DEVELOPMENT OF A THREE DIMENSIONAL MULTI-BLOCK STRUCTURED GRID DEFORMATION CODE FOR COMPLEX CONFIGURATIONS

Nguyen Anh Thi [(1)], Hoang Anh Duong [(2)]

(1) Full-time lecturer, Ho Chi Minh City University of Technology, Viet Nam

(2) Master student, Gyeongsang National University, South Korea

(Manuscript Received on February 24th, 2010, Manuscript Revised August 26th, 2010)

ABSTRACT: *In this study, a multi-block structured grid deformation code based on a hybrid of transfinite interpolation algorithm and spring analogy has been developed. The combination of spring analogy for block vertices and transfinite interpolation for interior grid points helps to increase the robustness and makes it suitable for distributed computing. Elliptic smoothing operator is applied to the block faces with sub-faces to maintain the grid's smoothness and skewness. The capability of the developed code is demonstrated on a range of simple and complex configuration such as airfoil and wing body configuration.*

Keyword: *transfinite interpolation (TFI), spring analogy, grid deformation, multi-block structured grid.*

1. INTRODUCTION

The numerical simulation of unsteady flow with multi-block structured grid arises in many engineering applications such as fluid-structure interaction (FSI), control surface movement and aerodynamic shape optimization design. One critical part in these applications is updating computational grid at each time step. The new mesh can be either regenerated or dynamically updated. The first approach is a natural choice that consists in regenerating computational grid at each time step during time integration. However, grid generation for complex configuration is by itself a nontrivial and time-consuming task. Even though there are still some robustness problems for large deformation to be solved, the second approach is inexpensive and appropriate for practical problems.

Development of an efficient and robust grid deformation methodology that still maintains the quality of the initial grid (smoothness, skewness,...) generated by a commercial grid generation package is the subject of various studies in the past. Many methodologies such as transfinite interpolation (TFI), isoparametric mapping, elastic-based analogy and spring analogy have been proposed [1-7]. Some of them are computationally efficient but less robust with respect to the crossover cells while others are more robust but very computationally expensive. An algebraic method was used by Bhardwaj et al. [1] to deform the grid by redistributing grid points along grid lines that

are in the normal direction of the surface. Jones et al. [1] had used transfinite interpolation (TFI) method to regenerate the structured grid. Dubuc et al. [7] had provided the detail analysis of TFI method and discussed pros and cons of this method for multi-block structured grids. Algebraic methods are fast but work well only for small deformation [2]. Large deformation may cause the crossover of grid lines or produce poor quality grid. A spring-analogy method initially proposed by Nakahashi and Deiwert [4] was applied to aero-elasticity problems by Batina [11]. The comparison between spring-analogy and elliptic grid generation approach was presented by Bloom [4]. It is well known that the standard spring analogy will result in the inversion of elements for large deformation. To overcome this drawback, numerous schemes such as torsional, semi-torsional and ortho-semi-torsional spring analogies have been suggested [5,6]. This method as well as the elastic analogy can adapt to significant surface deformations but their computational cost is expensive for complex problems with large number of grid points. It has been also widely applied to unstructured grid deformation [4,11].

Hybrid approach, a useful compromise between algebraic and iterative approaches, is proposed in the recent years [1-3,8,9]. Tsai et al. [1] provided a new scheme which combines the spring analogy and TFI method in Algebraic and Iterative Mesh 3D (AIM3D) code. Based on this scheme, Spekreijse et al. [2] introduced a new methodology which replaces spring-analogy by volume spline interpolation. Although these schemes provide relatively good results, there is still a major drawback involving sub-faces problem, which has been not solved yet. To overcome this disadvantage, Potsdam and Guruswamy [3] have proposed a point-by-point methodology. Instead of computing the displacement of block vertices, the nearest surface distances is used to define the deformed surfaces of block. In order to improve the orthogonality of the grid lines near the configuration surfaces, Samareh [9] introduces quaternion methodology. Although many algorithms were developed, considerable effort has been devoting to the development of robust and efficient general techniques for grid deformation. Reference [8] proposed a new methodology that combines the definition of material properties and transfinite interpolation to generate the deformed mesh.

Another important problem of multi-block structured grid deformation is the handling of blocks, in general connected in an unstructured fashion, in distributed computing context, wherein the blocks are usually distributed over different processors. Therefore, a grid deformation method should allow deformation to be accomplished on each processor without having to gather all of the blocks on one processor and with little communication between processors. This problem was first discussed and solved by Tsai et al. [1]. Another problem that one must face to is the matching between block faces in the matched multi-block structured grid concept.

In this study, an efficient and robust deformed grid code, substantially based on the technique proposed by Tsai et al. [1], is developed. This algorithm is the combination of spring analogy and TFI methods and can also be easy to implement in distributed parallel computing context. In the first step, the configuration surface is parameterized using Bezier surface. The second step consists in determining the displacement of all blocks' corner points by using the spring analogy. In general, the number of blocks, and thus, the number of vertices are far fewer than the volume grid points so that the computational cost for this step is small. Once new coordinates of the corner points are determined, TFI method will be used to compute the deformation of edges, face and volume grid points in each block separately. The current approach does not ensure the quality of block faces which are constituted by several patches having different boundary conditions. To solve this problem, instead of block faces, TFI method is applied to each patch of block faces. Elliptic smoothing operator with only one or two iterations is applied to these patches to improve the grid quality on these block faces. To ensure the matching on the block interfaces, mesh points are redistributed using an averaging of mesh point coordinates between two neighboured interfaces.

In the next sections, the shape parameterization, the spring analogy technique, and then the arc-length-based TFI technique will be presented. Various numerical results of grid deformation of some simple and complex configurations such as airfoil and wing-body configuration will be presented to demonstrate the capability of developed grid deformation code.

2. SHAPE PARAMETERIZATION

In design optimization problem, parameterization of configuration is one of the most outstanding issues of concern. One must compromise between the accuracy of parameterization technique and the number of required parameters. Among these techniques, Bezier curve/ surface is one of the most popular approaches. The design parameters for this case are the positions of control points of Bezier curves.

A Bezier curve/surface [10] in $\Re^d$ ($d = 2$ or 3) of degree n supported by a control polygon of $n+1$ control points $p_k \in \Re^d$ (with $k = 0, 1, \ldots n$) is:

$$x(t) = \sum_{i=0}^{k} B_n^k(t) p_k \qquad (1)$$

Here $B_n^k(t)$ is the *Bernstein* polynomial:

$B_n^k(t) = C_n^k t^k (1-t)^{n-k}$ in which $C_n^k = \dfrac{n!}{k!(n-k)!}$ and the parameter t varies from 0 to 1

The procedure used to compute the coordinate of control points from configuration surfaces is proposed in [13]. The formula of Bezier curve can be written in matrix form:

$$[X(t_i)] = [B_{i,k}][p_k] \qquad (2)$$

Multiplying the transpose of matrix B to this equation yields:

$$[B_{i,k}]^T[B_{i,k}][p_k] = [B_{i,k}]^T[X(t_i)] \qquad (3)$$

Solution of this system of linear equations is the coordinates of control points. For the Bezier surface, similar process can also be applied.

To demonstrate the capability of this approximation method, Bezier curves are used to represent the upper and lower surfaces of RAE2822 airfoil. Seventeen control points are used for each surface. The condition that the first and last control points of two Bezier curves are the same ensures the coincidence of two surfaces.

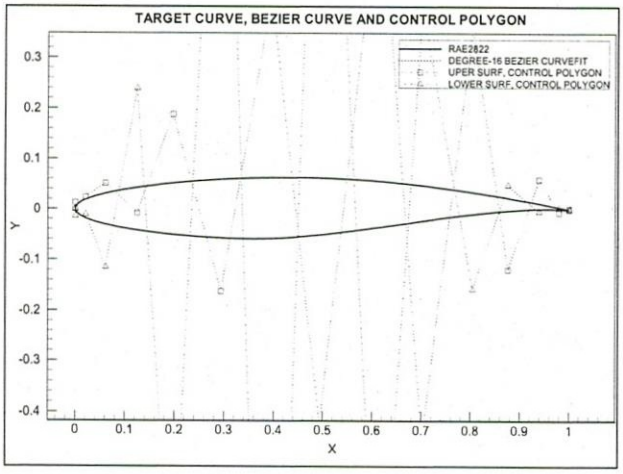


Figure 1. RAE2822 airfoil, 16-degree Bezier curve-fits, and control polygons of upper and lower surfaces

To examine the accuracy of shape parameterization technique, the tolerance between the Bezier curves and initial RAE2822 airfoil is formulated as:

$$TOL = \sum_{i=1}^{n} \frac{\sqrt{(x_B - x_i)^2 + (y_B - y_i)^2}}{N}$$

in which N is number of discrete points of airfoil (4)

In this example the tolerance is about 1E-3. It has been demonstrated that this error is adequate for optimization design [10].

While this method offers the acceptable accuracy and the small number of required parameters, it still has a minor drawback. If design surface is represented by a finite number of patches, the matching between these patches must be guaranteed. Because of the computational error, Bezier surface can not handle this problem. In order to solve matching problem, special coding logic should be written to eliminate this error.

3. MULTI-BLOCK STRUCTURED GRID DEFORMATION APPROACH

The grid deformation code developed in this study is substantially based on the combination of algebraic and iterative methods proposed by Tsai et al. [1]. Algebraic method such as transfinite interpolation (TFI) is inexpensive to run but they can not solve large deformation problems. This drawback can be surmounted by using iterative method such as spring analogy. Unfortunately, this method requires expensive computational cost. A hybrid approach, combining these two approaches, will naturally inherit the robustness of iterative method and the efficiency of algebraic one.

The first step of hybrid method used in this study consists in computing the displacement of all vertices of each block. In multi-block structured grid context, the arrangement of blocks is generally unstructured so that the motion of these corner points will be determined by spring analogy. TFI is then applied to compute the displacement of the interior grid points in each block.

3.1. Spring analogy

The concept of spring analogy as proposed in [4] is adopted for determining the moving of blocks' vertices. Spring analogy models are categorized into two types: vertex model and segment model. In this study, the segment model was adopted. The corner points are viewed as a network of fictitious springs with the stiffness defined as follows:

$$k_{ij} = \frac{\lambda}{\left[\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2\right]^{\beta}} \quad (5)$$

Spring stiffness is computed for all 12 edges and 4 cross-diagonal edges of a block. These cross-diagonal edges are used for controlling the shearing motion of grid cells. The coefficients λ and β are used to control the stiffness of grid cells. Typically, the coefficients λ and β are taken to be 1 and 0.5, which means that the stiffness is inversely proportional to the length of connecting edges [1].

It is assumed that the displacement of the configuration surface is prescribed. The motion of the corner points of each block is determined by solving the equations of static equilibrium:

$$\sum_{j=1}^{N_{ei}} k_{ij}\left(\delta_i^n - \delta_j^n\right) = 0 \quad (6)$$

The static equilibrium equations are iteratively solved as follows:

$$\left(\delta x\right)_i^{n+1} = \frac{\sum_{j=1}^{N_{ei}} k_{ij}\left(\delta x\right)_j^n}{\sum_{j=1}^{N_{ei}} k_{ij}}, \left(\delta y\right)_i^{n+1} = \frac{\sum_{j=1}^{N_{ei}} k_{ij}\left(\delta y\right)_j^n}{\sum_{j=1}^{N_{ei}} k_{ij}}, \left(\delta z\right)_i^{n+1} = \frac{\sum_{j=1}^{N_{ei}} k_{ij}\left(\delta z\right)_j^n}{\sum_{j=1}^{N_{ei}} k_{ij}} \quad (7)$$

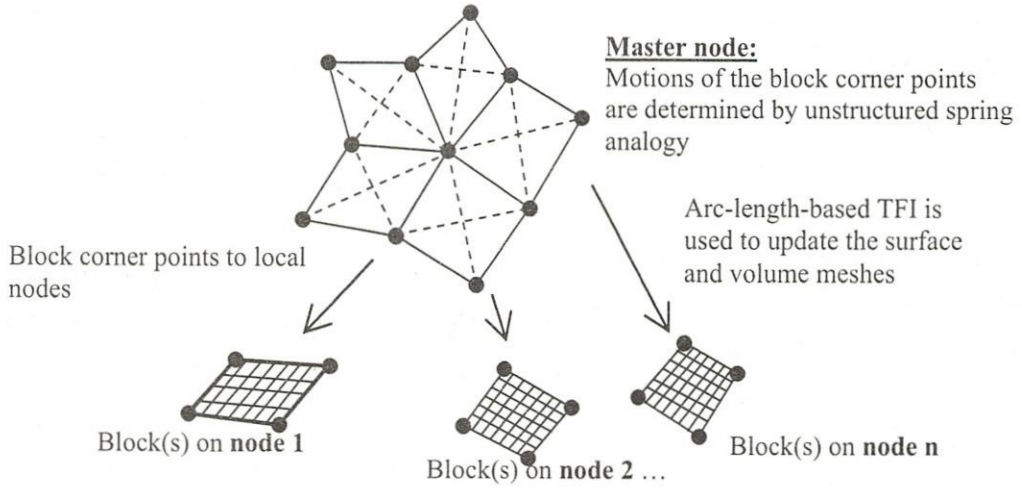


Figure 2. Strategy for parallel multi-block structured grid deformation

3.2. Transfinite interpolation (TFI)

After computing the moving of all blocks' vertices, the volume grid in each block can be determined by using the arc-length-based TFI method described below. It has been demonstrated [1] that this method preserves the characteristics of the initial mesh. The process to implement TFI method proposed in [1] includes following steps:

- Parameterize all grid points.

- Compute grid point deformations by using one, two and three dimensional arc-length-based TFI techniques.

- Add the deformations obtained to the original grid to obtain new grid.

A multi-block structured grid consists of a set of blocks, faces, edges and vertices. Each block has its own volume grid defined as follows:

$$X^n = \left\{ \bar{x}_{i,j,k} \mid i = 1,...,i\max; j = 1,...,j\max; k = 1,...,k\max \right\}$$

In parameterization process, the normalized arc-length-based parameter for each block along the grid line in i direction is defined as follows:

$$s_{1,j,k} = 0$$

$$s_{i,j,k} = s_{i-1,j,k} + \sqrt{\left(x_{i,j,k} - x_{i-1,j,k}\right)^2 + \left(y_{i,j,k} - y_{i-1,j,k}\right)^2 + \left(z_{i,j,k} - z_{i-1,j,k}\right)^2} \qquad (8)$$

$$F_{i,j,k} = \frac{s_{i,j,k}}{s_{i\max,j,k}}$$

Similarly, the parameters $G_{i,j,k}$ and $H_{i,j,k}$ for j and k directions can be defined.

The second stage is computing the displacement of the edges, surfaces and block points based on one, two and three dimensional TFI formula, respectively. From the

displacement of the configuration surfaces, the interpolated values of the deformation is created by using TFI method and so that the new grid, which is obtained by adding the deformations to the initial mesh, can maintain the quality of the original grid.

The one dimensional TFI in the i direction is simply defined by:

$$\Delta E_{i,1,1} = \left(1 - F_{i,1,1}\right)\Delta P_{1,1,1} + F_{i,1,1}\Delta P_{i\max,1,1} \quad (9)$$

Here ΔP is the displacement of the two corner points of block's edge. The displacement of block's surface (for example the surface in the plane $k = 1$) is computed by the two dimensional TFI formula:

$$\begin{aligned}\Delta S_{i,j,1} &= \left(1 - F_{i,j,1}\right)\Delta E_{1,j,1} + F_{i,j,1}\Delta E_{i\max,j,1} \\ &+ \left(1 - G_{i,j,1}\right)\left(\Delta E_{i,1,1} - \left(1 - F_{i,j,1}\right)\Delta P_{1,1,1} - F_{i,j,1}\Delta P_{N,1,1}\right) \\ &+ G_{i,j,1}\left(\Delta E_{i,j\max,1} - \left(1 - F_{i,j,1}\right)\Delta P_{1,N,1} - F_{i,j,1}\Delta P_{N,N,1}\right)\end{aligned} \quad (10)$$

After computing the deformation of all surfaces and edges, a standard three dimensional TFI formula is used to determine the displacement of all volume grid points:

$$\Delta V_{i,j,k} = V1 + V2 + V3 - V12 - V13 - V23 + V123 \quad (11)$$

where

$$V1 = \left(1 - F_{i,j,k}\right)\Delta S_{1,j,k} + F_{i,j,k}\Delta S_{i\max,j,k}$$

$$V2 = \left(1 - G_{i,j,k}\right)\Delta S_{i,1,k} + G_{i,j,k}\Delta S_{i,j\max,k}$$

$$V3 = \left(1 - H_{i,j,k}\right)\Delta S_{i,j,1} + H_{i,j,k}\Delta S_{i,j,k\max}$$

$$\begin{aligned}V12 &= \left(1 - F_{i,j,k}\right)\left(1 - G_{i,j,k}\right)\Delta E_{1,1,k} + \left(1 - F_{i,j,k}\right)G_{i,j,k}\Delta E_{1,j\max,k} \\ &+ F_{i,j,k}\left(1 - G_{i,j,k}\right)\Delta E_{i\max,1,k} + F_{i,j,k}G_{i,j,k}\Delta E_{i\max,j\max,k}\end{aligned} \quad (12)$$

$$\begin{aligned}V13 &= \left(1 - F_{i,j,k}\right)\left(1 - H_{i,j,k}\right)\Delta E_{1,j,1} + \left(1 - F_{i,j,k}\right)H_{i,j,k}\Delta E_{1,j,k\max} \\ &+ F_{i,j,k}\left(1 - H_{i,j,k}\right)\Delta E_{i\max,j,1} + F_{i,j,k}H_{i,j,k}\Delta E_{i\max,j,k\max}\end{aligned}$$

$$\begin{aligned}V23 &= \left(1 - G_{i,j,k}\right)\left(1 - H_{i,j,k}\right)\Delta E_{i,1,1} + \left(1 - G_{i,j,k}\right)H_{i,j,k}\Delta E_{i,1,k\max} \\ &+ G_{i,j,k}\left(1 - H_{i,j,k}\right)\Delta E_{i,j\max,1} + G_{i,j,k}H_{i,j,k}\Delta E_{i,j\max,k\max}\end{aligned}$$

$$V123 = (1-F_{i,j,k})(1-G_{i,j,k})(1-H_{i,j,k})\Delta P_{1,1,1} + (1-F_{i,j,k})(1-G_{i,j,k})H_{i,j,k}\Delta P_{1,1,k\max}$$
$$+(1-F_{i,j,k})G_{i,j,k}(1-H_{i,j,k})\Delta P_{1,j\max,1} + (1-F_{i,j,k})G_{i,j,k}H_{i,j,k}\Delta P_{1,j\max,k\max}$$
$$+F_{i,j,k}(1-G_{i,j,k})(1-H_{i,j,k})\Delta P_{i\max,1,1} + F_{i,j,k}(1-G_{i,j,k})H_{i,j,k}\Delta P_{i\max,1,k\max}$$
$$+F_{i,j,k}G_{i,j,k}(1-H_{i,j,k})\Delta P_{i\max,j\max,1} + F_{i,j,k}G_{i,j,k}H_{i,j,k}\Delta P_{i\max,j\max,k\max}$$

3.3. Smooth operator: elliptic differential equation

There are cases in which only a certain portion(s) of a surface is distorted extremely. To accommodate such problem, a smooth operator is locally applied to alleviate this distortion. In this study, elliptic different equation is used to smooth the deformed grid.

$$a_{22}r_{11} + a_{11}r_{22} - 2a_{12}r_{12} = 0 \qquad (13)$$

With $r_{11} = \begin{bmatrix} x_{\xi\xi} \\ y_{\xi\xi} \\ z_{\xi\xi} \end{bmatrix}$, $r_{22} = \begin{bmatrix} x_{\eta\eta} \\ y_{\eta\eta} \\ z_{\eta\eta} \end{bmatrix}$, $r_{12} = \begin{bmatrix} x_{\xi\eta} \\ y_{\xi\eta} \\ z_{\xi\eta} \end{bmatrix}$

$$a_{11} = x_\xi^2 + y_\xi^2 + z_\xi^2$$
$$a_{22} = x_{\eta\eta}^2 + y_{\eta\eta}^2 + z_{\eta\eta}^2 \qquad (14)$$
$$a_{12} = x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta$$

$$x_\xi = 0.5(x_{i+1,j} - x_{i-1,j})$$
$$x_\eta = 0.5(x_{i,j+1} - x_{i,j-1})$$
$$x_{\xi\xi} = x_{i+1,j} - 2x_{i,j} + x_{i-1,j}$$
$$x_{\eta\eta} = x_{i,j+1} - 2x_{i,j} + x_{i,j-1}$$
$$x_{\xi\eta} = 0.25(x_{i+1,j+1} - x_{i+1,j-1} - x_{i-1,j+1} + x_{i-1,j-1})$$

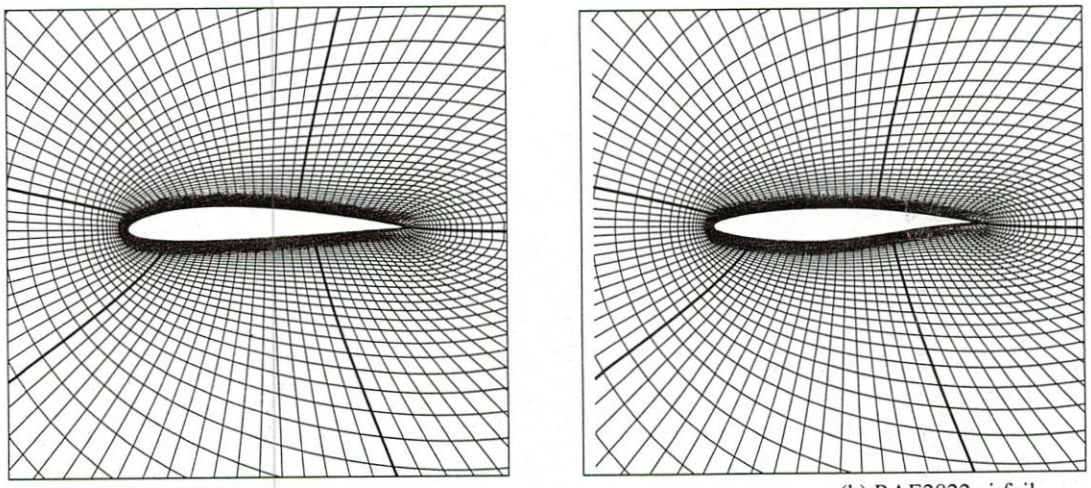
Elliptic operator is used only for the sub-faces to eliminate possible distortions after applying TFI method. To maintain the efficiency of this code, only one or two elliptic smoothing iterations are used. Because TFI method is already used, one or two iteration is enough enhance the smoothness of deformed grid. When elliptic smoothing operator is applied, the computational time is in general just 5% higher than the original time required by standard methodology but the grid quality is drastically improved.

4. COMPUTATIONAL RESULTS

4.1. Airfoil deformation

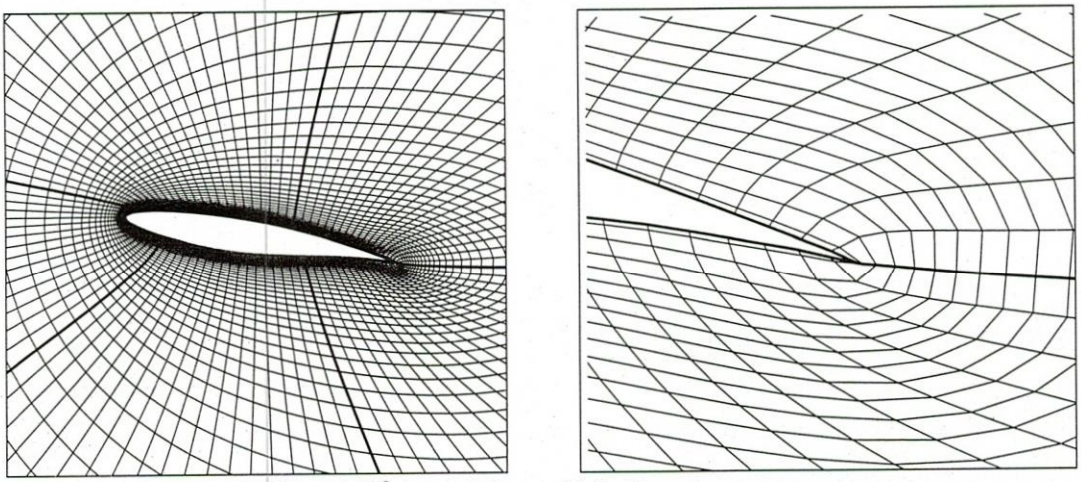
The following test cases demonstrate the efficiency and the robustness of developed grid deformation code. The performance of the developed grid deformation code is first demonstrated on the grid around RAE2822 airfoil. The O-typed initial grid generated by commercial package GRIDGEN® has 5 blocks with 95790 grid points, and 85260 cells (see Figure 3(a)). In addition to this initial grid, information concerning the grid topology is required as input for grid deformation program.

To evaluate the usability of this code for design optimization problem, one tries to adapt the grid for RAE2822 airfoil from the grid originally generated for NACA2412 airfoil. Figure 3(a) shows the grid around NACA2412 airfoil and Figure 3(b) is the grid around RAE2822 airfoil obtained by simply replacing NACA2412 airfoil by RAE2822 airfoil into the original grid. The grid update takes only several seconds on a common desktop.



(a) NACA2412 airfoil (b) RAE2822 airfoil

Figure 3. Multi-block grids around airfoil: five blocks, close-up view

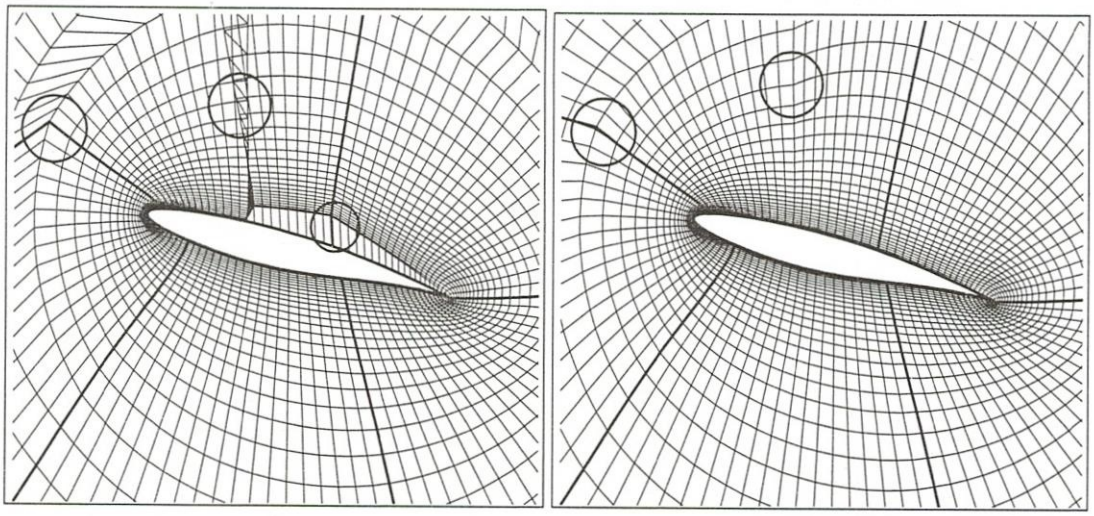


(a) RAE2822 with 10^0 dgree pitch up (b) Trailing edge

Figure 4. RAE2822 mesh with 10^0 pitch up: five blocks, close-up view and detail at the trailing edge

To evaluate the performance of this code, a more difficult situation is tested. RAE2822 airfoil is now rotated 10^0 around its quarter line. The grid around new configuration can be updated within several seconds (see Figure 4(a)). In Figure 4(b), the close-up view at the trailing edge shows that there is no cross-over of cells for this case. In multi-block structured grid deformation concept, the matching between two blocks is a critical problem. Figure 4(a) and 3(b) show that grid lines are perfectly matched at block-to-block interfaces. These results confirm that the approach suggested by Tsai et al. [1] automatically guarantees the matching between blocks interfaces. This is however not the case if grid topology includes sub-faces, especially when block face is constituted by solid wall patches and non-solid patches. In these cases, the standard algorithm suggested by Tsai et al [1] can give inadequate result as shown in Figure 5(a). One can observe clearly in Figure 5(a), non-matching between blocks interfaces with sub-faces. Because only solid-type patches of

block face is deformed when applying TFI, the discontinuity occurs at the transition between solid and non-solid patches. This discontinuity will result in the inversion of mesh cells. In this study, in order to solve this non-matching problem, TFI method is applied to sub-faces rather than block face. Figure 5(b) shows the final grid obtained by using new technique is free of discontinuity and non-matching problems.

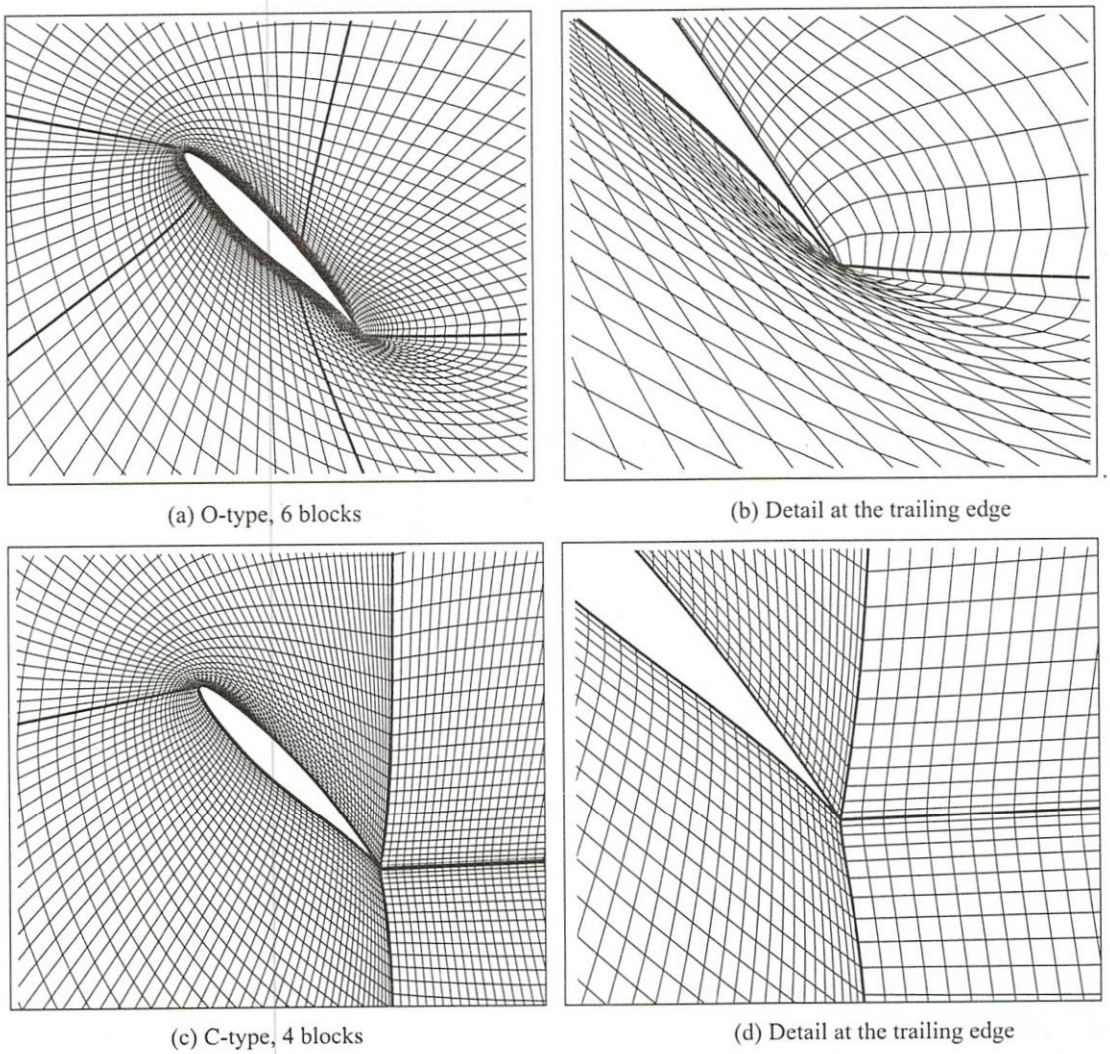


(a) Standard TFI method (b) Modified TFI method

Figure 5. RAE2822 mesh with 10^0 pitch up: five blocks (topology with sub-faces)

Figure 6(a) shows another case, the grid update for RAE2822 airfoil after a pitch up of 45^o. In this case, O-type grid topology was used. The deformed grid is visibly subjected to a crossover at the trailing edge (see Figure 6(b)). This can be avoided if C-grid topology is used. The detail at the trailing edge presented in Figure 6(d) shows a high quality grid without any crossover. These results clearly demonstrate that the quality of final grid partially depends on the grid topology originally adopted. This is understandable, since the spring analogy is used to determine the movement of block vertices before applying TFI. Further study is under progress to elevate grid crossover problem for large deformation problem.

To evaluate the robustness of current code, more critical situations are tested. Figure 7 demonstrates the grid update for RAE2822 airfoil Navier-Stokes-typed mesh with 10^0 pitch up. For Navier-Stokes calculations, where the mesh near the solid wall must be refined to resolve the high gradients of flow properties in these regions, the first mesh point's distance to the solid wall is order of 10^{-6} mm for commonly encountered aerodynamic problems. To handle these fine grids are a delicate problem. Figure 7 however shows that the code can be used equally well for Navier-Stokes mesh. The close-up view of trailing edge region shows no cross-over of mesh cells.



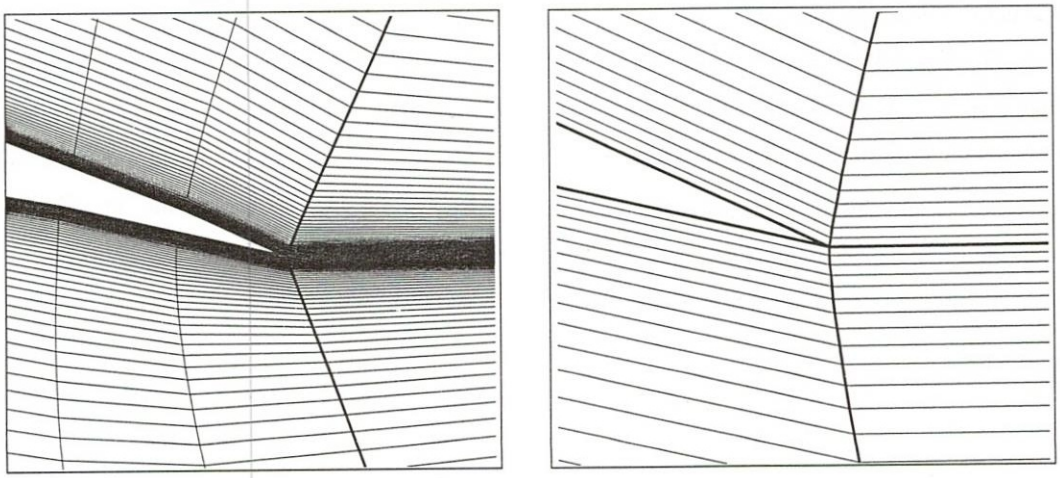
(a) O-type, 6 blocks

(b) Detail at the trailing edge

(c) C-type, 4 blocks

(d) Detail at the trailing edge

Figure 6. RAE2822 mesh with 45^0 pitch up with different topology



(a) Close-view at the trailing edge

(b) Detail at the trailing edge

Figure 7. RAE2822 Navier-Stokes mesh with 10^0 pitch up

4.2. DLR-F4 wing body deformation

This code has been also successfully tested for complex three-dimensional multi-block structured grids. Following is the deformation of grid around DLR-F4 wing-body configuration, which is used to evaluate the accuracy of Navier-Stokes solvers in the frame of AIAA CFD Drag Prediction Workshop. This grid has 24 blocks with 216678 grid points. The topology of grid generated by GRIDGEN package is shown in Figure 8.

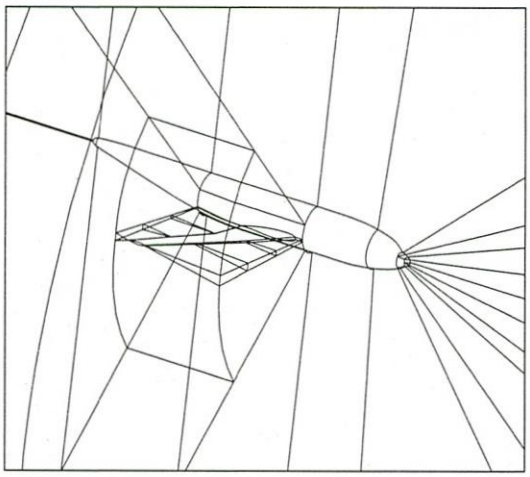
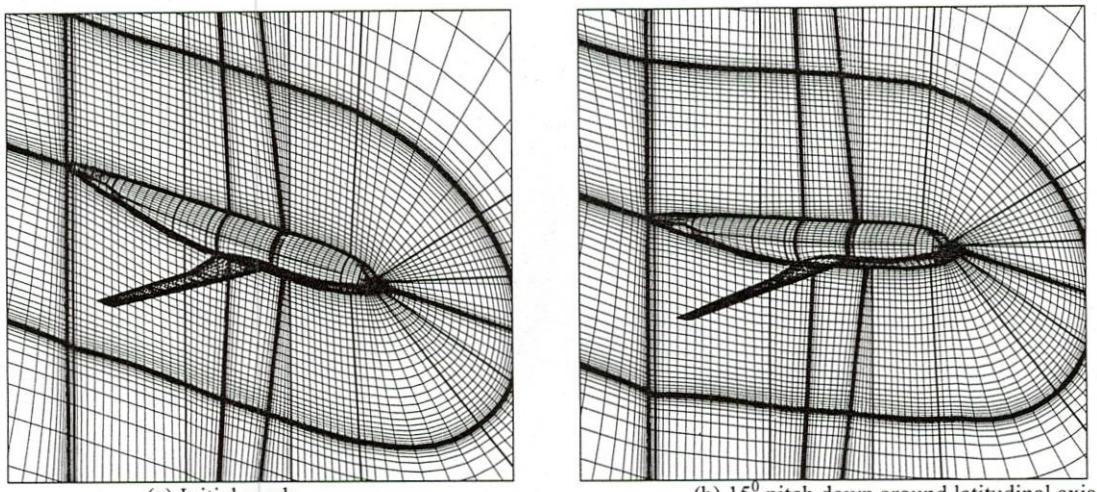


Figure 8. DLR-F4 wing body topology and mesh: 24 blocks, close-up view

Figure 9(b) shows the deformed grid in which the wing-body configuration rotates about its latitudinal axis by 15^0. This result shows that this code can successfully update the grid of complex configuration with arbitrary grid topology. In this case, the advantage of grid deformation is demonstrated clearly. It takes about 2-3 weeks to generate the initial grid but it needs only 40 seconds to determine the deformed grid on a desktop.

Figure 10 and Figure 11(a) show the detail of this deformed grid at the nose and tail of body. As mentioned in above sections, TFI method does not ensure the grid smoothness and orthogonality at the block interfaces with sub-faces. Figure 11(a) shows that there is some distortion in grid cell near the tail of wing body. In this study, the elliptic differential equation is applied as the smoothing operator to solve this problem. Figure 11(b) shows the final grid after applying the elliptic solver. It is clear that, with elliptic smoothing operator, the quality of deformed grid is drastically improved. In this case, the application of elliptic smoothing operator increases the computational time to 5%.



(a) Initial mesh (b) 15^0 pitch down around latitudinal axis

Figure 9. DLR-F4 wing body mesh

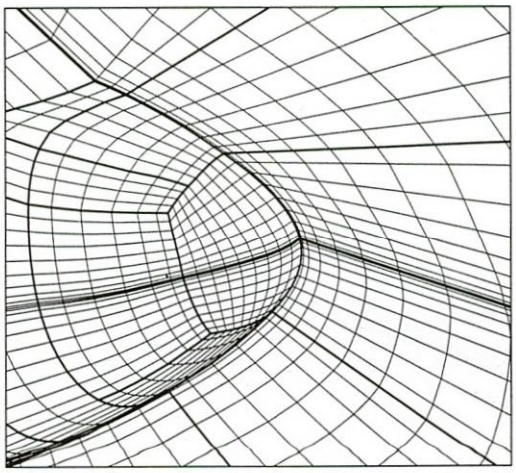
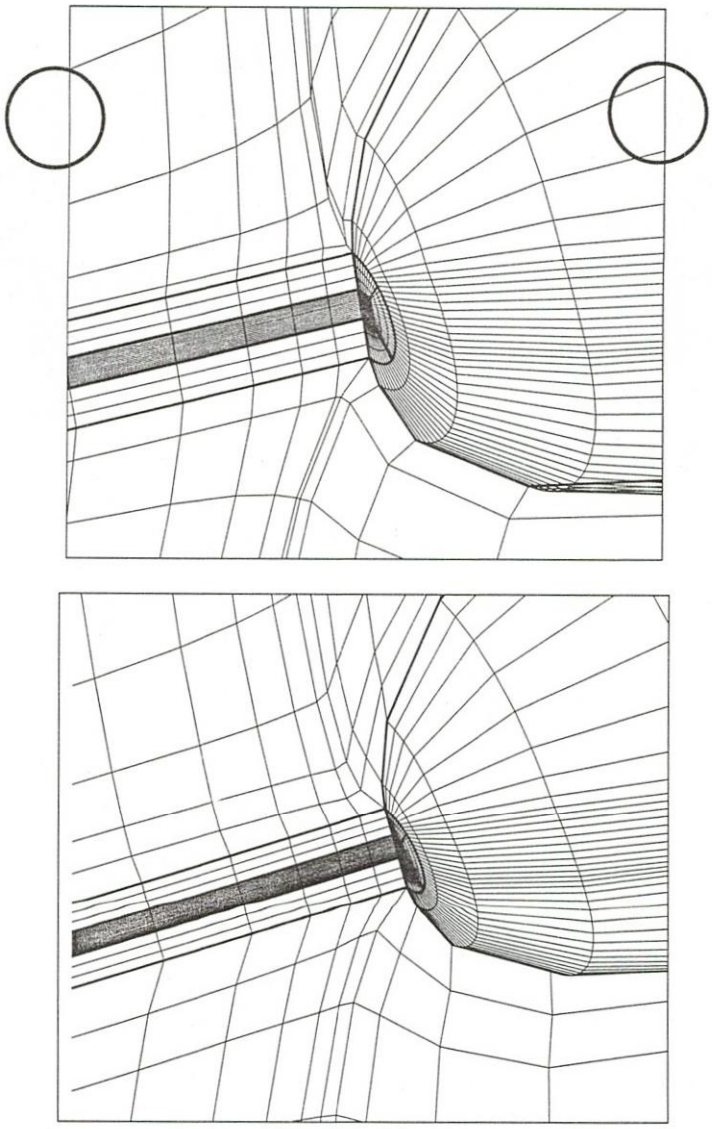


Figure 10. Detail of grid in the nose region of DLR-F4 wing body configuration



(a) Without smoothing operator (b) With elliptic smoothing operator

Figure 11. Detail of grid in the tail region of DLR-F4 wing body configuration

5. CONCLUSION

A deformation grid code has been developed and tested for two and three dimensional multi-block structured grid. This code, which is based upon a hybrid of algebraic and iterative methods, is demonstrated to be very efficient and robust enough for moderate deformation. The deformed grid still maintains the qualities of the initial grid such as smoothness and skewness. Because spring analogy is used for computing the deformation of all blocks' vertices and TFI technique is separately applied to the volume grid points (without having to gather all grid data on a processor), this code is easily to be applied for distributed computing context. This method also guarantees automatic matching of edges and surfaces between two blocks. Some modifications such as elliptic smoothing

operator (with only one or two iterations) and TFI for sub-faces are implemented to improve the quality of the deformed grid. It has been shown that adding smoothing operator does not penalize the computational time so much while the quality of deformed grid is drastically enhanced. Further researches have been under developing to improve the robustness of current code for large deformation problems.

Acknowledgement: This research work is partially supported by Vietnam's National Foundation for Science and Technology Development (NAFOSTED) (Grant #107.03.30.09) and by Korea Research Foundation Grant No. KRF-2005-005-J09901 and the 2nd Stage Brain Korea 21 project.

XÂY DỰNG CHƯƠNG TRÌNH BIẾN DẠNG LƯỚI CẤU TRÚC ĐA KHỐI BA CHIỀU ÁP DỤNG CHO CÁC CẤU HÌNH PHỨC TẠP

Hoàng Ánh Dương [(1)] , Nguyễn Anh Thi [(2)]

(1) Đại Học Quốc Gia Gyeongsang, Hàn Quốc

(2) Đại học Bách Khoa, ĐHQG-HCM

TÓM TẮT: *Trong nghiên cứu này, chương trình biến dạng lưới dựa trên giải thuật lai xây dựng trên cơ sở hai giữa giải thuật TFI và giải thuật tương tự lò xo đã được phát triển. Kết hợp giữa phương pháp tương tự lò xo ứng dụng cho các đỉnh của các khối và TFI cho các điểm nội của các khối giúp gia tăng độ bền vững của giải thuật. Đồng thời giải thuật sử dụng thích ứng cho ứng dụng trong môi trường tính toán phân bố. Toán tử làm trơn dạng elliptic được áp dụng cho các mặt của khối được làm bởi nhiều mảnh con nhằm bảo đảm tính trơn của lưới, đồng thời giảm sự nhọn hóa của lưới. Khả năng của chương trình phát triển đã được minh chứng cho một số trường hợp biến dạng từ đơn giản đến phức tạp.*

Từ khóa: *giải thuật TFI, chương trình biến dạng lưới*

REFERENCES

[1]. [H. M. Tsai, A. S. F. Wong, J. Cai, Y. Zhu and F. Liu, *Unsteady flow calculation with a parallel moving mesh algorithm*, *AIAA Journal*, Vol. 39, No. 6, pp. 1021-1029 (2001)

[2]. S. P. Spekreijse, B. B. Prananta and J. C. Kok, *A simple, robust and fast algorithm to compute deformations of multi-block structured grids*, *Technical Report - National Aerospace Laboratory NLR*, NLR-TP-2002-105 (2002)

[3]. M. A. Postdam and G. P. Guruswamy, *A parallel multi-block mesh movement*

scheme for complex aeroelastic application, AIAA Paper, AIAA-2001-0716 (2001)

[4]. F. J. Blom, *Considerations on the spring analogy, International Journal for Numerical Methods in Fluid*, Vol. 32, pp. 647-668 (2000)

[5]. D. Zeng and C. R. Ethier, *A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains, Finite Elements in Analysis and Design*, Vol. 41, pp. 1118-1139 (2005)

[6]. G. A. Markou, Z. S. Mouroutis, D. C. Charmpis, M. Papadrakakis, *The ortho-semi-torsional (OST) spring analogy method for 3D mesh moving boundary problems, Computer Methods in Applied Mechanics and Engineering*, Vol. 196, pp. 747-765 (2006)

[7]. L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben, K. J. Badcock and B. E. Richards, *A grid deformation technique for unsteady flows computation, International Journal for Numerical Method in Fluids*, Vol. 32, pp/ 285-311 (2000)

[8]. R. E. Bartels, *Finite macro-element mesh deformation in structured multi-block Navier-Stokes code, Technical Note – NASA,* NASA/TM-2005-213789 (2005)

[9]. J. A. Samareh, *Application of quaternions for mesh deformation, Technical Note – NASA,* NASA/TM-2002-211646 (2002)

[10]. J.-A. Désidéri and A. Janka, *Multilevel shape parameterization for aerodynamic optimization – Application to drag and noise reduction of transonic/supersonic jet, European Congress on Computational Methods in Applied Sciences and Engineering (2004)*

[11]. J. T. Batina, *Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA Journal*, Vol. 28, No. 8, pp. 1381-1388 (1990)

[12]. Oyama, *Wing design using evolutionary algorithms, Ph.D. Thesis*, Department of Aeronautical and Space Engineering, Tohoku University (2000)

[13]. G. O, *Shape optimization for two-dimensional transonic airfoil by using the coupling of FEM and BEM*, Ph.D. Thesis, Department of Mathematics, University of Stuttgart (2006)

[14]. M. Farrashkhalvat and J. P. Miles, *Basic structured grid generation with an introduction to unstructured grid generation,* Butterworth-Heinemann (2003)

[15]. J. F. Thompson, Bharat K. Soni, Nigel P. Weatherill, *Handbook of grid generation*, CRC Press (1999)

[16]. T. J. Chung, *Computational fluid dynamics*, Cambridge University Press (2002)

[17]. AGARD, 1994, *A selection of experimental test cases for the validation of CFD codes*, AGARD-AR-303, Vol. II (1994)