Comparison of Various Spring Analogy Related Mesh Deformation Techniques in 2-D Airfoil Design Optimization

Yosheph Yang and Serkan Özgen***
**Graduate Student in Dept. of Aerospace Engineering*
Middle East Technical University, Ankara 06800, Turkey
***Professor in Dept. of Aerospace Engineering*
Middle East Technical University, Ankara 06800, Turkey

Abstract

During the last few decades, CFD (Computational Fluid Dynamics) has developed greatly and has become a more reliable tool for the conceptual phase of aircraft design. This tool is generally combined with an optimization algorithm. In the optimization phase, the need for regenerating the computational mesh might become cumbersome, especially when the number of design parameters is high. For this reason, several mesh generation and deformation techniques have been developed in the past decades. One of the most widely used techniques is the Spring Analogy. There are numerous spring analogy related techniques reported in the literature: linear spring analogy, torsional spring analogy, semi-torsional spring analogy, and ball vertex spring analogy. This paper gives the explanation of linear spring analogy method and angle inclusion in the spring analogy method. In the latter case, two different solution methods are proposed. The best feasible method will later be used for 2-D Airfoil Design Optimization with objective function being to minimize sectional drag for a required lift coefficient at different speeds. Design variables used in the optimization include camber and thickness distribution of the airfoil. SU2 CFD is chosen as the flow solver during the optimization procedure. The optimization is done by using Phoenix ModelCenter Optimization Tool.

Nomenclature

$F_{x_{ij}}$	=	Force in x-direction along edge $i-j$
$F_{y_{ij}}$	=	Force in y-direction along edge $i-j$
k_{ij}	=	Spring stiffness for edge $i-j$
Δx_i	=	Increment in x-direction for node i
Δy_i	=	Increment in y-direction for node i
F_{x_i}	=	Force in x-direction exerted on node i
F_{y_i}	=	Force in y-direction exerted on node i
n_i	=	Number of neighbor nodes around node i
θ_{ij}	=	Angle made along edge $i-j$

1. Introduction

The presence of dynamic mesh concept has greatly evolved in this past decades. The idea to deform the mesh has been greatly developed in order to perform some computations easier, especially the ones dealing with aeroelasticity computations, airfoil oscillations, or even shape optimizations. In all those purposes, the mesh has to be updated according to the updated boundary domain. Lin, et al [1] categorized some methods that can be applied to update the mesh: remeshing, mesh deformation, and combination of both remeshing and mesh deformation. Compared to mesh deformation, the concept of remeshing is more expensive, since new mesh needs to be updated for each updated boundary domain case, especially for a complex boundary problem. On the other hand, the mesh deformation technique, the computational mesh is updated to the new displaced boundary without changing the mesh connectivity [2].

Mesh deformation technique reported in the literature, in general can be classified into several methods such as: partial differential equation (PDE) methods, physical analogies methods, and algebraic methods [2]. In the partial differential equation methods, Laplacian or bi-harmonic operators are generally used. The physical analogies [3] methods itself considers each edge of the mesh to behave as a spring, which has its own stiffness value. On the other hand, the algebraic methods compute the movement of grid nodes as a function of boundary nodes which has no actually attached physical meaning. Sample of algebraic methods that have been reported in the literature include: Inverse Distance Weighting Interpolation [4], Delaunay Mapping [5], and Radial Basis Function Interpolation [6]. Among these mesh deformation techniques, the physical spring analogy method is the most commonly used.

The mesh deformation technique with linear spring analogy has evolved since firstly introduced by Batina [3] for the 2-D unstructured mesh. Some improvement in the methods have been greatly developed such as: the torsional spring [7], semi-torsional spring [8], ball-vertex spring analogy [9]. The same authors have also conducted another research about improvement in spring analogy technique by using ball-center spring analogy [10].

In this paper, a detailed comparison between a spring analogy method and modified spring analogy method by considering angle into account is explained. Furthermore, in the case where edge angle is considered in the formulation, two different solution methods are presented. These two methods are later compared in terms of their accuracy and effectiveness.

From the developed methods used in the various spring analogy methods considered, the best viable method is used for an airfoil shape optimization. The airfoil shape optimization is accompanied by several basic design parameterization comprised of camber, thickness, and their combination. In the optimization procedure, the objective function is to minimize the sectional drag of an airfoil by specifying the required airfoil's lift coefficient. The optimization is conducted by the aid of Gradient Optimizer of Phoenix Model Center.

2. Spring Analogy Mesh Deformation

Based on the spring analogy mesh categorization by Blom [11], there are two major categorizations: vertex spring method and segment spring method. In the vertex method, the nodal position is considered, while on the segment method, the nodal displacement is considered. The first spring analogy developed by Batina [3] is considered as the segment method. The improvement in the traditional spring analogy considered in here is to include the angle made by each spring edge. This enhancement is similar to the approach that also suggested by Burg [12]. However, a different solution technique is explained here.

2.1 Basic Spring Analogy Method

In the spring analogy segment method, the idea is to consider each edge of the grid as a spring, which has a spring stiffness. Mathematically, the force exerted on this spring is computed as:

$$F_{x_{ij}} = k_{ij}(\Delta x_i - \Delta x_j)$$
$$F_{y_{ij}} = k_{ij}(\Delta y_i - \Delta y_j) \quad \textbf{(1)}$$

The stiffness of the spring is assumed to be inversely proportional to the length of edge $i - j$. Equation **(2)** shows the mathematical formulation of the stiffness of the spring.

$$k_{ij} = \frac{1}{\sqrt{(x_j - x_i)^2 + (x_j - x_i)^2}} \quad \textbf{(2)}$$

The updated coordinates of the mesh nodes are computed by imposing force equilibrium for both directions on each node. As a result, the force exerted on node i by neighbor nodes j can be computed as:

$$F_{x_i} = \sum_{j=1}^{n_i} k_{ij}(\Delta x_i - \Delta x_j)$$
$$F_{y_i} = \sum_{j=1}^{n_i} k_{ij}(\Delta y_i - \Delta y_j) \quad \textbf{(3)}$$

The equilibrium can be achieved by equating the right hand side of Equation **(3)** to zero. Based on that, the updated nodal coordinates can be found iteratively as:

$$\Delta x_i^{k+1} = \frac{\sum_{j=1}^{n_i} k_{ij}\Delta x_j^k}{\sum_{j=1}^{n_i} k_{ij}}$$
$$\Delta y_i^{k+1} = \frac{\sum_{j=1}^{n_i} k_{ij}\Delta y_j^k}{\sum_{j=1}^{n_i} k_{ij}} \quad \textbf{(4)}$$

2.2 Angle Inclusion in Spring Analogy Method

The previous mentioned technique lacks control for overlapped nodes since the method does not include the angle into the formulation. In this section, a detailed explanation regarding the inclusion of angle in the spring analogy formulation is briefly explained.

By taking into account the angle into the consideration, interaction between abscissa and ordinate of the node will be present. This interaction is modelled in the same manner like formulation used for Finite Element Analysis [13] for a beam element. In the formulation, each edge is modelled as a beam element which has its own stiffness matrix. This stiffness matrix can be computed as a function of angle made by the edge and is shown below.

$$\begin{bmatrix} F_{x_i} \\ F_{y_i} \\ F_{x_j} \\ F_{y_j} \end{bmatrix} = k_{ij} \begin{bmatrix} \cos^2\theta_{ij} & \cos\theta_{ij}\sin\theta_{ij} & -\cos^2\theta_{ij} & -\cos\theta_{ij}\sin\theta_{ij} \\ \cos\theta_{ij}\sin\theta_{ij} & \sin^2\theta_{ij} & -\cos\theta_{ij}\sin\theta_{ij} & -\sin^2\theta_{ij} \\ -\cos^2\theta_{ij} & -\cos\theta_{ij}\sin\theta_{ij} & \cos^2\theta_{ij} & \cos\theta_{ij}\sin\theta_{ij} \\ -\cos\theta_{ij}\sin\theta_{ij} & -\sin^2\theta_{ij} & \cos\theta_{ij}\sin\theta_{ij} & \sin^2\theta_{ij} \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ \Delta x_j \\ \Delta y_j \end{bmatrix} \quad \textbf{(5)}$$

The solution for this improvement method can be divided into two different categories; direct and indirect method. In the direct method, the above formulation is solved iteratively for each node located on the mesh domain. On the other hand, the solution by indirect method is solved by constructing a big square matrix corresponding to all unknown displacements in the grid.

Direct Method

In the direct method, the idea is to solve the unknown displacements, namely Δx and Δy for each nodes iteratively. This can be achieved by summing all forces exerted on node i from all the neighbor nodes j based on Equation **(5)**. Mathematically, the force exerted on node i by only one neighbor node j can be computed as:

$$F_{x_i} = k_{ij}\cos^2\theta_{ij}\left(\Delta x_i - \Delta x_j\right) + k_{ij}\cos\theta_{ij}\sin\theta_{ij}\left(\Delta y_i - \Delta y_j\right)$$
$$F_{y_i} = k_{ij}\cos\theta_{ij}\sin\theta_{ij}\left(\Delta x_i - \Delta x_j\right) + k_{ij}\sin^2\theta_{ij}\left(\Delta y_i - \Delta y_j\right) \quad \textbf{(6)}$$

In order to find the resultant forces exerted on node i, all the spring forces coming from neighbor nodes j are summed up. The total forces exerted on node i is shown in Equation **(7)**.

$$\begin{aligned} F_{x_{i_{total}}} &= \left(\sum_{j=1}^{n_i} k_{ij}\cos^2\theta_{ij}\right)\Delta x_i + \left(\sum_{j=1}^{n_i} k_{ij}\cos\theta_{ij}\sin\theta_{ij}\right)\Delta y_i - \left(\sum_{j=1}^{n_i} k_{ij}\cos^2\theta_{ij}\,\Delta x_j\right) - \left(\sum_{j=1}^{n_i} k_{ij}\cos\theta_{ij}\sin\theta_{ij}\,\Delta y_j\right) \\ F_{y_{i_{total}}} &= \left(\sum_{j=1}^{n_i} k_{ij}\,cos\,\theta_{ij}\,sin\,\theta_{ij}\right)\Delta x_i + \left(\sum_{j=1}^{n_i} k_{ij}\sin^2\theta_{ij}\right)\Delta y_i - \left(\sum_{j=1}^{n_i} k_{ij}\,cos\,\theta_{ij}\,sin\,\theta_{ij}\,\Delta x_j\right) - \left(\sum_{j=1}^{n_i} k_{ij}\sin^2\theta_{ij}\,\Delta y_j\right) \end{aligned} \quad \textbf{(7)}$$

The unknown displacements Δx_i and Δy_i are solved iteratively by imposing Dirichlet's boundary condition on the moving nodes. These unknowns are solved by equating Equation **(7)** to zero. The final expression for the solution method used is shown in Equation **(8)**. This equation can be solved by using Cramer's rule. The solution is said to be converged if there exists no more variation in the nodal displacements.

$$\begin{bmatrix} \sum_{j=1}^{n_i} k_{ij}\cos^2\theta_{ij} & \sum_{j=1}^{n_i} k_{ij}\cos\theta_{ij}\sin\theta_{ij} \\ \sum_{j=1}^{n_i} k_{ij}\,cos\,\theta_{ij}\,sin\,\theta_{ij} & \sum_{j=1}^{n_i} k_{ij}\sin^2\theta_{ij} \end{bmatrix} \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n_i} k_{ij}\cos^2\theta_{ij}\,\Delta x_j + \sum_{j=1}^{n_i} k_{ij}\cos\theta_{ij}\sin\theta_{ij}\,\Delta y_j \\ \sum_{j=1}^{n_i} k_{ij}\,cos\,\theta_{ij}\,sin\,\theta_{ij}\,\Delta x_j + \sum_{j=1}^{n_i} k_{ij}\sin^2\theta_{ij}\,\Delta y_j \end{bmatrix} \quad \textbf{(8)}$$

<u>Indirect Method</u>

Unlike the previous method on which the solution is achieved by solving the displacement for each node iteratively, the indirect method proposes a method by solving the unknown displacement for whole nodes iteratively. This idea is very similar to the solution of Finite Element Analysis [13].

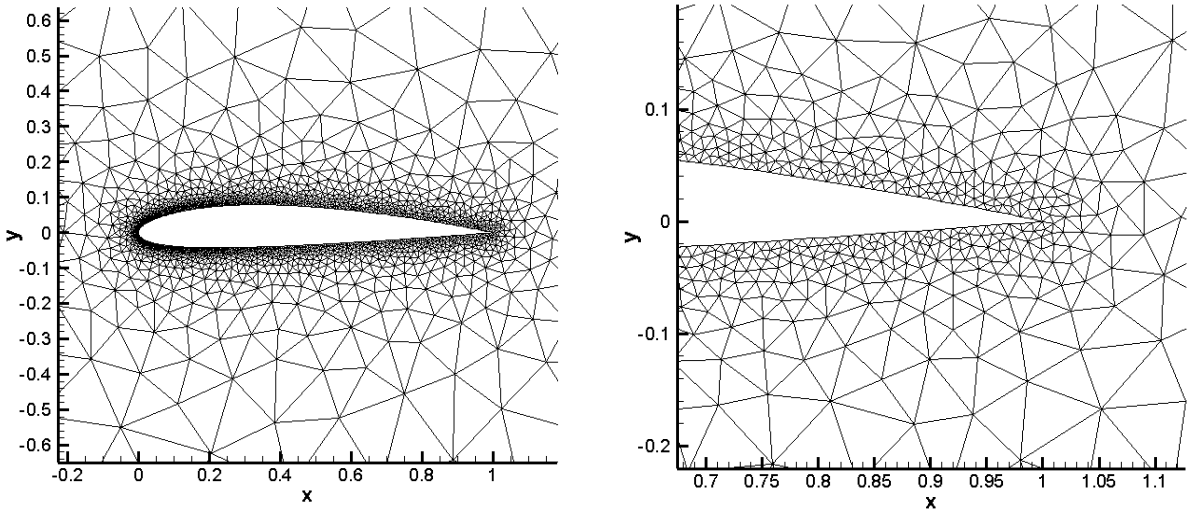
As mentioned earlier, each edge of the mesh has its own local stiffness matrix shown in Equation **(5)**. These local matrices are later assembled into a global stiffness matrix based on the global node numbering used in the solution. The node numbering is made in such a way that nodes corresponding to the moving boundary are numbered first and the nodes corresponding to the active nodes (movable nodes) are numbered later. Based on this numbering, the global stiffness matrix is partitioned. This global stiffness matrix, K can be written and partitioned as:

$$\begin{aligned} K &= \begin{bmatrix} K_{bb} & K_{ba} \\ K_{ab} & K_{aa} \end{bmatrix} \\ \begin{bmatrix} K_{bb} & K_{ba} \\ K_{ab} & K_{aa} \end{bmatrix} \begin{bmatrix} q_b \\ q_a \end{bmatrix} &= \begin{bmatrix} R_b \\ 0 \end{bmatrix} \\ K_{ab}q_b + K_{aa}q_a &= 0 \\ q_a &= -K_{aa}^{-1}[K_{ab}q_b] \end{aligned} \quad \textbf{(9)}$$

where q_b define the boundary displacements and q_a define the active displacements. K_{bb} and K_{aa} are square matrices that are partitioned matrices based on node numbering defined earlier. The solutions for active displacements, q_a are computed based on the last row of Equation **(9)**. In our case, the solution is computed based on the Conjugate-Gradient Method.

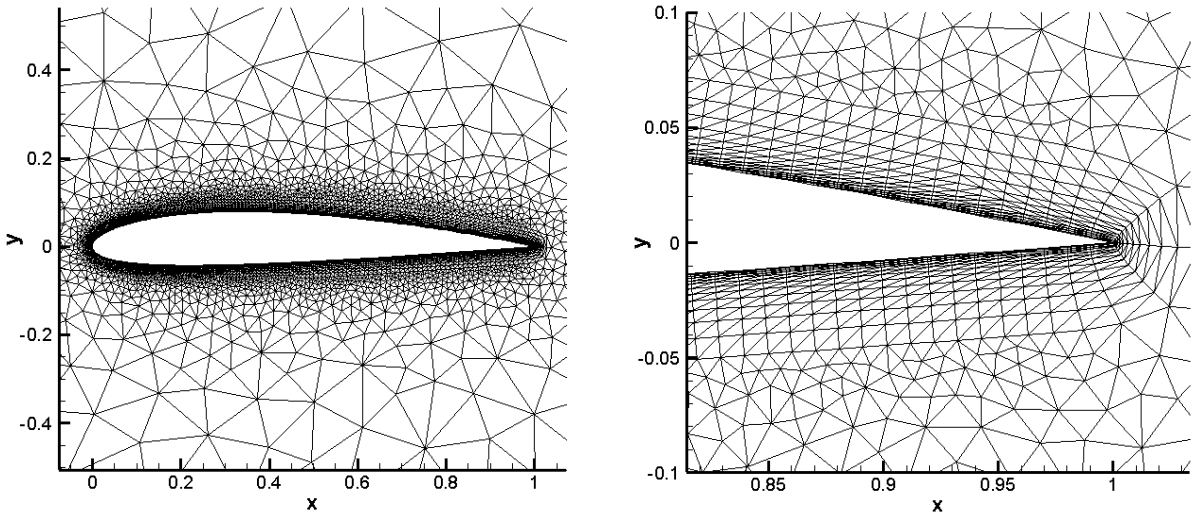
3. Mesh Deformation Results

In this section, the proposed methods defined earlier are tested for a rotating airfoil case. NACA 2412 is chosen as the initial airfoil geometry. There are two different mesh types considered during the analysis: inviscid mesh and viscous mesh. Figure 1 and Figure 2 illustrate the initial baseline mesh for these two mesh types, respectively. The zoom view near trailing edge is made as well since near that region a huge displacement is expected to occur and the region where spring analogy might fail. The mesh deformation capability is checked by rotating the airfoil by a fixed amount with different mesh deformation capabilities: basic spring analogy, angle inclusion in spring analogy with direct method, and angle inclusion in spring analogy with indirect method.



(a) (b)

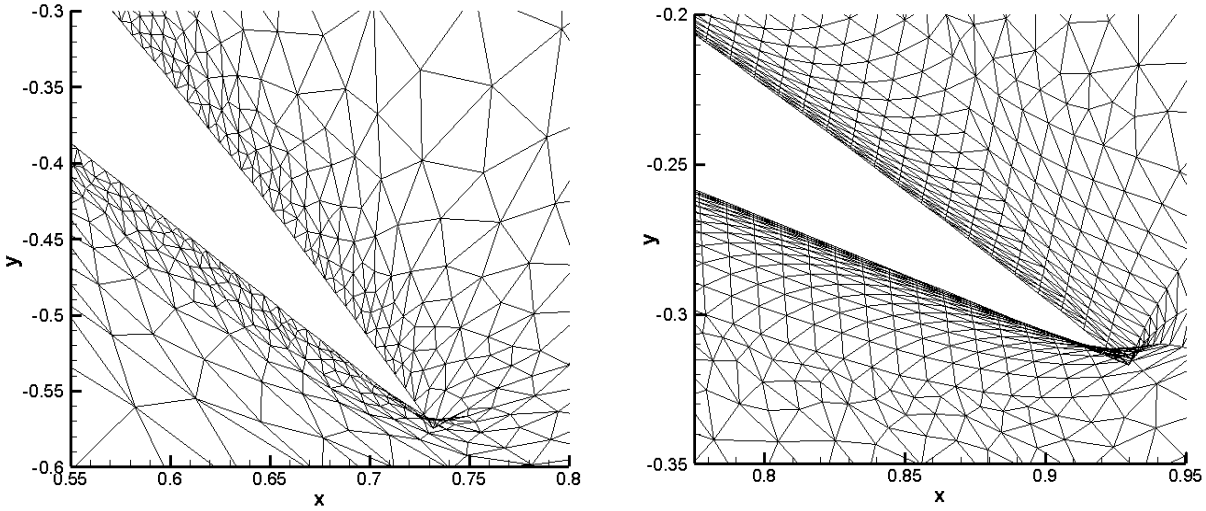
Figure 1: Initial Baseline Inviscid Mesh around NACA 2412 (a) Outer View (b) Zoom View Near Trailing Edge



(a) (b)

Figure 2: Initial Baseline Viscous Mesh around NACA 2412 (a) Outer View (b) Zoom View Near Trailing Edge

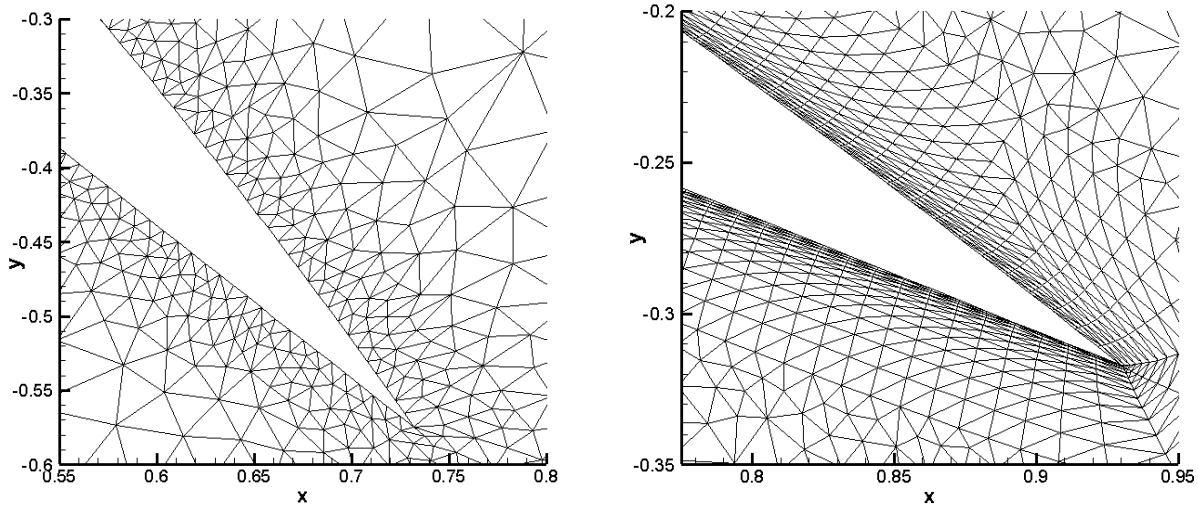
The deformed mesh using basic spring analogy is depicted in Figure 3. It can be seen clearly that this method cannot handle a huge displacement for the inviscid case. In the viscous case, this method also fails in terms of not being able to maintain the right angle that viscous cells near the boundary have. In order to overcome this problem, an improvement in terms of angle consideration is applied into the basic spring analogy. The deformed meshes based on this method are depicted in Figure 4 and Figure 5.



(a) (b)

Figure 3: Deformed Mesh Case Using Basic Spring Analogy
(a) 50° rotated inviscid mesh airfoil (b) 25° rotated viscous mesh airfoil

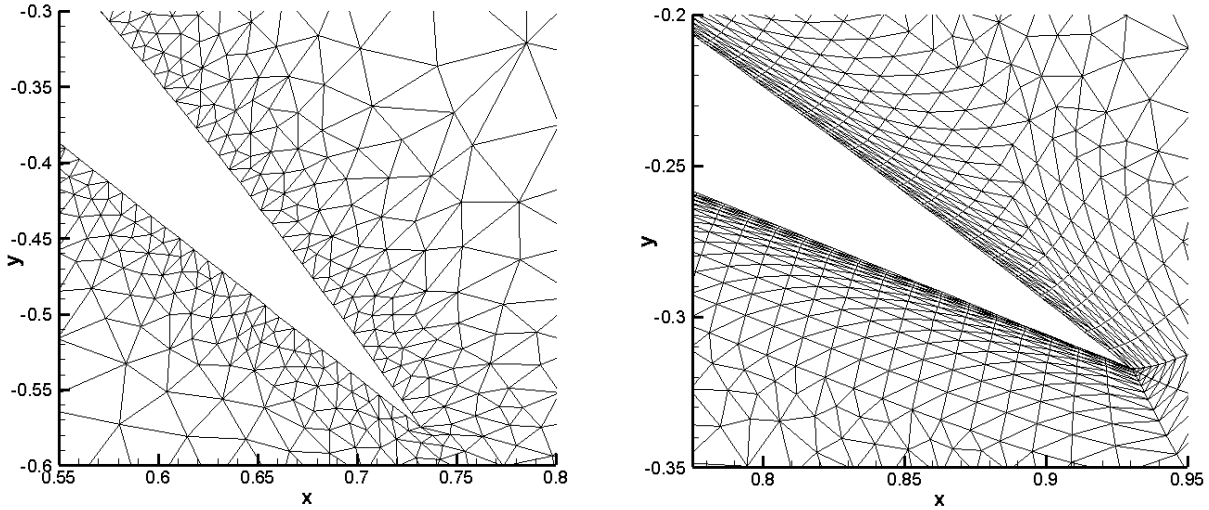
It can be seen clearly that by considering edge angle into the spring analogy formulation, a 50° can still be applied for inviscid mesh and 25° can be applied for viscous mesh. Viscous mesh can't be rotated more than 25° since the presence of high aspect ratio cell near the boundary becomes a hindrance to the spring analogy formulation to perform a large deformation. However, it can be seen clearly that in the rotated viscous mesh condition, the right angle condition can still be maintained. Although the rotation made by viscous mesh is smaller, the required deformation for shape parameterization is already encompassed by this modified spring analogy method.



(a) (b)

Figure 4: Deformed Mesh Case Using Angle Inclusion in Spring Analogy with Direct Method
(a) 50° rotated inviscid mesh airfoil (b) 25° rotated viscous mesh airfoil

Another thing that can be observed that the solution obtained from direct and indirect methods are almost similar to one another. This is already expected that the solution method does not change the main idea of the spring analogy methods. However, in terms of efficiency, the direct method is much faster compared to the indirect method. The fact that indirect method deals with solving a huge matrix requires a huge memory and consumes much time to get a convergence. On the other hand, solving 2 x 2 matrix in direct methods requires relatively smaller memory and less time. As a result, the angle inclusion spring analogy with direct method is implemented during the airfoil shape optimization.



(a) (b)

Figure 5: Deformed Mesh Case Using Angle Inclusion in Spring Analogy with Indirect Method
(a) 50° rotated inviscid mesh airfoil (b) 25° rotated viscous mesh airfoil

4. Shape Parameterization and Optimization

The best method described in the previous section is later coupled with CFD solver to perform an airfoil optimization. In the optimization, several shape parameterizations are considered: camber variation, thickness variation, and combined camber and thickness variation.

In all these shape parameterizations, initial camber line and thickness distribution should be determined. The camber line is estimated as the average between the ordinate of the upper and lower airfoil surfaces with the same abscissa value. In the case where airfoil points are not located on the same abscissa, spline interpolation is utilized. The final expression for the camber line is shown in Equation **(10)**.

$$y_{camber} = \frac{y_{upper} + y_{lower}}{2} \tag{10}$$

On the other hand, the airfoil thickness is estimated as the ordinate difference between the camber line and the airfoil surface. The expression for airfoil thickness is shown in Equation **(11)**.

$$y_{thick_{upper}} = y_{upper} - y_{camber}$$
$$y_{thick_{lower}} = y_{lower} - y_{camber} \tag{11}$$

Based on this parameterization, the design variables that can be applied are: camber factor which changes the camber variation of the airfoil and thickness factor which changes the thickness variation of the airfoil. The new camber and thickness are calculated by multiplying the initial camber and thickness by camber and thickness factor, respectively. These factors have their own ranges which are tabulated in Table 1.

Table 1: Range of Design Variables used in the Shape Parameterization

Design Variable	Lower Limit	Upper Limit
Camber Factor	0	3
Thickness Factor	0.6	3

After computing the new camber and thickness distribution, the airfoil's ordinate value is updated. The updated value for the airfoil's ordinate is computed based on

$$y_{upper_{final}} = y_{camber_{final}} + y_{thick_{upper}}$$
$$y_{lower_{final}} = y_{camber_{final}} + y_{thick_{lower}} \tag{12}$$

The optimization itself was conducted by using a gradient optimizer provided by Phoenix ModelCenter, OPTLIB Gradient Optimizer. During the optimization, the airfoil drag is considered as the objective function and a specified lift coefficient is considered. The lift coefficient is computed based on the required sectional lift for the airfoil and chord length of the airfoil. In this study, the sectional lift is taken as 61.3125 N/m with a chord length of 0.6 m.

There are two different flight conditions considered during the optimization scheme, loiter and take-off. In each case, different angle of attack constraint is also imposed. In loiter phase, the angle of attack is constrained to be between 0 and 7 degrees. On the other hand, angle of attacks are constrained to be between -3 and 6 degrees for take-off flight condition. Detail of the flow parameters for these two flight conditions are tabulated in Table 2.

Table 2: Summary of Flow Parameters Used in the Optimization

	Loiter Phase	Take-Off Phase
Flight Speed [m/s]	15.28	21.16
Reynolds Number	605075	858441
Density [kg/m^3]	1.1895	1.225
Mach Number	0.0451	0.0622
Altitude [ft]	1000	0

The lift and drag coefficients of the airfoil are computed based on CFD analysis using SU2 CFD Solver. Viscous solver equipped with Spalart Allmaras is chosen as the flow solver. The input mesh used for the analysis is based on the deformed mesh obtained from the mesh deformation analysis.

The optimization scheme using Phoenix ModelCenter contains three different modules: input module, mesh deformation module, and CFD solver module. The relation between each module is shown in Figure 6.

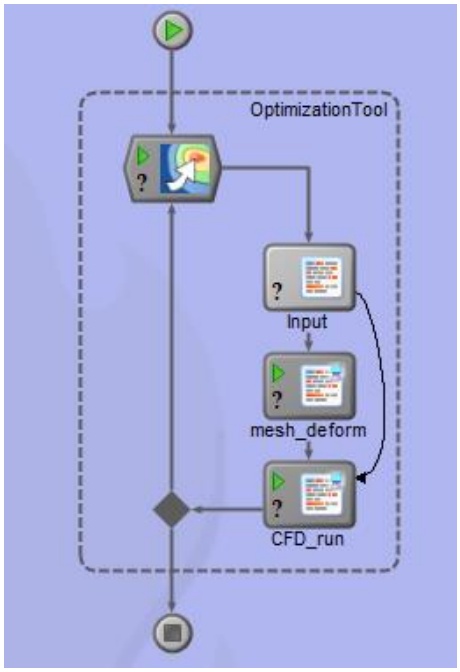


Figure 6: Flow Chart Used in the Optimization Scheme

5. Results

This section presents the optimization results based on shape parameterization described in the earlier section. The optimization is done for both loiter and take-off flight conditions defined earlier. In each case, NACA 2412 airfoil is chosen as the initial airfoil shape.

5.1 Loiter Phase Optimization

Based on the flow parameters described in the previous section, it is found that the required lift coefficient for this flight condition is 0.7361. The iteration history for this phase optimization with three different shape parameterizations are depicted in Figure 7.

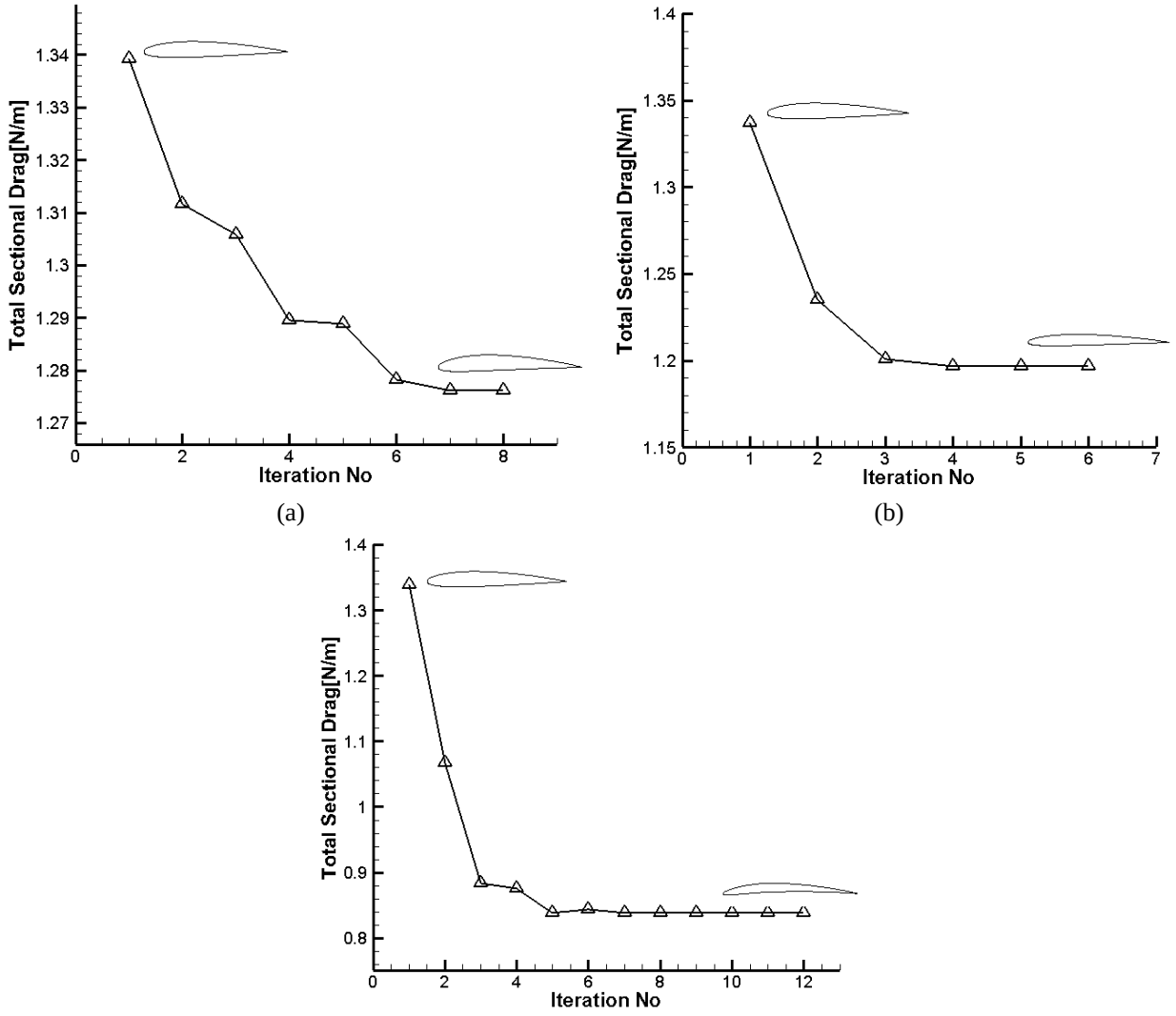


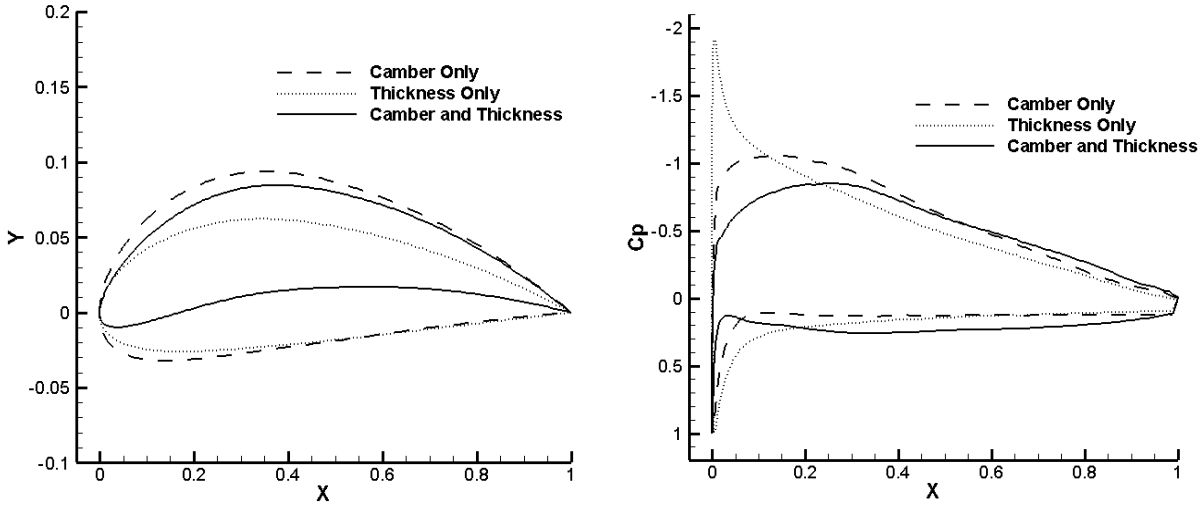
Figure 7: Optimization Iteration Summary for Loiter Phase
(a) Camber Only (b) Thickness Only (c) Camber and Thickness Combined

By considering the camber effect only in the optimization, the optimum airfoil has a relatively increase in camber by a factor of 1.747. On the other hand, by considering thickness only, the optimum airfoil has a relatively decrease in thickness by a factor of 0.714. Finally, the combined camber and thickness consideration in the optimization scheme leads to an optimum airfoil which has an increase in camber by a factor of 2.485 and decrease in thickness by a factor of 0.6, which is the minimum value.

The drag and angle of attack values corresponding to initial and final configurations are tabulated in Table 3. It can be seen clearly that the optimization which considers both effect of camber and thickness variation leads to a better optimum value among the proposed shape parameterization schemes. Furthermore, the optimum airfoil has a relatively low angle of attack compared to the initial airfoil. The shapes of optimum airfoils with their pressure distribution are shown in Figure 8.

Table 3: Summary of Optimization Results for Loiter Phase

Parameterization Scheme	Total Drag [N/m]		Alpha [deg]	
	Initial	Optimum	Initial	Optimum
Camber	1.3391	1.2761	5.417	4.030
Thickness	1.3391	1.1967	5.417	5.347
Camber and Thickness	1.3391	0.8376	5.417	2.395

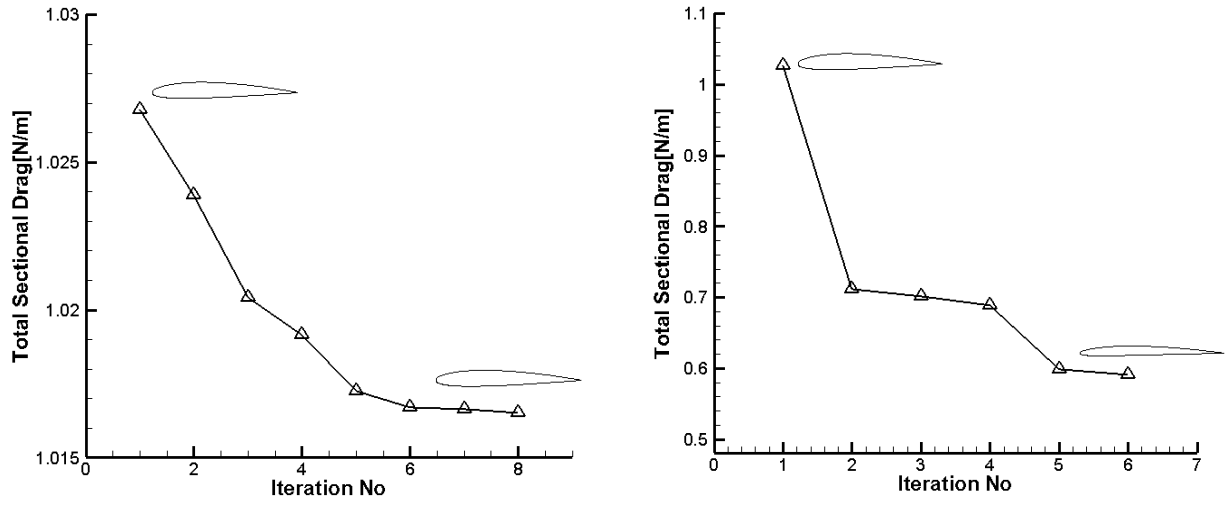


(a) (b)

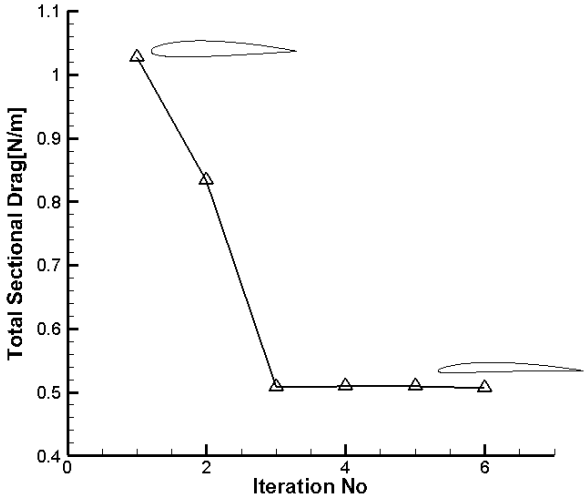
Figure 8: Optimized Design for Loiter Configuration with Several Shape Parameterizations
(a) Optimum Airfoil Shapes (b) Cp Distribution for Optimum Airfoil

5.2 Take-Off Phase Optimization

The required airfoil coefficient for this flow parameters is computed as 0.3725. Summary of iteration history for this optimization is shown Figure 9.



(a) (b)



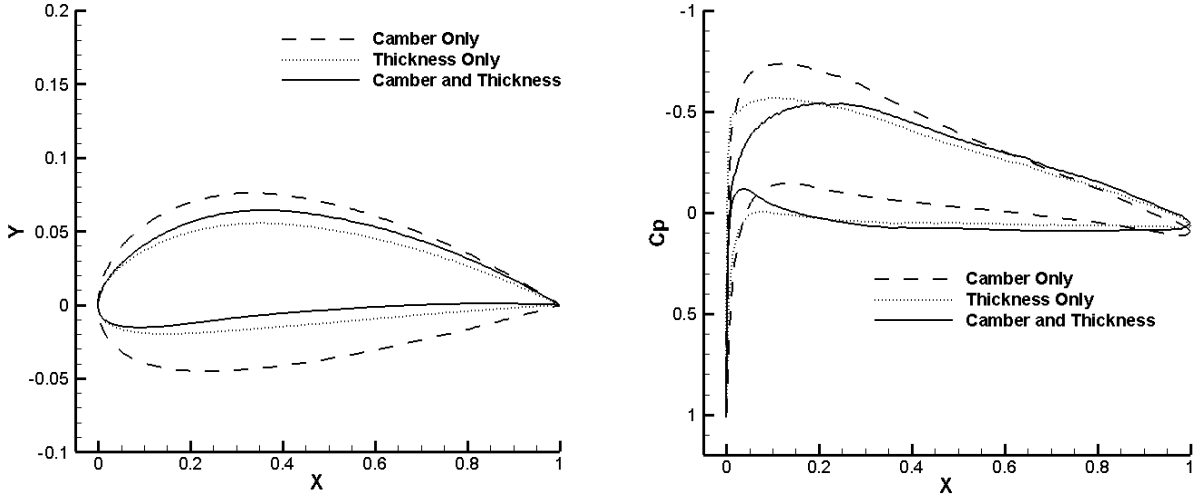
(c)

Figure 9: Optimization Iteration Summary for Take-Off Phase
(a) Camber Only (b) Thickness Only (c) Combined Camber and Thickness

The summary of optimization results for take-off phase is tabulated in Table 4. It can be seen clearly that the optimization conducted by considering both camber and thickness variation leads a better design. For a better illustration, the shape and pressure distributions for the optimum airfoils are shown in Figure 10.

Table 4: Summary of Optimization Results for Take-Off Phase

Parameterization Scheme	Total Drag [N/m]		Alpha [deg]	
	Initial	Optimum	Initial	Optimum
Camber	1.0267	1.0165	1.834	2.137
Thickness	1.0267	0.5907	1.834	1.768
Camber and Thickness	1.0267	0.5064	1.834	0.898



(a) (b)

Figure 10: Optimized Design for Take-Off Configuration with Several Shape Parameterizations
(a) Optimum Airfoil Shapes (b) Cp Distribution for Optimum Airfoil

Unlike the loiter phase optimization, where an optimum airfoil has a high camber, the optimum airfoil for take-off phase by considering the presence of camber only has a camber reduction. This is due the fact that the baseline airfoil has a relatively high thickness and the required lift coefficient is quite low. In fact, the optimum airfoil has a reduction in camber by a factor of 0.8333. In the case where shape modification is solely based on thickness factor, the optimum airfoil for this flight phase has a similar trend to loiter phase optimization. The optimum airfoil has a reduction in thickness by a factor of 0.6, which is the minimum allowable value. For the optimization by considering both camber

and thickness variation, the optimum airfoil has increasing in camber by a factor of 1.452 and decreasing in thickness by a factor 0.6.

6. Conclusions

This paper has elaborated the idea of mesh deformation technique by using spring analogy and its modification by considering the angle made by the edge of the mesh. This modified method can be solved by using either direct solution method which solves unknown displacement iteratively for each node or indirect method by constructing a huge matrix corresponds to the whole unknown displacements. It is found that the modified spring analogy can deform inviscid mesh to a higher amount of deformation and viscous mesh to a smaller amount of deformation. Furthermore, the solution achieved by using direct method is better than indirect method in terms of computation time.

In the airfoil CFD design optimization for two different flight conditions, it is found that performing an optimization by considering both camber and thickness variation leads to a better optimum design. In both cases of flight conditions, it is found that the optimum airfoil has a relatively high camber and low thickness distribution. However, the loiter optimization requires a high camber compared to take-off optimization since loiter phase is accompanied by a relatively high lift coefficient.

References

[1] Lin, T. J., Z. Q. Guan, J. H. Chang, and S. H. Lo. 2014. Vertex-Ball Spring Smoothing: An efficient method for unstructured dynamic hybrid meshes. *J. Comput. Struct.* 136: 24–33.

[2] Zhou, X. and S. Li. 2013. A new mesh deformation method based on disk relaxation algorithm with pre-displacement and post-smoothing. *J. Comput. Phys.* 235: 199–215.

[3] Batina, J. T. 1990. Unsteady Euler Solutions Using Unstructured Dynamic Meshes. *AIAA J.* 28: 1381-1388.

[4] Witteveen, J. A. 2010. Explicit and Robust Inverse Distance Weighting Mesh Deformation for CFD. In: *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*.

[5] Liu, X., N. Qin, and H. Xia. 2006. Fast dynamic grid deformation based on Delaunay graph mapping. *J. Comput. Phys.* 211: 405–423.

[6] de Boer, A., M. S. van der Schoot, and H. Bijl. 2007. Mesh deformation based on radial basis function interpolation. *J. Comput. Struct.* 85: 784–795.

[7] Farhat, C. 1998. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *J. Comput. Methods Appl. Mech. Eng.* 25: 231–245.

[8] Zeng, D. and C. R. Ethier. 2005. A semi-torsional spring analogy model for updating unstructured meshes in 3D moving domains. *J. Finite Elem. Anal. Des.* 41: 1118–1139.

[9] Bottasso, C. L., D. Detomi, and R. Serra. 2005. The ball-vertex method : a new simple spring analogy method for unstructured dynamic meshes. *J. Comput. Methods Appl. Mech. Eng*. 194: 4244–4264.

[10] Yang, Y., and S. Özgen. 2015. Implementation of Ball-Center Spring Analogy Mesh Deformation Technique with CFD Design Optimization. In: *22nd AIAA Computational Fluid Dynamics Conference.*

[11] Blom, F. J. 2000. Considerations on the spring analogy. *Int. J. Numer. Methods Fluids*. 32: 647–668.

[12] Burg, C. 2004. A Robust Unstructured Grid Movement Strategy Using Three-Dimensional Torsional Springs. In *34th AIAA Fluid Dynamics Conference and Exhibit*.

[13] Cook, R. D., D. S. Malkus, and M. Plesha. 1989. Concepts and Applications of Finite Element Analysis. 2nd Ed.