ANALYSIS OF MOVING MESHES ON WENO SCHEMES PERFORMED BY A TOPOLOGICAL DATA STRUCTURE

Fernanda Paula Barbosa, fernandapaulab@gmail.com
Antonio Castelo Filho, castelo@icmc.usp.br
José Alberto Cuminato, jacumina@icmc.usp.br
Instituto de Ciências Matemáticas e de Computação, USP- ICMC, São Carlos, SP, 13566-590, Brazil

João Luiz F. Azevedo, azevedo@iae.cta.br
Instituto de Aeronáutica e Espaço, CTA - IAE - ALA, São José dos Campos, SP, 12228-903, Brazil

Abstract. *The present paper addresses the adaptation of high order of accuracy weighted essentially non oscillatory (WENO) schemes when using unstructured meshes that are deformed thought time. WENO schemes are able to provide high order of accuracy on general unstructured grids. However, computational costs tend to rise quite sharply as the order of accuracy is increased due to the obvious increase in the size of the computational molecule and, at least partially, due to the inherent additional search work required to perform calculations with such increased molecule in unstructured grids. Moreover, WENO schemes are characterized by adaptive computational molecules, which may change from one iteration to the next, for the same grid point or control volume. In such context, this work will improve the applicability of high-order WENO schemes by coupling a standard cell centered, unstructured grid, finite volume method with an implicit topological data structure, the MateFace, which will, then, handle all the grid-related functions, including moving mesh operations. The computations are performed considering compressible flow of a perfect gas, modeled by the 2-D Euler equations, written in conservation-law form. Spatial discretization uses a 4th-order WENO scheme, implemented for a cell centered, finite volume method on general unstructured grids. Time march uses explicit Runge-Kutta schemes for solution advancement in time. The MateFace data structure is used to represent the mesh, controlling every single access to the mesh points and provides a complete set of operators for moving meshes. The MateFace is an implicit topological data structure specialized in representing meshes formed by different elements, such as triangles, quadrilaterals, or even mixed mesh elements. Some numerical experiments are performed involving unsteady flows around NACA0012 airfoil, and the results are presented to show the result of the moving mesh during the simulation.*

Keywords: *Moving Meshes, WENO Schemes, Finite Volume Method, Topological Data Structure*

1. INTRODUCTION

A essential step in a numerical simulation is the definition of a suitable discretization of the domain where the simulation will be performed. In many applications, one of the most common aspects to the CFD problem is the complexity of domain geometry where the analysis of the fluid flow is required. In such context, the geometrical model places an important role, once related problems in this field, like representation of complex domains, spatial decompositions, manipulation and movement of meshes, are naturally managed by methods developed in geometric modeling research. Topological data structures offer several advantages when performing a deformation on a mesh. These structures allow movement throughout the mesh without modifying its topology and there is always the possibility of merging it simulation/deformation cycle on a completely automatic and efficient form.

Over the past several years, the Computational Aerodynamics Laboratory of Instituto de Aeronáutica e Espaço (IAE) has been developing CFD solvers for two and three dimensional systems (Scalabrin, 2002) and (Basso and Azevedo, 2000). One branch of the development effort is aimed at the implementation of high-order methods suitable for problems of interest to the Institute, i.e., external high-speed aerodynamics. The current 2-D code has implementations of ENO and WENO schemes up to fourth order (Wolf and Azevedo, 2006). The essentially non-oscillatory schemes (ENO) and the weighted essentially non-oscillatory schemes (WENO) are implemented in cell centered, finite volume context for unstructured meshes. The ENO and WENO schemes have been developed with the purpose of accurately capturing discontinuities appearing in problems governed by hyperbolic conservation laws. The computations are performed considering compressible flow of a perfect gas, modeled by the 2-D Euler equations, written in conservation-law form. Spatial discretization uses a WENO scheme (Wolf and Azevedo, 2006), implemented for a cell centered, finite volume method on general unstructured grids. Time march uses explicit Runge-Kutta schemes for solution advancement in time.

The LCAD[1] group has been working with the development of topologic data structure and mesh generation. The Mate Face data structure, (Cunha, 2009, Cunha et. all., 2008) is used to represent the mesh, controlling every single access

[1] http://www.lcad.icmc.usp.br/

to the mesh points by providing a complete set of iterators and operators. The Mate Face is an implicit topological data structure specialized in representing meshes formed by different elements, such as triangles, quadrilaterals, or even mixed mesh elements.

The first study within the context of this work was improving the computational scheme of high-order WENO schemes by coupling a standard cell centered, unstructured grid, finite volume method with an implicit topological data structure, the Mate Face, which will, then, handle all the grid-related functions and operations (Barbosa et. all., 2010).

The main motivation of this paper is to include the procedure of moving meshes inside the CFD code. The results obtained from experiments based on a euler formulation for unsteady CFD code using unstructured grids are discussed.

The dynamic meshes are commonly used in the simulation of problems on domains whose geometry varies in time. Virtual springs are placed in the mesh to rearrange its vertexes whenever the domain is changed. The most commonly used types of springs are: longitudinal, torsional and semi-torsional. Here, will be used the type semi-torsional.

2. THEORICAL FORMULATION

The formulation used in this work is based on the twodimensional Euler equations. Due to the use of unstructured meshes and a finite volume discretization, these equations are used in the Cartesian form, they can be written as

$$\frac{\partial}{\partial t}\int\int_V \mathbf{Q}dxdy + \int_S (\mathbf{F}dy - \mathbf{G}dx) = 0, \tag{1}$$

where V represents the volume of the control volume, S is its surface, $\mathbf{Q}$ is the vector of conserved properties, $\mathbf{F}$ and $\mathbf{G}$ are the flux vectors in the x and y directions, respectively, defined as

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho U \\ \rho u U + p \\ \rho v U \\ U(e+p) + x_t p \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho V \\ \rho u V \\ \rho v V + p \\ V(e+p) + y_t p \end{bmatrix}, \tag{2}$$

where ρ is the density, u and v are the cartesian velocity components and e is the total energy per unity of volume. The pressure p is given by the perfect gas equation, written as

$$p = (\gamma - 1)\left[e - \frac{1}{2}\rho\left(u^2 + v^2\right)\right], \tag{3}$$

and γ represents the ratio of specific heats. The contravariant velocity components U and V are determined by

$$U = u - x_t \quad \text{and} \quad V = v - y_t, \tag{4}$$

where u and v are the cartesian components of the fluid velocity, while x_t and y_t are the cartesian components of the mesh velocity in the unsteady case.

3. NUMERICAL FORMULATION

In the present work, the 2D Euler equations are solved in their integral form as

$$\frac{\partial}{\partial t}\int_V \mathbf{Q}dV + \int_V (\nabla \cdot \mathbf{P})\, dV = 0, \tag{5}$$

where $\mathbf{P} = \mathbf{F}\vec{i} + \mathbf{G}\vec{j}$. Applying the Gauss theorem in Equation (5), we have

$$\frac{\partial}{\partial t}\int_V \mathbf{Q}dV + \int_V (\mathbf{P} \cdot \mathbf{n})\, dS = 0, \tag{6}$$

where V represents the volume of the cell, S represents the surface of the cell and $\mathbf{n}$ is the unit vector normal to the surface S. The Equation (6), discretized in a finite volume context centered in cells, can be rewritten for a control volume i as

$$\frac{\partial \mathbf{Q_i}}{\partial t} = -\frac{1}{V_i}\int_{S_i} (\mathbf{P} \cdot \mathbf{n})\, dS, \tag{7}$$

where $\mathbf{Q_i}$ is the mean value of $\mathbf{Q}$ at time t, in the control volume V_i, defined as

$$\mathbf{Q_i} = \frac{1}{V_i}\int_{V_i} \mathbf{Q}dV. \tag{8}$$

Therefore, a final formulation for a discretization of Euler equations in two dimensions inside a finite volume context is of form

$$\frac{\partial \mathbf{Q_i}}{\partial t} = -\frac{1}{V_i}\sum_{k=1}^{nf}\left(\mathbf{F_k}\vec{\mathbf{i}} + \mathbf{G_k}\vec{j}\right)\cdot \mathbf{S_k}, \tag{9}$$

where nf is the number of faces, or edges in the two dimensions space, of the volume control.

The control volumes considered in this work are triangles and they can be decomposed into a finite number of line segments Γ_j. One should observe that the control volumes could be composed by any type of polygon, because the really important aspect is that its bounding contour could be decomposed into a finite number of line segments. The surface integral from Equation (6) can be discretized using N -point Gaussian integration formula

$$\int_{S_i}(\mathbf{P}\cdot\mathbf{n})\,dS \approx \sum_j |\Gamma_j| \sum_{l=1}^{N} w_l \mathbf{P}\left(\mathbf{Q_j}(\mathbf{G_l}), t\right)\cdot\mathbf{n}, \tag{10}$$

where $\mathbf{G_l}$ and w_l are, respectively, the Gaussian points and the weights on the Γ_j line segment. For the second-order accuracy scheme just one Gauss point is necessary for the integration. Given the coordinates of the points in the corner of the control volume face, z_1 and z_2, it can be obtained the centroid point of the face,

$$\mathbf{G_1} = \frac{\mathbf{z_1} + \mathbf{z_2}}{2}. \tag{11}$$

In the case, the weight, w_1, is chosen as $w_1 = 1$. For the third-order schemes, two Gaussian points are necessary along each line segment. Their values are given by

$$G_1 = \frac{\sqrt{3}+1}{2\sqrt{3}}z_1 + \left(1 - \frac{\sqrt{3}+1}{2\sqrt{3}}\right)z_2 \qquad \text{and} \qquad G_2 = \frac{\sqrt{3}+1}{2\sqrt{3}}z_2 + \left(1 - \frac{\sqrt{3}+1}{2\sqrt{3}}\right)z_1, \tag{12}$$

where the respective weights, w_1 and w_2 are given by $w_1 = w_2 = 1/2$

Using the method described above, one can compute values of Q_i in some instant, t, and then, from these mean values, one can reconstruct polynomials that represent the primitive variables ρ, u, v and p. Finally, it is possible to compute values of the conserved variables in the Gaussian points. Due to the discontinuity of the reconstructed values of the conserved variables over the cell boundaries, one must use a numerical flux function to approximate these flux values on the cell boundaries. In this paper, we have used the Roe flux difference splitting method (Roe, 1981) to compute such approximations. An explicit Runge-Kutta scheme of five stages with second order of accuracy was used to obtain the time solution of the governing equations, proposed by Mavriplis (Mavriplis, 1988).

4. THE DATA STRUCTURE TOPOLOGICAL - MF

The Mate Face (MF) is an implicit topological data structure designed to store unstructured meshes. It implements many operators and iterators used to manipulate the mesh. The MF data structure allows the representation of two-dimensional meshes (simple 2D or superficial) and three-dimensional meshes (also called volumetric meshes). For a two-dimensional mesh, it can be of simple or hybrid type (triangle and/or quadrilateral), and a volumetric mesh can be only the simple type (which includes tetrahedron, prism, pyramid or hexahedron elements). The MF uses half-edges for superficial meshes and half-faces for volumetric meshes. The representation of vertexes, edges and cells is explicit, while topological operations are implicit.

The MF is a flexible data structure that can represent different mesh types, including meshes with different element types at the same time, such as triangles and quadrilaterals together. It implements a technique of partial allocations of vectors in order to represent vector of elements. In this way, memory blocks are dynamically set, as new elements are added in the mesh. This strategy eases the problem of memory space for vector of elements.

The geometric information about the mesh is stored in the vertexes, where each one is associated to position of the space, base for the most of geometric operations. Each vertex in the MF also store a reference to its mate cell through an identification number. The treatment of singularities is also implemented, facilitating diverse query types as star queries and operations on non convex meshes. The edges in MF are represented explicitly, and also stores the adjacent cells that share it. This representation has the advantage that information can be directly stored in the edge, as for example the id of adjacent cells of the edge.

The representation of cells is obtained by storing references to vertexes, edges (and faces in the three-dimensional case). Beyond that, mate cells can be retrieved through incident edges or faces.

One consideration about the organization in memory of mesh elements in two-dimensional meshes is that for each cell, the number of vertexes is equal to the number of edges and neighbors. This feature eases the implementation and representation of relationships of neighborhood. Then, by using a counter clock orientation, every neighbor cells are indexed in the mesh.

Besides the available traverse queries, geometric operators are also available, and are very used in the simulation code, such as the distance between two points, the area of a cell, the localization of a given query point (like inside a triangle or on the edge of it), the coordinates of a centroid of a cell, among others. The MF supports loading many mesh file formats, such as VTK and VRML.

4.1 Interface and the Integration Numeric - Fortran and C++

As separated research results, the simulator program and the data structure library were developed at different programming languages. In order to the simulation program (FORTRAN code) makes use of the data structure library (C++ code), an integration scheme was performed at compilation time. The proposed solution for the integration of technologies was the development of a mapping layer, acting as an interface between technologies, which maps operations of the Mate Face into functions that can be used by the simulator program. Figure 1 shows the integration scheme used to map and link both programs. The interface translate and controls every access to the mesh by the simulator program. All information exchanged is made at runtime between both codes using main memory for sharing data.

The interface is defined as a set of functions in such a format that can be called by the simulator program. Every call of functions in the interface share a controlled portion of main memory to pass information in and out. The processed information is formatted by the compiler, which performs a correct alignment of bytes in memory. By calling a set of functions in the interface, the simulator program can access the information of mesh, either performing all operations available in the Mate Face structure.

There are many advantages of centering all the mesh information and access in a data structure. One advantage is that the data structure is specialized in mesh control and consumes less memory to represent the same information previously stored inside the simulator program. Instead of wasting memory and time searching lots of tables of redundant mesh information, a single data structure optimizes and centralizes the mesh usage. The new scheme separates the mesh representation from the simulator core code.

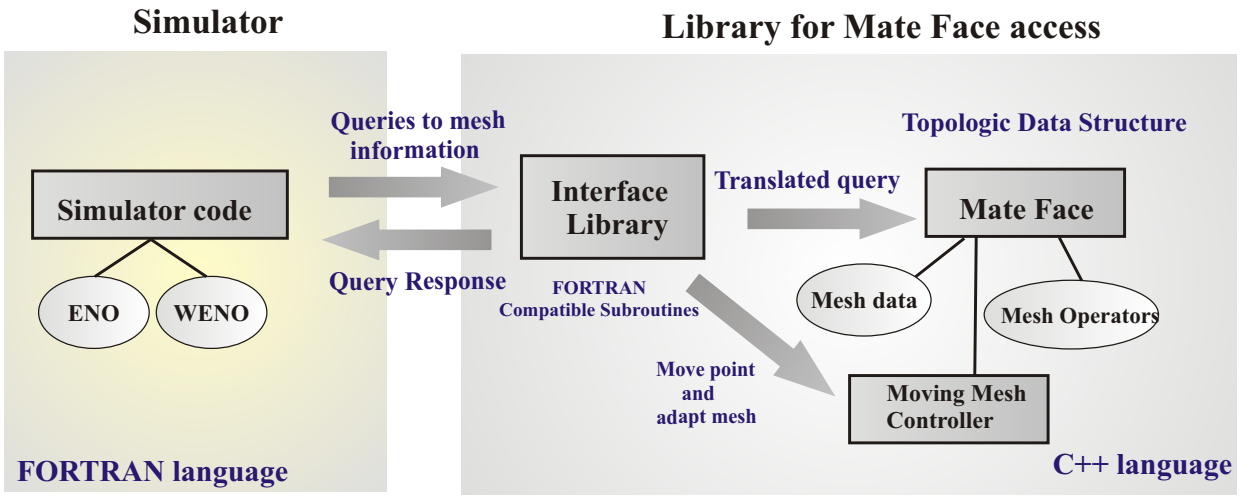


Figure 1. Integration scheme of the solver with the topological data structure library.

The interface layer maps all the iterators available in the Mate Face structure, organized in sets of functions to start, accesses current information, moves to the next item, and verifies if the iterator has ended. Examples would be simple queries of the type “loop over all cells (edges, vertexes)” to more complex queries such “retrieve all cells in the star (neighborhood) of a given vertex”. Beyond such queries, the interface provides common mesh information such as area value of the triangles of the mesh.

Another advantage of using a data structure is that Mate Face supports hybrid cells, such as triangles together with quadrilaterals, in the same mesh. All elements of the mesh will be automatically managed along with the proper iterators. Future functionalities include moving meshes controlled by the data structure triggered by simulator program data.

5. Dynamic Meshes

Dynamic meshes are created by methods that seek to replace the mesh vertexes thought simulations of physical forces of attraction and repulse. They are commonly used in the simulation of problems on domains whose geometry varies in time. In this case, the mesh is seen as a domain where the vertexes (seen as particles) move until the system reach a configuration of balance of forces. In general, such methods simulate a spring system, where each edge connecting two

vertexes is seen as a spring which has a width of equilibrium. The springs contribute in the movement of the vertexes in order to achieve the balance configuration (Batina, 1990) (Degand and Farhat, 2002).

Assuming that the incident forces in a vertex had been calculated, it is necessary to employ a movement equation in order to controll the dynamic behavior of the vertexes. From the second law of Newton, we have:

$$m\ddot{x} = \vec{F} \quad , \tag{13}$$

where x is the vector of positions of the vertexes in the mesh, M is the corresponding mass of vertexes and $\vec{F}$ are the forces applied on vertexes. This model can be extended for the case mass-spring-damper, where the force $\vec{F}$ is computed though the forces applied on the vertexes, given by:

$\vec{f}$: force between vertexes, analogous to a spring, and affect only adjacent vertexes.

$\vec{f_d}$: damper force, against the movement and contribute to the system convergence.

The system final equation can be written as the following second order ordinary differential equation:

$$m\ddot{x} + d\dot{x} = \vec{f} \quad . \tag{14}$$

A first-order differential equation can also be used to model the problem. Although more likely to park in a local minimum, This formulation has better numerical stability. The movement of the vertexes is determined by the equation:

$$\vec{f} - d\dot{x} = 0 \tag{15}$$

The problem can be simplified even more in a system known as quasi-static of form:

$$\vec{f} = 0. \tag{16}$$

The formulation of $\vec{f}$ is the key for the system solution. In the case of elastic mesh, we will write $\vec{f_i}$ as the resulting force in a vertex i under the influence of other vertexes j. There are many ways to calculate this force value, but the analogy with springs are the most known. Such approach can be described as follows. Springs are assigned between vertexes (or in edges between them), and the force value is determined by the Hook law:

$$\vec{f} = -k\vec{\delta} \quad , \tag{17}$$

where k is the spring stiffness coefficient and $\vec{\delta}$ is the displacement.

Each mesh edge is considered as being a spring having a stiffness coefficient. Such spring is called as a longitudinal spring. Considering that the resulting force value in a vertex is calculated combining the springs that are incident to it, the mesh connectivity will determine which vertexes j will contribute in the calculus of incident forces in vertex i, as the following equation shows:

$$\vec{f_i} = \sum_{j \in V(x_i)} k_{ij}(\vec{x}_j - \vec{x}_i - \vec{l}_{ij}) \quad , \tag{18}$$

where $\vec{l}_{ij}$ is the balance vector between vertexes i and j, $V(x_i)$ represents the set of vertexes that are adjacent to i.

In this type of problem, it is interesting that some properties of original mesh are preserved, for example, the proportion between the edges width. In this case, the vector $\vec{l}_{ij}$ of Equation (18) is defined from the edge between the vertexes i and j from the original mesh (Blom, 2000) and the force equation can be written as:

$$\vec{f_i} = \sum_{j \in V(x_i)} k_{ij}(\vec{\delta}_j - \vec{\delta}_i) \quad , \tag{19}$$

where $\vec{\delta}_i$ is the displacement of vertex i.

In order to preserve the size proportion between elements, the spring coefficient can be calculated as **?**

$$k_{ij} = \frac{1}{|\vec{l}^0_{ij}|} \tag{20}$$

where $|\vec{l}^0_{ij}|$ is the width of the edge between the vertexes i and j from original mesh, i.e., before the boundary deformation.

One problem with this type of modeling is that it cannot be guaranteed that a vertex do not move thought an edge, which can generate inconsistent elements, i.e., elements with negative area values. The main cause of the longitudinal spring deficiency is that these springs type do not consider the area value elements or the angle between edges. One solution for this problem consist to aggregate torsional springs (Farhat et. all., 1998) to the system. Consider the triangle

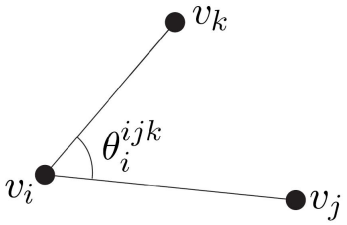


Figure 2. Torsional spring scheme.

t_{ijk} which vertexes are designed by v_i, v_j and v_k, as shown in Figure 2. Let θ_i^{ijk} the angle between the edges e_i and e_{ik}. Let $A(t_{ijk})$ the area of triangle t_{ijk}. The stiffness coefficient of a torsional spring in the vertex v_i of triangle t_{ijk} is given by:

$$C_i^{ijk} = \frac{1}{1 + cos(\theta_i^{ijk})} \frac{1}{1 - cos(\theta_i^{ijk})} = \frac{1}{sen(\theta_i^{ijk})^2} = \frac{(e_{ij})^2 (e_{ik})^2}{4A(t_{ijk})^2}, \tag{21}$$

Aiming at simplifying the torsional spring based formulation, adding information about internal angles of triangles, into systems based on longitudinal springs, the concept of semi-torsional spring was proposed by (Blom, 1990). The idea beyond semi-torsional formulation is to promote the influence of internal angles of triangles over the stiffness coefficient of longitudinal springs, which implies the exclusive presence of longitudinal springs in the system. This influence in the internal angles of triangles over longitudinal springs is determined by the ratio between the stiffness coefficient of the longitudinal spring and the value of the opposite angle.

Consider the Figure 3 a). The longitudinal spring coefficient in the edge e_{ij} is divided by the internal angle of triangle t_{ijk} in $v_k(\theta_k^{ijk})$. The resistivity coefficient of the semi-torsional spring of Blom, is given by:

$$k_{ij}^* = \frac{k_{ij}}{\theta_k^{ijk}} \tag{22}$$

Note that the value of k^* is minimally altered when the computed triangle is equilateral, once that $\theta_i^{ijk} = \theta_j^{ijk} = \theta_k^{ijk}$. The more acute the angle, the more tight is the spring. But, the more obtuse the angle is, the less tight is the spring, which it is not desirable.

An improvement in the semi-torsional springs was introduced by (Zeng and Ethier, 2001). The basic idea consist of add the coefficient of a torsional spring to the opposite longitudinal spring coefficient. In this manner, the total longitudinal stiffness coefficient k_{ij}^t of the spring between vertexes v_i and v_j (Note Figure 3 b)) is given by

$$k_{ij}^t = k_{ij} + C_k^{ijk} + C_l^{ijl}, \tag{23}$$

where C is the spring torsional stiffness coefficient given by Equation 21.

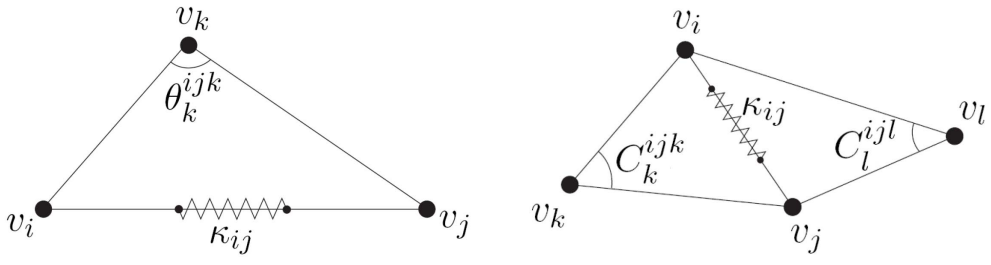


Figure 3. a) Semi-Torsional spring proposed by Blom; b) The torsional springs in v_k and v_l are combined to the longitudinal spring in the edge e_{ij}.

The advantage of the semi-torsional formulation is that it is applicable when there are great deformations in the mesh. Another advantage when compared to torsional springs is lower number of computations operations during its execution. However, the stiffening of torsional springs by using semi-torsional springs do not avoid the appearing of invalid elements for all cases.

5.1 Execution of Dynamic Mesh

The solution of the governing equations system of the dynamic mesh leads to the displacements bounded to the vertexes inside the moving mesh. This system is linear and can be solved iteratively by the method of Gauss-Seidel, where the new displacement of the iterative vertex v_i, $\vec{\delta}_i^{new}$, is dependent of the previous calculated displacement $\vec{\delta}^{old}$. For the case of semi-torsional springs, for example, the value of $\vec{\delta}_i^{new}$ is given by

$$\vec{\delta}_i^{new} = \frac{\sum_{v_w \in V(\star_{v_i})} k_{iw} \vec{\delta}_w^{old}}{\sum_{v_w \in V(\star_{v_i})}} \tag{24}$$

where w are values from the mesh connectivity.

From the Equation (24) it can be seen that $\vec{\delta}_i^{new}$ equals the mean of the neighbor displacements, weighted by the stiffness coefficient of the spring. Once the vertexes are dislocated, the spring torsional coefficients that composes the semi-torsional coefficient α must be updated after each iteration.

Having as parameters the mesh to be deformed (vertexes are displaced), the set of vertexes from boundary of the mesh and the set of displacements of vertexes, it is directly achieved that an iterative process can be applied which uses the Equation (24), where the values of the new displacements of vertexes are updated at each iteration. The stop criteria of the iterative process is the maximum absolute value from the relative difference between the values of the displacements obtained between an iteration and the next for all vertexes in the mesh. Therefore, it can be noted that during the process there is no effective displacement of vertexes, only the new displacements updated at each iteration. The final coordinates are calculated from the final displacements once the process converges.

5.1.1 Active Vertexes

The concept of relative displacement previously mentioned as a stop criteria can be expressed by the following equation:

$$\epsilon = \frac{\left\| \vec{\delta}_w^{(t)} - \vec{\delta}_w^{(t-1)} \right\|}{\left\| \vec{\delta}_w^{(t)} \right\|} \tag{25}$$

where ϵ represents the tolerance level defined by the specialist of the domain, and w is the iterated vertex. Although this strategy analyzes vertexes of refined regions and of less refined regions of the mesh with the same measure, the stop process can adapted, once less refined regions of the mesh have bigger edges being less sensitive to the relative displacement than the refined ones. Therefore, it is more appropriated that the influence of the relative displacement vary according to the local refinement of the moving mesh:

$$max_{v_w \in T} \left(\frac{\left\| \vec{\delta}_w^{(t)} - \vec{\delta}_w^{(t-1)} \right\|}{e_{min}^w} \right) \geq \epsilon \tag{26}$$

where e_{min} represents the width of the smaller edge incident to the iterated vertex v_w, T refer to the set of vertexes of the mesh. The calculus of value of e_{min}^w is performed only at the beginning of the process, before the displacement of the boundary.

In this way the value of ϵ becomes more intuitive to analyze, i.e., for $\epsilon = 0.01$ the stop criteria will be satisfied only if, for each vertex $v_w \in T$, the relative displacement is less than $0.01 e_{min}^w$. The value of ϵ can be seen as a percentile factor of the smallest edge, which is related to the refinement level of the mesh. There is a relaxation of the stop criteria in regions where the elements of the mesh are less refined.

An efficient way to perform the above mentioned process controlls the number of times in which the vertexes are visited and also controlls the sequence that it occurs, influencing directly the processing time and in the convergence of the employed numeric method. A more interesting process sequence in the case consists of iterate the vertexes in layers, starting from the border to the center of the mesh. In this approach, the vertexes are labeled in the following way:

- Active: vertex ready for processing,
- Inactive : vertex not being processed,
- Fixed: still vertex.

Once each vertex can have only one label, we can separate all vertexes from the mesh into three disjoint sets: $\mathcal{A}$, $\mathcal{I}$ and $\mathcal{F}$, representing the set of active, inactive and fixed vertexes in the same order. Initially, all vertexes of the mesh are labeled as follows:

1. Vertexes belonging to the border of the mesh are labeled s fixed;
2. Vertexes directly connected to the fixed vertexes are labeled as active, having displacement non zero;
3. All non-labeled vertexes in the two steps before are labeled inactive.

In this way, only the active labeled vertexes are iterated: $\mathcal{A}$ is initially composed of the inner vertexes, adjacent to the border of the mesh, leading to a major displacement at the first iterated vertexes. In this new methodology, it is not unusual the alternation between the active and inactive label to the same vertex during the iterative process: when its relative displacement becomes insignificant, an active vertex become inactive. Nevertheless, if its displacement is significant, its status will remain as active, as well as all vertexes inactive adjacent to it.

6. RESULTS

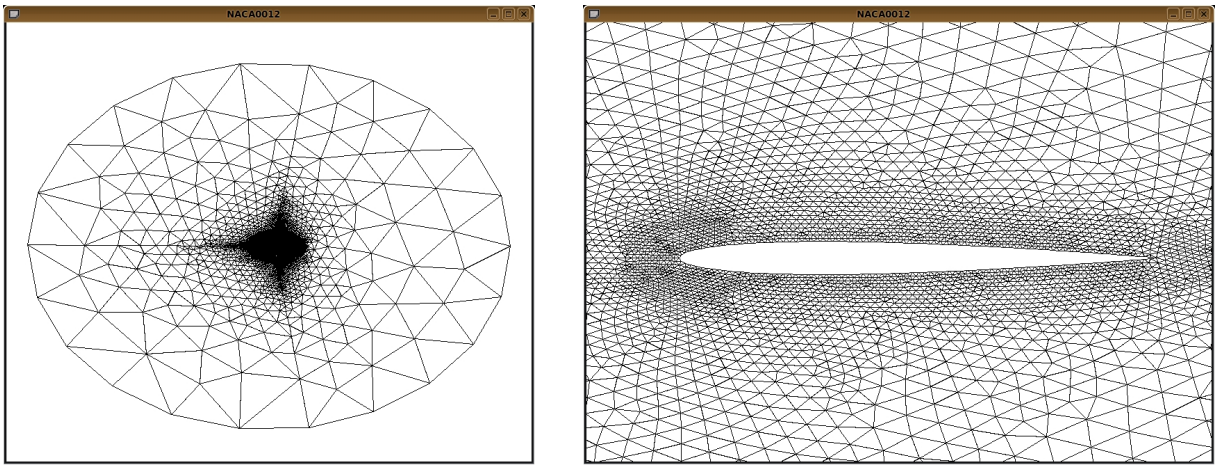
In this section we show some results from the simulations with moving meshes. Our focus is to analyze the behavior of the discussed algorithm for displacement of the mesh. Then, we show at different stages of simulation, the resulting mesh of each iteration of the solver. The chosen mesh represents a NACA 0012 airfoil section, object of test cases in many related works. The experiment moves the mesh and analyzes its integrity by checking good triangles. The mesh for tests is composed of 5908 vertexes, 11634 cells and 17542 edges.

The Figure 4 shows an example of mesh for the NACA 0012 airfoil section. The results here shown correspond to the harmonic movement with forced pitching. For the tests the airfoil harmonic movement is defined by the incidence angle equation as a time function:

$$\alpha(t) = \alpha_0 + \alpha_m sen(\omega t) \tag{27}$$

where α_m is the amplitude of the oscilatory movement, α_0 is the initial incidence and ω is the angular frequency of the movement.

The boundary conditions in the external border are of the type non-reflexive to guarantee that do not exist reflection in the information that propagate outside of the field.



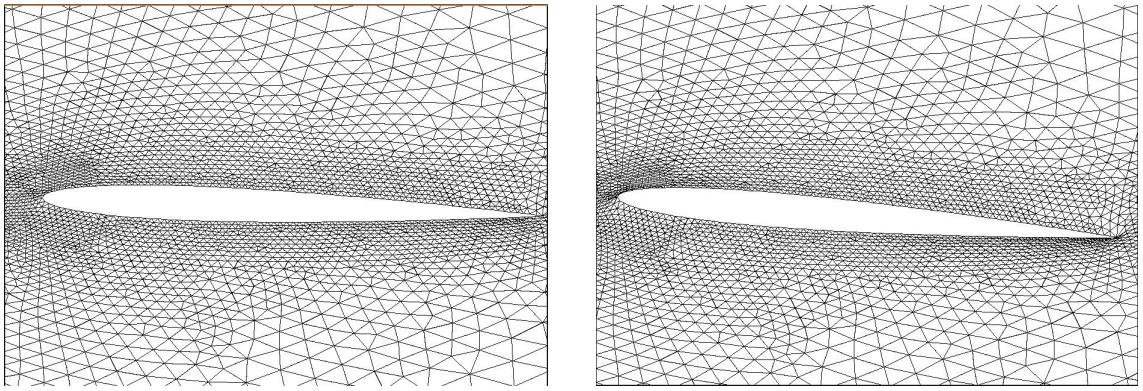
(a) (b)

Figure 4. a) General view of the computational mesh unstructured generated for the NACA 0012 airfoil; b) enlarged airfoil region showing refinement details.

An example of displacement of the mesh that represents the airfoil NACA 0012 can be seen in Figure 5, under simulation conditions. In 5(a), it is shown that the deformed mesh after 800 steps of solver execution, while in 5(b) is shown the same mesh processed after 1200 steps. It is important to analyze that the displacement process must adapt the elements in the deformed region, updating the new areas of triangles and the new mesh velocities (measured from the mean of vertexes velocities from each cell). In the resulting meshes, all triangles have positive area and none cross edges.

7. CONCLUSION

In many applications, one of the most common aspects to the CFD problem is the complexity of domain geometry where the analysis of the fluid flow is required. In such context, the geometrical model places an important role, once related problems in this field, like representation of complex domains, spatial decompositions, manipulation and movement of meshes, are naturally managed by methods developed in geometric modeling research. Topological data structures



(a) (b)

Figure 5. Deformations of the mesh representing the NACA 0012 airfoil, using α_m = 10 and ω = 0.12165 a) Simulation after 800 steps e b) Simulation after 1200 steps.

offer several advantages when performing a deformation on a mesh. These structures allow movement throughout the mesh without modifying its topology and there is always the possibility of merging it simulation/deformation cycle on a completely automatic and efficient form.

The dynamic meshes are commonly used in the simulation of problems on domains whose geometry varies in time. Virtual springs are placed in the mesh to rearrange its vertexes whenever the domain is changed. The most commonly used types of springs are: longitudinal, torsional and semi-torsional. Here, we used the semi-torsional type.

Results showed different mesh displacements over the discussed procedure for simulations in the NACA 0012 airfoil. In the meshes shown it can be seen the arrangement performed around the internal moving border, keeping cells area proportion on dense areas. Future work is to use the mesh velocities information in the CFD code and perform aeroelastic analysis.

8. ACKNOWLEDGEMENTS

The authors acknowledge the partial support provided to the present research by Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under Grant No. 2008/01544-6 and partial support for the present work was also provided by Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under the Research Project Grant No. 312064/2006-3 and 301408/2009-2 and it is greatly appreciated.

9. REFERENCES

Barbosa, F. P.; Castelo Filho, A. ; Cuminato, J. A. ; Azevedo, J. L. F.; Breviglieri, C., 2010, “A Computational Study of WENO Schemes Using a Topological Data Strucuture”, 13th Brazilian Congress of Thermal Sciences and Engineering, Uberlândia, MG - Brazil.

Basso, A. P. A. and Azevedo, J. L. F., 2000, Three Dimensional Flow Simulations Over a Complete Satellite Launcher with a Cluster Configuration, “Proceedings of the 18th AIAA Applied Aerodynamics Conference and Exhibit”, Vol. AIAA Paper No. 2000-4514, pp. 805-813, Denver, CO.

Batina, J. T. , 1990, “Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes”. AIAA Journal, 28(8):1381Ű1388.

Blom, F. J. , 2000, “Considerations on the Spring Analogy”. International Journal for Numerical Methods in Fluids, 32(6).

Camilo, E., 2007, “Aeroelasticidade Computacional Transônica em Aerofólios com Modelo Estrutural Não Linear”. PhD thesis, Tese de Doutorado, Universidade de São Paulo.

Cunha, I. L. L., 2009, “ Estrutura de dados Mate Face e Aplicações em Geração e Movimento de Malhas”. PhD thesis, Dissertação de Mestrado, Universidade de São Paulo.

Cunha, I. L. L., Lacassa, A. de ; Polizelli-Jr., V. ; Gois, J. P. ; Nonato, L. G. ; Castelo Filho, A, 2008, “Adaptive Algebraic Mesh Generation from Implicit Functions”, Iberian Latin American Congress on Computational Methods in Engineering, Maceió, Brazil.

Degand, C. and Farhat, C., 2002, “A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes”. Computers and Structures, 80:305Ű316.

Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., 1998, “Torsional Springs for Two-Dimensional Dynamic Unstructured Fluid Meshes”. Computer Methods in Applied Mechanics and Engineering, 163:231Ű245.

Wolf, W.R., and Azevedo, J.L.F., 2006, “High-Order Unstructured Essentially Nonoscillatory and Weighted Essen-

tially Nonoscillatory Schemes for Aerodynamic Flows", AIAA Journal, Vol. 44, No. 10, pp. 2295-2310.

Scalabrin, L. C., 2002, Numerical Simulation of Three-Dimensional Flows over Aerospace Configurations, Master Thesis.

Soares, I. P., 2007, "Movimento de Malhas e Remalhamento de Malhas Superficiais". PhD tesis, Tese de Doutorado, Universidade de São Paulo.

Zeng, D. and Ethier, C. R., 2001, " A Semitorcional Spring Analogy Model for Updating Unstructured Meshes". Proceedings of 9th Annual Conference of the CFD Society of Canada.

10. Responsibility notice

The author(s) is (are) the only responsible for the printed material included in this paper