tetrahedra or computation of overlapping terahedra regions, which is faster. Yet depending on the division and numbering of nodes and tetrahedra these overlapping regions can become very large especially as the number of processors grows. Once the viscous nodes were eliminated from the spring method region, the METIS[12] domain decomposition library was used to divide the grid into nearly equal tetrahedron regions. METIS division can be performed with an option of keeping the boundaries between the regions at minimum, which allows having smaller overlapping regions. METIS C language routines can be linked to the code or used as a stand alone module to divide the grid between the processors before the deformation code is loaded. To solve the system of equations in parallel, the AZTEC[13] library was used. AZTEC is an iterative parallel solver that can use a number of methods and preconditioners. AZTEC C language library was linked to the torsional spring Fortran 90 code. The Message Passing Interface (MPI) library is used to pass information between processors in the code (including AZTEC). Shown in Fig. 14a are the results of the torsional spring deformation code parallel performance for three different size grids on a Pentium IV, 2.8 GHz cluster. The number of cells in Fig. 14a is the number of the tetrahedra that were deformed using the torsional spring analogy method. Using the above described viscous layer deformer significantly reduces (at least by half) the number of cells that have to be deformed using the torsional spring analogy method. For the considered range of number of processors, each curve deviates slightly from the initial parallel performance as the number of processors is increased. This is because the overall share of the overlapping regions for matrix computations and the amount of data passed between processors for solving a linear system of equations increases as the number of processors is increased. This decline in performance is clearly seen in Fig. 14b, where on one of the platforms the number of processors used varies from 8 to 128.



a) **b)**

Fig 14. Parallel Performance: a) Grids of different size on a cluster; b) Same grid on two clusters.

D. Computation Performance Comparing to Flow Solver

Next let us compare the time that a torsional spring analogy mesh deformation code takes compared to the flow solver. In Ref. 2, the three-dimensional torsional spring analogy method took 52% of the total time of the AGARD 445.6 wing aeroelastic simulations on a 128 thousand tetrahedra grid. Thus, the grid deformation time was at least equal to the flow solver time. In the current work on 32 processors (see Fig. 14a) it takes roughly **30** sec to handle the mesh deformation of the 4.7 million AGARD 445.6 wing tetrahedra grid with the torsional spring analogy method (without using a boundary layer deformer). The flow solver that is used in our numerical simulations, such as in the tail buffet calculations of Ref. 9, is the commercial version of *Cobalt* developed by Cobalt Solutions.[14] It is a parallel, implicit, unstructured Euler/Navier-Stokes flow solver. In a typical run a time step with three subiterations on the same 32 processors takes the *Cobalt* flow solver about **60** sec. Note that combining the tetrahedra in the viscous layers into prisms the flow solver operates with only 3.6 million cells. Whether there is a need to update the grid on subiterations is not yet determined. In the best case scenario, when a grid doesn't have to be deformed during the subiterations, one obtains **1/2** ratio of the grid deformation module time to the flow solver time. However, after passing the 15 viscous layers present in this grid to the viscous deformer, the torsional spring method needs to deform only the remaining 1.6 million tetrahedra (which is about one-third of the initial amount of tetrahedra). Thus the ratio of

the consumed time of the current grid deformation module to the flow solver becomes **1/6** in the best case scenario or **4/6** in the worst case scenario, when a mesh update is necessary on each subiteration of the flow solver. Also, it must be noted that of the two operations in the torsional spring analogy deformation method, the construction of the stiffness matrix consumes significantly more time compared to the time it takes for the linear system of equations to converge. The exact solution of these equations is not at all necessary – they are equations of an imaginary spring system that reacts to given surface deformations to move the mesh. Thus, for example, within five to fifteen Gauss-Seidel iterations a linear system of equations solver converges to a sufficient precision for the value of updated mesh node coordinates. That time is about 20% of the time that it takes to build the stiffness matrix. However, the construction of the matrix is done with the previous coordinates of the grid nodes. Thus, one can be computing the stiffness matrix on mesh deforming processors at the same time the flow and structural solvers update the new locations of aerodynamic surfaces on the other processors. Therefore, it is very plausible that separate processors are running a mesh deformation module concurrently with the flow and structural solver. The time that the aeroelastic calculations are slowed by the mesh update reduces dramatically in this parallel environment..

IV. CONCLUSIONS

In this paper we presented a mesh deformation method for unstructured computational fluid dynamic grids that are employed in aerodynamic and aeroelastic studies of wings or complete aircraft configurations submerged in a high Reynolds number flow. Given a surface deformation, the method efficiently recalculates new locations of high aspect ratio cells that make the viscous grid layer and deforms the inviscid part of the grid using an established method of insertion of torsion springs in the cells. A means for monitoring of the mesh deformation with respect to detecting the changes in quality of the grid during deformation is also considered. Grid deformation results are discussed for a benchmark system as well as for a larger real application type system (a viscous AGARD 445.6 wing grid) under the deformations exhibited in experiments. Discussion of the parallel performance and comparison of the running time of the mesh deformation code to that used by the flow solver is made. The method showed overall promising results. The next step is to validate the method on other configurations. Finally, the ultimate performance assessment of this strategy will be revealed when the method is coupled with the flow and structural solvers in the aeroelastic simulations.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Christoph Degand[2] for sharing his experience with the torsional spring analogy method and Dr. Ray Tuminaro of Sandia National Laboratories for help with the AZTEC package. Finally, the authors would like to express their gratitude for the support of this project by Dr. John Schmisseur of AFOSR.

REFERENCES

[1] Batina, J. T., "Unsteady Euler Spring Method for Three-Dimensional Unstructured Dynamic Meshes," AIAA Paper 89-0150, Jan. 1989.

[2] Degand, C., and Farhat, C., "A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes," Computers and Structures, Vol. 80, pp. 305-316, 2002.

[3] Pirzadeh, S. Z., "An Adaptive Unstructured Grid Method by Grid Subdivision, Local Remeshing, and Grid Movement," AIAA Paper 99-3255, Jun.-July 1999.

[4] Murayama, M., Nakahashi, K., and Matsushima, K., "Unstructured Dynamic Mesh for Large Movement and Deformation," AIAA Paper 02-0122, Jan. 2002.

[5] Burg, C. O. E., Sreenivas, K, Hyams, D. G., and Mitchell, B., "Unstructured Nonlinear Free Surface Flow Solutions: Validation and Verification", AIAA Paper 02-2977, June 2002.

[6] Cavallo, P. A., Sinha, N., and Feldman, G. M., "Parallel Unstructured Mesh Adaptation for Transient Moving Body and Aeropropulsive Applications," AIAA Paper 04-1057, 42nd AIAA Aerospace Sciences Meeting, Reno, Jan. 2004.

[7] Stein, K., Tezduyar, T. E., and Benney, R., "Automatic mesh update with the solid-extension mesh moving technique", Comput. Methods Appl. Mech. Engrg. 139, pp. 2019-2032 , 2004.

[8] Burg, C. O. E., "A Robust Unstructured Grid Movement Strategy Using Three-Dimensional Torsional Springs", AIAA Paper 04-2529, 34th AIAA Fluid Dynamics Conference, Portland, June-July 2004.

[9] Morton, S. A., Cummings, R. M., and Kholodar, D. B., "High Resolution Turbulence Treatment of F/A-18 Tail Buffet", AIAA Paper 04-1676, Apr. 2004.

[10] Pirzadeh, S. Z., "Unstructured Grid Generation for Complex 3D High-Lift Configurations", AIAA Paper 01-5557, Oct. 1999.

[11] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional Springs for Two-Dimensional Unstructured Fluid Meshes", Comput. Methods Appl. Mech. Engrg., 163, pp. 231-245, 1998.

[12] Karypis, G., and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", SIAM Journal of Scientific Computing, Vol. 20, No. 1, pp. 359-392, 1999.

[13] Tuminaro, R. S., Heroux, M., Hutchinson, S. A., and Shadid, J. N., "Official Aztec User's Guide", Dec. 1999.

[14] Strang, W.Z., Tomaro, R.F., and Grismer, M.J., "The Defining Methods of Cobalt: A Parallel, Implicit, Unstructured Euler/Navier-Stokes Flow Solver," AIAA Paper 99-0786, Jan. 1999.

[15] Melville, R., and Morton, S., "Fully Implicit Aeroelasticity on Overset Grid Systems," AIAA Paper 98-0521, 36th AIAA Aerospace Sciences Meeting, Reno, Jan. 1998.