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

A.D. Hoang,[1] Y.M. Lee,[1] S.K. Jung,[1] A.T. Nguyen[2] and R.S. Myong*[3]

복잡한 형상에 관한 삼차원 변형 Multi-Block 정렬격자 프로그램 개발

A.D. Hoang,[1] 이 영 민,[1] 정 성 기,[1] A.T. Nguyen,[2] 명 노 신*[3]

In this study, a multi-block structured grid deformation code based on a hybrid of a transfinite interpolation algorithm and spring analogy was developed. The configuration was modeled by a Bezier surface. A combination of the spring analogy for block vertices and the transfinite interpolation for interior grid points helps to increase the robustness and makes it suitable for distributed computing. An elliptic smoothing operator was applied to the block faces with sub-faces in order to maintain the grid smoothness and skewness. The capability of this code was demonstrated on a range of simple and complex configurations including an airfoil and a wing-body configuration.

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

1. INTRODUCTION

Numerical simulation of unsteady flow with multi-block structured grids 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 the updating of the computational grid at each time step. The new grid can either be regenerated or dynamically updated. The first approach is a natural choice that consists of regenerating the computational grid at each time step during time integration. However, a grid generation for a complex configuration is by itself a nontrivial and time-consuming task. Although some robustness problems for large deformation remain to be solved, the dynamic grid is inexpensive and appropriate for practical problems.

접수일: 2007년 8월 30일, 심사완료일: 2007년 11월 9일.
1 경상대학교 대학원 기계항공공학부 항공우주공학전공
2 Ho Chi Minh City University of Technology, Vietnam
3 종신회원, 경상대학교 기계항공공학부 및 항공기부품기술연구소
* Corresponding author, E-mail: myong@gnu.ac.kr

The development of an efficient and robust grid deformation methodology that can maintain the quality of the initial grid generated by a commercial grid generation package has been 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. 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 to deform the grid by redistributing grid points along grid lines that are in the normal direction of the surface[1]. The transfinite interpolation (TFI) method had also been used to regenerate a structured grid. A detail analysis of TFI method and pros and cons of this method for multi-block structured grids were given by Dubuc et al.[2]. Algebraic methods are fast but work well only for small deformation[3]. Large deformation can cause a crossover of grid lines or produce a poor quality grid. A spring analogy method initially proposed by Nakahashi and Deiwert was applied to aero-elasticity problems by Batina[4]. A comparison of the spring analogy and an elliptic grid generation was presented by Bloom[5]. It is

well known that the standard spring analogy will result in an inversion of elements for large deformation. To overcome this drawback, numerous schemes such as torsional, semi-torsional and ortho-semi-torsional spring analogies were suggested[6,7]. This method as well as the elastic analogy can adapt to significant surface deformations but their computational cost is expensive for complex problems with a large number of grid points. In addition, it has been widely applied to unstructured grid deformation[8].

The hybrid approach, a useful compromise between algebraic and iterative approaches, has been proposed in the recent years. 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.[3] introduced a new methodology that replaces the spring analogy using the volume spline interpolation. Although these schemes provide relatively good results, a major drawback remains involving the sub-faces problem. To overcome this disadvantage, Potsdam and Guruswamy[9] proposed a point-by-point methodology. Instead of computing the displacement of block vertices, the nearest surface point, the surface spline, the blending function and the decay function are combined to define the deformed surfaces of a block. In order to improve the orthogonality of the grid lines near the configuration surfaces, Samareh[8] introduced a quaternion methodology. Although many algorithms were developed, considerable efforts have been devoted to the development of robust and efficient techniques for grid deformation. Bartels[10] proposed a new methodology that combines a definition of material properties and transfinite interpolation to generate a deformed mesh.

Another important problem with multi-block structured grid deformation is the handling of blocks, in general connected in an unstructured fashion, in a distributed computing context wherein the blocks are typically 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 is to ensure 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] was developed. This algorithm is a combination of the spring analogy and TFI methods. It is easy to implement this algorithm in a distributed parallel computing context. In the first step, the configuration surface is parameterized using a Bezier surface. The second step consists of determining the displacement of all block corner points by using the spring analogy. In general, the number of blocks, and hence the number of vertices are far fewer than the volume grid points implying that the computational cost for this step is small. Once new coordinates of the corner points are determined, the TFI method is used to compute the deformation of edge, face and volume grid points in each block separately. The previous approach does not ensure the quality of block faces that are constituted by several patches with different boundary conditions. To solve this problem, instead of block faces, the TFI method is applied to each patch of block faces. An 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 on the interfaces.

In the next section, the shape parameterization, the spring analogy technique, and then the arc-length-based TFI technique are presented. Various numerical results of grid deformation of several simple and complex configurations such as an airfoil and a wing-body configuration are presented in order to demonstrate the capability of the proposed grid deformation code.

2. SHAPE PARAMETERIZATION

In a design optimization problem, parameterization of the configuration is one of crucial areas of concern. It is necessary to compromise between the accuracy of a parameterization technique and the number of required parameters. Among the approximation techniques, a Bezier curve/surface is one of the most popular approaches. The design parameters for this case are the positions of the control points of the Bezier curves.

A Bezier curve/surface[11] in R^d ($d=2$ or 3) of degree n supported by a control polygon of $n+1$ control points $p_k \in R^d$ (with $k=0,1,\cdots,n$) is

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

Here, $B_n^k(t)$ is the Bernstein polynomial $B_n^k(t)=$

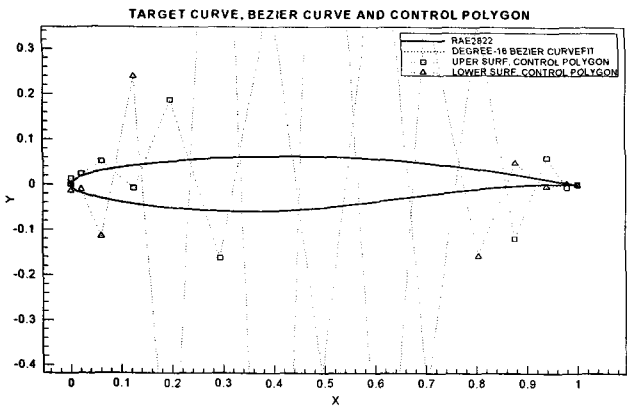


Fig. 1 RAE2822 airfoil, 16-degree Bezier curve-fits, and control polygons of upper and lower curves

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

A procedure used to compute the coordinate of the control points from configuration surfaces is proposed by O[12]. The formula of the Bezier curve can be written in a matrix form

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

Multiplying the transpose of matrix B to this equation yields

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

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

To demonstrate the capability of this approximation method, Bezier curves were used to represent the upper and lower curves of the RAE2822 airfoil. Seventeen control points are used for each curve. The condition in which the first and last control points of two Bezier curves are identical ensures the coincidence of the two curves.

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

$$TOL = \sum_{i=1}^{n} \frac{\sqrt{(x_B - x_i)^2 + (y_B - y_i)^2}}{N} \tag{4}$$

in which N is a number of discrete points of the airfoil. In this example the tolerance is approximately 10^{-3}. It was demonstrated that this error is adequate for an optimization design.

While this method offers an acceptable accuracy and a small number of required parameters, a minor drawback exists nonetheless. If a design surface is represented by a number of patches, the matching between these patches must be guaranteed. Due to the computational error associate with the Bezier surface, it is not feasible for this problem. In order to solve the matching problem, an averaging of mesh point coordinates between two adjacent patches should be applied to eliminate the computational error.

3. MULTI-BLOCK STRUCTURED GRID DEFORMATION APPROACH

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

The first step of the hybrid method used in this study consists of computing the displacement of all vertices of each block. In a multi-block structured grid context, the arrangement of blocks is generally unstructured so that the motion of these corner points is determined by the 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 the spring analogy[4] is adopted for determining the movement of the block vertices. Spring analogy models can be categorized into two types: vertex model and segment model. In this study, the segment model is applied to the grid deformation code. The corner points are viewed as a network of fictitious springs with the stiffness defined as follows

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

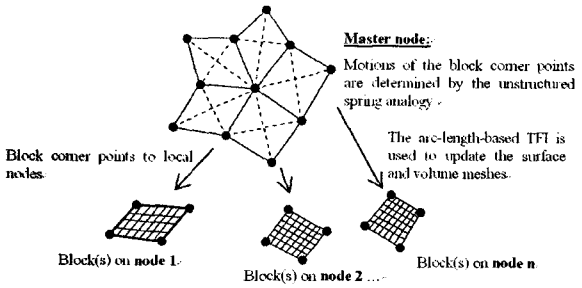


Fig. 2 A suitable strategy for parallel multi-block structured grid deformation

Spring stiffness is computed for all 12 edges and 4 cross-diagonal edges of a block. These cross-diagonal edges are utilized in the control of the shearing motion of the grid cells. The coefficients λ and β are used to control the stiffness of the grid cells. Typically, the coefficients λ and β are assumed to be 1 and 0.5, respectively, which implies that the stiffness is inversely proportional to the length of the 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

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

The static equilibrium equations are iteratively solved as follows

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

3.2 TRANSFINITE INTERPOLATION (TFI)

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

- Parameterize all grid points.
- Compute the grid point deformations by using the one-, two- and three-dimensional arc-length-based TFI techniques
- Add the deformations to the original grid to obtain the 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

$$X^B = \{\overrightarrow{x_{i,j,k}} \mid i = 1,\cdots,imax; \quad j = 1,\cdots,jmax; k = 1,\cdots,kmax\}$$

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

$$s_{1,j,k} = 0, \quad s_{i,j,k} = s_{i-1,j,k} + \sqrt{\left(x_{i,j,k} - x_{i-1,j,k}\right)^2 + \left(y_{i,j,k} - y_{i-1,j,k}\right)^2 + \left(z_{i,j,k} - z_{i-1,j,k}\right)^2}$$
$$F_{i,j,k} = \frac{s_{i,j,k}}{s_{i\max,j,k}} \tag{8}$$

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

The second stage involves computing the displacement of the edges, surfaces and block points based on the one-, two- and three-dimensional TFI formulas, respectively. From the displacement of the configuration surfaces, the interpolated values of the deformation are created by using the TFI method, allowing the new grid obtained by adding the deformations to the initial mesh to maintain the quality of the original grid.

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

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

Here, $\triangle P$ is the displacement of the two corner points of the block edge. The displacement of the block surface (for example, the surface in the plane $k = 1$) is computed by the two-dimensional TFI formula

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

After computing the deformation of all surfaces and edges, the standard three-dimensional TFI formula[13] 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

$$\begin{aligned}
V1 &= (1-F_{i,j,k})\Delta S_{1,j,k} + F_{i,j,k}\Delta S_{i\max,j,k} \\
V2 &= (1-G_{i,j,k})\Delta S_{i,1,k} + G_{i,j,k}\Delta S_{i,j\max,k} \\
V3 &= (1-H_{i,j,k})\Delta S_{i,j,1} + H_{i,j,k}\Delta S_{i,j,k\max} \\
V12 &= (1-F_{i,j,k})(1-G_{i,j,k})\Delta E_{1,1,k} \\
&\quad +(1-F_{i,j,k})G_{i,j,k}\Delta E_{1,j\max,k} \\
&\quad +F_{i,j,k}(1-G_{i,j,k})\Delta E_{i\max,1,k} \\
&\quad +F_{i,j,k}G_{i,j,k}\Delta E_{i\max,j\max,k} \\
V13 &= (1-F_{i,j,k})(1-H_{i,j,k})\Delta E_{1,j,1} \\
&\quad +(1-F_{i,j,k})H_{i,j,k}\Delta E_{1,j,k\max} \\
&\quad +F_{i,j,k}(1-H_{i,j,k})\Delta E_{i\max,j,1} \\
&\quad +F_{i,j,k}H_{i,j,k}\Delta E_{i\max,j,k\max} \\
V123 &= (1-F_{i,j,k})(1-G_{i,j,k})(1-H_{i,j,k})\Delta P_{1,1,1} \\
&\quad +(1-F_{i,j,k})(1-G_{i,j,k})H_{i,j,k}\Delta P_{1,1,k\max} \\
&\quad +(1-F_{i,j,k})G_{i,j,k}(1-H_{i,j,k})\Delta P_{1,j\max,1} \\
&\quad +(1-F_{i,j,k})G_{i,j,k}H_{i,j,k}\Delta P_{1,j\max,k\max} \\
&\quad +F_{i,j,k}(1-G_{i,j,k})(1-H_{i,j,k})\Delta P_{i\max,1,1} \\
&\quad +F_{i,j,k}(1-G_{i,j,k})H_{i,j,k}\Delta P_{i\max,1,k\max} \\
&\quad +F_{i,j,k}G_{i,j,k}(1-H_{i,j,k})\Delta P_{i\max,j\max,1} \\
&\quad +F_{i,j,k}G_{i,j,k}H_{i,j,k}\Delta P_{i\max,j\max,k\max}
\end{aligned} \quad (12)$$

3.3 Smooth Operator : Elliptic Differential Equation

There are cases in which only a certain portion of a surface is extremely distorted. To accommodate such problems, a smooth operator is locally applied to alleviate the distortion. In this study, an elliptic differential equation is used to smooth the deformed grid[14].

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

with

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

$$\begin{aligned}
a_{11} &= x_\xi^2 + y_\xi^2 + z_\xi^2 \\
a_{22} &= x_{\eta\eta}^2 + y_{\eta\eta}^2 + z_{\eta\eta}^2 \\
a_{12} &= x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta \\
x_\xi &= 0.5(x_{i+1,j} - x_{i-1,j}) \\
x_\eta &= 0.5(x_{i,j+1} - x_{i,j-1}) \\
x_{\xi\xi} &= x_{i+1,j} - 2x_{i,j} + x_{i-1,j} \\
x_{\eta\eta} &= x_{i,j+1} - 2x_{i,j} + x_{i,j-1} \\
x_{\xi\eta} &= 0.25(x_{i+1,j+1} - x_{i+1,j-1} - x_{i-1,j+1} + x_{i-1,j-1})
\end{aligned} \quad (14)$$

An elliptic operator is used only for the sub-faces to eliminate possible distortion after applying the TFI method. To maintain the efficiency of this code, only one or two elliptic smoothing iterations are used. As the TFI method has previously been used at this point, one or two iterations are sufficient to enhance the smoothness of the deformed grid. When the elliptic smoothing operator is applied, the computational time is in general only 10% higher than the original time required by the standard methodology. However, the grid quality is significantly improved.

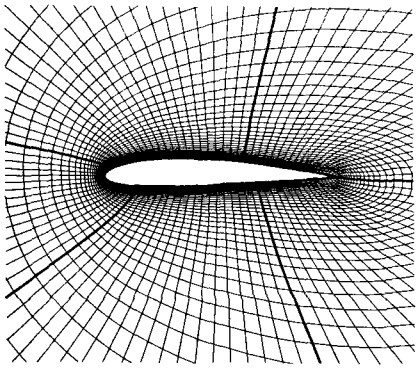
4. Computational Results

4.1 Airfoil Deformation

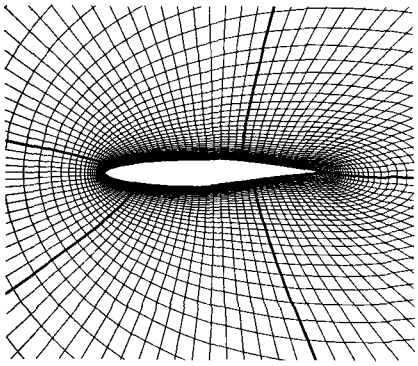
The following test cases demonstrate the efficiency and the robustness of the proposed grid deformation code. The performance of this code is first demonstrated on a grid around the RAE2822 airfoil. An O-typed initial grid generated by the commercial package GRIDGEN has 5 blocks with 95,790 grid points, and 85,260 cells (see Figure 3(b). In addition to this initial grid, the information concerning the grid topology is required as an input for the grid deformation program.

To evaluate the usability of this code for the design optimization problem, it is necessary to adapt the grid for the NACA2412 airfoil from the grid originally generated for the RAE2822 airfoil. Figure 3(b) shows the grid around the RAE2822 airfoil, while Figure 3(a) shows the grid around the NACA2412 airfoil obtained by simply replacing the RAE2822 airfoil with the NACA2412 airfoil in the original grid. The grid update requires only several seconds on a common desktop computer.

To evaluate the performance of this code, a more difficult situation was tested. The RAE2822 airfoil in this test was rotated 10° around its quarter line. The grid around new configuration could be updated in seconds

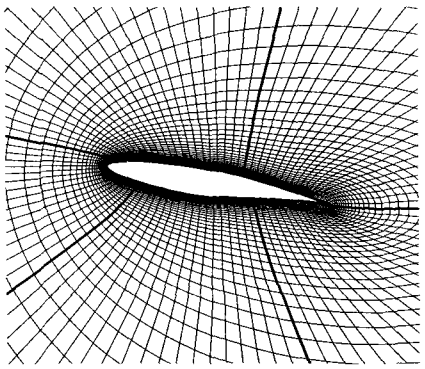


a) NACA2412

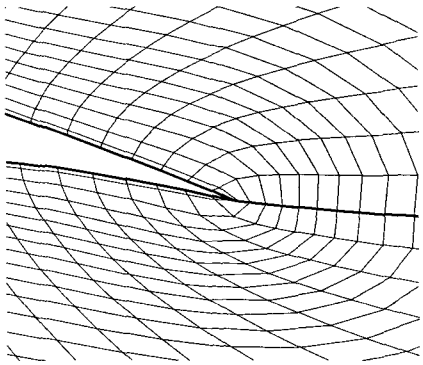


b) RAE2822

Fig. 3 Multi-block grids around an airfoil: 5 blocks in a close-up view



a) RAE2822 with 10° pitch up



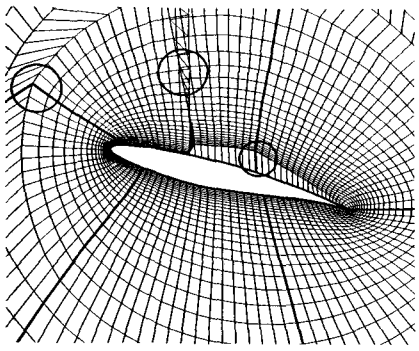
b) Detail at the trailing edge

Fig. 4 RAE2822 mesh with 10° pitch up: 5 blocks in a close-up view and detail at the trailing edge

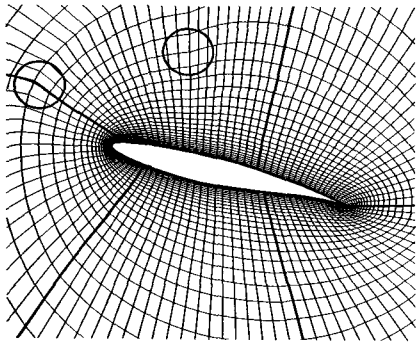
(see Figure 4(a)). In Figure 4(b), the close-up view at the trailing edge shows that there is no cross-over of cells for this case. In the multi-block structured grid deformation concept, the matching between two blocks is a critical problem. Figure 4(a) and 3(a) show that grid lines are perfectly matched at block-to-block interfaces. These results confirm that the approach suggested by Tsai et al.[1] automatically guarantees the matching on the block interfaces. This is, however, not the case if the grid topology includes sub-faces, especially when the block face consists of solid patches and non-solid patches. In these cases, the standard algorithm suggested by Tsai et al[1] can give an inadequate result as shown in Figure 5(a). As can be observed clearly in Figure 5(a), non-matching between block interfaces with sub-faces exists. As only solid-type patches of block faces are deformed when applying the TFI method, 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, the TFI method was applied to sub-faces rather than to block face. Figure 5(b) shows that the final grid obtained by using the new technique is free of discontinuity and non-matching problems.

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

To evaluate the robustness of the current code, a more critical situation was tested. Figure 7 demonstrates the grid update for the RAE2822 airfoil Navier-Stokes-typed mesh with 10° pitch up. For Navier-Stokes calculations, where the mesh near the solid wall must be refined to resolve the high gradients of the flow properties in these regions, the distance to the solid wall of the first mesh point is in the order of 10^{-6} mm for commonly encountered



a) Standard TFI method



b) Modified TFI method

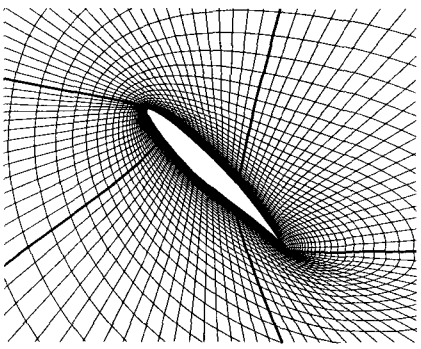
Fig. 5 RAE2822 mesh with 10° pitch up: five blocks (topology with sub-faces)

aerodynamic problems. Handling these fine grids is a delicate problem. Figure 7 however shows that the code can be used equally well for a Navier-Stokes mesh. A close-up view of the trailing edge region shows no cross-over of the mesh cells.

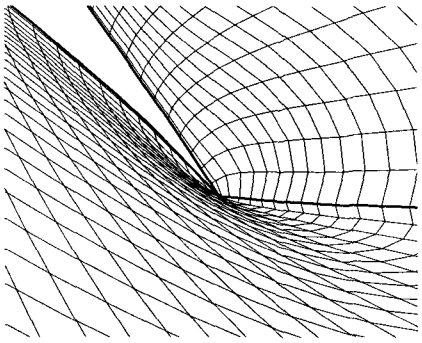
4.2 DLR-F4 WING BODY DEFORMATION

This code was also successfully tested for complex three-dimensional multi-block structured grids. The following outlines the deformation of a grid around the DLR-F4 wing-body configuration, which was used to evaluate the accuracy of the Navier-Stokes solvers in the frame of the AIAA CFD Drag Prediction Workshop[15]. This grid has 24 blocks with 216,678 grid points. The topology of a grid generated by the GRIDGEN package is shown in Figure 8.

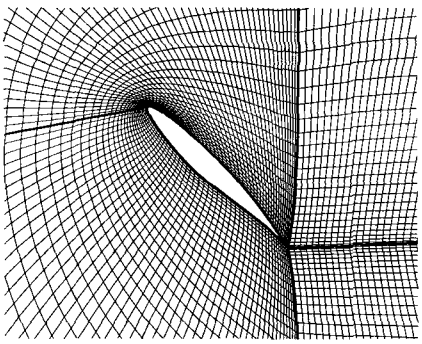
Figure 9(b) shows the deformed grid in which the wing-body configuration rotates about its latitudinal axis by 15°. This result shows that the current code can successfully update the grid of a complex configuration with an arbitrary grid topology. In this case, the advantage of the grid deformation code is demonstrated clearly. It takes approximately 2-3 weeks to generate the initial grid



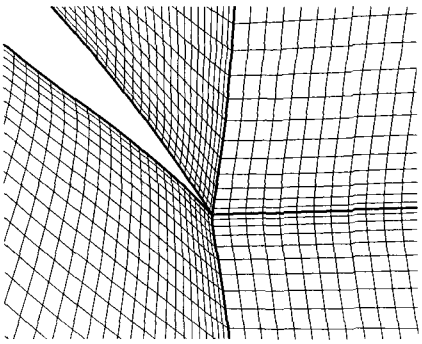
a) O-type, 6 blocks



b) Detail at the trailing edge

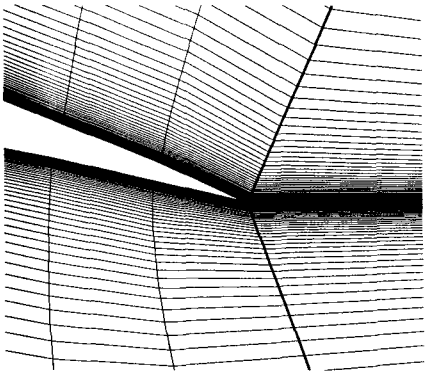


c) C-type, 4 blocks

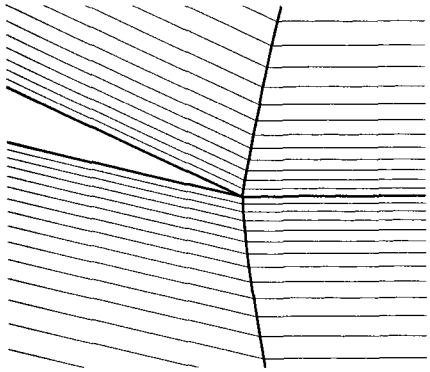


d) Detail at the trailing edge

Fig. 6 RAE2822 mesh with a 45° pitch up with different topology



a) Close-up view at the trailing edge



b) Detail at the trailing edge

Fig. 7 RAE2822 Navier-Stokes mesh with a 10° pitch up

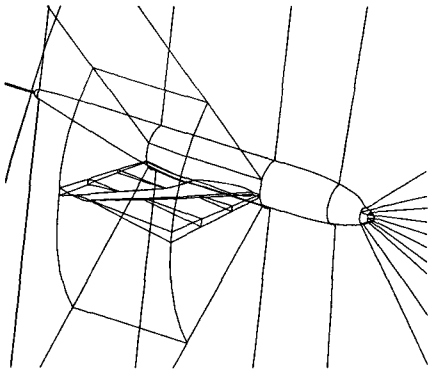
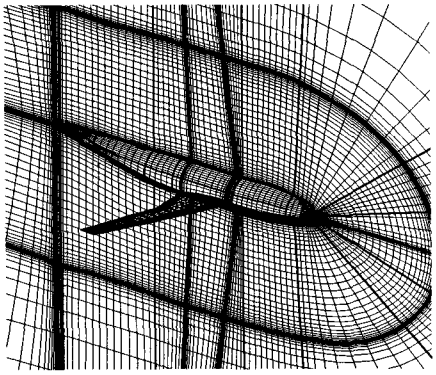
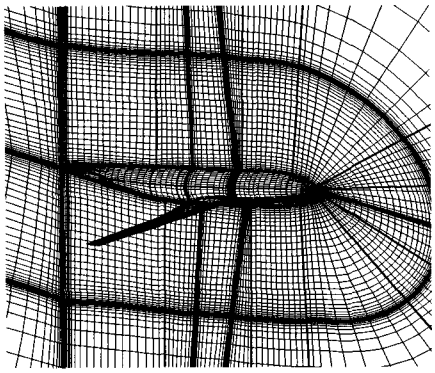


Fig. 8 DLR-F4 wing body topology: 24 blocks, close-up view



a) Initial mesh



b) 15° pitch down around latitudinal axis

Fig. 9 DLR-F4 wing body mesh

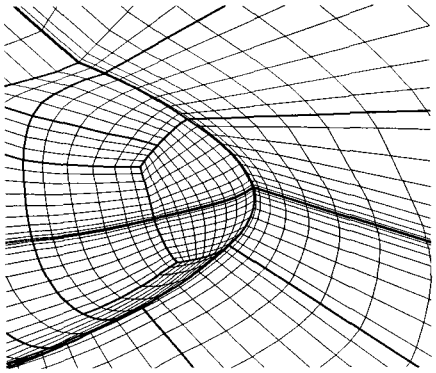


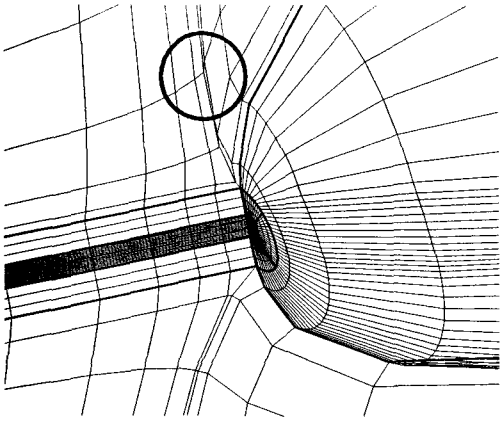
Fig. 10 Detail of grid in the nose region of DLR-F4 wing body configuration

but requires only 40 seconds to get the deformed grid on a desktop computer.

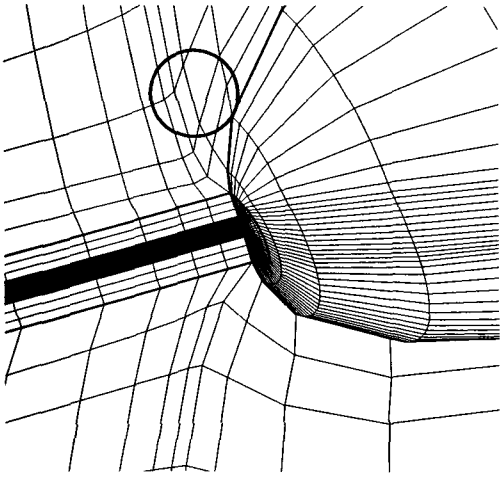
Figure 10 and Figure 11(a) show the details of this deformed grid at the nose and tail of the wing-body. As mentioned in above sections, the TFI method does not ensure the grid smoothness and orthogonality at the block interfaces with sub-faces. Figure 11(a) shows that there is some distortion in the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

5. CONCLUSION

A grid deformation code was developed and tested for



a) Without the smoothing operator



b) With the elliptic smoothing operator

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

the two- and three-dimensional multi-block structured grids. This code, which is based on a hybrid of the algebraic and iterative methods, was demonstrated to be very efficient and sufficiently robust for moderate deformation. The deformed grid still maintains the qualities of the initial grid including the smoothness and skewness. As the spring analogy is used to compute the deformation of all block vertices and the TFI technique is applied separately to the volume grid points (without the need to gather all grid data on a processor), this code can easily be applied in a distributed computing context. This method also guarantees the automatic matching of edges and surfaces between two blocks. Some modifications, including the elliptic smoothing operator (with only one or two iterations) and the modified TFI method for sub-faces, were implemented to improve the quality of the deformed grid. It was shown that addition of the smoothing operator does not increase the computational time much but does greatly improve the quality of the deformed grid. Further research is in progress in an effort to improve the robustness of the current code for large deformation problems.

ACKNOWLEDGEMENTS

This work was supported by Korea Research Foundation Grant No. KRF-2005-005-J09901 and the 2nd Stage Brain Korea 21 project.

REFERENCES

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

[2] 2000, Dubuc. L., Cantariti. F., Woodgate. M., Gribben. B., Badcock. K.J. and Richards. B.E., "A grid deformation technique for unsteady flows computation," *International Journal for Numerical Method in Fluids*, Vol.32, pp.285-311.

[3] 2002, Spekreijse. S.P., Prananta. B.B. and Kok. J.C., "A simple, robust and fast algorithm to compute deformations of multi-block structured grids," *National Aerospace Laboratory NLR*, NLR-TP-2002-105.

[4] 1990, Batina. J.T., "Unsteady Euler airfoil solutions using unstructured dynamic meshes," *AIAA Journal*, Vol.28, No.8, pp.1381-1388.

[5] 2000, Bloom. F.J., "Considerations on the spring analogy," *International Journal for Numerical Methods in Fluid*, Vol.32, pp.647-668.

[6] 2005, Zeng. D. and Ethier. C.R., "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.

[7] 2006, Markou. G.A., Mouroutis. Z.S., Charmpis. D.C. and Papadrakakis. M., "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.

[8] 2002, Samareh. J.A., "Application of quaternions for mesh deformation," *NASA/TM-2002-211646.*

[9] 2001, Postdam. M.A. and Guruswamy. G.P., "A parallel multi-block mesh movement scheme for complex aeroelastic application," *AIAA-2001-0716.*

[10] 2005, Bartels. R.E., "Finite macro-element mesh deformation in structured multi-block Navier-Stokes

code," *NASA/TM-2005-213789.*

[11] 2004, Désidéri. J.A. and Janka. A., "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.*

[12] 2006, O, C.G., "Shape optimization for two-dimensional transonic airfoil by using the coupling of FEM and BEM," *Ph.D. Thesis*, Department of Mathematics, University of Stuttgart.

[13] 1999, Thompson. J.F., Soni. B.K. and Weatherill. N.P., *Handbook of grid generation*, CRC Press.

[14] 2002, Chung. T.J., *Computational fluid dynamics*, Cambridge University Press.

[15] 1994, AGARD, "A selection of experimental test cases for the validation of CFD codes," *AGARD-AR-303*, Vol.II.