IMPLEMENTATION OF TWO-DIMENSIONAL TIME ACCURATE FLOW SIMULATION ON MOVING MESH EXPLOITING MESH RUPTURING

By

Ya'nan Gong

Steve Karman
Professor of Computational
Engineering
(Chair)

Kidambi Sreenivas
Professor of Computational Engineering
(Committee Member)

Yu Liang
Associate Professor of Computer Science
(Committee Member)

IMPLEMENTATION OF TWO-DIMENSIONAL TIME ACCURATE FLOW SIMULATION ON MOVING MESH EXPLOITING MESH RUPTURING

By

Ya'nan Gong

A Thesis Submitted to the Faculty of the University
of Tennessee at Chattanooga in Partial
Fulfillment of the Requirements of the
Degree of Master of Science :
Engineering

The University of Tennessee, Chattanooga
Chattanooga, Tennessee

December 2014

ABSTRACT

A two-dimensional dynamic moving mesh algorithm is implemented for simulating aerodynamic oscillating airfoils. The airfoil is inserted into a uniform unstructured background mesh by exploiting an new technique named mesh rupturing, which allows for significant geometry movement without grid regeneration or interpolation between grids. Then unsteady flow simulation results are presented and validated by several test cases. Previous methodologies for unsteady flow simulation are listed and compared, advantages and disadvantages of the new mesh technique are discussed and advice for further research as well.

DEDICATION

This thesis is dedicated to Baoxiang Pan, my Mom, without whom I would not have been able to complete it.

ACKNOWLEDGEMENTS

I would like to thank Dr. Steve Karman for his guidance and patience during the research. Also, I would like to thank Dr. James Newman for his advice in completing the work, Jianfeng Yan for his time helping me in the discussion and Arash Ghasemi for the help to complete this document.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ξ, computational coordinates

x, y, physical coordinates

s, clustering function

u, v, perturbation in x and y direction

Γ, integral length of a surface

Ω, integral area

α, θ, Linear-Elastic coefficients

E, Young's Modulus

ν, Poisson's Ratio

∇, differential operator

n, normal vector for edge

f, scalar variable

t, external edge normal of a control volume

J, Jacobian of triangle

CN, condition number

$\vec{s}_n$, sensitivity vector

$\vec{p}_n$, perturbation vector

C_e, cost of element

$\vec{W}_s$, velocity vector of face

ϕ, coefficients in BDF function

δ, kronecker delta

γ, heat capacity ratio

F, flux

ρ, density

u, v, velocity vector in x and y direction

$\bar{U}$, averge velocity

p, pressure

E, energy

a, speed of sound

M, Mach number

Δ, delta

R, residual

V, volume of control volume

α, angle of attack

k, reduced frequency

w, angular velocity

c, chord length

U_∞, farfield velocity

CHAPTER 1
INTRODUCTION

Literature Review

In the field of computational fluid dynamics, significant progress has been made over past decades in unsteady flow simulation, which solve the problems of oscillating objects and moving bodies around each other. The difference between a steady simulation and an unsteady one is that the latter solves fluid equations on a continuously changing domain, especially when boundaries are moving. A numbers of algorithms have been developed for the flow simulations on moving mesh.

The first intuitive method would be just move the entire mesh as a rigid body, where element volumes will result in no change because no deformation of grid occurs, the reason leading to difference compared to steady cases is the integrations of grid speed term into flow calculations. Furthermore, unsteady problems could also be solved by merely exploiting grid smoothing and optimizing methods when only small boundary movement happens.

When large scale body motion occurs, the earliest and easiest approach is to regenerate grid domain after each time step of solution, like Gationde[1] proposed, which is desirable for structured grid. A recent improved technique is as described Alauzet [2] who exploits mesh regeneration or reconnection to implement mesh deformation. This approach shows a preference of node-centered scheme, where nodes could be removed and reconnected to retain mesh quality. For cell-centered scheme, interpolation will be required to complete the transfer from old mesh to new mesh. Either earlier or modern techniques, mesh regeneration

is a time consuming process, and a lot of interpolation is required to maintain solution accuracy.

In order to replace the idea of grid regeneration, local restructuring as in ref [3] was proposed. As its name shows, grid is restructured instead of regenerated, and it happens only in parts which are adjacent to moving boundaries. The disadvantage of this approach is that after several time steps, small cells will emerge in these adjacent parts. However the process to remove these elements is costly. To reduce the cost, a smoothening process is called, for instance the one in ref [4]. This idea of spring connectors was first proposed by Batina [5]. Connectors between nodes are treated as springs, a uniform grid is generated via the balance of spring forces after each time step of boundary movement. The spring factor K, which is the most crucial parameter in this approach, has been studied in different papers. Application of this methodology could be seen in ref [6]. However, the integrity of grid is destroyed for large boundary motions, so small scale of regeneration is still needed.

Since quality of grid around moving boundaries is difficult to maintain, it can be solved by surrounding the body with high-quality grid as showcased by Zhang [7]. Let surrounding grid moves synchronously as the body, so only the regions which are far from body will be affected.

For complex geometry motion on structured grids, the approach of overset-grid (Chimera) was first proposed in ref [8]. In this approach, each body has its own grid, hence there are some regions where the grid overlaps solid body with neighboring grids. These regions are defined as chimera holes. Therefore, high-quality grid is maintained around each moving body in simulation as a compensation of interpolations. Nowadays, the majority of problems which requires significant geometry movement are solved with Chimera grids or sliding interfaces, although they are not conservative, and require a defined direction of motion needing human interference to modify.

Another hybrid grid based on Chimera is also being applied in body motion problems as shown in ref [9], where flow domain is partitioned into three nested zones. The first zone,

a circular domain, includes the moving body and enables the rotational motion of it. The second zone, a square domain, includes the first zone and enables the translational motion of the body. The third zone is a background grid in which the second zone moves. Any two-dimensional motion of a body can be modeled with almost no grid insertion or deletion under this settings of grids. However, some elements would be merged or split during some phases of motion.

This thesis is an implementation of time accurate flow simulation of moving mesh. The mesh technique is based on O'Connell and Karman [10] who proposed a new mesh technique called mesh rupturing. It facilitates significant and relative geometry motion while keeping a single mesh, and no procedure of altering mesh topology or interpolation between grids is needed. Because of the moving boundaries, deformations of grid will occur, so it leads to Geometry Conservation Law(GCL) being applied to maintain the global conservation while doing flow calculations. For flow solution technique, a spatial discretization based on flux-vector splitting(FVS) of Van-Leer which takes into account of the wave-propagation characteristics of the flow and naturally dissipative is applied. Furthermore, an implicit time integration scheme is used for temporal discretization, applying a Gauss-Seidel relaxation. Therefore, it allows to select step size based on the temporal accuracy. Unsteady results are presented for the NACA0012 airfoil pitching about the quarter chord. The details to build this Euler solver will be described along with results and comparisons presented in this thesis.

Chapter Summaries

In Chapter 2, one dimensional arc length parametric distribution, Linear-Elastic and adaptive optimization-based smoothing methods are presented, which are used to retain mesh quality after boundaries are moved.

In Chapter 3, mesh rupturing technique is presented. It includes two aspects, mesh insertion and mesh movement, the basic algorithm for this approach is discussed.

In Chapter 4, the procedure to build this two-dimensional Euler flow solver are presented, and Geometry Conservation Law is applied.

In Chapter 5, results of airfoil pitching and moving validation cases are presented and compared.

Chapter 6 is a summary of the advantages and disadvantages of this mesh technique and some suggestions on further research.

CHAPTER 2

BACKGROUND

Arc Length Parametric Distribution

Geometry insertion and motion is implemented as a process of splitting and merging the background mesh. During these phases, nodes might be added to or removed from geometry, which will lead to deformation of geometry. Thus, an approach of arc length parametric distribution is needed to redistribute the grid points along the surface of geometry to maintain the original features of it.

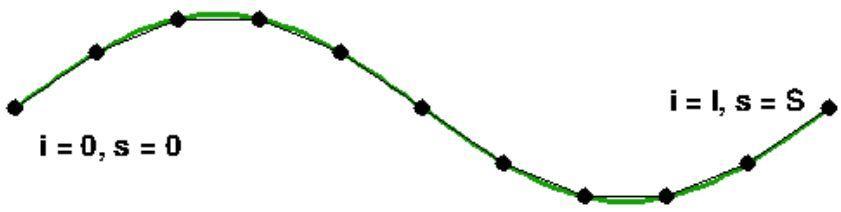

Figure 2.1 Grid points distribution using arc length parametric distribution

A curve which has $I + 1$ points start from 0 to I and position from 0 to S, as shown in Figure 2.1, the procedure for determining desired points on line is as follows:

1. $\xi_i = (i - 1)/(I - 1)$

2. $s_i = s(\xi_i) * S$

3. $x_i = x(s_i)$, $y_i = y(s_i)$

where $s(\xi)$ is the desired clustering function(transformation), and its value varies from 0 to 1. The procedure to get values of $s(\xi)$ will be discussed in following. For known functions $x(s)$ and $y(s)$, only first two steps are needed to determin the grid point coordinates. However, if the two functions are unknown, a collection of grid positions are known instead, then the following process is taken:

1. Given $x(k)$ and $y(k)$ for discrete values k = 0, 1, ..., K

2. Compute arc length at each discrete i location,

   $s_0 = 0$

   $s_k = s_{k-1} + \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}$

   then normalize s from 0 to 1 using total arc length.

3. The known values of $x_k(s_k)$ and $y_k(s_k)$ provide discrete representations of $x(s)$ and $y(s)$. Curve fit $x(s)$ and $y(s)$ to provide a continuous representation for $x(s)$ and $y(s)$. This curve fit can be a linear function, a subdivision curve or a higher order spline.

4. For each i compute s_i using the transformation $s(\xi)$.

   $s_i = s(\xi_i)$

5. Use the curve fits to compute (x_i, y_i) from $x(s_i)$ and $y(s_i)$.

It is desirable to place more points in regions where large solution changes exist, and a lot of grid point clustering can be used to achieve this goal. Two-Sided Hyperbolic Tangent Clustering to Side Boundaries is used in this thesis. The transformation is represented by $s(\xi)$ as shown in Equation 2.1, and the variable s_t is as shown in equation 2.2, and the function clusters away from the cluster point ξ_t.

$$s(\xi) = s_t(1 + \frac{\tanh(a(\xi - \xi_t))}{\tanh(a\xi_t)}) \tag{2.1}$$

$$s_t = \frac{\tanh(a\xi_t)}{\tanh(a\xi_t) + \tanh(a(1 - \xi_t))} \tag{2.2}$$

$$s_\xi = \frac{as_t}{\tanh(a\xi_t)} \operatorname{sech}^2(a(\xi - \xi_t)) \tag{2.3}$$

Linear Elastic Smoothing

When geometry motion happens, the Linear-Elastic relationships are used to move interior mesh points. The L-E equations represent the perturbation field on the inside of the domain based on the perturbations prescribed on the boundaries. The governing equations are:

$$\frac{\partial}{\partial x}(\alpha_{11}u_x) + \frac{\partial}{\partial y}(\alpha_{12}u_y) + \frac{\partial}{\partial x}(\theta_{11}\nu_y) + \frac{\partial}{\partial y}(\theta_{12}\nu_x) = 0 \tag{2.4}$$

$$\frac{\partial}{\partial x}(\alpha_{21}\nu_x) + \frac{\partial}{\partial y}(\alpha_{22}\nu_y) + \frac{\partial}{\partial x}(\theta_{21}u_y) + \frac{\partial}{\partial y}(\theta_{22}u_x) = 0 \tag{2.5}$$

$$\alpha_{11} = \alpha_{22} = \frac{E}{1 - \nu^2} \tag{2.6}$$

$$\alpha_{12} = \alpha_{21} = \theta_{12} = \theta_{21} = \frac{E}{2(1 + \nu)} \tag{2.7}$$

$$\theta_{11} = \theta_{22} = \frac{\nu E}{1 - \nu^2} \tag{2.8}$$

The corresponding perturbation components in the x and y directions are represented by u and v. The coefficients, α and θ, are functions of two parameters, E and ν which are Young's Modulus and Poisson's Ratio in structural mechanics. For our purpose, we will assume the leading coefficients are constants, which are not in reality, and pull them out of the partial derivative operators, thereby making the equations a set of linear partial

differential equations:

$$\alpha_{11}u_{xx} + \alpha_{12}u_{yy} + \theta_{11}\nu_{xy} + \theta_{12}\nu_{xy} = 0 \tag{2.9}$$

$$\alpha_{21}\nu_{xx} + \alpha_{22}\nu_{yy} + \theta_{21}u_{xy} + \theta_{22}u_{xy} = 0 \tag{2.10}$$

Finite Volume Discretization of the Linear-Elastic Equations

In order to solve the L-E equations numerically, a finite volume method of discretization has been used. The differential governing equations are first to be recast in integral form. If the coefficients α, θ are assumed constants, then the L-E equations become:

$$\alpha_{11}\iint u_{xx}d\Omega + \alpha_{12}\iint u_{yy}d\Omega + \theta_{11}\iint \nu_{xy}d\Omega + \theta_{12}\iint \nu_{xy}d\Omega = 0 \tag{2.11}$$

$$\alpha_{21}\iint \nu_{xx}d\Omega + \alpha_{22}\iint \nu_{yy}d\Omega + \theta_{21}\iint u_{xy}d\Omega + \theta_{22}\iint u_{xy}d\Omega = 0 \tag{2.12}$$

where Ω is an incremental area two-dimensional. By applying Gauss' theorem, double integrals are converted to line integrals as:

$$\iint \nabla \bullet \vec{F}d\Omega = \oint \vec{F} \bullet \hat{n}d\Gamma \tag{2.13}$$

If one defines

$$\vec{\nu} = u_x\hat{i} + 0\hat{j} \tag{2.14}$$

Then divergence of this vector becomes

$$\nabla \bullet \vec{\nu} = u_{xx} \tag{2.15}$$

Now the double integral becomes

$$\iint u_{xx}d\Omega = \oint (u_x\hat{i} + 0\hat{j}) \bullet (n_x\hat{i} + n_y\hat{j})d\Gamma = \oint u_x n_x d\Gamma \tag{2.16}$$

If one redefines

$$\vec{\nu} = u_y\hat{i} + 0\hat{j} \tag{2.17}$$

Then divergence of this new vector becomes

$$\nabla \bullet \vec{\nu} = u_{xy} \tag{2.18}$$

Now the double integral becomes

$$\iint u_{xy} d\Omega == \oint u_y n_x d\Gamma \tag{2.19}$$

Finally define

$$\vec{\nu} = 0\hat{i} + u_y\hat{j} \tag{2.20}$$

Then divergence of this new vector becomes

$$\nabla \bullet \vec{\nu} = u_{yy} \tag{2.21}$$

Now the double integral becomes

$$\iint u_{yy} d\Omega == \oint u_y n_y d\Gamma \tag{2.22}$$

Same operations will be applied on each double integral in both equations, by replacing all the double integrals with line integral, then the governing equations become:

$$\alpha_{11} \oint u_x n_x d\Gamma + \alpha_{12} \oint u_y n_y d\Gamma + \theta_{11} \oint \nu_y n_x d\Gamma + \theta_{12} \oint \nu_x n_y d\Gamma = 0 \tag{2.23}$$

$$\alpha_{21} \oint \nu_x n_x d\Gamma + \alpha_{22} \oint \nu_y n_y d\Gamma + \theta_{21} \oint u_y n_x d\Gamma + \theta_{22} \oint u_x n_y d\Gamma = 0 \tag{2.24}$$

By applying the approach to a control volume with node-centered scheme, because it is defined as a set of unstructured triangles connected to the center node, the sum is around all the external sides on the control volume.

$$\sum_{i=1}^{ns} \alpha_{11} u_{xi} n_{ix} + \alpha_{12} u_{yi} n_{iy} + \theta_{11} \nu_{yi} n_{ix} + \theta_{12} \nu_{xi} n_{iy} = 0 \tag{2.25}$$

$$\sum_{i=1}^{ns} \alpha_{21} \nu_{xi} n_{ix} + \alpha_{22} \nu_{yi} n_{iy} + \theta_{21} u_{yi} n_{ix} + \theta_{22} u_{xi} n_{iy} = 0 \tag{2.26}$$

The x and y terms in equation are first derivatives defined for the triangle connected to that external edge, and n is the external normal vector.

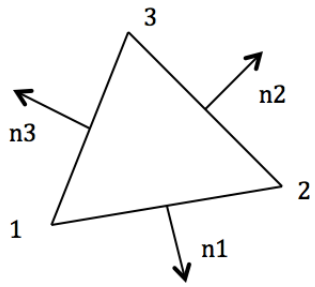

Figure 2.2 Normal vectors for triangle

where

$$n_{1x} = y_3 - y_2, n_{1y} = x_2 - x_3$$

$$n_{2x} = y_1 - y_3, n_{2y} = x_3 - x_1$$

$$n_{3x} = y_2 - y_1, n_{3y} = x_1 - x_2$$

For a general scalar variable f, the numerical form of the first derivative components of f across a triangle can be computed using Gauss' theorem:

$$\frac{\partial f}{\partial x} = \frac{1}{2A}\big[f_1(n_{2x} + n_{3x}) + f_2(n_{1x} + n_{3x}) + f_3(n_{1x} + n_{2x})\big] \tag{2.27}$$

$$\frac{\partial f}{\partial y} = \frac{1}{2A}\big[f_1(n_{2y} + n_{3y}) + f_2(n_{1y} + n_{3y}) + f_3(n_{1y} + n_{2y})\big] \tag{2.28}$$

Besides:

$$n_{1x} + n_{2x} + n_{3x} = 0 \tag{2.29}$$

$$n_{1y} + n_{2y} + n_{3y} = 0 \tag{2.30}$$

As such, by subsituting equation 2.29 and 2.30 back into equation 2.27 and 2.28, the derivative formulae can be rearranged as:

$$\frac{\partial f}{\partial x} = -\frac{1}{2A}\big[f_1 n_{1x} + f_2 n_{2x} + f_3 n_{3x}\big] \tag{2.31}$$

$$\frac{\partial f}{\partial y} = -\frac{1}{2A}\big[f_1 n_{1y} + f_2 n_{2y} + f_3 n_{3y}\big] \tag{2.32}$$

All first derivatives in equation 2.25 and 2.26 are replaced with the discrete formula respectively in equation 2.31 and 2.32. And to avoid confusion between face normals and external edge normals, the latter ones are treated as t in following:

$$\begin{aligned}\sum_{i=1}^{ns}\{&\frac{\alpha_{11}}{2A_i}(-u_1 n_{1x} - u_2 n_{2x} - u_3 n_{3x})_i t_{ix} + \frac{\alpha_{12}}{2A_i}(-u_1 n_{1y} - u_2 n_{2y} - u_3 n_{3y})_i t_{iy} + \\ &\frac{\theta_{11}}{2A_i}(-\nu_1 n_{1y} - \nu_2 n_{2y} - \nu_3 n_{3y})_i t_{ix} + \frac{\theta_{12}}{2A_i}(-\nu_1 n_{1x} - \nu_2 n_{2x} - \nu_3 n_{3x})_i t_{iy}\} = 0\end{aligned} \tag{2.33}$$

$$\sum_{i=1}^{ns}\{\frac{\alpha_{21}}{2A_i}(-\nu_1 n_{1x}-\nu_2 n_{2x}-\nu_3 n_{3x})_i t_{ix}+\frac{\alpha_{22}}{2A_i}(-\nu_1 n_{1y}-\nu_2 n_{2y}-\nu_3 n_{3y})_i t_{iy}+$$
$$\frac{\theta_{21}}{2A_i}(-u_1 n_{1y}-u_2 n_{2y}-u_3 n_{3y})_i t_{ix}+\frac{\theta_{22}}{2A_i}(-u_1 n_{1x}-u_2 n_{2x}-u_3 n_{3x})_i t_{iy}\}=0 \qquad (2.34)$$

Summate all the triangle around center node of the control volume, equation 2.33 and 2.34 will become the following:

$$\sum_{i=1}^{ns}\begin{bmatrix} -\frac{(\alpha_{11}n_{1x}t_x+\alpha_{12}n_{1y}t_y)}{2A}u_1-\frac{(\alpha_{11}n_{2x}t_x+\alpha_{12}n_{2y}t_y)}{2A}u_2-\frac{(\alpha_{11}n_{3x}t_x+\alpha_{12}n_{3y}t_y)}{2A}u_3 \\ -\frac{(\theta_{11}n_{1y}t_x+\theta_{12}n_{1x}t_y)}{2A}\nu_1-\frac{(\theta_{11}n_{2y}t_x+\theta_{12}n_{2x}t_y)}{2A}\nu_2-\frac{(\theta_{11}n_{3y}t_x+\theta_{12}n_{3x}t_y)}{2A}\nu_3 \end{bmatrix}_i=0 \qquad (2.35)$$

$$\sum_{i=1}^{ns}\begin{bmatrix} -\frac{(\alpha_{21}n_{1x}t_x+\alpha_{22}n_{1y}t_y)}{2A}\nu_1-\frac{(\alpha_{21}n_{2x}t_x+\alpha_{22}n_{2y}t_y)}{2A}\nu_2-\frac{(\alpha_{21}n_{3x}t_x+\alpha_{22}n_{3y}t_y)}{2A}\nu_3 \\ -\frac{(\theta_{21}n_{1y}t_x+\theta_{22}n_{1x}t_y)}{2A}u_1-\frac{(\theta_{21}n_{2y}t_x+\theta_{22}n_{2x}t_y)}{2A}u_2-\frac{(\theta_{21}n_{3y}t_x+\theta_{22}n_{3x}t_y)}{2A}u_3 \end{bmatrix}_i=0 \qquad (2.36)$$

This is the discrete finite volume form of the Linear-Elastic equations, and they are applied using the control volume for each node. To simplify the equations, w_{11}-w_{16} and w_{21}-w_{26} in following equations 2.37 and 2.38 are treated as corresponding weights.

$$\sum_{i=1}^{ns}[w_{11}u_1+w_{12}u_2+w_{13}u_3+w_{14}\nu_1+w_{15}\nu_2+w_{16}\nu_3]_i=0 \qquad (2.37)$$

$$\sum_{i=1}^{ns}[w_{21}\nu_1+w_{22}\nu_2+w_{23}\nu_3+w_{24}u_1+w_{25}u_2+w_{26}u_3]_i=0 \qquad (2.38)$$

After weights are computed, add them to corresponding matrices to achieve the global matrix. The global matrix stores values in a compressed row storage system which is a three-dimensional array where stores 2x2 matrices, and three special one-dimensional integer arrays named ja, ia and iau will be needed to achieve this goal.

The ja array contains all the entries in the augmented node-to-node hash table, created by simply conjugating the lists in a row. The dimension of ia array is the number of nodes

plus one, it contains the index in the ja array where the augmented node-to-node list for each node begins. The last entry is added to simplify the indexing. Finally, the iau array is dimensioned by the number of nodes, it contains the index in the ja array that contains the node itself. This will be the location in the global matrix where the 2x2 matrices are stored. The code segments listed below are to demonstrate the use of ja, ia and iau arrays and how to add weights $w_{11} - w_{16}$ and $w_{21} - w_{26}$ to the global matrix.

```
mat[iau[n]][0][0] += w11;
mat[iau[n]][0][1] += w14;
mat[iau[n]][1][0] += w24;
mat[iau[n]][1][1] += w21;

for (k = ia[n]; k < ia[n+1]; k++) {
    if (ja[m] == n2) {
        mat[k][0][0] += w12;
        mat[k][0][1] += w15;
        mat[k][1][0] += w25;
        mat[k][1][1] += w22;
        break;
     }
}

for (k = ia[n]; k < ia[n+1]; k++) {
    if (ja[m] == n3) {
        mat[k][0][0] += w13;
        mat[k][0][1] += w16;
        mat[k][1][0] += w26;
```

```
            mat[k][1][1] += w23;
            break;
        }
    }
```

By solving this global matrix, perturbations for each interior node will be achieved.

Adaptive Optimization-Based Smoothing

This is a new version of optimization-based smoothing approach as in ref [19], and the version in ref [20] will be published in January 2015. But only optimization-based smoothing part will be used in this thesis. Since the very new material hasn't been published yet, details will be presented as follows. The basic optimization-based mesh smoothing method is to unify a cost function and perturb the nodes to minimize the cost. The cost of an element is on basis of the status of it. And if the element is inverted, condition number cannot recognize it, so two formulations are used.

$$C = \begin{cases} 1 - J, & if J \leq 0.0 \\ 1 - \frac{1}{CN}, & if J > 0.0 \end{cases} \tag{2.39}$$

Frietag and Knupp[21] used multiple perturbation directions and mesh quality function evaluations to determine the new position of each node, after several trials, the better quality mesh will be attained. Instead of trying different directions, in this new approach, the directions of node perturbation are achieved by calculating the sensitivity of the element cost function with respect to each node of the element. These element-based sensitivities work fine for the derived cost function. The vectors emanating from the corners of the triangle are the sensitivity vectors for each node. They are the gradient vectors for each

node for increasing the cost of the element. Perturbation directions for each node are simply the inverse of gradient vector as shown in Figure 2.3.

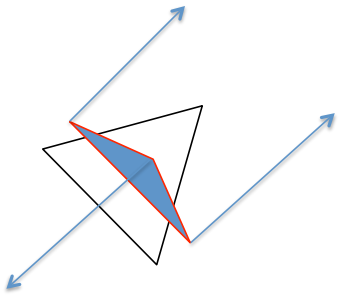

Figure 2.3 Sensitivity vectors for element cost function

A nodal perturbation vector is constructed by a cost-weighted average of the sensitivity vectors for that node from all the elements given by $\vec{s_n}$. These are weighted by the square of the element cost function for the number of elements surrounding that node. The cost-weighted average vector will typically be smaller in magnitude than maximum element-based vectors. The perturbation amount is restricted to a fraction of the smallest distance to any neighboring node, typically 1% to 5%. The magnitude of the average perturbation vector is also limited to one or less. Multiple iterations are required to "converge" the mesh to a stable configuration.

$$\vec{p}_n = -\frac{\sum_{e=1}^{ne} (\vec{s}_n)_e C_e^2}{\sum_{e=1}^{ne} C_e^2} \tag{2.40}$$

The sensitivities can be computed a number of ways. Differentiating the cost function formulae with respect to X and Y coordinate of the three corner nodes generates the analytic derivatives. The chosen method for computing them uses finite difference to compute the function value and the twelve derivative quantities through numerical chain rule differentiation.

CHAPTER 3

MESH RUPTURING

Mesh rupturing consists of two parts, geometry insertion and geometry motion. It is used as the mesh technique in this thesis, and the two parts will be discussed in details as follows.

Geometry Insertion

In the geometry insertion phase, a uniform unstructured background mesh as shown in Figure 3.1 and a geometry as shown in Figure 3.2 are needed. The resolution of this geometry is shown as in Figure 5.5, and the resolution of the background mesh will make the resolution of newly generated airfoil as shown in Figure 5.6.

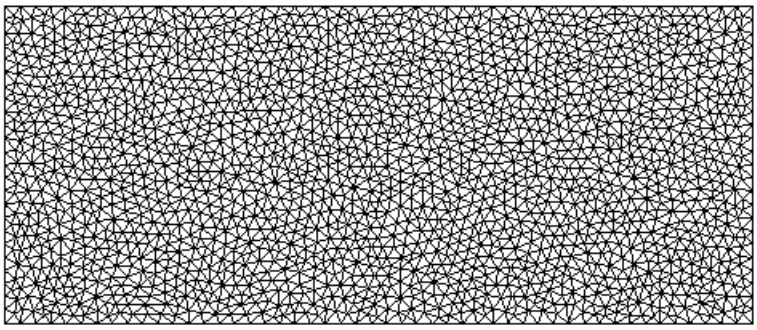

Figure 3.1 Uniform background mesh

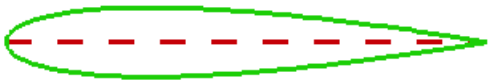

Figure 3.2 Geometry with bisected centerline

Crackle

The first step is to bisect the geometry. Whether the geometry is symmetric or not, it will have a centerline which bisects it, and this centerline is just simply composed of midpoints of paired nodes describing the geometry. Bisected centerline is shown as red in the Figure 3.2. Before inserting geometry into background mesh, the head node is found by a walking algorithm which is the closet to the beginning of the centerline. Start from the head node, edges from it to every neighboring node is tested to see which is best aligned with the direction of centerline, and then it will serve as the next seed to repeat the edge search. This process will come to a stop when all the edges have been traced down and reach the end of centerline. The grid points and edges aligning crackle line are duplicated, thus a closed hole is created in background mesh topology where geometry could be inserted into. And mesh rupturing is named after this phase because a jagged line in the background mesh

will be obtained and it is like the rupture caused by an earthquake. The blue line is the crackle line as shown in Figure 3.3.

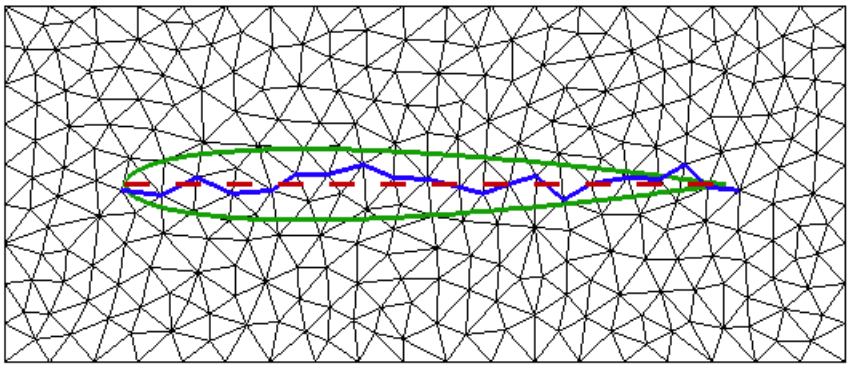

Figure 3.3 Geometry position and crackle line in background mesh

Split, Snap and Pop

After crackling, the crackle line consists of a head node, a tail node and other nodes in between which are called interior nodes. The background mesh is going to be split along this crackle line by duplicating all the interior nodes of the crackle line, and the new ones are going to have a new node number. Cells adjacent to this line will be chosen to be one of the two sides. Cells on either side will take the original nodes or the duplicated ones and the connectivity will also be split along this line. The two curves of geometry are projected onto the two sides, and then parametric distribution is used to smooth the two curves. This snapping process will cause invalid elements to pop out after which the whole mesh would be smoothed by methods including Linear-Elastic smoothing, Winslow smoothing and

optimization-based smoothing. As shown in Figure 3.4, the red edge is an example, it is split into two by duplicating one node, and connectivity has been changed after this procedure.

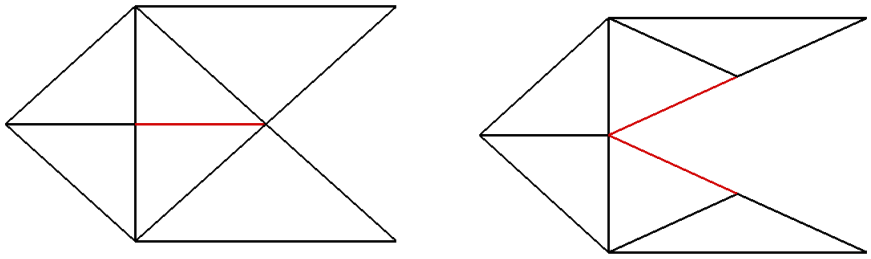

Figure 3.4 Edge split and merging process as it is inversed

Geometry Motion

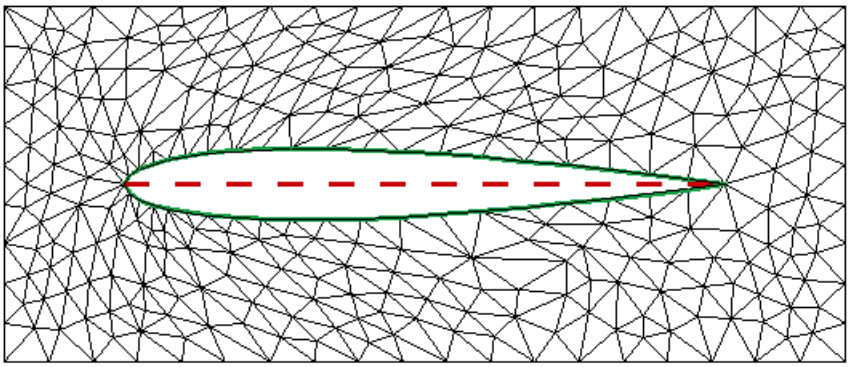

Figure 3.5 Geometry inserted into background mesh

Once the geometry is inserted as in Figure 3.5, it is ready to move. Instead of using reinsertion of the geometry, which can enable any kind of motion, such as translation, rotation

or stretching. This new mesh movement scheme will maintain nodes on each side as the airfoil moves. And for each time step, the magnitude of the motion would be one edge length at most. The whole geometry would be translated at the same length. The mesh would be split around the edge, and merged at the tail. Compared to mesh regeneration, this method is less flexible because it only allows for small range of motion. The new physical locations of end nodes would be the seeds for the walking algorithm to search the new head and tail nodes. At the head, the old head node will split into two, while at the tail, the two old nodes adjacent to the tail will be collapsed into one with the tail node merging into background mesh. The splitting process at leading edge is the reverse the one at trailing edge. The process is shown in the following figures cited from ref[10].

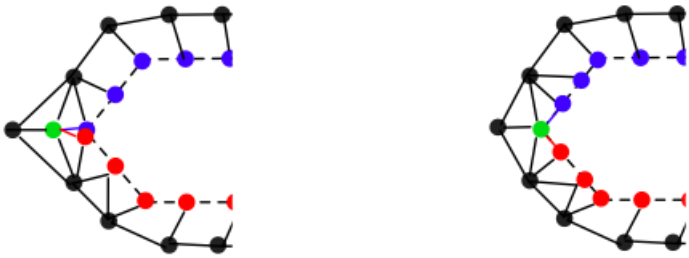

Figure 3.6 Old head node split into two

New head node selection

The walking algorithm to find new head node is based on the crackled mesh, and the new boundaries will be updated. Otherwise, if the node is selected from the smoothed mesh after each time step, then the body will keep pushing the nodes towards its motion direction which might cause infinity loops to find new head nodes. This process is important to keep mesh quality, because a bad choice will cause invalid elements. For each selection phase, a list of

neighboring nodes of current head node is built. Two types of nodes will be removed from selection, first type is the nodes which are on geometry, second type is the ones if added to the list would cause degenerated elements. All the nodes remaining in the list will be tested to see which one is the best option since all of them will keep a valid topology.

With all details described, the procedure for Mesh Rupturing is presented below with a flowchart in Figure 3.7.

1: Read in background mesh and geometry
2: Find crackle line in background mesh by sampling from bisected line of geometry
3: Duplicate interior nodes of crackle line
4: Apply arc length parametric distribution on closed hole to get new geometry
5: Apply Linear-Elastic smoothing and optimization-based smoothing to get high-quality grid
6: **for** $i \leq$ movement loops **do**
7: Select new head node and tail node
8: Split head node and merge two nodes adjacent the tail node
9: Apply arc length parametric distribution
10: Apply Linear-Elastic smoothing and optimization-based smoothing
11: **end for**
12: Output mesh

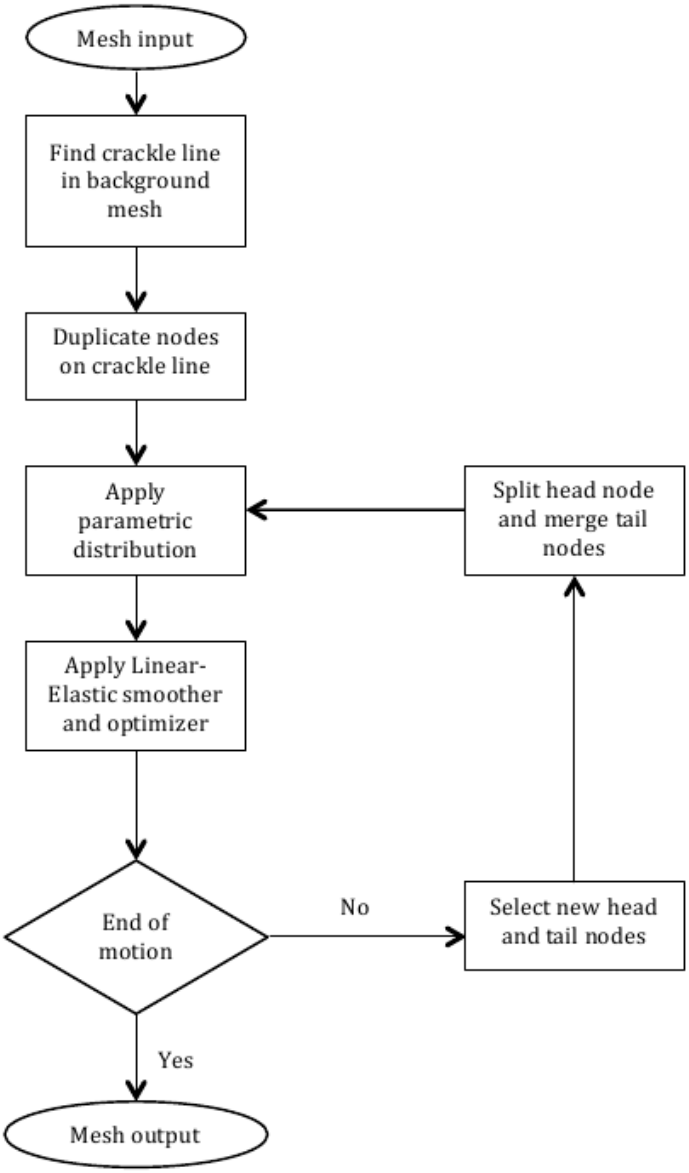

Figure 3.7 Flow chart of mesh rupturing process

CHAPTER 4

FLOW SOLUTION

This chapter will present the procedure to build the Euler solver from three aspects, the governing equations, spatial discretization and temporal discretization. Based on the fact that finite volume method is exploited in this thesis, all the equations will be stated in integral form.

Euler Equations

In this thesis, the flow is assumed to be governed by the two-dimensional time-dependent Euler equations written as:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{4.1}$$

The integral form of equation 4.1 is:

$$\iint (\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}) dxdy = 0 \tag{4.2}$$

For steady-state problems, the time term could be written just as:

$$\iint \frac{\partial Q}{\partial t} dxdy = \frac{d}{dt} \iint Q dxdy \tag{4.3}$$

While for unsteady ones, the time term is different due to grid speed, which would be written as:

$$\iint \frac{\partial Q}{\partial t} dxdy = \frac{d}{dt} \iint Q dxdy - \iint div Q \vec{W}_s dxdy \tag{4.4}$$

where $\vec{W}_s$ is the local control volume face velocity. By substituting unsteady time term back into Euler equations 4.2, it can be rearranged as:

$$\frac{d}{dt}\iint Q dxdy + \iint (\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} - divQ\vec{W}_s)dxdy = 0 \tag{4.5}$$

The flux terms should be put in divergence form so the volume integral can be converted into a surface integral. This is achieved by rewriting the flux terms as a dot product of the gradient operator, ∇, and the flux tensor, $\vec{F}$.

$$\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = \nabla \bullet \vec{F} \tag{4.6}$$

where

$$\nabla = \frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j} \tag{4.7}$$

and

$$\vec{F} = F\vec{i} + G\vec{j} \tag{4.8}$$

By exploiting Gauss' theorem, the volume integral is converted into a surface integral and equation 4.5 becomes:

$$\frac{d}{dt}\iint_{\Omega} Q d\Omega + \oint_{\Gamma} (\vec{F} - Q\vec{W}_s) \bullet \hat{n} d\Gamma = 0 \tag{4.9}$$

where $\vec{F}$ is the flux, and the three terms of the surface integral are expanded below.

$$\vec{F} = \begin{bmatrix} \rho(u\hat{i} + v\hat{j}) \\ \rho u(u\hat{i} + v\hat{j}) + p\delta_{im}\hat{m} \\ \rho v(u\hat{i} + v\hat{j}) + p\delta_{jm}\hat{m} \\ (E+p)(u\hat{i} + v\hat{j}) \end{bmatrix} \tag{4.10}$$

$$Q\vec{W}_s = \begin{bmatrix} \rho\vec{W}_s \\ \rho u\vec{W}_s \\ \rho v\vec{W}_s \\ E\vec{W}_s \end{bmatrix} \tag{4.11}$$

with

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix} \tag{4.12}$$

Spatial Discretization of Flux-Vector Splitting

Within this thesis, the convective fluxes are computed with Van-Leer flux-splitting scheme. For second order accuracy, the values at dual-cell interfaces are reconstructed using gradients at each node computed with a least-square technique. Limiters may be applied for flow with strong shocks when reconstructing values. For all results presented in this paper, the convective flux scheme is second order with unlimited reconstruction.

The spatial discretization is based on a node-centered scheme where the conservative variables are stored at each node, and the control volume is composed of median duals around each node. Since the flux is approximated by FVS of Van-Leer, the flux vectors are split into forward and backward contributions. Because of grid speed term integrating into the scheme, the flux calculation on dynamic mesh will use the ones in ref[11], which is

derived as following:

$$F^{\pm} = \begin{pmatrix} f^{\pm}_{mass} \\ f^{\pm}_{mass}(\hat{n}_x \frac{-\bar{U} \pm 2a}{\gamma} + u) \\ f^{\pm}_{mass}(\hat{n}_y \frac{-\bar{U} \pm 2a}{\gamma} + v) \\ f^{\pm}_{energy} \end{pmatrix} \tag{4.13}$$

where

$$\bar{U} = \hat{n}_x u + \hat{n}_y v - \hat{W}_s \tag{4.14}$$

$$M = \frac{\bar{U}}{a} \tag{4.15}$$

$$f^{\pm}_{mass} = \pm \rho a (M \pm 1)^2 / 4 \tag{4.16}$$

$$f^{\pm}_{energy} = \{ (\frac{-(\gamma - 1)\bar{U}^2 \pm 2(\gamma - 1)\bar{U}a + 2a^2}{(\gamma^2 - 1)}) + \frac{u^2 + v^2}{2} - \frac{\hat{W}_s}{\gamma}(-\bar{U} \pm 2a) \} \tag{4.17}$$

The flux vector F is split in a two-dimensional fashion into forward F^+ and backward F^- vectors for $|\bar{U}| < a$ in above equations, for non split form,

$$F = \begin{pmatrix} \rho \bar{U} \\ \rho \bar{U} u + n_x p \\ \rho \bar{U} v + n_y p \\ E\bar{U} + p(\bar{U} + \vec{W}_s) \end{pmatrix} \tag{4.18}$$

Temporal Discretization

The following is a summary and derivation from details in ref [12]. The time-advancement scheme is as following, which is based on the steps for rigid mesh case, with extension to

allow for general deforming mesh.

$$\frac{\partial(QV)}{\partial t} = R \tag{4.19}$$

Evaluating at time level n+1, and writing the time derivative in BDF2 form, which give us second order accuracy in time, equation 4.19 becomes:

$$\frac{1}{\Delta t}[\phi_{n+1}(QV)^{n+1} + \phi_n(QV)^n + \phi_{n-1}(QV)^{n-1}] = R(Q^{n+1}) \tag{4.20}$$

where

$$\phi_{n+1} = \frac{3}{2}, \phi_n = -2, \phi_{n-1} = \frac{1}{2} \tag{4.21}$$

Similarly, the Geometric Conservation Law is discretized as:

$$\frac{1}{\Delta t}[\phi_{n+1}V^{n+1} + \phi_n V^n + \phi_{n-1}V^{n-1}] = R_{GCL}^{n+1} \tag{4.22}$$

The discretization of the time derivative term in the GCL must has the same sequence as the conservation equations. The Equation 4.20 may be rewritten as:

$$\frac{1}{\Delta t}[Q^n(\phi_{n+1}V^{n+1}+\phi_n V^n+\phi_{n-1}V^{n-1})+\phi_{n+1}(Q^{n+1}-Q^n)+\phi_{n-1}(Q^{n-1}-Q^n)] = R^{n+1} \tag{4.23}$$

Substituting GCL term back into equation 4.23, then it becomes:

$$Q^n R_{GCL}^{n+1} + \frac{1}{\Delta t}(\phi_{n+1}(Q^{n+1} - Q^n) + \phi_{n-1}(Q^{n-1} - Q^n)) = R^{n+1} \tag{4.24}$$

The right-hand side could be linearized about time level n, resulting in an implicit scheme for $\Delta Q = Q^{n+1} - Q^n$. However, this linearization will introduce time-step dependent error into the result. Since the nonlinear residual $R(Q)$ is hard to linearize exactly, so another error will result in this approximate linearization. In order to reduce the errors caused by

this linearization, a pseudo-time is added:

$$(V\frac{\partial Q}{\partial \tau})^{n+1} + Q^n R_{GCL}^{n+1} + \frac{1}{\Delta t}(\phi_{n+1}(Q^{n+1} - Q^n) + \phi_{n-1}(Q^{n-1} - Q^n)) = R^{n+1} \tag{4.25}$$

where this pseudo time τ is constrained as $0 < \tau < \infty$ during each physical time step. Now, Q is a function of t and τ while V is only decided by t. By discretizing this term with a first-order backward difference about pseudo-time level $m+1$, since the previous values of the solution do not depend on the current pseudo-time level, it gives us:

$$(V^{n+1}\frac{Q^{n+1,m+1} - Q^{n+1,m}}{\partial \tau}) + Q^n R_{GCL}^{n+1} + \frac{1}{\Delta t}(\phi_{n+1}(Q^{n+1,m+1} - Q^n) + \phi_{n-1}(Q^{n-1} - Q^n)) = R^{n+1,m+1} \tag{4.26}$$

Noting that there is no significant advantage of using a higher-order discretization of the pseudo-time term, first-order temporal discretization is typically used with local-time stepping methods to advance time-dependent problem to a steady state. In addition, the use of pseudo time here is exactly analogous to local-time stepping for steady flows.

The nonlinear residual at the $m+1$ pseudo-time level may be linearized about the m^{th} level as:

$$R^{n+1,m+1} = R^{n+1,m} + (Q^{n+1,m+1} - Q^{n+1,m})\frac{\partial R^{n+1,m}}{\partial Q} \tag{4.27}$$

Subtracting $\phi_{n+1}V^{n+1}Q^{n+1,m}/\Delta t$ from both sides and defining $\Delta Q^{n+1,m} = Q^{n+1,m+1} - Q^{n+1,m}$, the final form of equation yields after rearranging:

$$\begin{aligned}\left[(\frac{V^{n+1}}{\Delta \tau} + \frac{\phi_{n+1}V^{n+1}}{\Delta t})I + (\frac{\partial R}{\partial Q})^{n+1,m}\right]\Delta Q^m = &-R(Q^{n+1,m}) + Q^n R_{GCL}^{n+1} \\ &- \frac{\phi_{n+1}V^{n+1}(Q^{n+1,m} - Q^n)}{\Delta t} \\ &- \frac{\phi_{n-1}V^{n-1}(Q^{n-1} - Q^n)}{\Delta t}\end{aligned} \tag{4.28}$$

Geometry Conservation Law

The crucial difference of steady-state problems and unsteady ones lies in the deformation of element. In order to maintain the global conservation and compute the local volume element, GCL is first brought up by Thomas and Lombard [13], formulated to improve the method of boundary-conforming coordinate transformations to solve unsteady problems.

Boundary Conditions

To ensure the flow tangency along the surface of the airfoil, boundary conditions are imposed. The conservative variables on boundary edge are determined by the arithmetic average of the two nodes which form the edge on surface. Use this average as the input to compute the real conservative variables and we will get results where the normal velocity component is zero. At the farfield, a characteristic analysis based on Riemann invariants is applied to determine the conservative variables. Details are in ref[14], which is modified to suit two-dimensional grid.

CHAPTER 5
COMPARISONS

Initial Condition for unsteady Cases

The flow solution of steady-state is obtained on the NACA0012 airfoil inviscidly at a freestream Mach number of $M_\infty = 0.755$ and an angle of attack of $\alpha_0 = 0.016$ degree. This will be the initial condition for unsteady cases.

Unsteady Flow Results

The periodic motion of the airfoil is defined by the angle of attack as a function of time with the amplitude $\alpha_m = 2.51$ degree as

$$\alpha(t) = \alpha_0 + \alpha_m sin(\omega t) \tag{5.1}$$

and oscillating at a frequency of 62.5 Hz, producing a reduced frequency $k = 0.0814$, where ω is related to the reduced frequency k by

$$k = \omega c / 2U_\infty \tag{5.2}$$

The number of time steps per cycle is 416, and the undimensional time for each time step is 0.125.

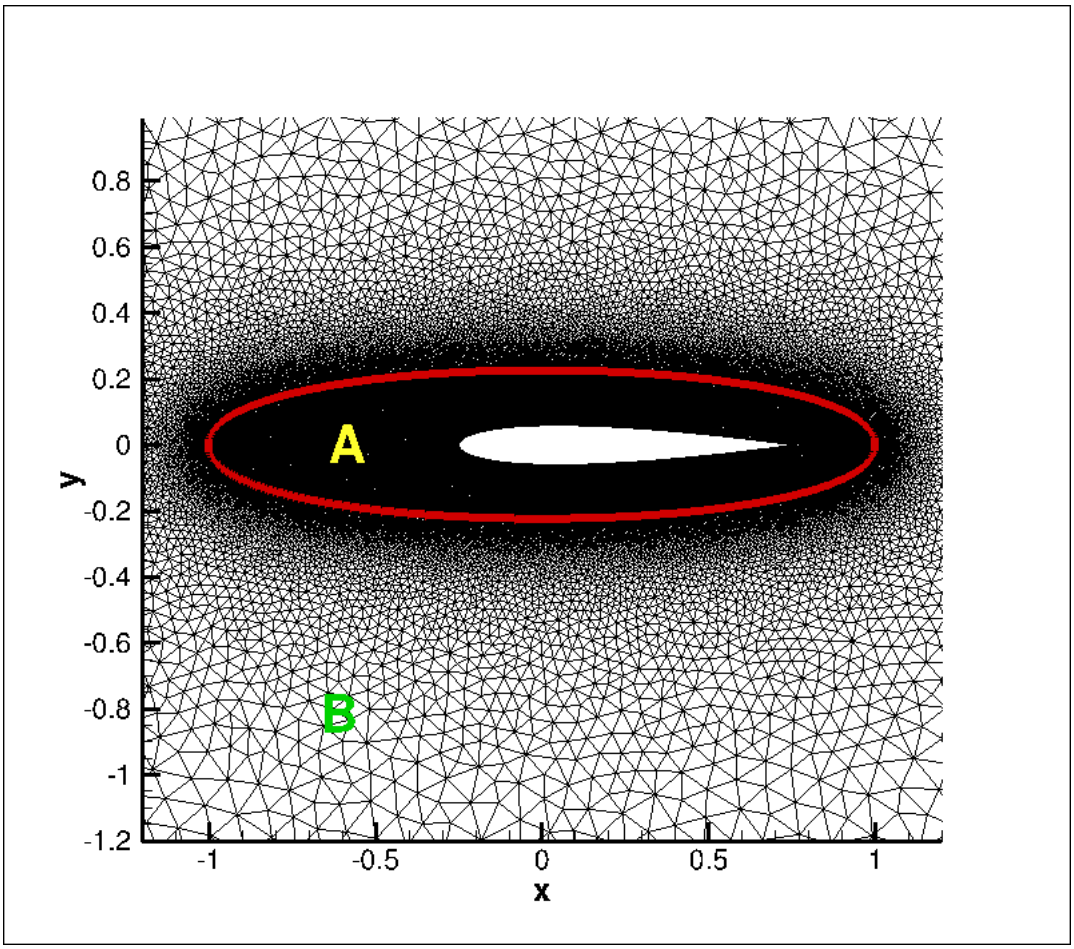

Figure 5.1 Whole mesh with geometry inserted

The airfoil is inserted into the background mesh by using mesh rupturing approach, and the result is shown as in Figure 5.1. The background mesh is divided into two parts by the red line, part A is where the airfoil resides in and moves, and all the nodes in this area are free to move, part B is where all the nodes are fixed. In this way, the movement of airfoil will only perturb the nodes in part A, which will save a lot of computations for unsteady case. Besides, due to the essence of mesh rupturing, part A is a uniform fine mesh while

part B is not. The shape of part A is arbitrary but the size of it should be restricted by the movement of the airfoil, since the geometry should not go across the red line. Therefore, if the area is too big, computational time would increase.

In order to show that this new mesh technique is well behaved combined with the flow solver, three validation cases have been run as follows:

- Case A : whole mesh oscillating as a rigid body at farfield Mach number 0.755 and angle of attack 0.016
- Case B : airfoil oscillating with boundary fixed at farfield Mach number 0.755 and angle of attack 0.016
- Case C : airfoil oscillating with boundary fixed at farfield Mach number 0.75 and angle of attack 0.016, and moving facing flow at speed of 0.005, which makes the relative Mach number equals 0.755

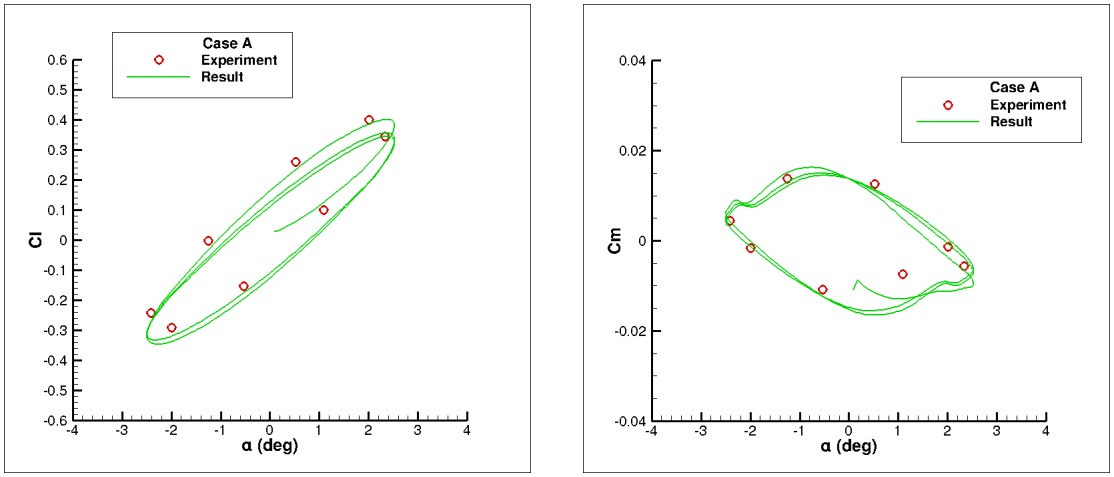

Figure 5.2 Lift coefficient and moment coefficient of Case A

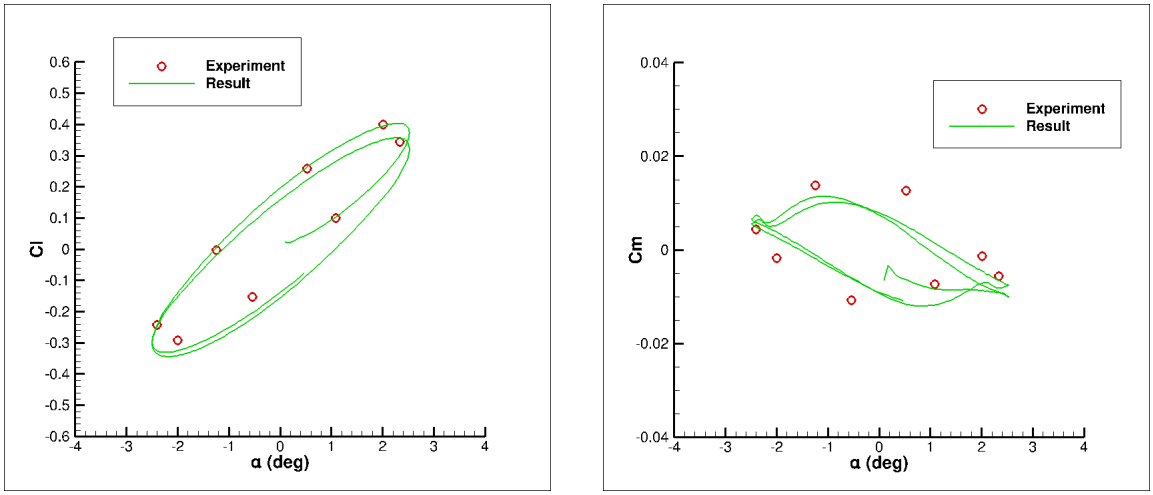

Figure 5.3 Lift coefficient and moment coefficient of Case B

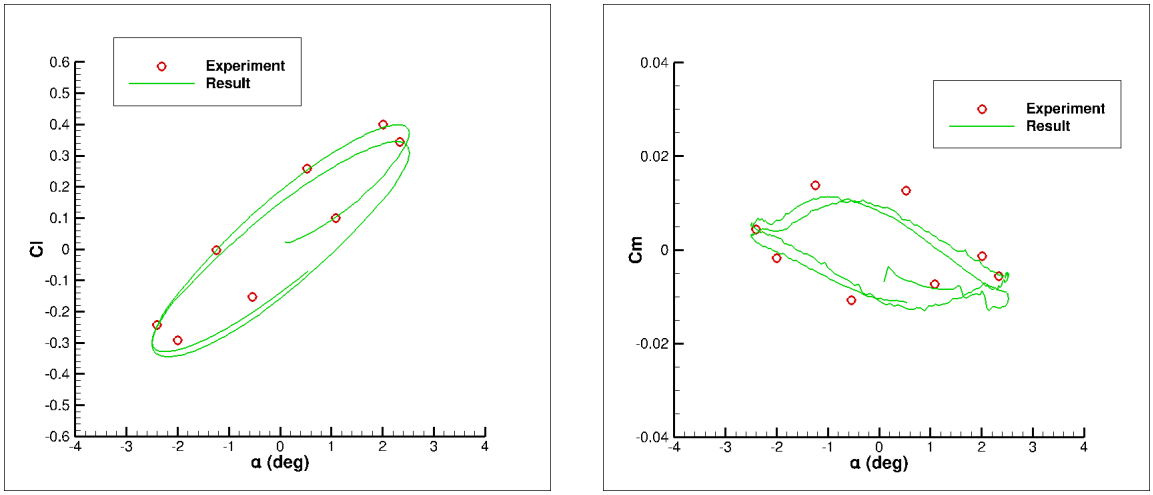

Figure 5.4 Lift coefficient and moment coefficient of Case C

The above Figures 5.2, 5.3 and 5.4 are the comparisons of experimental lift and moment coefficients versus the instantaneous angle of attack for the NACA0012 airfoil pitching implemented in three different ways. These coefficients show the changes due to the angle of attack during a cycle of motion, and all the three cases present no big difference compared with experimental data.

Slight difference of result compared to experimental data in lift coefficient plots exists, this is mainly caused by the resolution of background mesh. In this thesis, since mesh rupturing technique is exploited, then the number of grid points and their distribution on airfoil are decided by background mesh resolution and arc length parametric function. Once the original NACA0012 airfoil is inserted into the uniform background mesh, the new airfoil generated by the mesh technique will be different compared to the original one. Therefore, when calculating lift, moment and pressure coefficients, the difference will occur because of different grid points distribution. A bigger difference could be seen in moment coefficient plots for the three cases. The result changes smoothly and is close to experimental data in case A, it is also smooth but less close to experimental data in case B. However, in case C, the result is not smooth at all, but the tendency of variation is the same as previous two cases. One reason causing this unsmooth line is that the number and distribution of grid points on airfoil is changing as airfoil moves. Another reason cause this is the same as the one causing difference in pressure coefficient, it will be discussed in details later. The grid points distribution of original NACA0012 and new generated body are as follows.

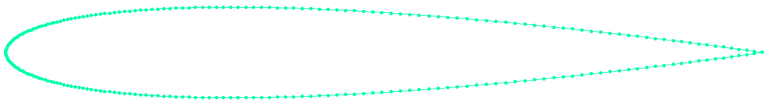

Figure 5.5 Grid points distribution of NACA0012

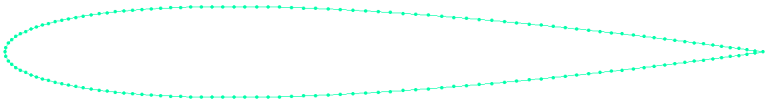

Figure 5.6 Grid points distribution of newly generated NACA0012

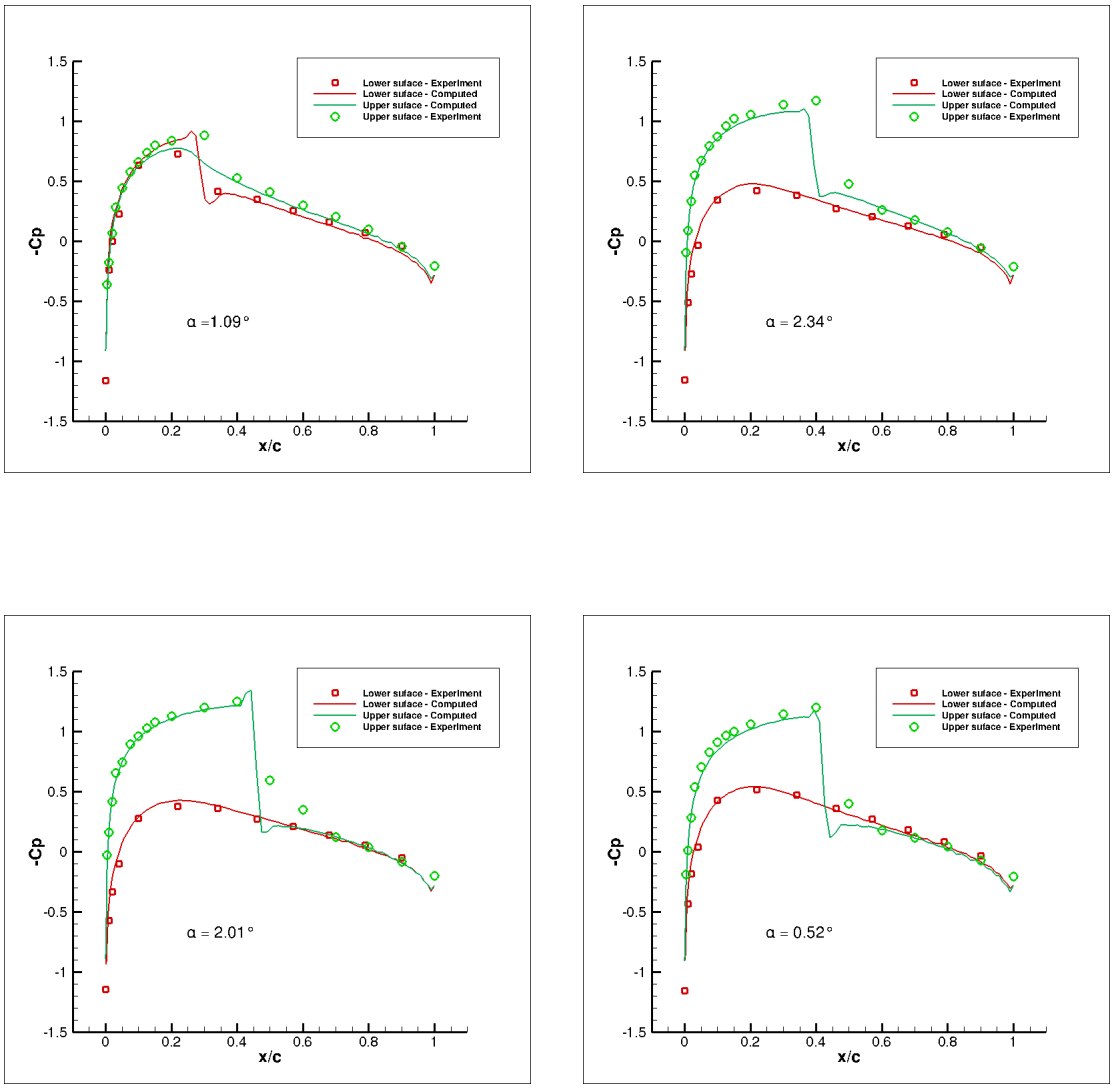

Figure 5.7 Pressure coefficient of Case A(1)

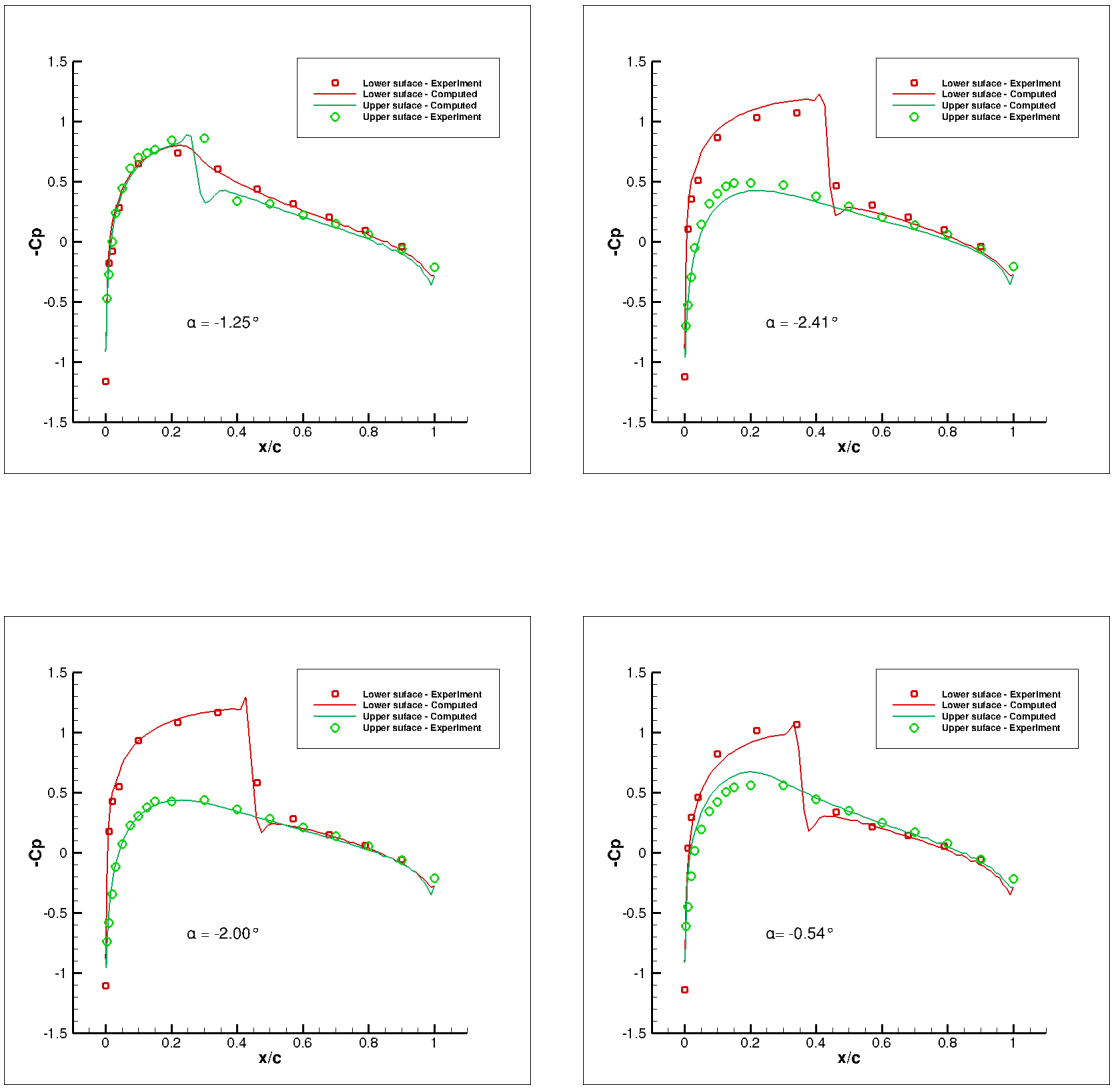

Figure 5.8 Pressure coefficient of Case A(2)

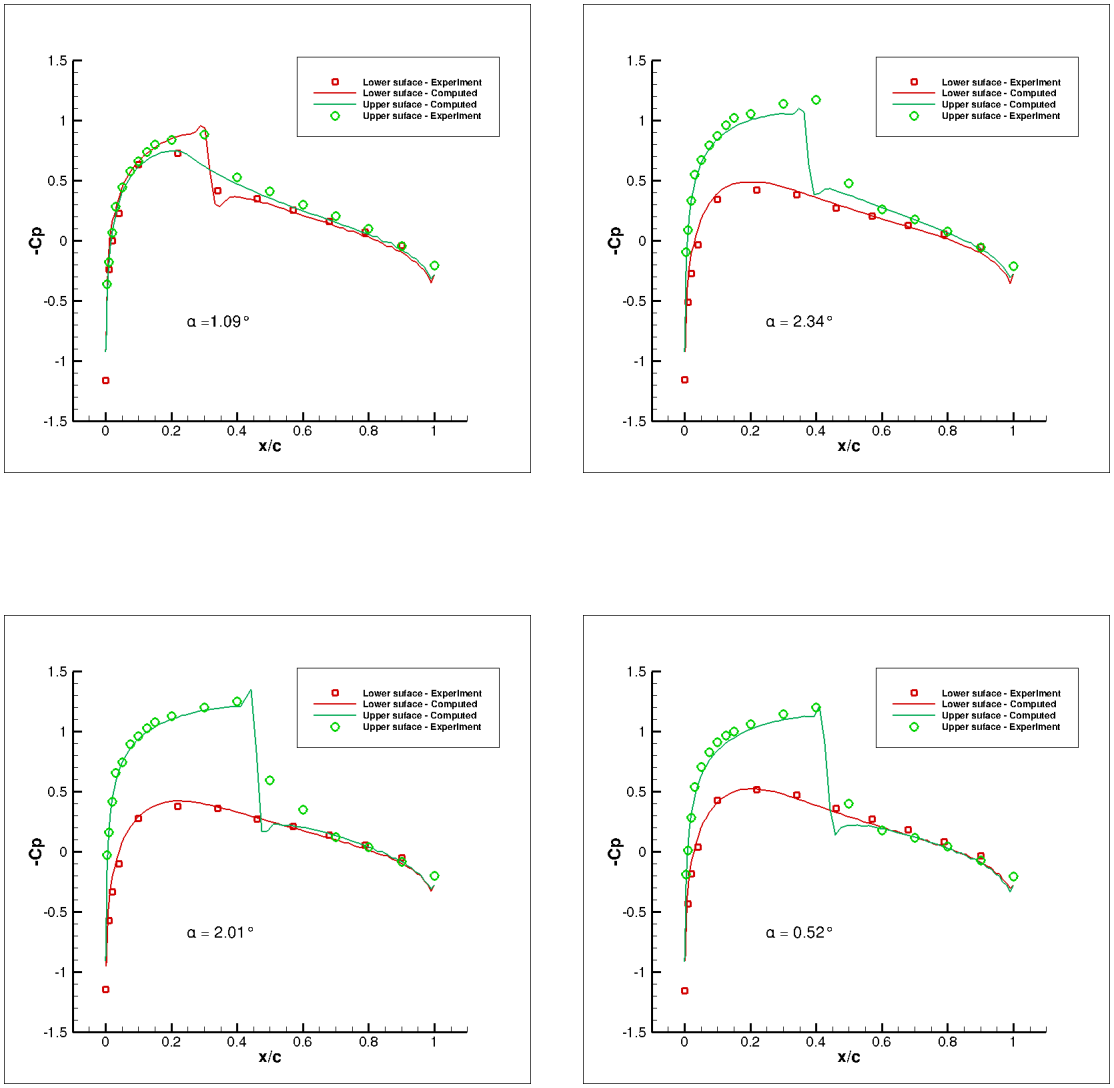

Figure 5.9 Pressure coefficient of Case B(1)

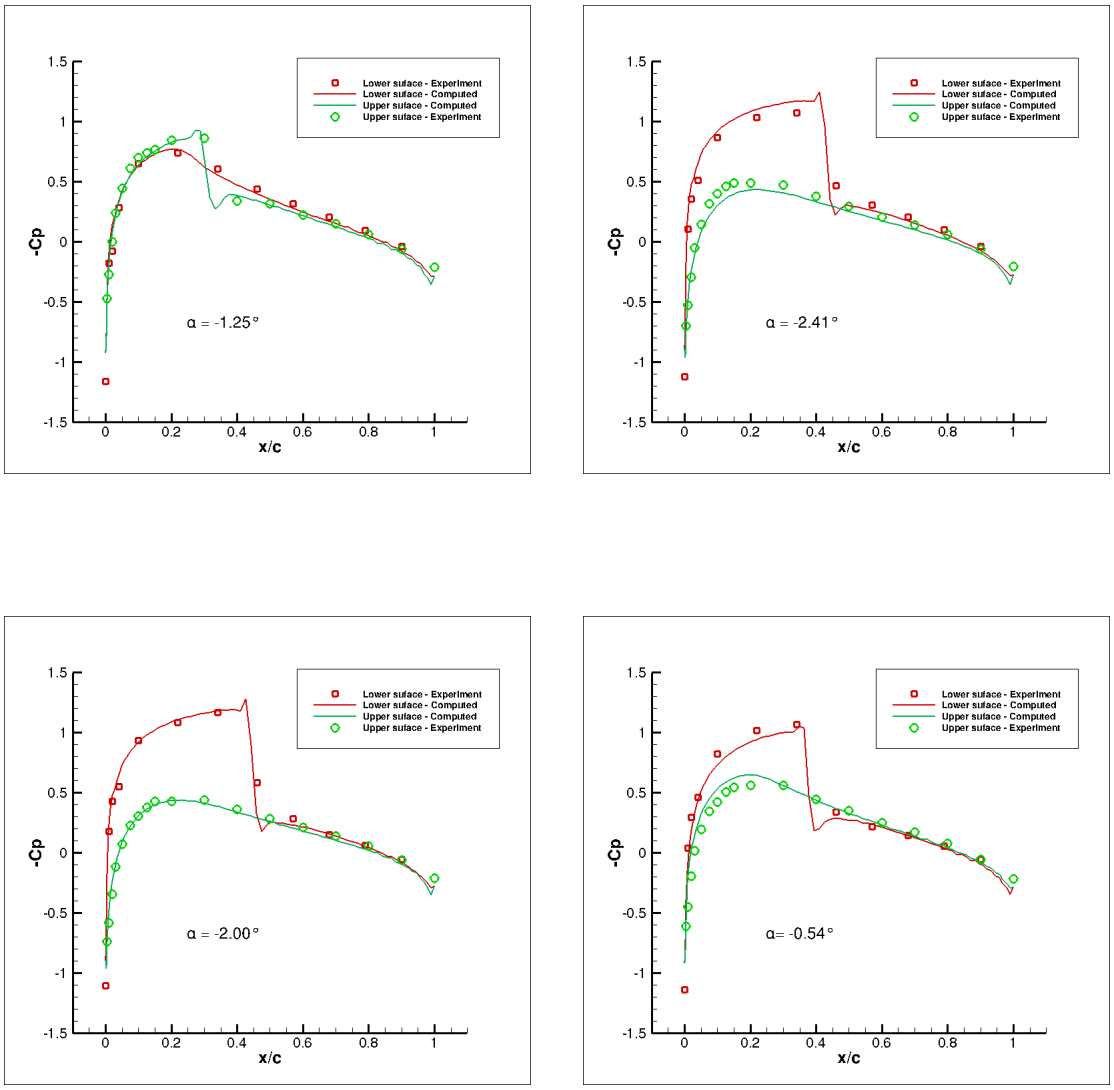

Figure 5.10 Pressure coefficient of Case B(2)

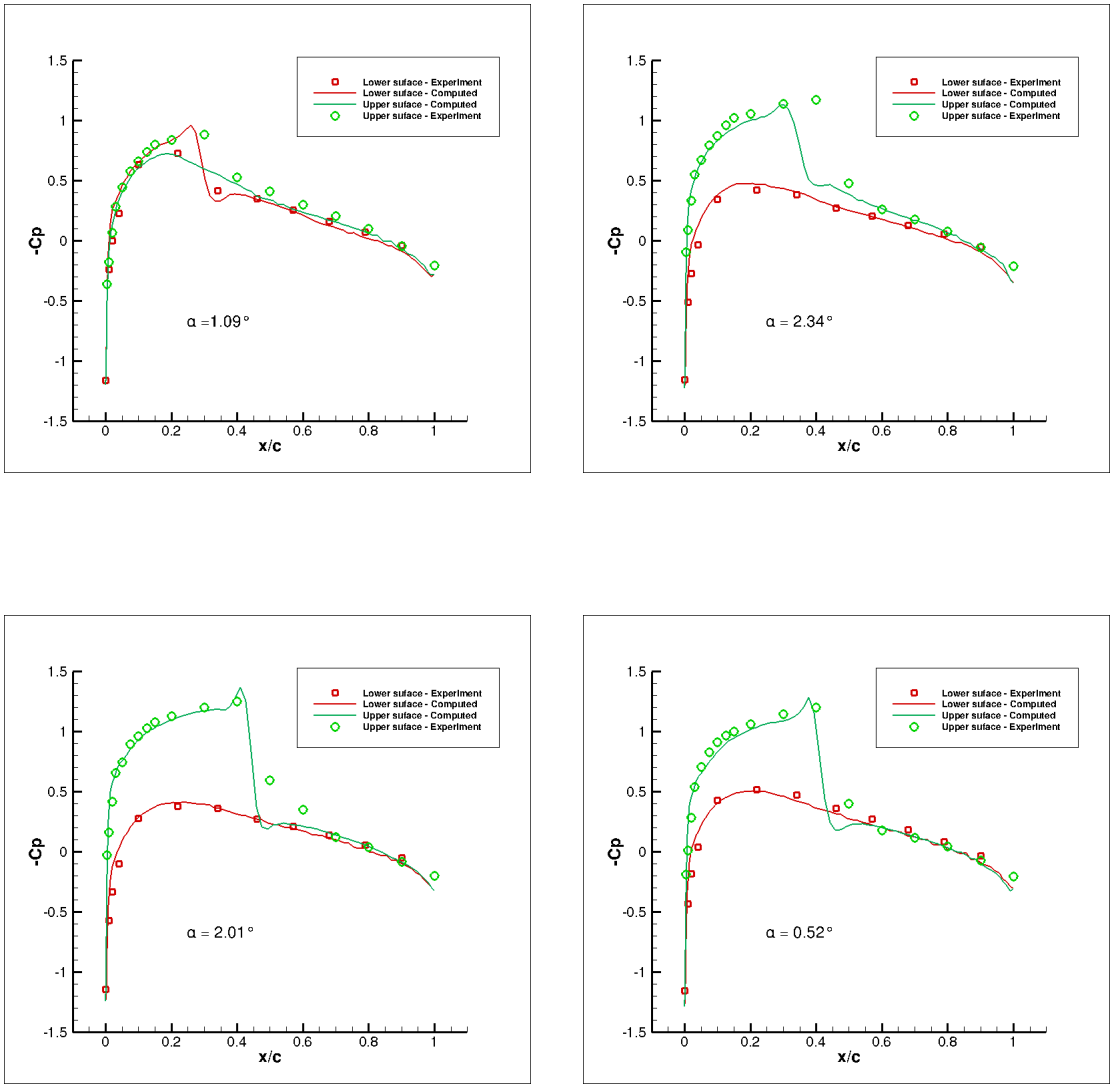

Figure 5.11 Pressure coefficient of Case C(1)

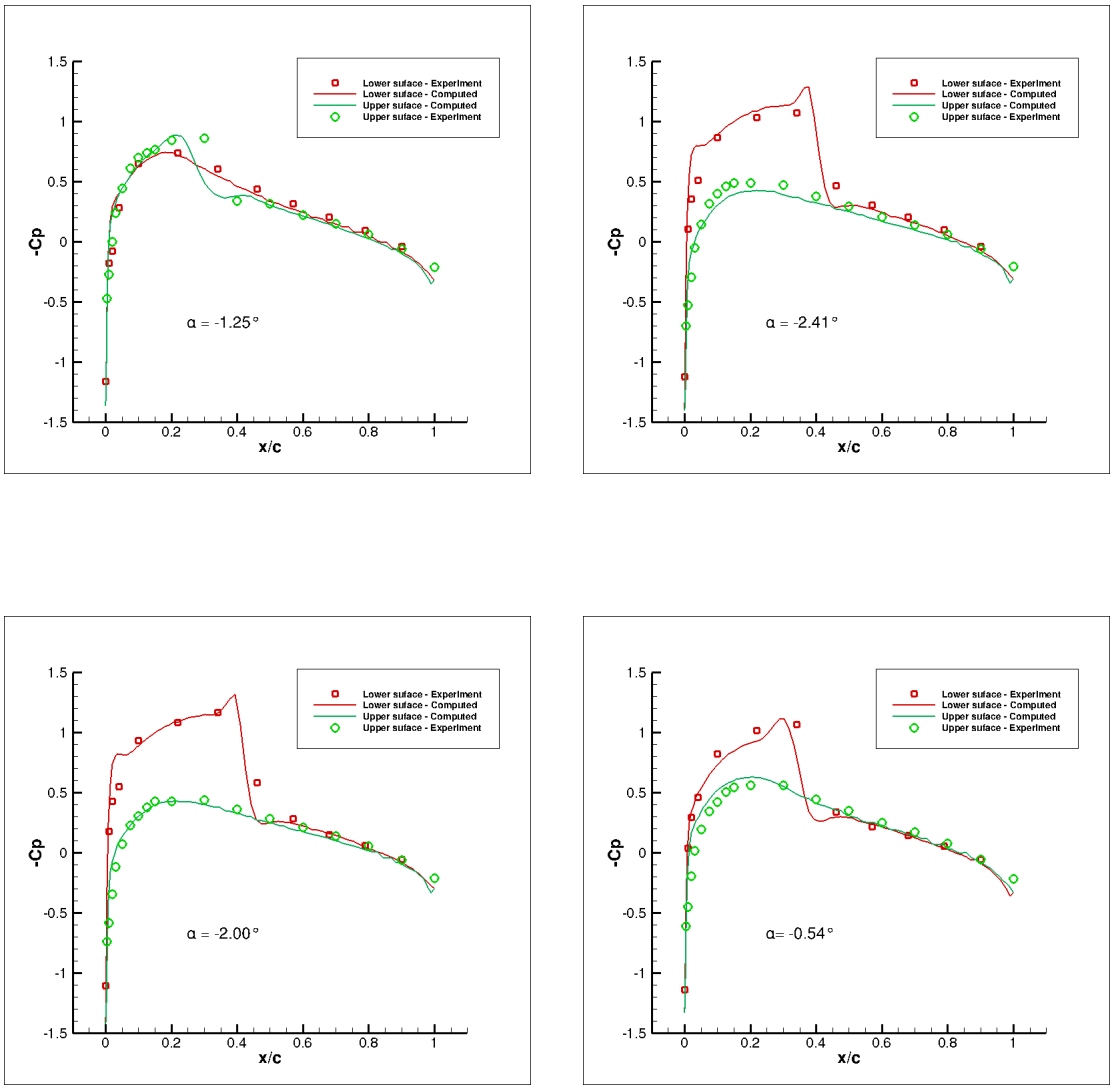

Figure 5.12 Pressure coefficient of Case C(2)

The above Figures 5.7 to 5.12 present instantaneous pressure coefficient distributions of three cases compared to experimental data at eight points during one cycle of motion. When the airfoil is moving towards positive angle of attack, there is a shock wave on the upper surface, and the flow over lower surface is less critical. When the airfoil is moving towards negative angle of attack, the shock wave moves to lower surface which will make the flow over

upper surface less critical. The plots above present that the shock wave appears around the quarter chord, the point where the airfoil oscillates around. Besides, an expected symmetry of flow appears over upper surface and lower surface when the airfoil is moving towards two opposite directions.

Although airfoils are moving in different ways in these three cases, they are implementing the same case, that is, the airfoil oscillating against a freestream at Mach number equals 0.755. Therefore, the three cases should ideally have the same solution results as presented in pressure coefficient distributions plots. As shown from the above plots, case A is the best compared with experimental data among the three cases, secondary is case B, and case C presented slight difference from experimental data.

For case A, there is no deformation of grid, because the whole grid is moving as a rigid body. For case B, the airfoil is oscillating around quarter chord, since the motion is not very large scale, and only smoothing methods would be exploited to implement motion, so small deformation of grid occurs around the airfoil. However, for case C, due to the approach of mesh rupturing technique, number of total grid points on surface of the body is changing as the airfoil moves against freestream. Hence, compared to previous two cases, larger scale deformation of grid takes place which would introduce more errors to solutions. This is the same reason which causing the big difference in moment coefficient plots.

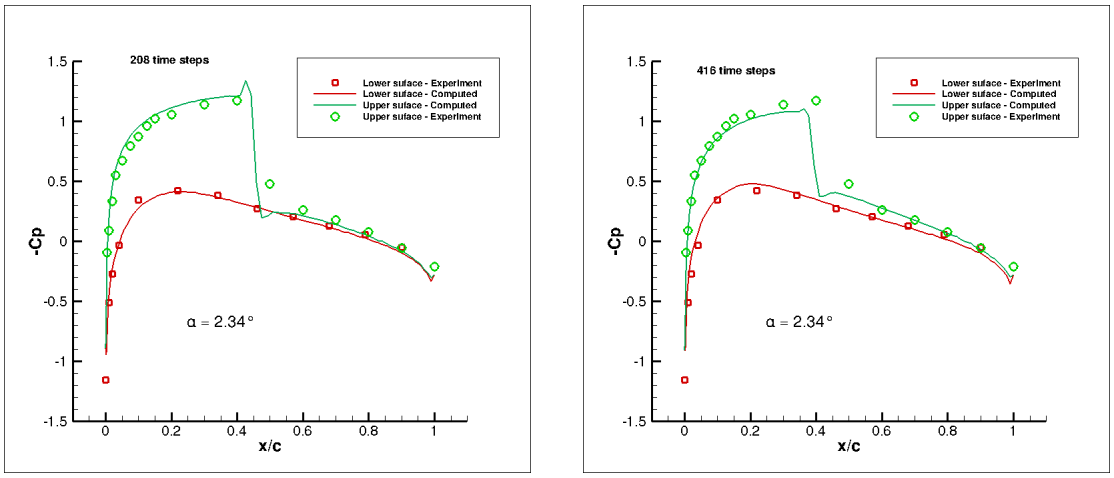

Figure 5.13 Pressure coefficient of case A for 208 and 416 steps per cycle

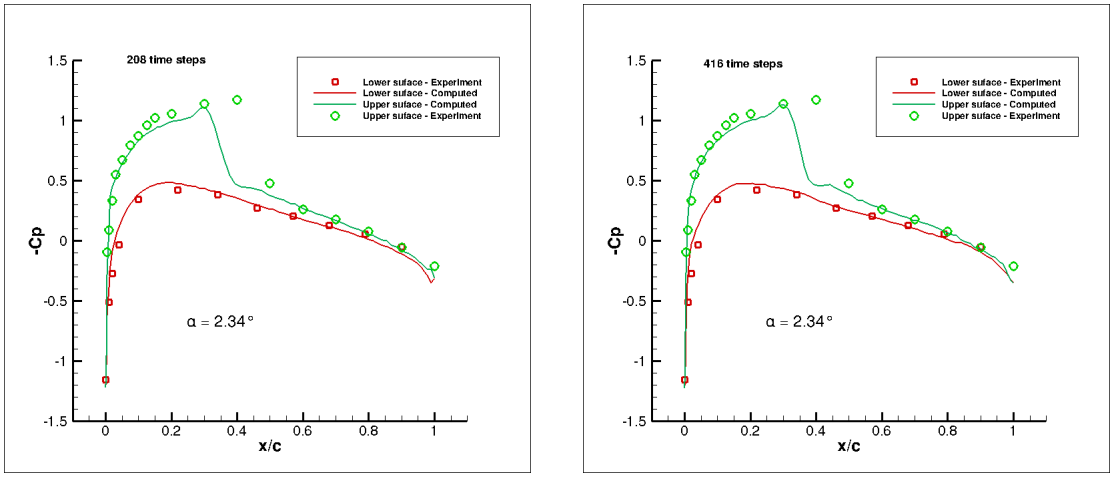

Figure 5.14 Pressure coefficient of case C for 208 and 416 steps per cycle

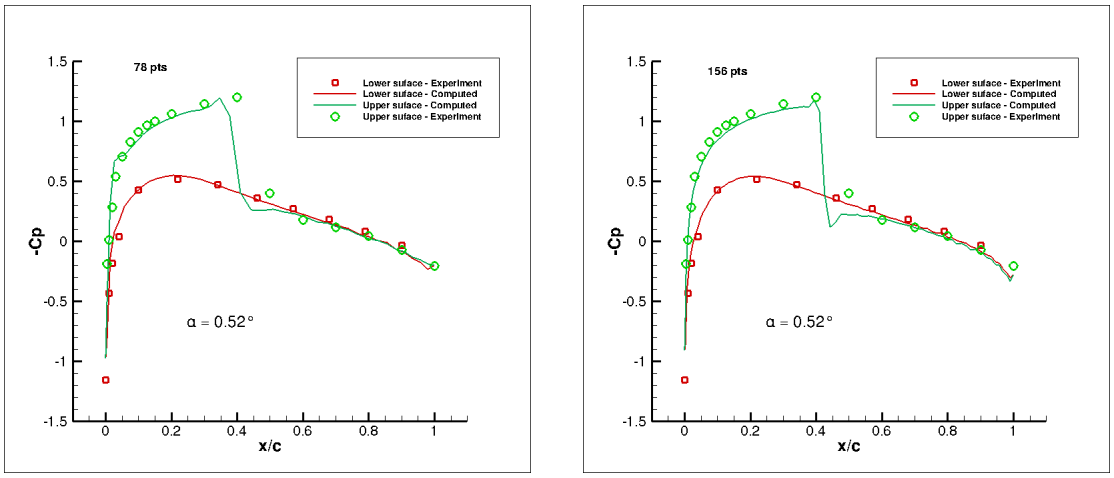

Figure 5.15 Pressure coefficient of case A for 78 and 156 grid points on airfoil

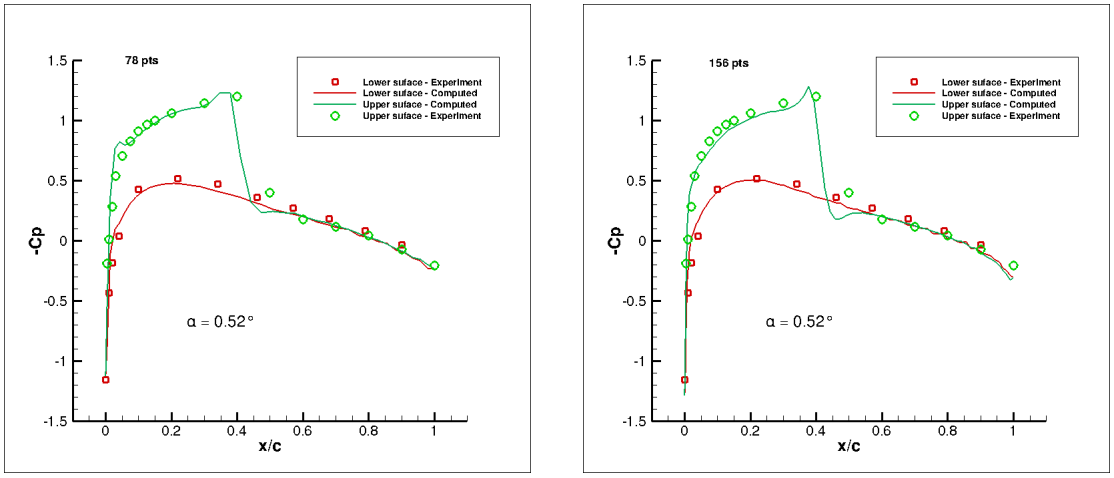

Figure 5.16 Pressure coefficient of case C for 78 and 156 grid points on airfoil

Above Figures 5.13 and 5.14 demonstrate that as the number of time steps per cycle increases, the more accurate results would be obtained. Figure 5.15 and Figure 5.16 demonstrate that as the number of grid points on airfoil increases, the more accurate results would be obtained.

Table 5.1 L2-norm of solutions on surface of airfoil compared to steady state

Angle of attack(degree)	case A	case B	case C
1.09	6.8474934e-02	7.3007368e-02	7.3557526e-02
2.34	7.5587570e-02	8.0232178e-02	8.0839881e-02
2.01	3.6110147e-02	3.4370655e-02	4.5296636e-02
0.52	5.0368728e-02	5.1800298e-02	5.3644513e-02
-1.25	6.8544827e-02	7.3476948e-02	7.6153300e-02
-2.41	6.0171991e-02	6.2529714e-02	6.8877501e-02
-2.0	4.0432858e-02	3.9328781e-02	5.0637232e-02
-0.54	6.1569833e-02	6.4990548e-02	6.7484987e-02

In order to show the accuracy of the three cases in unsteady state, solutions for them are used to compare with the same position as in steady state. Because interior nodes move as the airfoil moves, therefore, the difference between steady state and unsteady one is presented in L2-norm form of the q of grid points surface of airfoil. As shown in Table 5.1, solutions of all the three cases present slight difference compared to steady state. Case A is the best compared, case B is secondary, and case C shows only a little more difference, but they are all among the same level of magnitude regarding to the difference.

Table 5.2 Average computation cost of each time step of case A

		case A(sec)	percentage (%)	total (%)
mesh	move mesh	2.9999e-4	0	0
rupturing	smooth mesh	0	0	
flow solver		56.9960	100	100

Table 5.3 Average computation cost of each time step of case B

		case B(sec)	percentage (%)	total (%)
mesh	move mesh	0.1132	0.1281	46.28
rupturing	smooth mesh	47.5220	46.1519	
flow solver		47.4897	53.72	53.72

Table 5.4 Average computation cost of each time step of case C

		case C(sec)	percentage (%)	total (%)
mesh	move mesh	0.1112	0.1161	49.7528
rupturing	smooth mesh	47.5220	49.6367	
flow solver		48.1065	50.2472	50.2472

Since case A did not exploit mesh rupturing technique, flow solver part is predominant in computational cost. Table 5.3 and 5.4 show computational cost of mesh movement and flow calculation part of each time step for case B and case C. It is obvious that these two parts use almost half of the total amount of time for each time step respectively. Although moving mesh only needs a small amount of time, mesh smoothing part is predominant of computational cost in mesh rupturing technique. Case C used more time than case B in mesh rupturing part mainly because it had larger motion scale, while the flow solver part took almost same amount of time.

CHAPTER 6

CONCLUSIONS

As presented in this thesis, mesh rupturing is an approach which enables geometry insertion and motion, including large scale of motion. However the quality of the resulting mesh is dependent on the resolution of the uniform background mesh, so it should be have high enough resolution to capture the geometric features of body. When integrated into a time accurate CFD solver, the scale of motion is restricted since it changes boundary conditions at leading edge and trailing edge. Either at the leading edge where one edge might be split into two, or at the trailing edge where two edges might be merged into one, because of this deformation, special care has to be paid to adjust the flux across these edges as the result of changing of conserved variables. Therefore, in this thesis, the boundaries are restricted to move merely one edge length at one time step.

The biggest advantage of this mesh rupturing technique is that no interpolation or re-generation of grid is needed to implement mesh movement. So the computational cost in this part is highly reduced. Since the flow solver built in this thesis is inviscid one on unstructured mesh, so it is important to keep the triangles in mesh isotropic, which makes mesh smoothing methods important. Special care should be taken to keep high quality of the mesh, this will results in increasing cost in mesh adaption. Compared to traditional diverging elements from surface of airfoil to farfield, the mesh rupturing technique needs uniform background mesh. It is easier to be generated and because of the uniformity of the grid, the body could move arbitrarily which is a good aspect of this approach. But on the other side, this uniformity will increase the number of elements in mesh which will increase the computational cost to solve solution.

As shown in this thesis, in order to reduce the computational cost, if the scope of geometry motion is known, only that area should be filled with fine enough uniform grid, and the rest of the area could be filled with coarser grid. Once the geometry is inserted, it will move in surrounding fine grid area, which will ensure that all the features of geometry could be caught. And it is wasteful to do flow calculations on fine grid where little deformation happens.

From the results presented in Chapter 5, Case C shows the biggest difference in solution compared to steady state. It is obvious that because of the deformation of the surface of the airfoil, the number of nodes on the geometry is also changing as it moves. Because of changing distribution of grid points on airfoil, the line in moment coefficient plot is not smooth. Besides, although all the smoothing methods are able to optimize the interior nodes in the mesh, the quality of the grid surrounding the airfoil cannot be ensured to be high. For the future work, a layer of fine grid could be used to wrap the geometry that is going to be inserted into a coarser uniform background mesh, and the wrapping grid together with the geometry could be treated as a rigid body to facilitate the geometry motion. In this way, the quality around the geometry could be maintained, and same mesh rupturing technique can be applied to achieve geometry motion. From the computational cost presented in Chapter 5, mesh smoothing part occupies almost same amount of time compared to flow solver part. The cost in this part could be reduced by developing some new smoothing techniques.

The advantages of this new mesh technique make it easy to be combined with flow solver to implement body motion. The pitching airfoil in this thesis is just a simple case, it can also be used to simulate multiple moving geometries, for instance the store separation. All these cases discussed and presented here are two-dimensional unstructured mesh. If it is extended to three-dimensional, then the most challenging part maybe the split and merging of those tetrahedral elements, and the interpolation of new conservative variables for new elements after deformation of grid. In addition, the connectivity should be updated due to this grid

deformation. Much more work would be needed to achieve this since it is more complicated than simply duplicating or removing nodes in two-dimensional case.

REFERENCES

[1] A. L. Gationde, "A Dual-Time for Solution of the Unsteady Euler Equation" Aeronautical J., vol. 98, 1994, pp. 283-291 1

[2] Alauzet, F., "E cient Moving Mesh Technique Using Generalized Swapping," Proceedings of the 21st International Meshing Roundtable, San Jose, CA, 2012. 1

[3] A. Goswami, I. H. Parpia "Grid Restructuring for Moving Boundaries" AIAA Paper, No. 91-1589-CP, 1991. 2

[4] Batina, J. T., "Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-Aircraft Aerodynamic Analysis", *AIAA Journal*, vol. 29, No. 3, 1991, pp. 32733. 2

[5] Batina, J.T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes", *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 13811388. 2

[6] A. Jahangirian, and M. Hadidoolabi, "Unstructured Moving Grids for Implicit Calculation of Unsteady Compressible Viscous flow", International Journal for Numerical Methods in Fluids, 2005, 47, pp. 1107-1113. 2

[7] L. P Zhang., Z. J. Wang, "A Block LU-SGS Implicit Dual Time- Stepping Algorithm for Hybrid Dynamic Meshes", Computers & Fluids J., vol. 33, 2004, pp. 891-916. 2

[8] J. L. Steger, F. C Dougherty., and J. A. Benek, "A Chimera Grid Scheme" in Advances in Grid Generation, American Society of Mechanical Engineers, FED, New York, 1983, Vol. 5, pp. 59-69. 2

[9] Karimian, S.M.H., Salehi, F.S. and Alisadeghi H., "A Hybrid Overset Algorithm for Aerodynamic Problems with Moving Objects", World Academy of Science, Engineering and Technology, Vol. 3, 2009-10-24. 2

[10] O'Connell, M., and Karman, Jr., S.L., "Mesh Ruputuring: A Technique for Geometry Insertion and Significant Mesh Movement", AIAA-2013-0148, 51^{th} *AIAA Aerospace Sciences Meeting and Exhibit*, Grapevine, Texas, Jan. 07-10, 2013. 3, 20

[11] Anderson, W.K., Thomas, J.L. and Rumsey, C.L., "Extension and Application of Flux-Vector Splitting to Calculations on Dynamic Meshes", *AIAA Journal*, Vol. 27, No. 6, 1989, pp. 673-674. 25

[12] Biedron, R.T. and Thomas J.L., "Recent Enhancements to the FUN3D Flow Solver for Moving-Mesh Applications", AIAA Paper, 2009-1360. 26

[13] Thomas, P. D. and Lombard, C. K., "Geometric Conservation Law and Its Application to Flow Computations on Moving Grids", *AIAA Journal*, vol. 17, No. 10, 1979, pp. 1030-1037. 29

[14] Whitfield, D.L. and Janus, J.M., "Three-Dimensional Unsteady Euler Equations Solution Using Flux Vector Splitting", *AIAA* 17^{th} *Fluid Dynamics, Plasma Dynamics, and Lasers Conference*, Snowmass, Colorado, June 25-27, 1984. 29

[15] Bhardwaj, M. K., Kapania, R. K., Reichenbach, E., and Guruswamy, G. P., "Computational Fluid Dynamics/Computational Structural Dynamics Interaction Methodology for Aircraft Wings", *AIAA Journal*, Vol. 36, No. 12, 1998, pp. 2179 2186.

[16] Byun, C., and Guruswamy, G. P., "Aeroelastic Computation on Wing Body-Control Con gurations on Parallel Computers", Journal of Aircraft, Vol. 35, No. 2, 1998, pp. 288 294.

[17] Ji, S., and Liu, F., "Flutter Computation of Turbomachiner y Cascades Using a Parallel Unsteady Navier Stokes Code", *AIAA Journal*, Vol. 37, No. 3, 1999, pp. 320 327.

[18] Y. Zheng, R. W. Lewis, and D. T. Gethin, "Three-Dimensional Unstructured Mesh Generation. Part 1-3" Computer Methods in Applied Mechanics and Engineering J., vol. 134, 1966, , pp. 175-181.

[19] Sahasrabudhe, M., Karman, S. L., and Anderson, W., "Grid Control of Viscous Unstructured Meshes Using Optimization," 44^{th} Aerospace Sciences Meeting and Exhibit, January 2006, Reno, NV, AIAA-2006-0532 14

[20] Karman, S. L., "Adaptive Optimization-Based Smoothing for Tetrahedral Meshes," Accepted at AIAA SciTech 2015 conference, January 2015. 14

[21] Freitag, L. A., and Knupp, P. M. "Tetrahedral Element Shape Optimization via the Jacobian Determinant and Condition Number", *8th International Meshing Roundtable*. South Lake Tahoe, CA, 1999. 14

[22] Jameson, A., and Mavriplis, D.J., "Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh", *AIAA Journal*, Vol. 24, No. 4, 1986, pp. 611-618.

VITA

Ya'nan Gong was born in Nanchang, Jiangxi, China. She finished elementary school through high school in Nanchang. Wuhan University is where she completed undergraduate education with a bachelor's degree in Software Engineering in 2011. Yanan enrolled in the M.S. program at the University of Tennessee at Chattanooga SimCenter in fall of 2012.