---

**Article:**
Xiao, T., Qin, N., Luo, D. et al. (2016) Deformable Overset Grid for Multibody Unsteady Flow Simulation. AIAA Journal, 54 (8). pp. 2392-2406. ISSN: 0001-1452

https://doi.org/10.2514/1.J054861

---

# Deformable Overset Grid for Multi-Body Unsteady Flow Simulation

Tianhang Xiao[*]
*Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*

Ning Qin[†]
*The University of Sheffield, Sheffield, S1 3JD, UK*

Dongming Luo[‡]
*Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*

Shuanghou Deng[§]
*Delft University of Technology, Delft, the Netherlands, 2629HS, the Netherlands*

**A deformable overset grid method is proposed to simulate the unsteady aerodynamic problems with multiple flexible moving bodies. This method uses an unstructured overset grid coupled with local mesh deformation to achieve both robustness and efficiency. The overset grid hierarchically organizes the sub-grids into CLUSTERs and LAYERs, allowing for overlapping/embedding of different type meshes, in which the mesh quality and resolution can be independently controlled. At each time step, mesh deformation is locally applied to the sub-grids associated with deforming bodies by an improved Delaunay graph mapping method that uses a very coarse Delaunay mesh as the background graph. The graph is moved and deformed by the spring analogy method according to the specified motion and then the computational meshes are relocated by a simple one-to-one mapping. An efficient implicit hole-cutting and inter-grid boundary definition procedure is implemented fully automatically for both cell-centered and cell-vertex schemes based on the wall distance and an alternative digital tree (ADT) data search algorithm. This method is successfully applied to several complex multi-body unsteady aerodynamic simulations and the results demonstrate the robustness and efficiency of the proposed method for complex unsteady flow problems, particularly for those involve simultaneous large relative motion and self-deformation.**

---

[*] Associate Professor, College of Aeronautics and Astronautics, 29 Yudao Street; Visiting Scholar, Department of Mechanical Engineering, University of Sheffield.

[†] Professor of Aerodynamics, Department of Mechanical Engineering, Mappin Street, AIAA Associate Fellow.

[‡] Lecturer, College of Aeronautics and Astronautics, 29 Yudao Street.

[§] PhD researcher, Faculty of Aerospace Engineering, Kluyverweg 1. AIAA Student Member.

## Nomenclature

$c$ = speed of sound

$C_v$ = vertical force coefficient

$d$ = distance

$e$ = area/volume ratio coefficient

$E$ = internal energy per unit mass

$f$ = frequency

$\mathbf{F}$ = convective flux vector

$\mathbf{F}_v$ = viscous flux vector

$J$ = advance ratio

$k_{ij}$ = spring stiffness

$L$ = characteristic length/chord length

$Ma$ = Mach number

$\vec{n}$ = unit normal vector

$\mathbf{Q}$ = entropy variables

$Re$ = Reynolds number

$\mathbf{RES}$ = residual

$S$ = area or entropy

$St$ = Strohaul number

$t$ = physical time

$T$ = periodic time

$\mathbf{\Gamma}_Q$ = preconditioning matrix for entropy variables

$\mathbf{\Gamma}_W$ = preconditioning matrix for conservative variables

$U$ = free-stream velocity magnitude

$V$ = volume

$\vec{V}=[u,v,w]$ = velocity vector

$\mathbf{W}$ = conservative variables

$\boldsymbol{x}$ = Cartesian coordinate

$\rho$ = density

$\Omega$ = control volume

$\tau$ = pseudo-time

## I. Introduction

Unsteady flow simulation around multiple moving objectives poses numerical challenges for efficient and robust meshing strategies. Four methodologies are documented that can simulate the unsteady flow around bodies with locomotion, i.e., the re-meshing method, the immersed boundary method (IBM), the mesh deformation method and the overset grid.

Re-meshing during unsteady motion can be carried out to address the change of the solution domain. However there are two fundamental problems with such approach. Most importantly, re-meshing during time evolution suffers from an accuracy loss due to solution interpolation in the physical conservation laws, where the computational accuracy will be reduced because the new grid and the original one at the previous time step are not necessarily consistent. Secondly, re-meshing will require some significant additional computational effort for complicated geometries.

The IBM method [1] is extensively used in simulations involving moving bodies, particularly for low Reynolds number flows. It discretizes the governing equations entirely on a fixed Cartesian grid, which does not conform to the geometry of the boundaries. The presence of solid boundaries is represented by adding appropriate forcing to the flow equations. This method therefore avoids the complicated grid movements. Such a feature makes it attractive for simulating flows that involve moving bodies. However, imposing the wall boundary condition in IBM is not straightforward, and may negatively impact on the accuracy and conservation properties of the numerical scheme. In addition, it is noted that for high Reynolds numbers the computation becomes expensive in order to resolve the flow behavior in the boundary layer.

Different from the re-meshing method, mesh deformation methods deform the grid with respect to the specified motion where the grid connectivity is preserved. Generally, the mesh deformation techniques offer better computational efficiency and numerical accuracy compared with the re-meshing method. Mesh deformation can be mainly classified into two categories: physical analogy and interpolation method.

Physical analogies for mesh deformation, such as the spring analogy method first developed by Batina [2], use certain physics processes to propagate the perturbation of boundaries to the field mesh. The spring analogy method models the whole mesh as a network of linear springs, in which each grid edge is viewed as a spring with stiffness proportional to the reciprocal of the length, and the new position of mesh points are determined by solving a static equilibrium equation. The spring analogy method has been successfully applied to a wide range of unsteady and

optimization problems. However, the mesh quality will be difficult to be preserved by spring analogy when large displacement occurs since ill-conditioned or even negative cells may easily be created and thus abort the solution process. To overcome this problem, some efforts have been made for improving the robustness of spring analogy. Farhat *et al.* [3,4] introduced non-linear torsional springs to the spring analogy method to avoid the mesh crossing associated with the linear spring network and Murayama *et al.* [5] linked the grid edge stiffness with the angle between the faces to avoid the generation of squashed invalid elements. Elastic analogy [6,7], which can be viewed as an extension of the spring analogy, treats the grid as an elastic body following linear elasticity equations of solid mechanics. Theoretically, this method could be robust as it links the stiffness of a region to its volume and sets the boundary layer as a solid body. However, the performance of this method varies for different mesh types and magnitude of mesh deformation. All the spring analogy methods as mentioned above have to solve huge equations and become very expensive for large meshes.

An interpolation method for mesh deformation, by applying certain interpolation schemes, directly obtains the new position or the displacement of each mesh point so as to reflect geometric changes. Transfinite interpolation (TFI) [8] is an algebraic mesh generation method for generating structured meshes. It can also be used as a mesh deformation method for structured meshes if all the mesh generation parameters are kept the same as the geometry deforms. In principal, TFI interpolates the displacements of points on boundaries along mesh lines to the points in the interior domain. Combined with the multi-block structured grid, the ability of TFI can be enhanced to handle three-dimensional geometric perturbations. TFI has been widely used in aeroelasticity, aerodynamic optimization and multidisciplinary optimization to generate the dynamic structured grid. However, the efficiency and robustness of the TFI are limited to applications for structured meshes. There is also another type of interpolation methods developed without dependency on mesh topology and therefore they can be applied to different kind of meshes. This type of methods is typically represented by the Delaunay graph mapping method (DGM) proposed by Liu *et al.* [9] and the radial basis function (RBF) method proposed by de Boer [10] and further developed by Rendall and Allen [11,12,13]. The mesh deformation method based on Delaunay graph mapping [9] has been proved as a fast mesh deforming method due to the fact that it uses an explicit algebraic one to one mapping rather than solve a differential equation or a large linear system. The drawback of this method is that the mesh quality near the moving boundaries is difficult to be preserved when the moving bodies exhibit large rotation, which can lead to an invalid Delaunay graph. Alternatively, a mesh deformation method [10] using the radial basis functions interpolation [14,15] provides

a more robust moving mesh, which can handle larger mesh rotational deformation. However, the robustness is at the expense of computational cost with large mesh due to the fact that the interpolation of any mesh point is a global function involving position changes of all basis points on the surface. The size of the linear system to be solved is directly related to the number of the moving surface mesh points. To accelerate the computation, data reduction algorithms were proposed by Rendall and Allen [11, 12, 13], Sheng and Allen [16] and Wang *et al*. [17] to limit the RBF interpolation on a coarsened subset of surface mesh. To decrease the error on the surface points, they applied a greedy algorithm to select the optimum reduced set of surface mesh. Using only portion of the moving surface mesh points as basis points cannot fully recover the exact deformation, making it difficult to tackle aerodynamic problems which are sensitive to small-scale deformation. Most recently, Wang and Qin [18] developed a method combining the Delaunay graph mapping with local RBF, in which the Delaunay graph is used to group fluid mesh points, and the nodes of each graph element are treated as basis points of RBF for each group of fluid mesh points. This method can dramatically reduce the number of the basis points for RBF and hence results in higher computational efficiency and more robustness for mesh deformation. However, this method has to treat translation and rotation motion in separated ways, which may be difficult to distinguish in some applications.

It has to be mentioned that, though numerous efforts have been made within the methodology to improve the performance of the mesh deformation methods, these methods suffer from the problem with large deformation, in particular, with large relative motion of multiple bodies. Mesh quality is difficult to be preserved, or even, mesh could degenerate, when large displacements occur. A remedial measurement for the degenerated mesh quality is to locally or even fully regenerate the mesh. Zhang and Wang [19] used an unstructured grid to link the body-fitted grid and Cartesian grid and applied a local grid regeneration on this part when mesh deformation deteriorates. Zhang *et al*. [20] further extended this technique to three-dimensional applications where dynamic hybrid mesh deformation are applied in combination with local re-meshing. Still local grid regeneration is a costly part with penalty of complex algorithm design and accuracy loss at the same time.

In addition to the aforementioned grid regeneration and mesh deformation methodologies, the overset grid is a mature technology that has been used for decades to simplify the grid generation for complex geometries and as an embedding technique for simulations involving multiple bodies with relative movement. The overset grid method was firstly proposed by Steger [21] and subsequently extended by Nakahashi *et al*. [22] for its applications on unstructured grids. The overset grid can be applied to store separation, turbomachinery [23], rotary aircrafts [24,25]

and flapping wing aerodynamics [26,27]. For problems with boundary deformation, Fast and Henshaw [28] applied the overset grid in conjunction with a hyperbolic grid generator. In their work, a thin layer of body-fitted structured grid around the deforming boundary is overlapped on a fixed Cartesian grid covering the entire computational domain. At each time step, the body-fitted grid was regenerated for the deforming shape resulting from flow-structure interaction. By doing so, the global mesh regeneration switches to a local one, which improves the computational efficiency. However, local grid regeneration is still expensive, in particular, for complex three-dimensional large mesh systems.

There are many engineering applications, which exhibit simultaneous large relative displacement between bodies with self-deformation, such as flexible flapping wings, fish swarming, rotary wings coupled with structural dynamics, pose significantly challenge to the dynamic mesh techniques. Solely using mesh deformation methods or overset grid cannot satisfy the extreme scenarios mentioned above in terms of preserving the mesh quality and computational efficiency. To successfully simulate the unsteady flow field contains multiple bodies undergoing relative motion and deformation, a deformable overset grid by using unstructured overset grid technique coupled locally with an improved Delaunay graph mapping mesh deformation method is proposed in the present study. This paper is organized as follows: the numerical frame of an in-house developed unsteady Reynolds averaged Navier-Stokes (URANS) solver is introduced in Section 2; the dynamic mesh techniques, including the improved Delaunay graph mapping mesh deformation, unstructured overset grid method and deformable overset grid, are presented in Section 3; followed by several demonstration cases in Section 4 and the conclusions in Section 5.

## II. Numerical frame of the URANS solver

In this section, some key elements of an in-house unsteady Reynolds-averaged Navier-Stokes flow solver used in this study are briefly described.

### A. Governing equations in arbitrary Lagrangian Eulerian form

For the general problem of compressible flows in a moving and deformable computational domain $\Omega(t)$ with boundary $\partial\Omega(t)$, the integral form of unsteady compressible Navier-Stokes equations can be written as:

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \mathbf{W}\,\mathrm{d}V + \oint_{\partial\Omega(t)} \left(\mathbf{F}(\mathbf{W}) - \left(\dot{\boldsymbol{x}}\cdot\vec{n}\right)\mathbf{W}\right)\mathrm{d}S = \oint_{\partial\Omega(t)} \mathbf{F}_v\,\mathrm{d}S\,, \tag{1}$$

where $\mathbf{W}$ represents the vector of conservative variable (mass, momentum, and energy), $\mathbf{W}=\left[\rho \quad \rho u \quad \rho v \quad \rho w \quad \rho E\right]^{\mathrm{T}}$, $\mathbf{F}(\mathbf{W})$ represents the convective fluxes and $\mathbf{F}_v$ represents the viscous fluxes. $\dot{\boldsymbol{x}}$ and $\vec{n}$ are the velocity and the unit normal of the interface $\partial\Omega(t)$, respectively.

Defining $v_{\mathrm{gn}}=\dot{\boldsymbol{x}}\cdot\vec{n}$, the case $v_{\mathrm{gn}}=\vec{V}\cdot\vec{n}$ (where $\vec{V}=\left[u,v,w\right]$ is the vector of flow velocity) corresponds to a Lagrangian system, and $v_{\mathrm{gn}}=0$ is an Eulerian one. In the present formulation, $v_{\mathrm{gn}}$ is arbitrarily specified. The Spalart-Allmaras one-equation model and the Menter *k-ω* SST two-equation model are implemented in the developed code to close the governing equations for turbulent flows.

**B. Dual-time stepping with low Mach number preconditioning**

For the solution of unsteady flow, a dual-time stepping algorithm is employed in conjunction with low Mach number preconditioning intended for extending application of the solver to low speed flows. The governing equations with a preconditioned pseudo-time-derivative term introduced into Eq.(1) can be written as follows:

$$\boldsymbol{\Gamma}_W\frac{\partial}{\partial\tau}\int_{\Omega(t)}\mathbf{W}\,\mathrm{d}V+\frac{\partial}{\partial t}\int_{\Omega(t)}\mathbf{W}\,\mathrm{d}V+\oint_{\partial\Omega(t)}\left(\mathbf{F}(\mathbf{W})-v_{\mathrm{gn}}\mathbf{W}\right)\mathrm{d}S=\oint_{\partial\Omega(t)}\mathbf{F}_v\,\mathrm{d}S\ , \tag{2}$$

where $\tau$ and $t$ denote pseudo and physical time respectively, and $\boldsymbol{\Gamma}_W$ is the preconditioning matrix. This approach involves an inner iteration loop in each pseudo time step that is wrapped by an outer loop stepping through physical time, whereas convergence of the inner iterations in pseudo-time is optimized by preconditioning, local time stepping or other convergence enhancement techniques.

For the convenience of preconditioning analysis, primitive variables instead of the conservative variables are selected as the system variables. The present analysis is simplified by considering the entropy variables $\mathrm{d}\mathbf{Q}=\left[\mathrm{d}\,p/\rho c \quad \mathrm{d}\,u \quad \mathrm{d}\,v \quad \mathrm{d}\,w \quad \mathrm{d}\,S\right]$ where $\mathrm{d}\,S=\mathrm{d}\,p-c^2\,\mathrm{d}\,\rho$ is proportional to the change in entropy. In terms of conservative variables, the preconditioning matrix is $\boldsymbol{\Gamma}_W=\boldsymbol{\Gamma}_Q\dfrac{\partial\mathbf{Q}}{\partial\mathbf{W}}$ where $\boldsymbol{\Gamma}_Q$ represents the preconditioning matrix for the entropic variables designed as $\boldsymbol{\Gamma}_Q=\dfrac{\partial\mathbf{W}}{\partial\mathbf{Q}}\mathrm{diag}\left(\beta^2 \quad 1 \quad 1 \quad 1 \quad 1\right)$. In this paper, $\beta$ is designed for the purpose of improving robustness for unsteady flow with moving boundaries as follows,

$$\beta^2=\min\left(\max\left(Ma_{\mathrm{Local}}^2,\beta_{\mathrm{min}}^2\right),\ 1.0\right),$$

where $Ma_{\mathrm{Local}}$ is the local maximum relative Mach number defined as,

$$Ma_{\text{Local}} = \max\left(\frac{\left|\vec{V}-\vec{V}_g\right|_{\text{neighbors}}}{c}, \frac{\left|\vec{V}-\vec{V}_g\right|_{cell}}{c}\right),$$

and $\beta^2_{\min} = 3\max(Ma^2_\infty, Ma^2_{g,\text{mean}})$, $\vec{V}_g$ represents grid velocity vector, $Ma_\infty$ the incoming free-stream Mach number and $Ma_{g,\text{mean}}$ the mean Mach number of moving boundaries.

### C. Finite volume discretization: a unified approach for either cell-centered or cell-vertex scheme

In the code, a unified approach is used for either cell-centered or cell-vertex discretization applying on arbitrary type meshes (structured, unstructured, Cartesian or hybrid of them). A face-based data structure is employed which creates pointers from cell interfaces to adjacent control volumes as the only connectivity information. Cell-centered discretization using the primary mesh or cell-vertex discretization using the median dual mesh can be selected at run-time, the only difference being the preparation of the metric data. Eq. (2) can be discretized in a polygonal control volume $V_i$ as:

$$\boldsymbol{\Gamma}_{Wi}\frac{\partial(\mathbf{W}V)_i}{\partial\tau} + \frac{\partial(\mathbf{W}V)_i}{\partial t} = -\mathbf{RES}_i(\mathbf{W}) \tag{3}$$

$$\mathbf{RES}_i(\mathbf{W}) = \sum_{j=1}^{nface}\tilde{\mathbf{F}}(\mathbf{W})_{ij}S_{ij} - \sum_{j=1}^{nface}\mathbf{F}_{vij}S_{ij}$$

.

The inviscid flux, through the interface $S_{ij}$ between the control volume $V_i$ and $V_j$, is calculated using a reformulated Roe-type flux difference splitting scheme. The left and right state variables at both sides of a control volume face are reconstructed by a weighted least square or Green-Guass linear gradient reconstruction approach with Venkatakrishnan's limiter [29] applied to prevent oscillations near shock waves. For viscous fluxes computation, the velocity and temperature gradients at the interface are obtained by averaging the values of its adjacent control volumes with an additional correction in the direction from volume centroid *i* to volume centroid *j* to avoid odd-even decoupling.

When dynamic meshes are used, the grid velocities $\dot{\boldsymbol{x}}$ and the surface unit normal $\vec{n}$ need to be considered carefully so that the errors introduced by the mesh deformation do not degrade the accuracy of the flow computation. The discrete geometric conservative law [30, 31, 32] provides a guideline on how to evaluate these parameters.

### D. Temporal discretization and implicit iteration

The pseudo-time term in Eq.(2) is discretized with a first order backward difference and the physical time term is discretized in an implicit fashion by means of $k$-step backward difference respectively. The linearized equations system is finally given as,

$$\left(\mathbf{\Gamma}_{Wi}\frac{V_i^{n+1}}{\Delta\tau}+\frac{\phi_{n+1}V_i^{n+1}}{\Delta t}+\frac{\partial\mathbf{RES}}{\partial\mathbf{W}}\right)\Delta\mathbf{W}=-\mathbf{RES}_i(\mathbf{W}^m)-\frac{\phi_{n+1}\mathbf{W}^mV^{n+1}}{\Delta t}-\frac{1}{\Delta t}\sum_{h=0}^{k-1}\phi_{n-h}\left(\mathbf{W}V\right)^{n-h}, \tag{6}$$

where $m$ and $n$ denote pseudo-time and physical time steps, respectively. The choice of the sequence $\{\phi_n\}$ [33] governs the accuracy of the temporal discretization from the steady flow solver mode to unsteady schemes up to 3rd order time accuracy. The linear system of equations for the increments to the dependent variables, given by Eq.(6) is solved by an iterative Lower-Upper Symmetric Gauss-Seidel or Krylov subspace type Generalized Minimal Residual (GMRES) algorithm.

In this study, all the unsteady flow simulations were performed by using cell-vertex scheme for spatial discretization with Green-Gauss gradient reconstruction. Second-order temporal accuracy was achieved by implicit GMRES with the residual of each time-step reduced by at least two orders.

## III. Deformable Overset Grid

In this section, an improved Delaunay graph mapping strategy is first developed to achieve a robust and efficient mesh deformation. Secondly, the overset grid method and its combination with the local mesh deformation are presented.

### A. Improved Delaunay graph mapping for mesh deformation

The mesh deformation method base on Delaunay graph mapping [9] has been proved to be an efficient mesh deformation method since it uses an explicit algebraic one to one mapping. The original Delaunay background graph is generated by all the surface mesh points and a few boundary points at the outer boundary. A notable disadvantage of this choice is that the initial Delaunay graph will easily get invalid when large rotational deformation occurs. To solve the aforementioned problem, a finer Delaunay graph, yet a very coarse mesh is proposed to work as the background graph here. The spring analogy method is then used to deform the Delaunay graph according to specified movement. The original computational mesh is mapped into their new position using the algebraic one to one mapping [9]. By doing so, the robustness of the original Delaunay graph mapping method is improved without substantially increasing the computational cost, as the spring analogy method is only applied on a very coarse

background Delaunay graph. The algorithm of the proposed graph mapping is illustrated using a periodic pitching NACA0012 airfoil (seen in **Fig.1**) exhibits a sinusoidal kinematics: $a(t) = 90^{\circ}\sin(2\pi ft)$ as follow:

(1) **Generating background Delaunay graph in the computational domain.**

A good quality computational mesh is generated in the computational domain (**Fig. 1**(a)). In the same domain, a very coarse background Delaunay graph (**Fig. 1**(b)) is generated, and the background graph will be moved by spring analogy method on demand. For the Delaunay graph, the grid distribution on the moving boundaries (walls) should be identical to the computational mesh to preserve full integrity of boundary movement, while the entire domain is meshed with a large growth rate, yet resulting in much less cell amount when compared with the computational mesh. This can be helpful to propagate the deformation to the very far field while the mesh quality in the area near wall boundaries can be preserved.

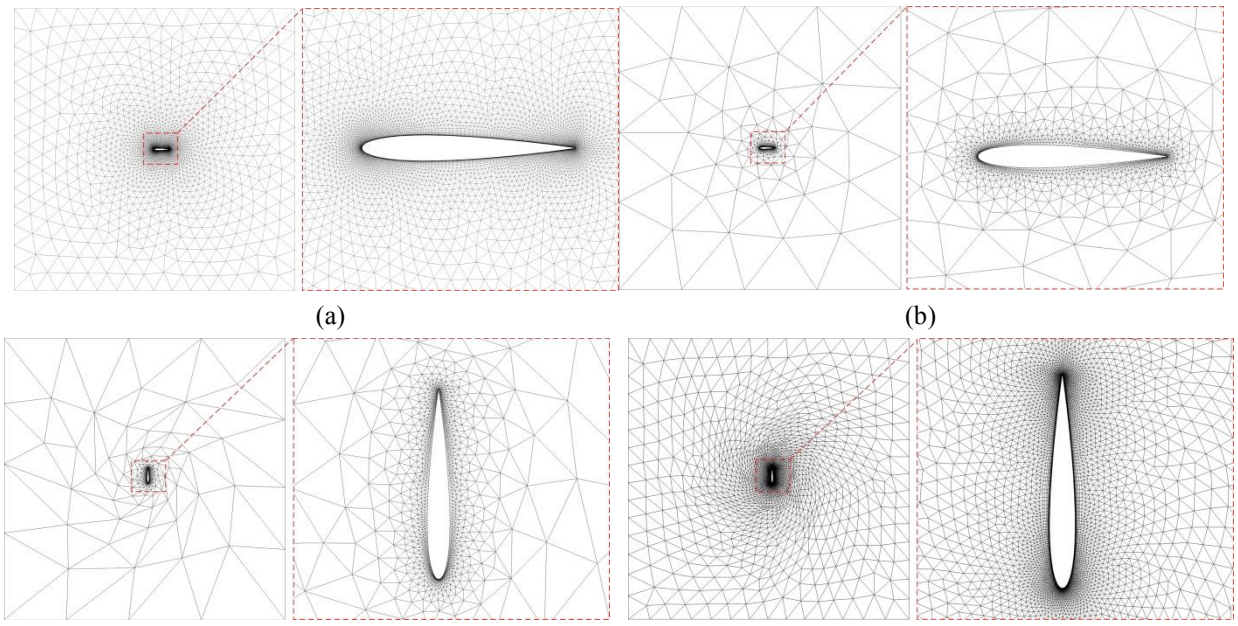



**Fig. 1 Improved Delaunay graph mapping method: (a) initial computational mesh; (b) coarse background Delaunay graph mesh; (c) deformation of the background graph; (d) deformation of the computational mesh after the one to one Delaunay graph mapping.**

(2) **Defining one-to-one mapping between the computational mesh and the Delaunay graph.**

For each computational mesh point *P*, by using the efficient searching Alternative Digital Tree (ADT) algorithm [34], the Delaunay graph element *E* is found where the mesh point *P* locates. As illustrated in **Fig. 2** for 2D case but without losing its generality, the corresponding area (2D) or volume (3D) ratios based on Eq. (7) [9] are then

calculated to obtain the uniquely defined one-to-one mapping between the mesh point and the graph element for two-dimensional or three-dimensional cases. The area or volume coefficients $e_i$, which uniquely determine the relative position of point $P$ in the graph element $E$, are defined as,

$$\begin{aligned} e_i &= S_i/S, \quad i=1,2,3 \quad \text{(2D)} \\ e_i &= V_i/V, \quad i=1,2,3,4 \quad \text{(3D)} \end{aligned}, \tag{7}$$

where $S_i$ denotes the area of the triangle formed by point $P$ and the edge of graph element $E$ for 2D case while $V_i$ is the volume of the tetrahedral formed by the point $P$ and the face of element $E$. $S$ or $V$ is total area or total volume of the graph element.

The calculated area/volume coefficients of each mesh point are stored along with the graph element number as a primary quantity before deforming the mesh and remain the same during the mesh deformation.

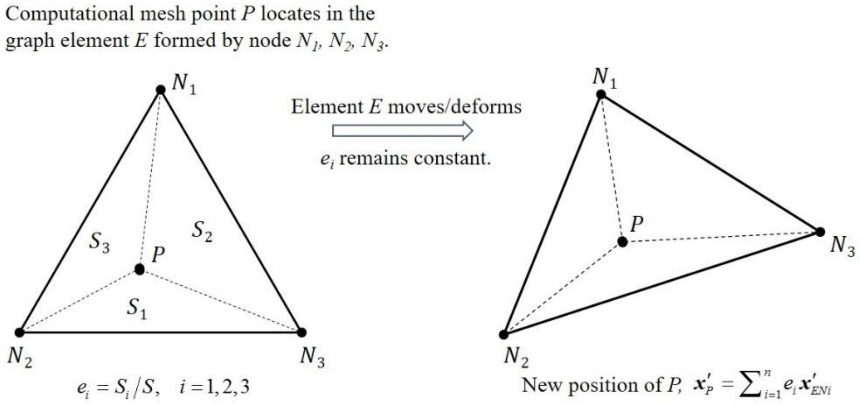



**Fig. 2 Illustration of Delaunay graph mapping method.**

(3) **Moving or deforming the Delaunay graph.**

Treated as a network of springs, the background Delaunay graph is moved by means of the spring analogy model with passive deformation accordingly, as shown in **Fig. 1**(c). Although more sophisticated methods including torsional springs are available, the commonly used spring analogy model of Batina [2] is employed in this procedure by considering the fact that the background graph is very coarse with large disparity of mesh density near the wall and far away from the wall and, therefore, this method is sufficient for deforming the Delaunay graph in a robust manner even for large displacements. With graph element edges modelled as springs, the static equilibrium equation for node $i$ can be written as,

$$\sum_{j=1}^{NE_i} k_{ij}\left(\delta \boldsymbol{x}_i - \delta \boldsymbol{x}_j\right) = 0$$

where $NE_i$ is the number of nodes directly connected to node $i$ by the element edges, $\delta \boldsymbol{x}$ represents node displacement vector and $k_{ij}$ is the linear spring stiffness for a given edge $ij$. In the present study, $k_{ij}$ is calculated as $k_{ij} = 1 \Big/ \left| \boldsymbol{x}_i - \boldsymbol{x}_j \right|^2$.

By applying the static equilibrium equation to all nodes in the Delaunay graph, a system of equations can be derived. Again, the efficient iterative methods used in the flow solver can be used here. In the present study, the derived linear system is iteratively solved by a preconditioned conjugated gradient algorithm with tolerance of $10^{-10}$.

(4) **Relocating the computational mesh points by the pre-calculated one-to-one mapping coefficients.**

After the background Delaunay graph is deformed, as illustrated in **Fig. 2**, the computational mesh points can be mapped into new position by the one to one mapping scheme. The new position $\boldsymbol{x}'_P$ of computational mesh point $P$ is calculated based on the pre-calculated area/volume ratio coefficients as,

$$\boldsymbol{x}'_P = \sum_{i=1}^{n} e_i \boldsymbol{x}'_{ENi} ,$$

where $\boldsymbol{x}'_{ENi}$ is the new position of the $i^{th}$ node of the moved graph element, and $n$=3 for two dimensional case, $n$=4 for three dimensional case. The deformed computational mesh around the NACA0012 airfoil can be seen in **Fig. 1** (d).

Comparing to the original Delaunay graph mapping method [9], using a coarse background mesh modeled by a spring analogy model as the mapping graph is principally able to improve the robustness in terms of preserving the mesh quality. The new Delaunay graph contains a small number of interior points, rather than the original one which only consists of surface mesh points and some outer boundary points. It can propagate the wall displacement, both translation and rotation, to the far field by spring analogy effectively and, therefore, can survive from graph element intersection for larger displacement. In the meantime, the very coarse Delaunay graph with large disparity of element size is beneficial to maintain the quality of the graph elements near wall boundaries. These small size elements are hardly deformed due to their strong stiffness while the large size elements in the far field absorb the most of the perturbation. As a result, the computational mesh quality is well maintained in the area near wall boundaries where the mesh quality is particularly important especially for viscous flow simulation. In addition, as the spring analogy is only applied on a very coarse graph, rather a dense computational mesh, the graph updating procedure does not dramatically scarify the computational time.

To assess the aforementioned capability of the improved method, evaluations were performed on two typical

test cases, a two-dimensional NACA0012 airfoil rotating around its ¼ chord axis, and a three-dimensional DLR-F6 model deforming following a prescribed blending motion. These two test cases were performed on a computer with Intel i7-4500@2.4G Hz CPU and 16GB RAM. For the sake of comparison, the same tests were run by both the original Delaunay graph mapping (hereafter referred to as DGM) and the improved Delaunay graph mapping method (hereafter referred to as Improved-DGM).

***Case1*** **Rotating NACA0012 airfoil**

In the first case, the two-dimensional NACA0012 airfoil rotates around its ¼ chord position with a constant angular speed at 1 degree/step within a fixed box boundary. In order to test the range of rotation the method can tolerate, the rotation continues until element crossing occurs in the Delaunay graph which is regarded as the rotating limit for valid mesh deformation. To address the effect of mesh density on the efficiency and deforming capability, four sets of meshes with different resolutions are compared in **Table 1**.

**Table 1 Mesh size of the NACA0012 airfoil**

| | Computational mesh | | | Background Delaunay graph | | |
|---|---|---|---|---|---|---|
| | Nodes on wall | Total nodes | Total cells | Nodes on wall | Total nodes | Total cells |
| Coarse | 200 | 7510 | 11443 | 200 | 1236 | 2249 |
| Medium | 400 | 16288 | 22042 | 400 | 2348 | 4261 |
| Fine | 800 | 35282 | 43281 | 800 | 4367 | 7893 |
| Extra fine | 1600 | 92976 | 104855 | 1600 | 6414 | 11583 |

**Fig. 3** shows the average and worst mesh quality of the computational mesh by both DGM and Improved-DGM. The grid cell shape/skew [35] is regarded as a criterion here to represent the grid quality ranges from 0 to 1 where 1 means an equilateral cell and 0 indicates the mesh is degenerated. As seen for all the mesh density, the mesh quality of DGM, either average or worst value, degrades rapidly as rotation angle increases while the Improved-DGM successfully maintains the average quality at a relative high level throughout with only a very small degradation even at very large rotational displacement. Considering the worst grid quality, a plateaus region is found at small rotating angles for the Improved-DGM, which indicates that the mesh quality can be well maintained with small rotating magnitude.

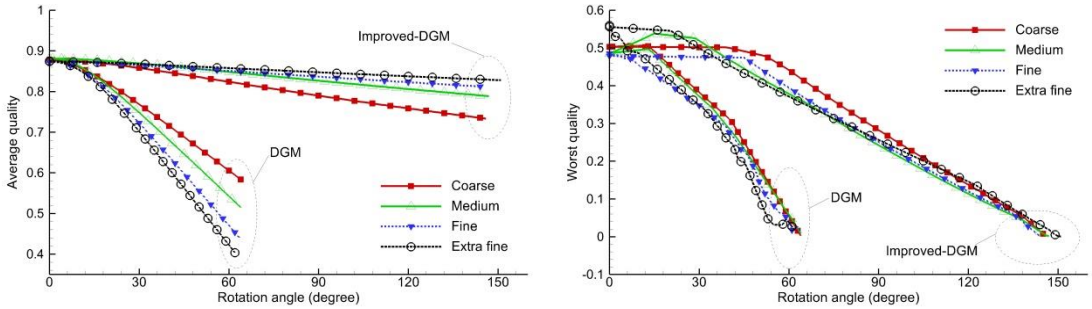



**Fig. 3 Mesh quality vs. rotation angle (Left: average mesh quality; Right: worst mesh quality)**

The rotation angle limit and average CPU time per step for both DGM and Improved-DGM is summarized in **Table 2**. As illustrated, the Delaunay graph are invalid at around 62-64 degree when using DGM, whereas the mesh system is admitted as valid with Improved-DGM even the airfoil rotates up to around 150 degree. Regarding the efficiency, the CPU time of Improved-DGM is about one order higher as compared to DGM, a price for robustness. Nevertheless, the Improved-DGM method is a few orders more efficient than the original spring analogy method for deforming the mesh. In addition, the extra computational cost decreases relatively with the increase of mesh size, as can be seen from the time ratio listed in **Table 2**.

**Table 2 Comparison of rotation angle limit and average CPU time per step**

| | Rotation angle limit | | Average CPU time per step | | |
|---|---|---|---|---|---|
| | DGM | Improved-DGM | DGM | Improved-DGM | ratio |
| Coarse | 64deg | 146deg | 6.23ms | 101.12ms | 16.23 |
| Medium | 64deg | 147deg | 12.66ms | 213.46ms | 16.86 |
| Fine | 64deg | 144deg | 27.02ms | 379.40ms | 14.04 |
| Extra fine | 62deg | 151deg | 69.60ms | 590.65ms | 8.49 |

To expand this point, **Table 3** shows the breakdown of the CPU time for the Improved-DGM. The CPU time of Improved-DGM can be principally divided into three parts: (1) moving boundary nodes based on the specified motion; (2) updating Delaunay graph; (3) relocating the computational mesh by the algebraic one-to-one mapping. As shown in **Table 3**, the ratio of the updating Delaunay graph procedure decreases with the increasing grid size.

**Table 3 Breakdown of the CPU time for Improved-DGM**

| | Moving Boundary nodes | updating Delaunay graph | relocating computational mesh nodes |
|---|---|---|---|
| Coarse | 0.12% | 93.84% | 6.04% |
| Medium | 0.10% | 94.07% | 5.83% |
| Fine | 0.11% | 92.88% | 7.01% |
| Extra fine | 0.13% | 88.22% | 11.65% |

**Fig. 4** plots the mesh quality contour of the computational mesh by both DGM and Improved-DGM. The DGM method cannot guarantee a good mesh quality near the moving boundary as shown in **Fig. 4**(a), while the Improved-DGM method is able to propagate the rotational displacement to the very far-field and hence the mesh quality can be well preserved (seen in **Fig. 4**(b)). The robustness of the Improved-DGM method is further evidenced by keeping rotating the airfoil to 120 degree situation as illustrated in **Fig. 4**(c) that the viscous mesh (boundary layer) demonstrates a good mesh quality distribution.

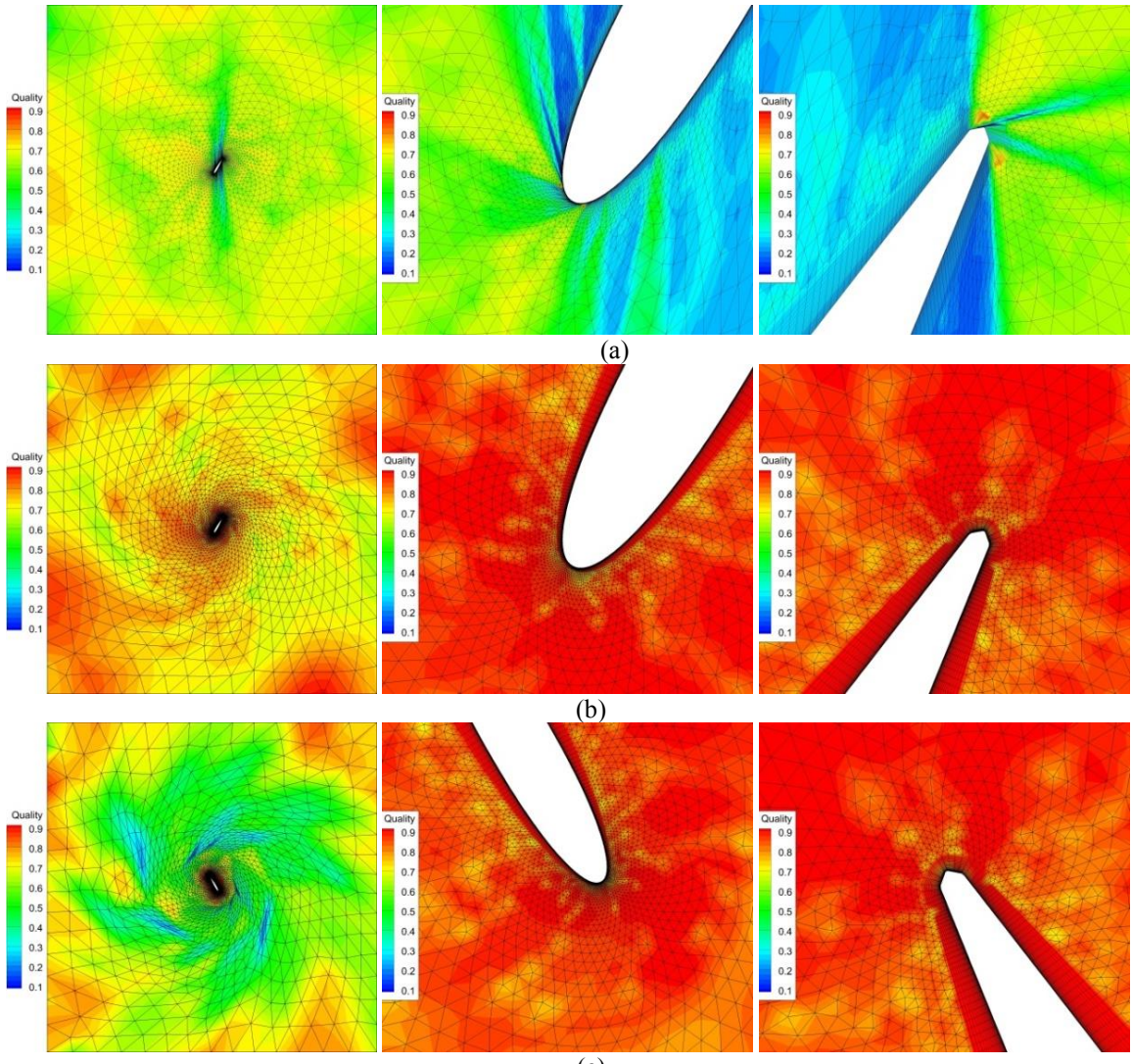



**Fig. 4 Mesh quality contours of the medium density computational mesh around the NACA 0012 airfoil: (a) DGM, 60 degree rotation angle; (b) Improved-DGM, 60 degree rotation angle; (c) Improved-DGM, 120 degree rotation angle.**

***Case2* Deforming DLR F6 wing model**

To further examine the capability of the improved mesh deformation strategy, a three-dimensional DLR F6 model with large bending deformation is used here as an extreme test case. **Fig. 5**(a, b) demonstrate the computational mesh around the model and its corresponding background Delaunay graph. The computational mesh consists of about 6.46 million nodes and 19.07 million cells while the graph contains 257869 nodes and about 1 million graph elements and the wall surface meshes are the same for the computational mesh and the graph.

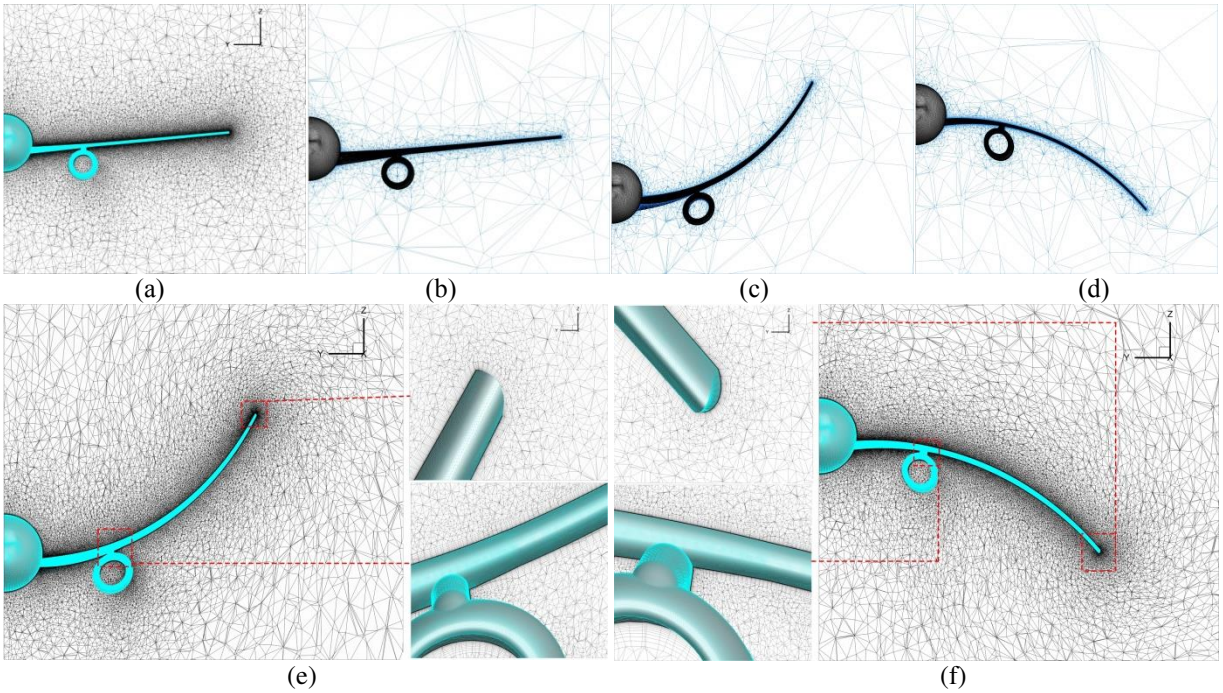



**Fig. 5 The application of improved Delaunay graph mapping method on a wing/body/pylon/nacelle model: (a) initial computational mesh; (b) initial Delaunay graph; (c, d) the deformed Delaunay graph; (e, f) the deformed computational mesh and zoom in view.**

**Table 4** presents the average mesh quality and worst quality of the computational mesh around the DLR F6 model when the wing of the model performs bending deformation with large vertical displacement of $L$, $2L$, $-L$ and $-2L$ at the wingtip. Compared to the initial mesh, the mesh quality does not deteriorate much even for such a large displacement. The breakdown of the CPU time of Improved-DGM for DLR F6 model is shown in **Table 5**. The CPU time for deforming the Delaunay graph decreased down to 83.67% of the full process which is about 5.3 times higher the one-to-one mapping, indicating that the robustness and good mesh quality of mesh deformation can be achieved by the Improved-DGM method at a moderate extra cost for large scale meshes. **Fig. 5**(e, f) presents the

deformed the computational mesh and the zoom-in views of the areas near the wingtip and the pylon, demonstrating that the mesh quality was well preserved.

**Table 4 mesh quality summarization for the computational mesh of DLR F6 model**

| | Initial mesh | $+L$ | $+2L$ | $-L$ | $-2L$ |
|---|---|---|---|---|---|
| Average quality | 0.881643 | 0.880797 | 0.878079 | 0.880720 | 0.878048 |
| Worst quality | 0.131632 | 0.110985 | 0.107877 | 0.110252 | 0.098935 |

**Table 5 Breakdown of the CPU time of mesh deformation for DLR F6 model**

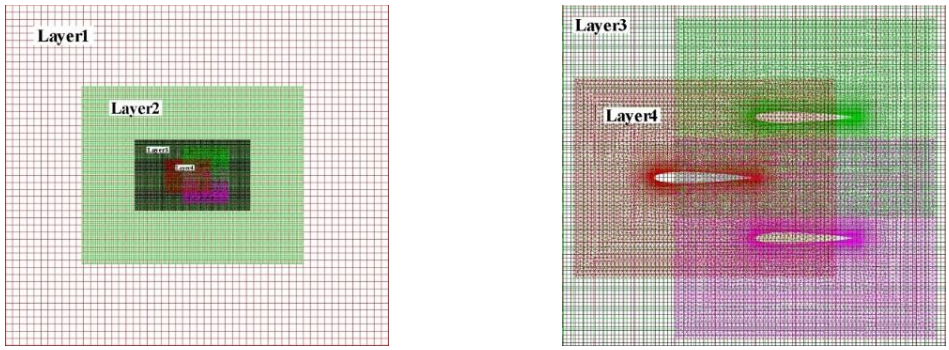
| | Moving Boundary nodes | updating Delaunay graph | relocating computational mesh nodes | Total |
|---|---|---|---|---|
| CPU Time (ms) | 291.62 | 45785.68 | 8647.35 | 54724.65 |
| Percentage | 0.53% | 83.67% | 15.80% | 100% |

**B. Hierarchically organized unstructured overset grid technique**

The developed solver uses an unstructured overset grid to simulate the flow involves complex geometries and multiple moving boundaries with large relative movement.

**(1) Generating and hierarchically organizing meshes**

The overset grid technique generates separate grids for each individual component, and one or more Cartesian/hybrid unstructured off-body grids as background grid if necessary. The generated grids are hierarchically organized into two levels as CLUSTER and LAYER according to [36]. A CLUSTER is usually a grid that covers a certain region of the flow field. It can be a body-fitting grid around geometrical component, such as wing, fuselage and tail, and also can be a background grid. A LAYER is a grid organizing level which consists of one CLUSTER or a number of overlapping CLUSTERs. One CLUSTER can overlap with other CLUSTERs in the same LAYER. One LAYER or several LAYERs are used to obtain an appropriate overall grid system that offers high densities in the near field of bodies but becomes coarser gradually towards the far field. Note that each LAYER can be only embedded into its immediate lower LAYER. **Fig. 6** illustrates the hierarchical overset grid system for a group of airfoils with relative movement. Layer1, Layer2 and Layer3 each contains one CLUSTER of Cartesian grid offering a smooth transition from the high resolution grid near the airfoils to the coarser grid in far field and Layer4 consists of three CLUSTERs of body-fitted unstructured meshes with each for individual airfoil.




(a) (b)

**Fig. 6 Hierarchical overset grid system for a group of moving airfoils and its zoom in view.**

**(2) Implicit hole-cutting and inter-grid boundary definition**

For flow field involves moving boundaries, a fully implicit automatic Chimera hole cutting and identification of inter-grid boundary procedure is implemented to exclude the mesh points or cells which do not participate in the computation. The grids in a higher LAYER are usually finer than the grids in a lower LAYER, or in the same LAYER, the grid elements closer to the body are finer than the far ones for resolving the flow structure in the near field region. Hence, the grid elements closer to the body are expected to be retained for the computation. For this reason, the wall-distance [22] is used as an indicator to decide the inter-grid boundary and thus provide the appropriate resolution near surfaces.

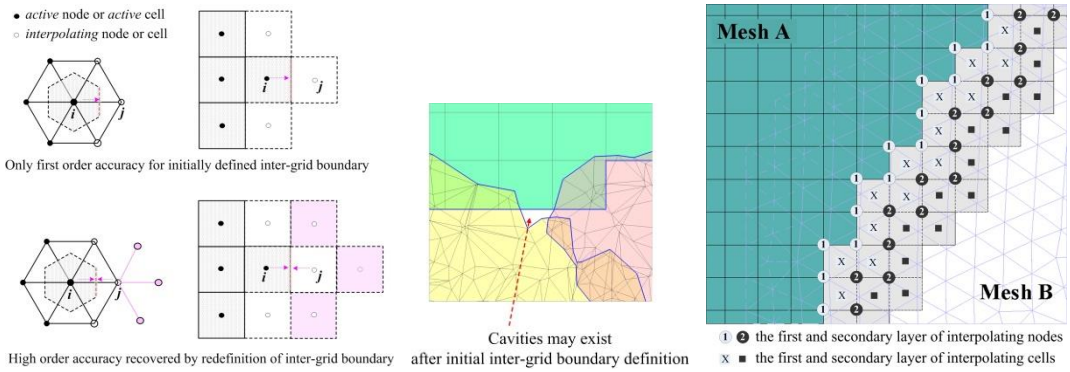
The minimum self-wall distance from each node to the body surfaces in the same CLUSTER is first measured. For the body-fitted CLUSTER, the real wall distances of the nodes are computed to the self-wall boundaries. For the body-off CLUSTERs served as background grid, the wall distances of each node are determined according to the LAYER's level as $\Delta d$, $2\Delta d$, $3\Delta d$……$n\Delta d$, where $n$ is the LAYER's level number and $\Delta d$ is decided beforehand by user. Second, search the donor cell for each node lying in the overlapping regions, and then compare the wall distances between the node and its donor cell. If the wall distance of the node is smaller, the node is defined as an *active* node (which can also be called computational node); otherwise it is defined as a *non-active* node or non-computational node. Then, by the nodal activity, all cells are classified into three groups: *active* cell with all nodes active, *non-active* cell with all nodes non-active and *inter-grid boundary* cell that has both active and non-active nodes. The inter-grid boundary cells are responsible to transfer the flow properties between different CLUSTERs.

During the procedure of hole-cutting, donor cell searching is the most time-consuming part. To accelerate the data searching, the ADT technique [34] is again employed which organizes the grid elements of each CLUSTER into a binary data structure according their spatial position along alternative dimensions. In the present study, grid

elements in each CLUSTER are hierarchically organized in several ADTs and each ADT only contains the elements overlapped with the other CLUSTERs. By doing so, the donor cell searching between two overset CLUSTERs is localized to the partial overlapped region and therefore the efficiency is further improved.

**(3) Redefinition of inter-grid boundary**

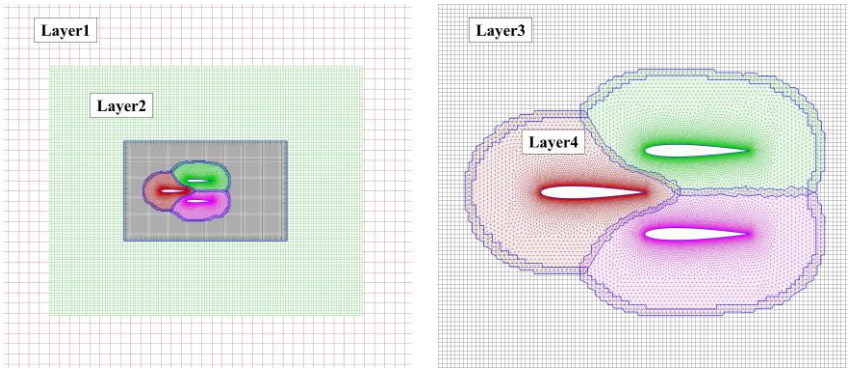
The inter-grid boundary cells are identified among the sub-grids for inter-grid communication by the above hole-cutting and inter-grid definition procedure. However, the band of these overlapping layers is not spatially sufficient for a higher order flux computation. As shown in **Fig. 7**(a), for the interface ***ij*** between control volume ***i*** and control volume ***j***, where ***i*** is active node/cell and ***j*** is interpolating node/cell determined by the initial inter-grid boundary defining step, only first-order accuracy of flux computation can be obtained as the flow gradient cannot be reconstructed due to the fact that some neighbors of control volume ***j*** are non-activated and excluded from flow computation. Therefore, a redefinition of inter-grid boundary, by which the non-active neighbors of volume ***j*** is activated as new interpolating nodes/cells, is needed to recover the high order accuracy of flux computation for volume ***i*** as demonstrated in **Fig. 7**(a). This procedure is also called for by another reason that cavities may exist after the initial definition of inter-grid boundary as shown in **Fig. 7**(b). An optimized redefining algorithm, that adds one or a few more layers of nodes/cells at each inter-grid boundary by advancing the inter-grid boundary to its non-active region, is implemented for each grid CLUSTER to recover the accuracy as illustrated in **Fig. 7**(c). Two layers of interpolating nodes/cells are enough for the present secondary-order accuracy, but higher order scheme may need more layers. The algorithm is summarized in a pseudo program presented in **Table 6** which can be applied to both cell-centered and cell-vertex scheme and higher order accuracy of spatial discretization.




(a) (b) (c)

**Fig. 7 Illustration of inter-grid boundary redefinition: (a) high order accuracy of flux computation can be recovered by redefinition of inter-grid boundary; (b) cavities may exist after initial inter-grid boundary definition; (c) inter-grid boundary is extended for Mesh A (from mesh A to mesh B).**

In the case of multiple sub-grids, there may be more than one candidate of donor cells for interpolation in the overlapping regions. In the current method, the active one with smallest/smaller cell volume is chosen as the optimum donor cell. The resulting overset mesh system of the airfoil group is illustrated in **Fig. 8**. The information transfer from a donor cell to a receiver cell/node is completed by an interpolation algorithm. $2^{nd}$ order accurate inverse distance based interpolation method is used here.




(a) (b)

**Fig. 8 The resulting overset grid system for a group of airfoils.**

**Table 6 Pseudo program for inter-grid boundary redefinition**

```
!After the initial inter-grid definition,
! all cells and nodes are classified as
!    cell-flag = 0 – non-active
!                1 – active
!                2 – BNDARY (interpolating)
! Node-flag = 0 – non-active
!             1 – active
do m = 0  number of extend layers
    for all cells with cell-flag = BNDARY + m do
        cell-flag = BNDARY + 1
        set non-active nodes of this cell as BNDARY + m
        add this cell to FrontCell-list
    end
    while FrontCell-list is not empty
        for all cells in the FrontCell-list do
            find its non-active face-adjacent neighbor cells
            add these neighbor cells to TempFrontCell-list
        end
        empty FrontCell-list
        for all cells in the TempFrontCell-list do
            if it contains nodes with node-flag = BNDARY + m
                set cell-flag = BNDARY + 1 + m
                add this cell to FrontCell-list
            end if
        end
        empty TempFrontCell-list
    end
end do
set all cells with cell-flag ≧ BNDARY as interpolating cell
set all nodes with node-flag ≧ BNDARY as interpolating node
```

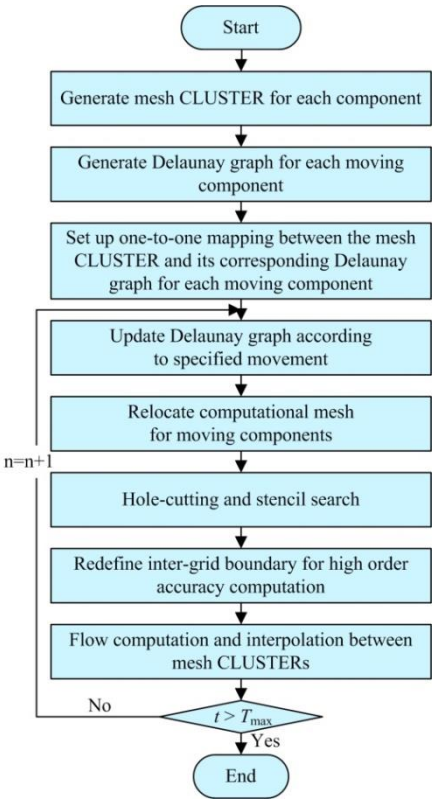



**Fig. 9 Flowchart of the deformable overset grid method.**

### C. Deformable overset grid

In engineering application, there are a large number of cases that involve multiple moving bodies with large relative displacement motion and self-body deformation at the same time, such as fish swarm swimming, flexible flapping wings and rotary wings coupled with structural deforming. Solely using overset grid or mesh deformation technique is not sufficient to satisfy the specified requirements. In the present study, an effective deformable overset grid is implemented by using the aforementioned hierarchical overset grid locally coupled with the improved Delaunay graph mapping method to perform the dynamic mesh movement. The movement of each component is decomposed to a relative motion and a self-deformation which are implemented by the overset grid and the mesh deformation technique, respectively.

The algorithm of the proposed deformable overset grid method is summarized in a flowchart in **Fig. 9**. Taking a group of deforming airfoils with relative motion for example, the deformable overset grid technique is illustrated in **Fig. 10**. The airfoil group, consisting of Airfoil A, B and C, moves from the state in **Fig. 10**(d) to the position of **Fig. 10**(e) or of **Fig. 10**(f) with each airfoil performing translation, pitching and self-deformation simultaneously. As can be seen, the relative movement between them is extremely large. The deformable overset grid method generates a coarse background Delaunay graph (see **Fig. 10**(a)) along with the computational mesh (see mesh CLUSTER in **Fig. 10**(d)) for each airfoil and then calculates the area/volume ratio coefficients to obtain the one-to-one mapping between them. The domain of the Delaunay graph can be larger than that of the computational mesh, which can be beneficial to preserve the computational mesh quality, as the surface deformation can be propagated to very far field by the large domain of Delaunay graph and, therefore, the mesh quality are maintained in the near-field. At each time step of unsteady computation, the Delaunay graph is moved according to the specified translating or rotating motion and is deformed by the spring analogy according to the surface deformation (**Fig. 10**(b), (c)) Then the mesh CLUSTER of each moving component is mapped into its new position by the pre-defined one-to-one mapping (**Fig. 10**(e), (f)). After the overset grid algorithm presented in Section II is performed (**Fig. 10**(h), (i)), the fluid flow computation can be fulfilled on each CLUSTER followed by flow interpolation between the CLUSTERs.

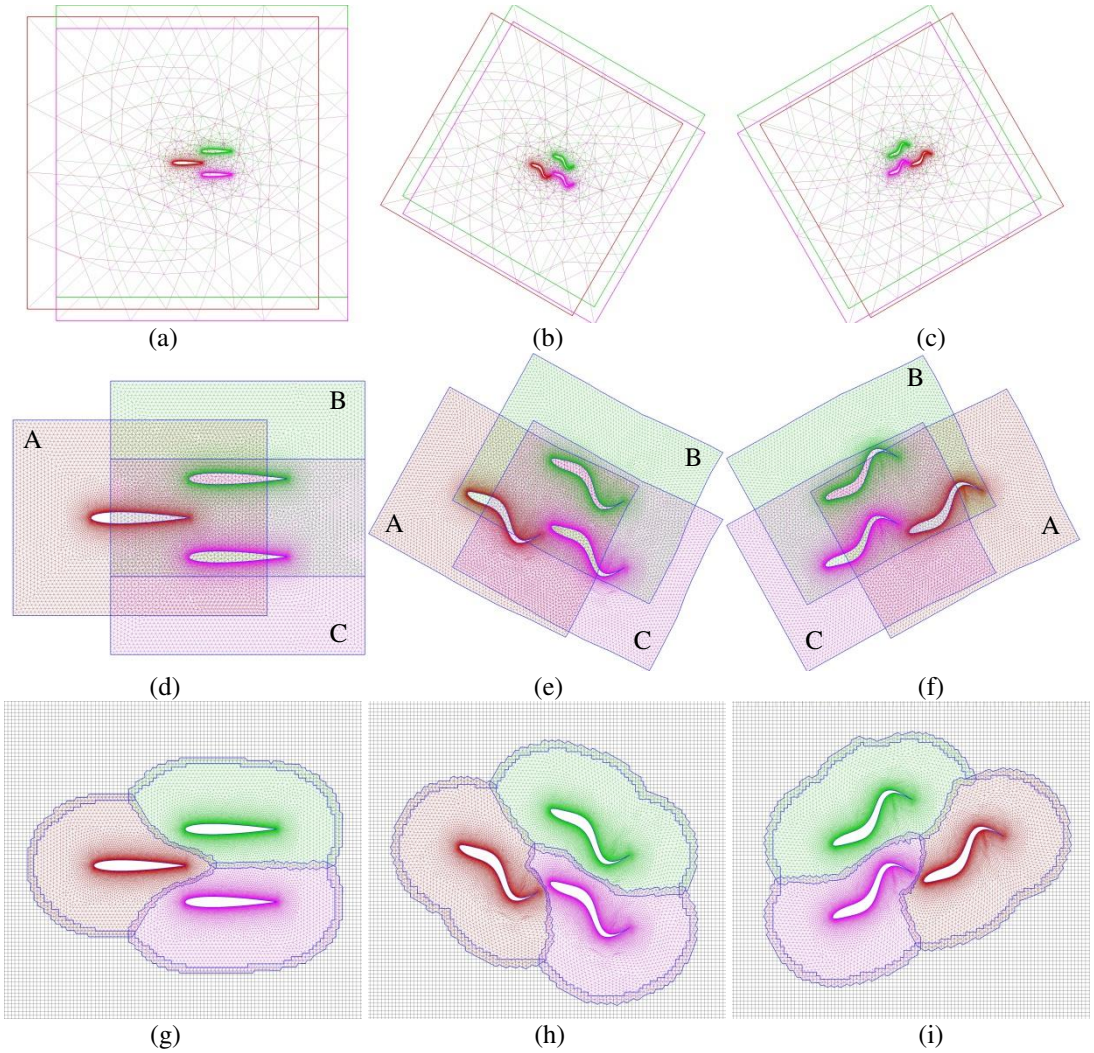



**Fig. 10 Illustration of the deformable overset grid for fish swarm swimming.**

## IV. Application to Multi-body Unsteady Aerodynamic Test Cases

This section performs several typical applications to examine its capability in multi-body unsteady aerodynamic simulations.

### A. Multi-flexible-body Problem with Relative Motion

To test the the robustness and efficiency of the proposed deformable overset grid method in solving multiple bodies with both large movement and self-deformation, the unsteady flow caused by multiple flexible airfoils with relative motion was simulated. As shown in **Fig. 11**, the two airfoils on the sides of the middle one move forward with a relative faster speed, resulting in a shear type motion between them, while each airfoil perform a periodic swimming-like motion defined by the deforming of their backbone as,

$$h(x,t) = a(x - x_0)\sin\left[2\pi\left(\frac{x - x_0}{\lambda} - \frac{t}{T}\right)\right], \tag{8}$$

where $a(x) = c_1 x + c_2 x^2$ is the curve envelop. The swimming law is characterized by four parameters: $c_1$, $c_2$, the wave length $\lambda$ and the frequency $f$. In this simulation, the free-stream conditions and the motion parameters were set to make the Reynolds number and the Strouhal number at $Re$=9.8×10$^3$, $St$=0.32, respectively.

Four grid CLUSTTERs were generated and organized into two LAYERs. LAYER 1 as background mesh has one hybrid grid CLUSTER consisting of 49,614 rectangle/triangle elements with uniform size of 0.04$L$ in the Cartesian grid zone. Three body-fitted grid CLUSTERs with refined boundary layer in LAYER 2 are related to the three deforming airfoils with each containing 10,801 grid cells. The relative motion of the airfoils, say the forward movement, was implemented by the overset grid at each time step; instantaneously, the self-deformation was locally handled with the improved Delaunay graph mapping method. Laminar flow was applied for this simulation. The deforming overset grid system and the flow field plotted with vorticity at different time instants are presented in **Fig. 11**. As can be seen, the flow field reveals the period shedding of vortices due to the deforming airfoils.

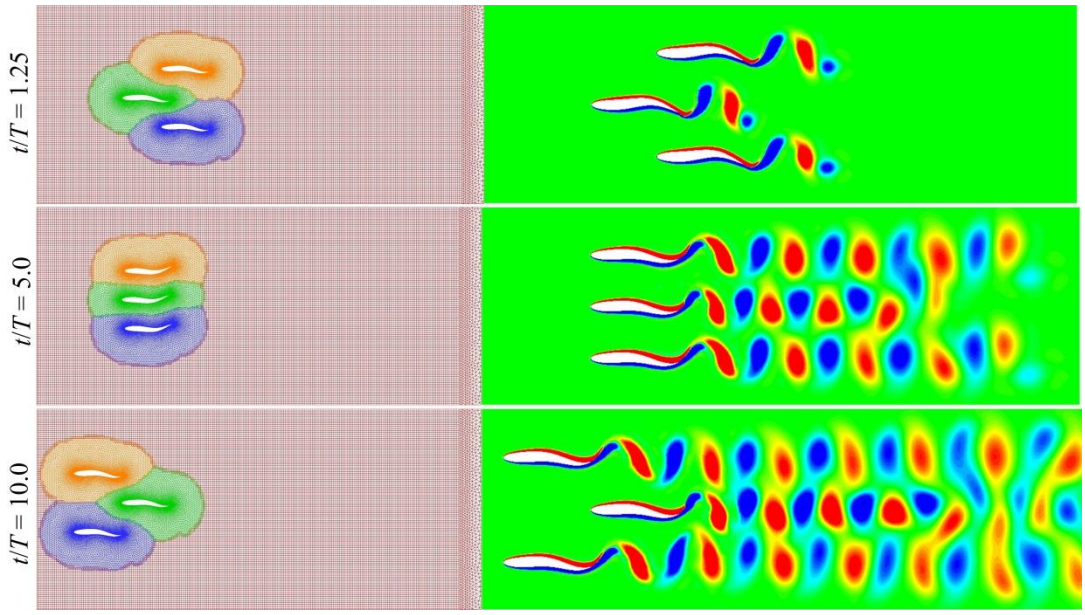



**Fig. 11 The simulation of unsteady flow caused by the multiple flexible airfoils with large relative motion (left column: deformable overset grid; right column: vorticity contours).**

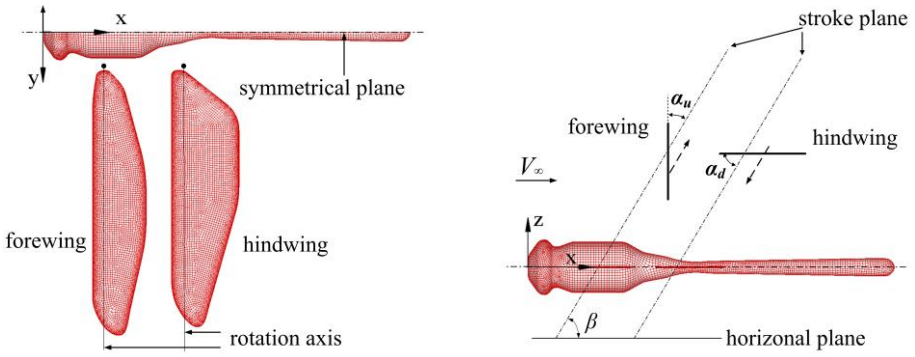
### B. Application to 3D dragonfly model in forward flight with flexible wing

Another application presented in this paper is a forward flight case of a 3D dragonfly model which was constructed according a picture of a real dragonfly as shown in **Fig. 12**(a). The kinematic parameters of the model's

flapping wings, illustrated in **Fig. 12**(b), are set up the same as the forward flight case (advance ratio $J$=0.3) presented in Wang and Sun [37] with the hind-wing leading the forewing in a phase of 180°. Two kinematics of the motion were simulated: one is for rigid wings; the other is for flexible wings with camber deformation according to the data measured by Wang and Zeng [38]. The scattered camber deformation data during one flapping cycle are fitted into a curve as shown in **Fig. 13**(a) so that a continuous deformation can be implemented for the simulation. The flapping motion and the camber deformation for either wing during one cycle are demonstrated in **Fig. 13**(b).

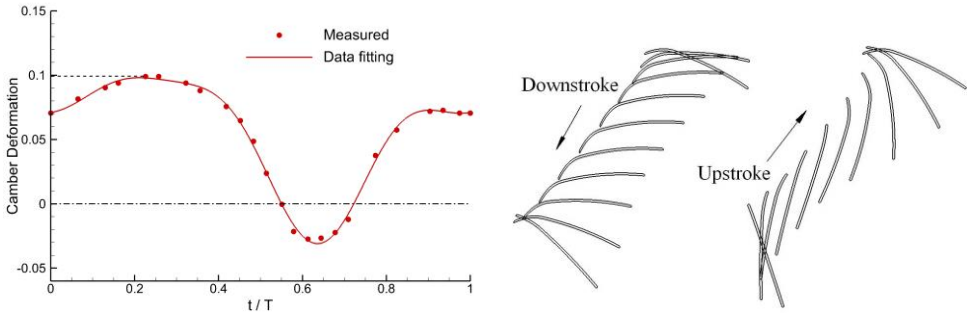
**Fig. 14**(a) and **Fig. 14**(b) show the grid system for the dragonfly model. A total four grid CLUSTERs were generated and assembled into two LAYERs. LAYER 1 has one CLUSTER serving as background grid while three body-fitted CLUSTERs in LAYER 2 are associated with the forewing, the hindwing and the body, respectively. A total four CLUSTERs consist of about 1.04 million nodes and 2.54 million cells.

Laminar model was employed for the simulation as the Reynolds number is only 1566 based on the mean chord length $L$ and the mean translating velocity $U_{ref}$ at 2/3 spanwise location of the forewing.



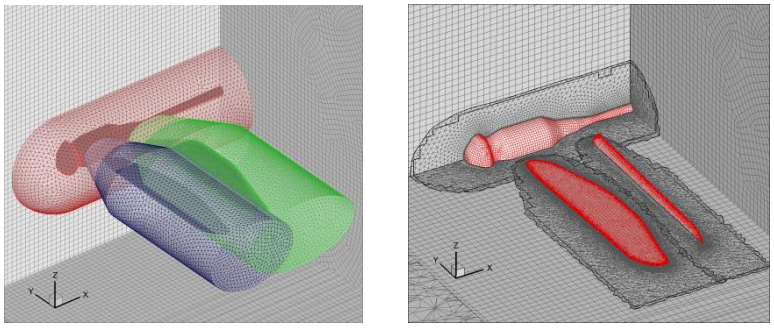

(a) (b)

**Fig. 12 Geometric and kinematic definition of a dragonfly model.**




(a) (b)

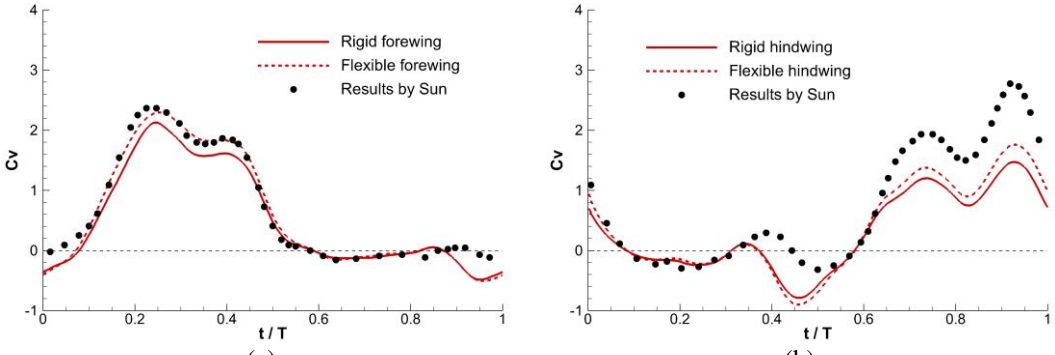
**Fig. 13 Camber deformation of the dragonfly flapping wings during one flapping cycle.**



(a) (b)

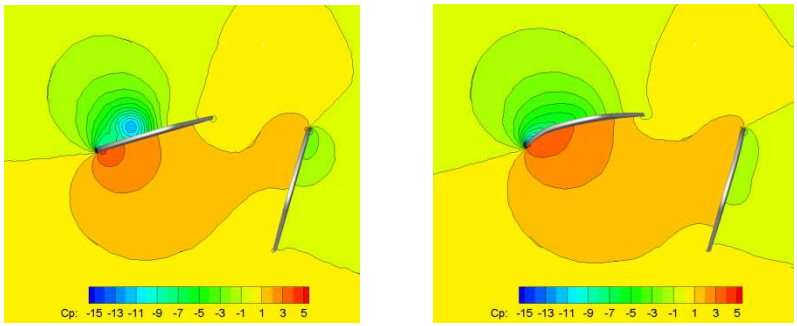
**Fig. 14 Overset grid system for the dragonfly model: (a) overset grid before hole-cutting; (b) overset grid after hole-cutting.**

The results for the vertical force coefficient caused by the forewing and the hindwing are shown in **Fig. 15**(a) and **Fig. 15**(b). The vertical forces computed by Wang and Sun [37] for rigid wings are also presented for comparison. As can be seen, good agreement in trend was obtained as the same kinematic motion of wings were employed by both studies, while the difference in the wing geometries (platform, aspect ratio and area) results in a difference in the predicted peaks of vertical forces. Comparison of vertical force between the rigid wings and flexible wings indicates that wing's flexibility benefits the aerodynamics of flapping wing. Spanwise pressure contours of the two styles of flow at 2/3 wing length are plotted in **Fig. 16**(a, b). It can be seen from the figure that the pressure contours pass smoothly across the overlapping region showing proper communication between different sub-grids.




(a) (b)

**Fig. 15 Time courses of vertical force coefficient caused by the wings of the dragonfly model.**




(a) (b)

**Fig. 16 Spanwise pressure contours at 2/3 wing span length of the dragonfly model: (a) rigid wings; (b) flexible wings.**

**Fig. 17** plots the iso-surface of the vorticity at different time instants during one flapping cycle, where the evolution of the starting vortex, leading edge vortex, wake vortex and tip vortex are clearly captured. The LEVs on the forewing and hindwing augment the vertical force generation with the first force peak at $t/T$ = 0.25 as seen in **Fig. 15**(a) and the second peak at $t/T$=0.7 in **Fig. 15**(b), respectively. The second peak of the forewing ($t/T$ = 0.4) and the third peak of the hindwing ($t/T$ = 0.9) might due to the generation of the RSV that create a low pressure region on the upper wing surface.

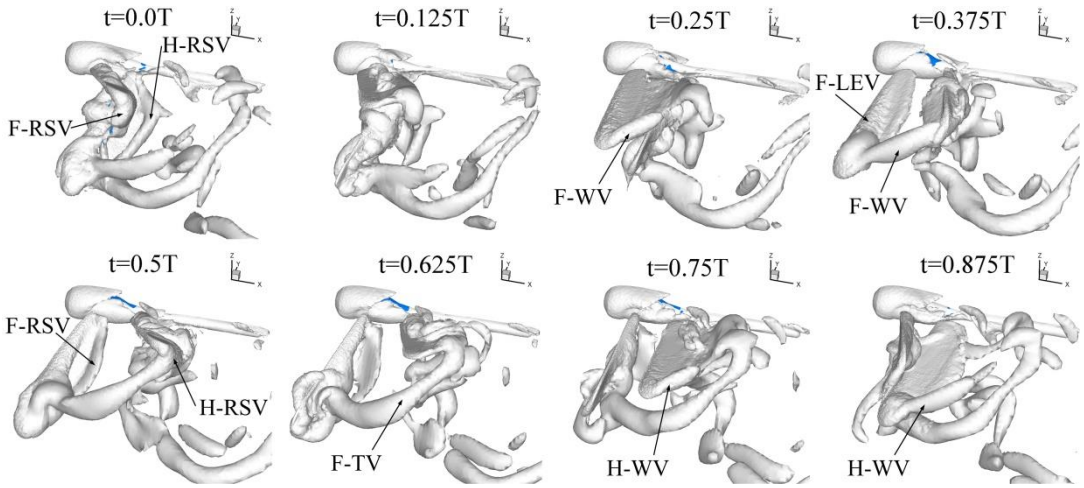



**Fig. 17 Vortical structures generated by the dragonfly model during one flapping cycle plotted by the iso-surface of the vorticity magnitude ($J$=0.3, $\psi$=180$^o$ ,$|\omega|$=0.72$U_{ref}$ /$L$), notations are RSV－Rotational Starting Vortex; LEV－Leading Edge Vortex; TV－Trailing Vortex; WV－Wingtip Vortex; F－Forewing, H－Hindwing.**

## V. Conclusion

A deformable overset grid method was proposed in this paper by combining an unstructured overset grid with an improved Delaunay graph mapping deformation method for simulating unsteady flows involving multiple moving bodies. The original Delaunay graph mapping method was firstly improved by creating a very coarse mesh as background graph which is deformed by means of spring analogy. As demonstrated by a two-dimensional rotating airfoil and a three-dimensional bending DLR F6 wing, the robustness (i.e. mesh quality) of the proposed mesh deformation method is significantly improved when comparing with the original method, particularly when the moving bodies experience large rotational deformation. The additional computational expense associated with the improved Delaunay graph mapping method is more than offset by the significantly improved robustness.

An unstructured overset grid technique, with sub-grids hierarchically organized into CLUSTERs and LAYERs, allows for overlapping and embedding of different type meshes, of which the mesh quality and resolution can be independently controlled. An efficient implicit hole-cutting and inter-grid boundary definition procedure was designed that allows a fully automatic implementation for either cell-centered or cell-vertex schemes. By locally coupled with the improved Delaunay graph mapping method on sub-grid CLUSTERs associated with deforming bodies, the deformable overset grid was implemented for multi-body unsteady problems with simultaneous large relative movement and surface deformation. This dynamic mesh method inherits the advantages of the overset grid method in handling large relative boundary movement, and also benefits from the efficiency of the mesh deformation technique based on Delaunay graph mapping for small deformation, especially when the mesh deformation is localized.

The deformable overset grid method was successfully applied to two unsteady aerodynamic problems, flows around multiple flexible moving airfoils and multiple deforming flapping wings, to demonstrate its capability for simulating multiple body flows undergoing both large relative movement and self-deformation. The results showed the robustness of this method for complex unsteady problems and suggested this method to be an efficient way to solve the problems that afflict the previous dynamic mesh methods.

## Acknowledgments

This work has been supported by the Fundamental Research Funds for the Central Universities (56XAA15012) and the National Science Foundation of China (No.11002072 and No.11502112). The first author acknowledges the support from China Scholarship Council (CSC Grant No. 201303070173) for funding his visiting scholarship.

## References


[1] Peskin, C. S., "The Immersed Boundary Method," *Acta Numerica*, Vol. 11, 2002, pp. 479-517.

[2] Batina, J. T., "Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-Aircraft Aerodynamic Analysis," *AIAA Journal*, Vol. 29, No. 3, 1991, pp. 327-333.

[3] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M., "Torsional Springs for Two-dimensional Dynamic Unstructured Fluid Meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol. 163, 1998, pp. 231-245.

[4] Degand, C., and Farhat, C., "A Three-Dimensional Torsional Spring Analogy Method for Unstructured Dynamic Meshes," *Computers & Structures*, Vol. 80, 2002, pp.305-316.

[5] Murayama, M., Nakahashi, K., and Matsushima, K., "Unstructured Dynamic Mesh for Large Movement and Deformation," *40th AIAA Aerospace Science Meeting & Exhibit*, AIAA 2002-0122, Reno, NV, 2002.

[6] Johnson, A. A., and Tezduyar, T. E., "Simulation of Multiple Spheres Falling in a Liquid-filled Tube," *Computer Methods in Applied Mechanics and Engineering*, Vol.134, No.3, 1996, pp.351-373.

[7] Yang, Z., and Mavriplis, D. J., "Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier-Stokes Equations," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2005-1222. Reno, NV, 2005.

[8] Thompson, J. F., Warsi, Z. U., and Mastin, C. W., "Numerical Grid Generation, Foundations and Applications," *Elsevier Science Publishing Company*, New York, NY, 1985.

[9] Liu, X., Qin. N, and Xia, H., "Fast Dynamic Grid Deformation based on Delaunay Graph Mapping," *Journal of Computational Physics*, Vol. 211, No. 2, 2006, pp. 405-423.

[10] de Boer, A., van der Schoot, M. S., and Bijl, H., "Mesh Deformation based on Radial Basis Function Interpolation," *Computers & Structures*, Vol. 85, 2007, pp.784-795.

[11] Rendall, T. C. S., and Allen, C. B., "Improved Radial Basis Function Fluid-Structure Coupling via Efficient Localised Implementation," *International Journal for Numerical Methods in Engineering*, Vol. 78, No. 10, 2009, pp. 1188–1208.

[12] Rendall, T. C. S., and Allen, C. B., "Efficient Mesh Motion Techniques using Radial Basis Functions with Data Reduction Algorithms," *Journal of Computational Physics*, Vol. 228, No.17, 2010, pp. 6231–6249.

[13] Rendall, T. C. S., and Allen, C. B., "Parallel Efficient Mesh Motion Using Radial Basis Functions with Application on Multi-Bladed Rotors," *International Journal for Numerical Methods in Engineering*, Vol. 81,No. 1, 2010, pp. 89–105

[14] Buhmann, M., "Radial Basis Functions (1st edition)," *Cambridge University Press*: Cambridge, 2005.

[15] Wendland, H., "Scattered Data Approximation (1st edition)," *Cambridge University Press*: Cambridge, 2005.

[16] Sheng, C., and Allen, C. B., "Efficient Mesh Deformation Using Radial Basis Functions on Unstructured Meshes," *AIAA Journal*, Vol. 51, No. 3, 2013, pp. 707-720.

[17] Wang, G., Mian, H. H., Ye, Z. Y., and Lee, D. L., "Improved Point Selection Method for Hybrid-Unstructured Mesh Deformation Using Radial Basis Functions," *AIAA Journal*, Vol. 53, No.4, 2015, pp. 1016-1025

[18] Wang, Y., Qin, N., and Zhao, N., "Delaunay Graph and Radial Basis Function for Fast Quality Mesh Deformation," *Journal of Computational Physics*, Vol. 294, 2015, pp. 149–172.

[19] Zhang, L.P., and Wang, Z.J., "A Block LU-SGS Implicit Dual Time-Stepping Algorithm for Hybrid Dynamic Meshes," *Computers & Fluids*, Vol. 33, No.7, 2004, pp. 891-916.

[20] Zhang, L. P., Chang X. H., Duan, X. P., and Zhao Z., "Applications of Dynamic Hybrid Grid Method for Three-dimensional Moving/Deforming Boundary Problems," *Computers & Fluids*, Vol. 65, No.15, 2012, pp. 45-63.

[21] Steger, J. L., Dougherty, F. C., and Benek, J. A., "A Chimera Grid Scheme," Advances in grid generation, K. N. Chua and U. Ghia, Eds., New York: ASME FED, Vol. 5, June, 1985.

[22] Nakahashi, K., Togashi, F., and Sharov, D., "Inter-grid Boundary Definition Method for Overset Unstructured Grid Approach," *AIAA Journal*, Vol. 38, No. 11, 2000, pp. 2077-2084.

[23] Wang, G., Duchaine, F., Papadogiannis,D, Ignacio Duran, I., Moreau,S., Laurent Y.M., and Gicquel, L., "An Overset Grid Method for Large Eddy Simulation of Turbomachinery Stages," *Journal of Computational Physics,* Vol.274, No.1,2014, pp.333-355.

[24] Chan, W. M., "Overset Grid Technology Development at NASA Ames Research Center," *Computers & Fluids,* Vol.38, No.3, 2009, pp.496-503.

[25] Biava, M., Khier, W., and Vigevano, L., "CFD Prediction of Air Flow past a Full Helicopter Configuration," *Aerospace Science and Technology*, Vol.19, No.1, 2012, pp.3-18

[26] Aono, H., and Liu, H., "Flapping Wing Aerodynamics of a Numerical Biological Flyer Model in Hovering Flight," *Computers & Fluids*, Vol.85, No.1, 2013, pp.85-92.

[27] Koomullil, R., Bhagat, N., Ito, Y., and Noack, R., "Generalized Overset Grid Framework for Incompressible Flows," *44th Aerospace Sciences Meeting and Exhibit*, Reno, NV, AIAA 2006-1146, 2006.

[28] Fast, P., and Henshaw, W. D., "Time-Accurate Computation of Viscous Flow around Deforming Bodies Using Overset Grids," *15th AIAA Computational Fluid Dynamics Conference*, Anaheim, CA, AIAA 2001-2640, 2001.

[29] Venkatakrishnan, V., "Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters," *Journal of Computational Physics*, Vol.118, No.1, 1995, pp.120-130.

[30] Lesoinne M., and Farhat C., "Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable Meshes, and their Impact on Aeroelastic Computations," *Computer Methods in Applied Mechanics and Engineering*, Vol.134, No.1-2, 1996, pp. 71-90.

[31] Koobus B., and Farhat C., "Second-Order Time-Accurate and Geometrically Conservative Implicit Schemes for Flow Computations on Unstructured Dynamic Meshes," *Computer Methods in Applied Mechanics and Engineering*, Vol.170, No.1-2, 1999, pp. 103-129.

[32] Mavriplis D.J., and Yang Z., "Construction of the Discrete Geometric Conservation Law for High-order Time Accurate Simulations on Dynamic Meshes," *Journal of Computational Physics*, Vol.21, No.2, 2006, pp. 557-573.

[33] Xiao, T., Ang, H., and Yu, S., "A Preconditioned Dual Time-stepping Procedure Coupled with Matrix-free LU-SGS Scheme for Unsteady Low Speed Viscous Flows with Moving Objects," *Internatioanl Journal of Computational Fluid Dynamics*, Vol.21, No. 3-4, 2007, pp. 165-173.

[34] Bonet, J., and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," *International Journal of Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1-17.

[35] Knupp, P. M., "Algebraic Mesh Quality Metrics for Unstructured Initial Meshes," *Finite Elements in Analysis and Design*, Vol.39, No.3, 2003, pp. 217-241.

[36] Cai, J., Tsai, H.M., and Liu, F., "A Parallel Viscous Flow Solver on Multi-block Overset Grids," *Computers & Fluids*, Vol. 35, No. 10, 2005, pp. 1290-1301.

[37] Wang, J. K., and Sun, M., "A Computational Study of the Aerodynamics and Forewing-Hindwing Interaction of a Model Dragonfly in Forward Flight," *Journal of Experimental Biology*, Vol. 208, No.19, 2005, pp. 3785-3804.

[38] Wang, H., Zeng, L. J., Liu, H., and Yin, C., "Measuring Wing Kinematics, Flight Trajectory and Body Attitude during Forward Flight and Turning Maneuvers in Dragonflies," *Journal of Experimental Biology*, Vol. 206, No.4, 2003, pp. 745-757.