Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 211 (2006) 405–423
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

# Fast dynamic grid deformation based on Delaunay graph mapping

Xueqiang Liu, Ning Qin *, Hao Xia

*Department of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK*

Received 1 November 2004; received in revised form 20 April 2005; accepted 12 May 2005
Available online 1 August 2005

**Abstract**

A simple and efficient dynamic grid deformation technique is proposed for computing unsteady flow problems with geometrical deformation, relative body movement or shape changes due to aerodynamic optimisation and fluid–structure interaction. A Delaunay graph of the solution domain is first generated, which can be moved easily during the geometric dynamic deformation, even for very large distortion. A one to one mapping between the Delaunay graph and the computational grid is maintained during the movement. Therefore the new computational grid after the dynamic movement can be generated efficiently through the mapping while maintaining the primary qualities of the grid. While most dynamic grid deformation techniques are iterative based on the spring analogy, the present method is non-iterative and much more efficient. On the other hand, in comparison with dynamic grid techniques based on transfinite interpolation for structured grids, it offers both geometric and cell topology flexibility, which is crucial for many unsteady flow problems involving geometric deformation and relative motions. The method is demonstrated through some typical unsteady flow test cases, including a pitching aerofoil in a fixed domain boundary, relative movement between multi-element aerofoils due to flap deployment for high lift, a deformable sphere in a fixed cube, and a three-dimensional flexible wing with large deformation.


*Keywords:* Dynamic grid; Moving mesh; Grid deformation; Delaunay graph

## 1. Introduction

Dynamic grids are essential for solving unsteady flow problems involving geometric deformation or relative motion of multiple bodies. For aeronautical applications, examples of such flows include the

* Corresponding author. Tel.: +44 114 222 7718.
*E-mail address:* n.qin@sheffield.ac.uk (N. Qin).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.05.025

deployment of high lift or flow control devices, deflection of control surfaces, fuel tank separation, weapon release and so on. Recent development in aerodynamic optimisation and multidisciplinary design optimisation including flexible wings with fluid structure interaction, and associated aeroelastics, and aeroservoelastics demands the simulation of deforming wings under aerodynamic loads or optimisation search involving potentially large deformation. The deforming geometries necessitate dynamically changing meshes to cover the changed solution domain. Both surface and volume mesh in the flow field need to be updates at each time step of the solution process. It is therefore very important to develop an efficient dynamic grid deformation/generation technique, which maintains the primary grid qualities at each time step or each design change.

For a given problem, geometric deformation can be defined by the specified motion, which is reflected as the perturbation of the surface grid. The dynamic grid generation can be viewed as a means to propagate this boundary perturbation into the whole solution domain. There are generally two categories of methods to achieve this: (i) by mesh regeneration; (2) by mesh deformation.

For its efficiency, the algebraic grid generation technique based on trans-finite interpolation (TFI) has been widely used for aeroelastic problems [1], aerodynamic optimisation [2], and multi-disciplinary optimisation [3] for mesh regeneration involving deforming boundaries for the dynamic structured grid regeneration. Based on TFI, Gaitonde and Fiddes [4] used exponential blending functions to control the quality and robustness of the mesh. TFI combines the efficiency of an algebraic grid generation method with the ability to handle fully 3D geometric perturbations by the multi-block structured grid. Numerous modifications have been made within the methodology for complicated geometric perturbations. However, the efficiency and robustness of this method are limited to applications for multi-block structured moving meshes.

For unstructured grids or structured grids generated by the solution of partial differential equations (often elliptic solvers) for desirable grid qualities, the mesh deformation approach is usually adopted. Grid regeneration at each time step or each optimisation iteration would be very expensive computationally. This leads numerous researchers to develop more efficient methods for dynamic meshing. For mesh deformation, the spring analogy scheme, first developed by Batina [5], is widely used, which models the mesh as a network of linear springs and solve the static equilibrium equations for this network to determine the new locations of the grid points. Applications of the spring analogy for grid deformation can be found for unsteady flow solutions [6] and for aerodynamic optimisation [7]. However, the method suffers problems in robustness for large deformations and refined grid, such as grids for viscous flows, resulting in grid crossing and negative volumes.

Farhat [8] proposed a modified spring analogy by adding additional non-linear torsion springs to avoid the negative cell volume problem associated with the linear spring network. A detailed analysis of the spring analogy is presented by Blom [9], demonstrating that the torsional spring is imperative to retain validity of a moving Navier–Stokes mesh. Murayama et al. [10] proposed a method to relate the spring stiffness with the angle between the faces in order to deal with the problems with large movement and deformation of surfaces. Chen [11] developed an exterior BEM solver that has a unified feature for the deforming flowfield grid generation. Assuming that the CFD mesh is to be embedded in an infinite linear elastic medium where the CFD surface grid is treated as a deformable hollow slit, a pseudo elastostatic problem with semi-infinite elastic domain can be formulated. Treating the grid as an elastic solid [12–14], which can be viewed as an extension of the spring analogy, improve substantially the robustness of the moving mesh (in the sense of avoiding mesh crossing and negative volumes) at the expense of a much higher computational cost. All the spring analogy based schemes for moving mesh require iterative solution of the static equilibrium equations to determine the new locations of the grid points at each time step or design change. For three-dimensional viscous problems, it is still time consuming to propagate the perturbation of boundaries to the volume grids in the flow field using the spring analogy based method, though a large improvement over grid regeneration.

In addition to the TFI methods for dynamic grid regeneration and the spring analogy based methods (including elastic solid based methods) for grid deformation, overset grids have also been used for dynamic grid problems, such as that proposed by Fast and Henshaw [15]. A thin layer of body-fitted grid near the deforming boundary is overset on a fixed Cartesian grid covering the computational domain. The body-fitted structured grid is regenerated at each time step using a hyperbolic grid generator. Therefore the grid regeneration is localised, comparing to full grid regeneration. Instead of overlapping the grids, Zhang and Wang [16] used an unstructured grid to link the body-fitted grid and the Cartesian grid. Localising the grid regeneration to the unstructured part of the grid with the application of the spring analogy. Still grid regeneration and interpolation of the flow properties at each time step is a costly part for both methods.

It is clear that, for moving grids, geometric complexity, computational efficiency and algorithm robustness are conflicting factors. For geometries with multi-block grids, TFI can be used to regenerate the grid at each time step or each design change with reasonable efficiency. For more complicated geometries when unstructured or hybrid grids are necessary, either spring analogy based grid deformation methods or grid regeneration are required with substantially increased computational effort. Within spring analogy based methods, increasing robustness and sophistication is at the expense of further reduction in computational efficiency.

In this paper, we propose a novel method to deform a mesh of any given topology dynamically for large shape changes. It is based on the creation of a Delaunay graph of the original mesh. The geometric movement is carried out using the Delaunay graph with ease and efficiency. The original grid is then mapped back onto the deformed Delaunay graph to provide the new mesh for the new time step or the new design cycle. The mapping guarantees that the original mesh topology and density distribution are maintained without mesh crossing or overlapping cells. In the following section, the dynamic grid generation method is described. Some examples of dynamic meshing are demonstrated in Section 3 including both inviscid and viscous grids.

## 2. Dynamic remeshing based on Delaunay graph

The method is divided into four steps: (a) generating the Delaunay graph; (b) locating the mesh points in the graph; (c) moving the Delaunay graph according to the specified geometric change; (d) relocating the mesh points in the new graph. In order to show the procedures of this method, a hybrid grid over a deforming aerofoil is chosen as shown in Fig. 1, with a highly stretched structured grid around the aerofoil and unstructured meshes away from the surface.

### 2.1. Generating the Delaunay graph

The computational domain is defined by the geometric boundary and the non-geometric boundaries, including far field, symmetry, singularity, and periodic boundaries, etc. In order to generate the Delaunay graph, some representative points are chosen on the boundaries of the computational domain. For this given set of boundary points, there exists a unique triangulation (for two-dimensional cases) or tetrahedralisation (for three-dimensional cases) using the Bowyer–Watson algorithm [17,18], known as the Delaunay criterion.

We define the grid generated in such a way as *a Delaunay graph* for the given moving grid problem. The graph covers the whole computational domain for the given configuration, including the interior elements. Such triangulation or tetrahedralisation is unique, maximizing the minimum angle of a triangle or tetrahedral. The Delaunay graph provides a unique mapping from the given boundary points to a coarse unstructured grid.

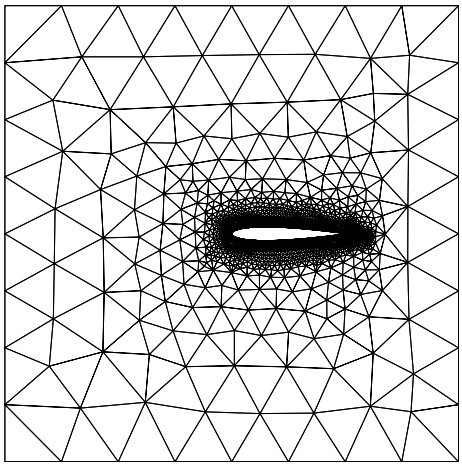

Fig. 1. Hybrid grid around an aerofoil.

For a computational grid, the geometrical boundary edges/faces should be a subset of the generated grid. It is well known that a main disadvantage of Delaunay grid generation is its inability to guarantee the geometry boundary integrity [19]. This is due to the fact that the unique Delaunay triangulation is only valid for convex domains and the final computational grid is an extraction of the whole Delaunay grid by deleting the interior cells/points inside the geometries in question. It is therefore important to make sure that the cells do not break through the prescribed geometry boundaries by constrained Delaunay triangulation [20] through edge swapping or judiciously adding surface points in the break through regions [21].

However, for the dynamic grid method presented in this paper, the Delaunay graph is only an intermediate map rather than the grid itself. The boundary faces (3D) and edges (2D) are maintained in the moving grid automatically as they are mapped back after the graph is moved. As the topology of the original grid is fully maintained in the method, so are the boundary faces and edges. This means that the Delaunay graph itself (not the grid) does not need to be geometrical surface preserving, i.e. breakthroughs at the geometry boundaries are allowable in the Delaunay maps, assuring the Delaunay property which is useful for validity of the moving graph.

For the grid shown in Fig. 1, the far field boundary can be defined by the four corner points and the aerofoil geometry by all the surface points. Note that since the whole aerofoil deforms, all surface points are required to maintain the integrity of the curved geometry at the grid level while four corner points are sufficient to maintain the integrity of the farfield boundary. If a curved farfield boundary needs to be maintained during the grid movement, more points may be necessary there for the Delaunay graph.

The Delaunay unstructured grid generation method as described in [22,23] are adopted, based on point insertion using the Delaunay criterion, i.e. the circum-circle for triangles or the circum-sphere for tetrahedrons should not include other points except the points which construct the triangle or tetrahedral. As we only use the boundary points, the cell number of triangles and tetrahedrons are comparatively much smaller than that for the whole grid and therefore the time in forming the graph is very small, even if more points are used for the farfield boundary. Because we only use the graph for mesh movement or deformation, we do not need to delete the unwanted grids that lie in the solid faces. For the deforming aerofoil case, Fig. 2(a) shows the solution domain boundaries and Fig. 2(b) the corresponding Delaunay graph. Note that the Delaunay graph includes the cells inside the aerofoil. This is important since there is no guarantee that

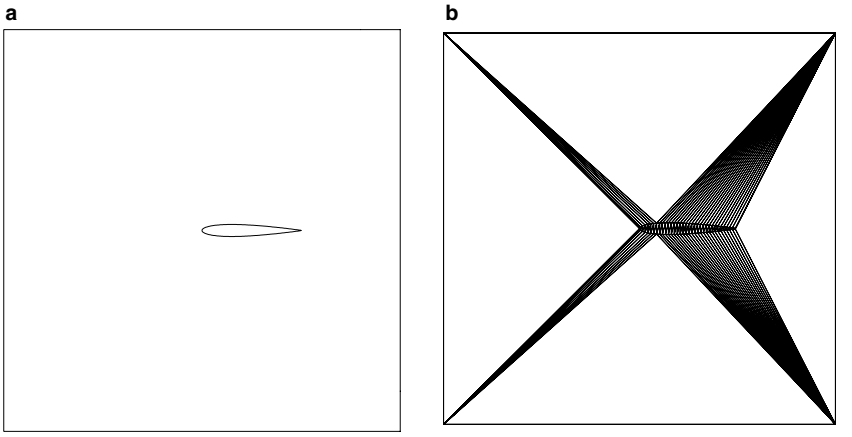


Fig. 2. Generation of Delaunay graph: (a) solution domain boundaries: geometry and farfield, (b) the Delaunay graph.

the geometrical boundary is not broken for the Delaunay graph although it is maintained in this case. If a breakthrough of the solid boundary happens, the Delaunay element strides the boundary will contain some the original grid points, which need to be mapped in the moving grid.

### 2.2. Locating the mesh points in the Delaunay graph

Given a Delaunay graph covering the original grid, different rules may be used to locate the mesh points in the graph, but these rules must satisfy that the positions of the mesh points in the graph is uniquely defined by a one-to-one mapping. Since the graph covers the whole solution domain, any point within it must belong to one of the triangles or tetrahedrons elements in the graph.

In the method proposed in this paper, we use the relative area coefficients to define the points for 2D and the relative volume coefficients to define points for 3D meshes. These coefficients have three components for two-dimensional cases and four components for two-dimensional cases.

Fig. 3(a) demonstrates the construction of the coefficients of a point in an element of the graph for two-dimensional cases. If a mesh point P is found in the graph element ABC, three area ratio coefficients, $e_1$, $e_2$, $e_3$, which are relative to the points A, B, C, respectively, are calculated by

$$e_i = \frac{S_i}{S}, \quad i = 1, 2, 3, \tag{1}$$

where the areas are given by

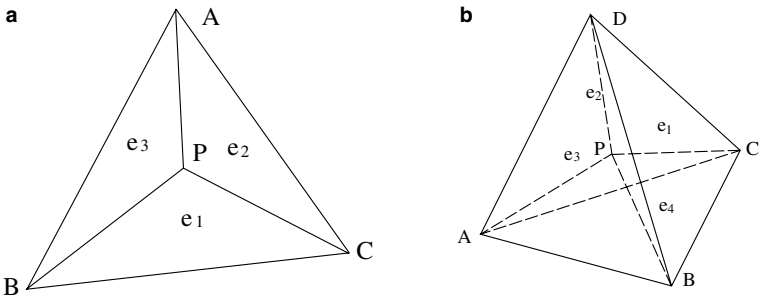


Fig. 3. Relation of the coefficients with the nodal points: (a) 2D, (b) 3D.

$$S_1 = \frac{1}{2}\begin{vmatrix} x_P & y_P & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}, \tag{2}$$

$$S_2 = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_P & y_P & 1 \\ x_C & y_C & 1 \end{vmatrix}, \tag{3}$$

$$S_3 = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_P & y_P & 1 \end{vmatrix}, \tag{4}$$

$$S = \frac{1}{2}\begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}. \tag{5}$$

As can be seen from the above formulae, $e_1$, $e_2$, $e_3$ are the relative area coefficients, $e_1$ being the ratio of the triangular area PBC to the triangular area ABC, and so on.

Similarly, Fig. 3(b) shows the situation for two-dimensional cases, where the position of point P in the graph element is defined by the four relative volume coefficients by

$$e_i = \frac{V_i}{V}, \quad i = 1, 2, 3, 4, \tag{6}$$

where the volumes are given by

$$V_1 = \frac{1}{6}\begin{vmatrix} x_P & y_P & z_P & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix}, \tag{7}$$

$$V_2 = \frac{1}{6}\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_P & y_P & z_P & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix}, \tag{8}$$

$$V_3 = \frac{1}{6}\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_P & y_P & z_P & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix}, \tag{9}$$

$$V_4 = \frac{1}{6}\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_P & y_P & z_P & 1 \end{vmatrix}, \tag{10}$$

$$V = \frac{1}{6}\begin{vmatrix} x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ x_C & y_C & z_C & 1 \\ x_D & y_D & z_D & 1 \end{vmatrix}. \tag{11}$$

For each grid point, the coefficients are calculated in relation to the graph element it occupies. Obviously, we have

$$\sum_i e_i = 1 \tag{12}$$

for both two-dimensional and three-dimensional cases.

From the definition of areas (2D) and volumes (3D), if and only if all the signs of the above coefficients are positive or zero, the point is in the graph element. Otherwise the point is classified outside the element. An efficient walk-through algorithm [24,25] is adopted for the purpose of locating the Delaunay element for a given mesh point. The complexity of the walk-through algorithm is of $O(n^{1/d})$, where $n$ is the total number of the Delaunay elements and $d$ is the spatial dimension. A brute force search would result in an algorithm complexity of $O(n)$, which could make locating element stage expensive for three-dimensional cases. The calculated mesh points' coefficients are stored along with the graph element number.

The idea of the above mapping between the Delaunay graph and the mesh points comes from the linear interpolation shape functions used in finite element methods (see e.g. [26]). To find a linear interpolation of a functional value $U$ at P in a triangular element from the values, $U_1$, $U_2$, $U_3$, at the nodal points $P_1$, $P_2$ and $P_3$, respectively, we have

$$U_P = \sum_i e_i U_i. \tag{13}$$

In the context of the present paper, by replacing the functional values with the co-ordinates for point P, ($x_P$, $y_P$), and the nodal points of the graph element, ($x_i$, $y_i$), $i = 1, 2, 3$, we have

$$x_P = \sum_i e_i x_i, \tag{14}$$

$$y_P = \sum_i e_i y_i. \tag{15}$$

Eqs. (14) and (15) along with (1) and (12) defines a one-to-one mapping from ($x_P$, $y_P$) in the Delaunay element, to the associated area ratio coefficients ($e_1$, $e_2$, $e_3$). Therefore, all the points within the identified Delaunay graph element are mapped to the coefficient sets. The 3D mapping can be similarly defined.

### 2.3. Moving the Delaunay graph

As discussed before, the graph elements are constructed based on Delaunay triangulation/tetrahedralisation of selected boundary points and geometrical points, which represent the moving/deforming body. The graph covers the whole solution domain. Generally the outer boundary points may stay fixed, and the geometrical points move according to a prescribed motion or deformation due to the underlined physics (e.g. structural deformation). The body geometrical points chosen for the construction of the Delaunay graph should be sufficient to provide an accurate description of the body motion and deformation and to preserve the surface integrity during the movement. In order to maintain geometrical accuracy, all the body surface geometrical mesh points are chosen for the Delaunay graph generation.

After the movement, the graph is no longer guaranteed to satisfy the Delaunay rules. For the purpose of grid mapping, this does not cause any problem. However, if the movement is so large that some points move across each other, it will create a problem for the mapping as some of the coefficients (areas or volumes in Eqs. (1) and (6)) become negative. When this happens, the movement of the Delaunay graph is defined to be invalid.

In this situation, the variation for the movement is halved and a single movement is broken up into two smaller steps. If the situation persists, the step can be further refined. Note that this does not necessarily mean a reduction in the time step for the flow solution or a change of design change steps. For most moving mesh cases, the unsteady flow physics needs to be captured with reasonable time step resolutions and this anomaly of invalid Delaunay movement rarely happens if the original graph is Delaunay, which maximises the internal angles as mentioned earlier. In the application of the method, a single Delaunay graph may be used for a sequence of movements for efficiency if the changes are relative small. For the robustness of the procedure for relatively larger changes, it is advisable to regenerate the Delaunay graph at each geometrical change, which is also useful for maintaining the grid quality.

### 2.4. Relocating the mesh points in the graph

After the graph movement, a new set of coordinates is generated for the nodal points for each graph elements. On the graph, the mesh points can now be located based on the area or volume ratio coefficients stored for the points with the associated graph element number.

We require that the distribution of the point within a graph element maintains the area or volume ratio as in the original graph, i.e. the coefficients are kept constants during mesh movement. For two-dimensional cases,

$$x'_{\mathrm{P}} = \sum_{i=1}^{3} e_i x'_i, \tag{16}$$

$$y'_{\mathrm{P}} = \sum_{i=1}^{3} e_i y'_i. \tag{17}$$

For three-dimensional cases, we have

$$x'_{\mathrm{P}} = \sum_{i=1}^{4} e_i x'_i, \tag{18}$$

$$y'_{\mathrm{P}} = \sum_{i=1}^{4} e_i y'_i, \tag{19}$$

$$z'_{\mathrm{P}} = \sum_{i=1}^{4} e_i z'_i. \tag{20}$$

Here $(x'_{\mathrm{P}}, y'_{\mathrm{P}})$ or $(x'_i, y'_i, z'_i)$ are the new coordinates of the mesh point and the element nodal points. During the movement, the area or volume ratio coefficients remain constant, computed using (1) or (6), respectively. This implies that the coefficients calculated in (1) or (6) can be used in relocating the mesh points by (16), (17) or (18)–(20), respectively. Fig. 4 shows such a mapping from P in the original grid to P′ in the moved grid for the 2D case.

By doing so, the relative position of a point in a graph element is maintained by the fixed area or volume ratios no matter how much the element deforms. No crossover of the points is possible, which is crucial for the original grid property to be carried over and for the robustness of the meshing mesh. Fig. 5 illustrates how a cell ($P_1$, $P_2$, $P_3$) in the original grid striding over two Delaunay elements (A, B, C) and (A, C, D) is transformed into a cell in the new grid after the movement along with the movement of the Delaunay elements.

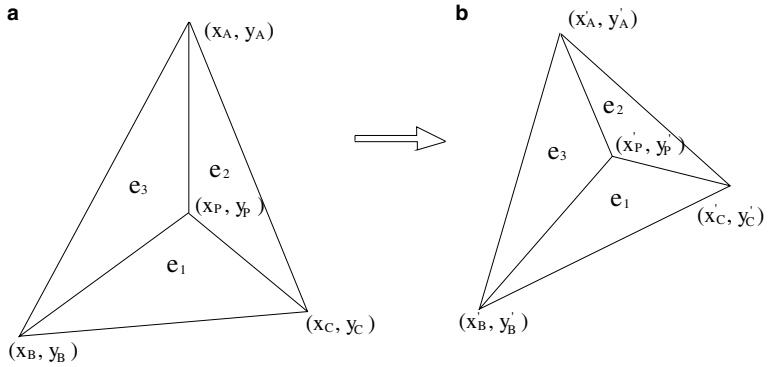


Fig. 4. Relocation of mesh point P within a Delaunay element during the mesh movement: (a) before movement, (b) after movement.

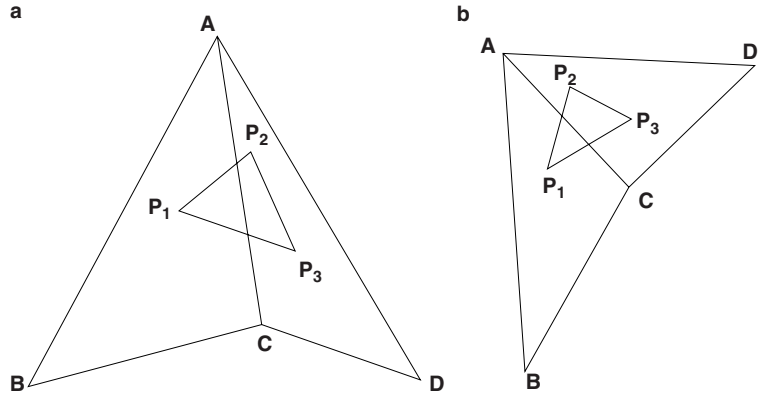


Fig. 5. Demonstration of the movement of a grid cell in the Delaunay graph: (a) before movement, (b) after movement.

Fig. 6 demonstrates a two-step deformation of a hybrid grid around an aerofoil using the Delaunay graph. Note that the quality of the original viscous grid is maintained with the high clustering of the points near the solid surface for boundary layer resolution. For this case, a single Delaunay graph on the initial symmetric geometry is generated for the sequence of two movements.

### 2.5. Computational procedure

In summary, the procedure of the moving mesh method based on the Delaunay graph can be written as

(1) Using selected solution domain boundary points and the Delaunay method to construct the Delaunay graph;
(2) Locating the graph elements for all the mesh points using the walk-through method; computing and storing the area or volume ratio coefficients for each mesh points using formulae (1)–(5) or (6)–(11), respectively;
(3) Moving the Delaunay graph according to the deformation of the solid surfaces or the relative movements between different bodies;
(4) Checking for any intersection between the graph elements, and if this happens, splitting the movement into two smaller movements and go to step (3);

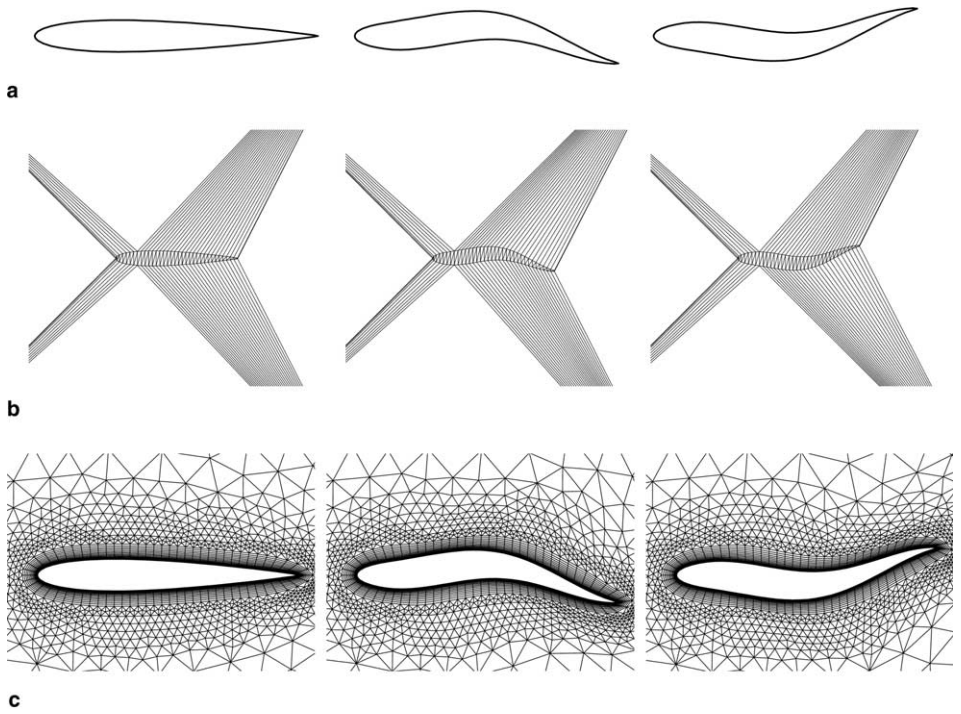


Fig. 6. Moving mesh for a hybrid viscous grid using the Delaunay graph: (a) solid surfaces deformation, (b) the Delaunay map and its movement, (c) original and moved meshes.

(5) Relocating the mesh points according to expressions (16), (17) or (18)–(20), respectively, using the coefficients computed in step (2) and the nodal coordinates of the new moved graph elements;
(6) Repeating (1)–(5) or (3)–(5) for the next time step or optimisation search shape.

The above moving mesh method is non-iterative and is therefore very efficient for unsteady flow solutions and design changes in optimisation involving deformation or relative motion. It is also worthwhile to point out that the procedure is valid for any type of meshes in 2D or 3D. In comparison, TFI is limited to structured meshes and methods based spring analogy often need special treatment for different type of meshes.

The procedure is more clearly illustrated in the following pseudo program:

```
begin
generate the Delaunay graph for current geometry location and solution boundaries
do n = 1, number of all mesh points
   set point n as Q
   choose a random Delaunay element k with centroid P
   walk-through from P to Q until Q's Delaunay element is found
   calculate the area or volume ratio coefficients in the graph element for Q
end do
1     generate new nodal points for the Delaunay graph after movement
do j = 1, number of graph elements
   if(area or volume of element. le. 0) then
```

```
      reduce the boundary movement
      goto 1
    end if
  end do
  do n = 1, number of all mesh points
    get graph element number: graph_element_number(n)
    get new graph element nodal coordinates
    get mapping coefficients for point n
    calculate the new mesh points
  end do
  end
```

## 3. Test cases

### 3.1. A pitching multi-element aerofoil

A two-element aerofoil is used for the pitching test case. Figs. 7(a), (b) and (c) demonstrate the moving mesh for the aerofoil from 0° incidence to −30°, and 30°, respectively, using the method described in Section 2. A single Delaunay graph is generated at 0°. No element cross over is observed in the Delaunay graph movements, indicating the robustness of the method for large movement. The original grid quality with a denser grid distribution around the aerofoil for inviscid flow solution is maintained after the movement. The moving meshes also show a good resistance to mesh distortion for these relatively large movements.

Additionally, in order to demonstrate the independence of the method on mesh topology, a grid, shown in Fig. 8(a), made of polygon cells with varying numbers of edges is created for the same two-element aerofoil. Such a grid can result from the application of some advanced grid generation techniques, e.g. that reported in Leatham et al. [27]. Fig. 8(b) shows the mesh with a pitching angle of 60°. Although, due to the much larger movement, the distortion of the cells is more visible in this case, the mesh is still of reasonable quality as an inviscid solution grid.

Note that this is a one-step movement solely for demonstration of the robustness of the method. For its implementation in dynamic flow calculations, many and much smaller steps (more regeneration of Delaunay graphs) are required to capture the flow physics and therefore the distortion should be much less severe.

### 3.2. Multi-element aerofoils with large dynamic flap deployment

The study of the dynamic effects of the flaps is important for aircraft control and high lift generation. For this case, a four-element aerofoil is chosen as a test case, for which there is a relative movement of multiple bodies. A large movement of the trailing edge flaps in relation to the main wing aerofoil is specified. Fig. 9(a) shows a local view of the original mesh for inviscid flow solution, with clustering around each of the four elements of the aerofoil. Fig. 9(b)–(f) plot a sequence of the mesh movement using the Delaunay graph mapping. A Delaunay graph is generated at each step before moving the grid for the next step. As can be seen, the mesh quality has been maintained at each stage of the movement.

For viscous flows, hybrid grids are often preferred because a high aspect ratio structured grid with a high degree of clustering near the solid surface is normally more suitable for capturing the boundary layer physics. For example, to resolve the turbulent boundary layers, the requirement of a $y$-plus value in the order of

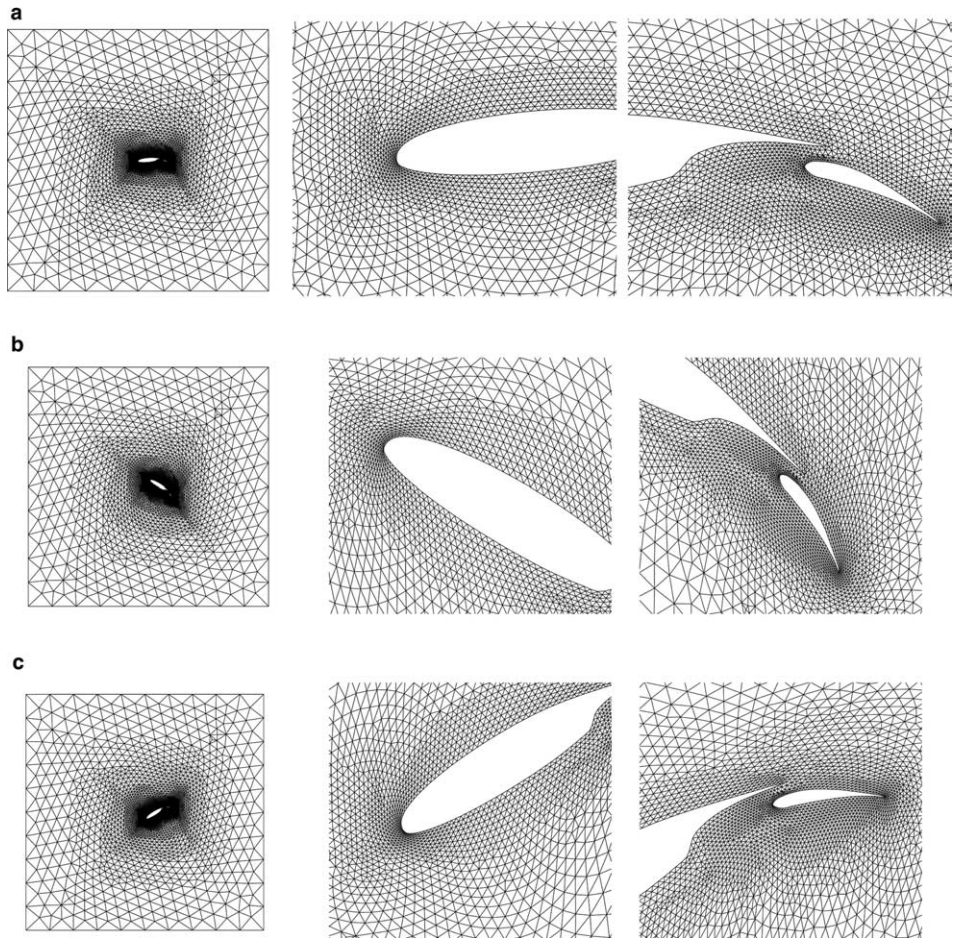


Fig. 7. Pitching aerofoil case: (a) original mesh, (b) moved mesh for 30° pitch, (c) moved mesh for −30° pitch.

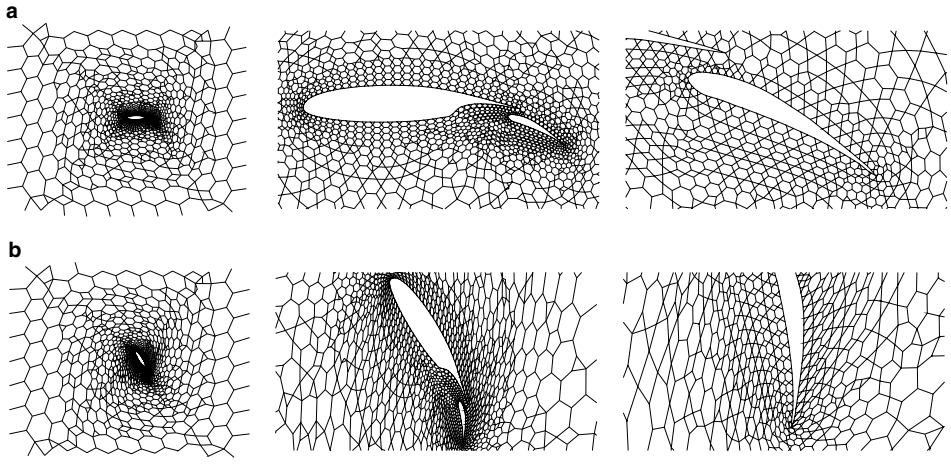


Fig. 8. Demonstration of moving mesh for arbitrary cell topology: (a) original mesh, (b) moved mesh with 60° pitch.

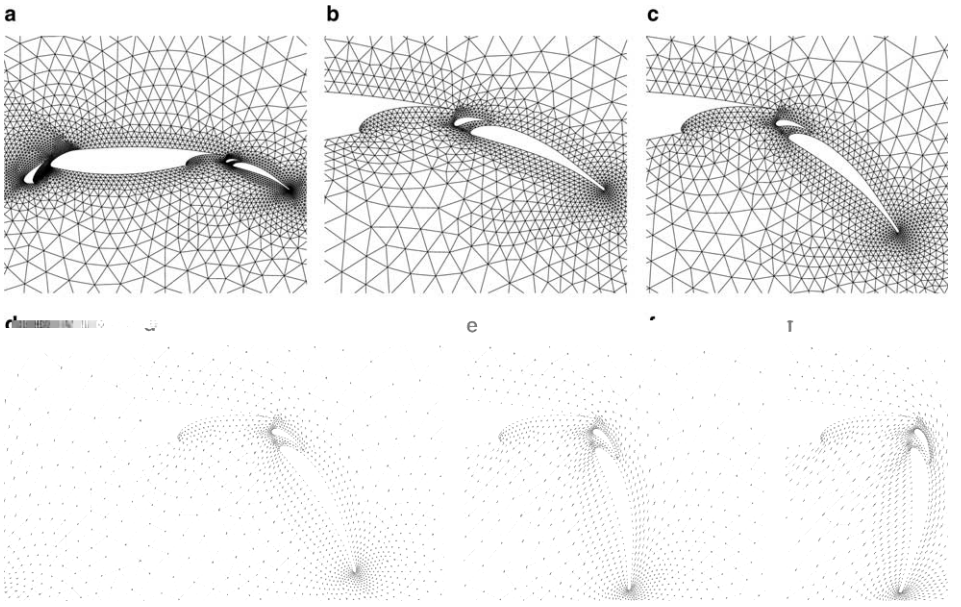


Fig. 9. Moving mesh around main foil and flaps.

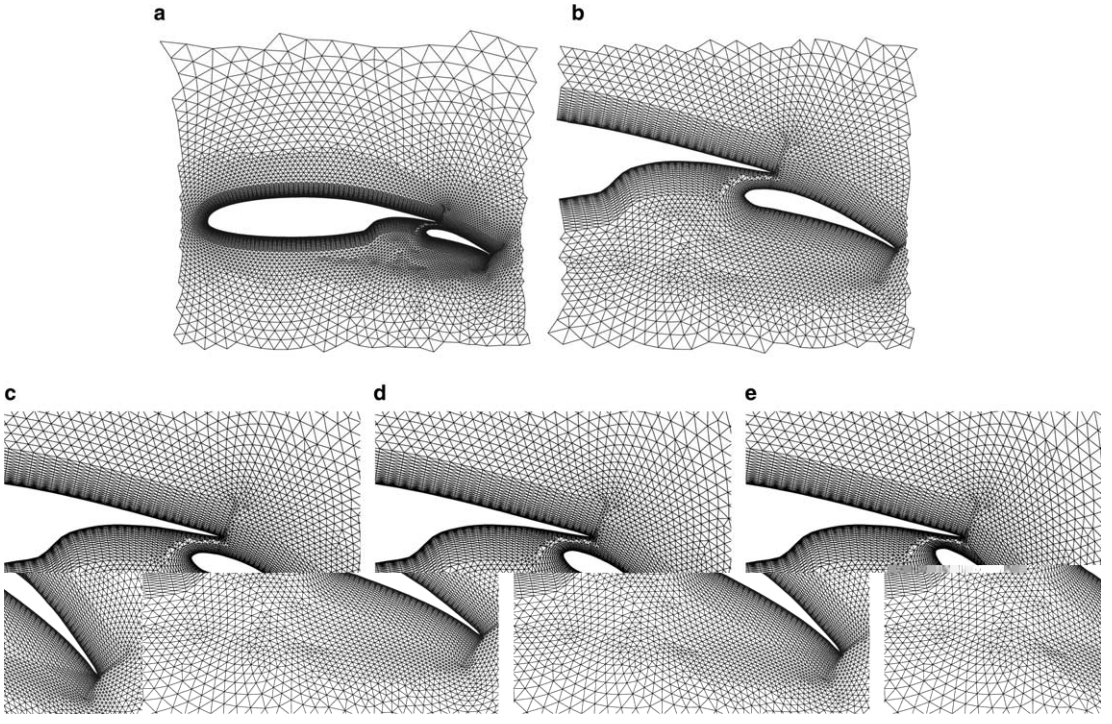


Fig. 10. Sequence of moving hybrid viscous meshes around a multi-element aerofoil.

O(1) necessitates an extremely fine mesh resolution in the wall normal direction, which can cause such problems as mesh line crossovers and slow convergence for methods based on spring analogy.

A hybrid viscous grid around a two-element aerofoil is shown in Fig. 10(a) and (b). Fig. 10(c)–(e) illustrates a sequence of the moving mesh using the proposed method by regenerating the Delaunay graph at each step. For this case, the maximum movement is limited by the gap between the two elements of the aerofoil. The quality of the hybrid grid for viscous flow simulation is maintained throughout the movement using the Delaunay graph mapping.

Figs. 11 and 12 show a local view of the Delaunay graph and corresponding local grid after the movement for this case. As can be seen, although the surface integrity is violated in the Delaunay graph, the moved grid ensures the boundary integrity and the quality of the viscous grid near the wall.

### 3.3. Movement of a deformable sphere in a cube

Dynamic grids for a deformable sphere moving in a fixed cube are studied, demonstrating the capability of the proposed method for three-dimensional cases. The sphere moves to the right and at the same time elongates in the vertical direction. The original hybrid grid is shown in Fig. 13(a). Fig. 13(b) shows the mesh

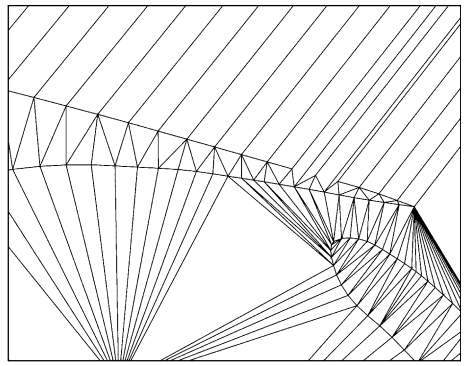

Fig. 11. Delaunay graph without boundary integrity.

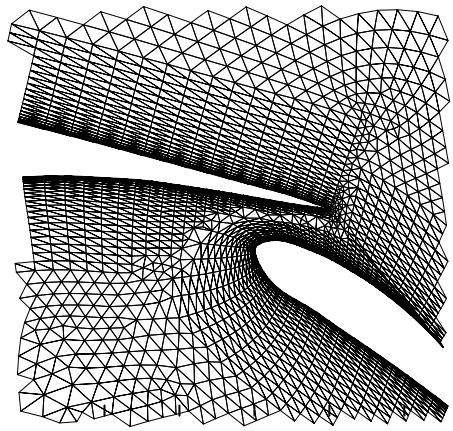

Fig. 12. The mapped mesh.

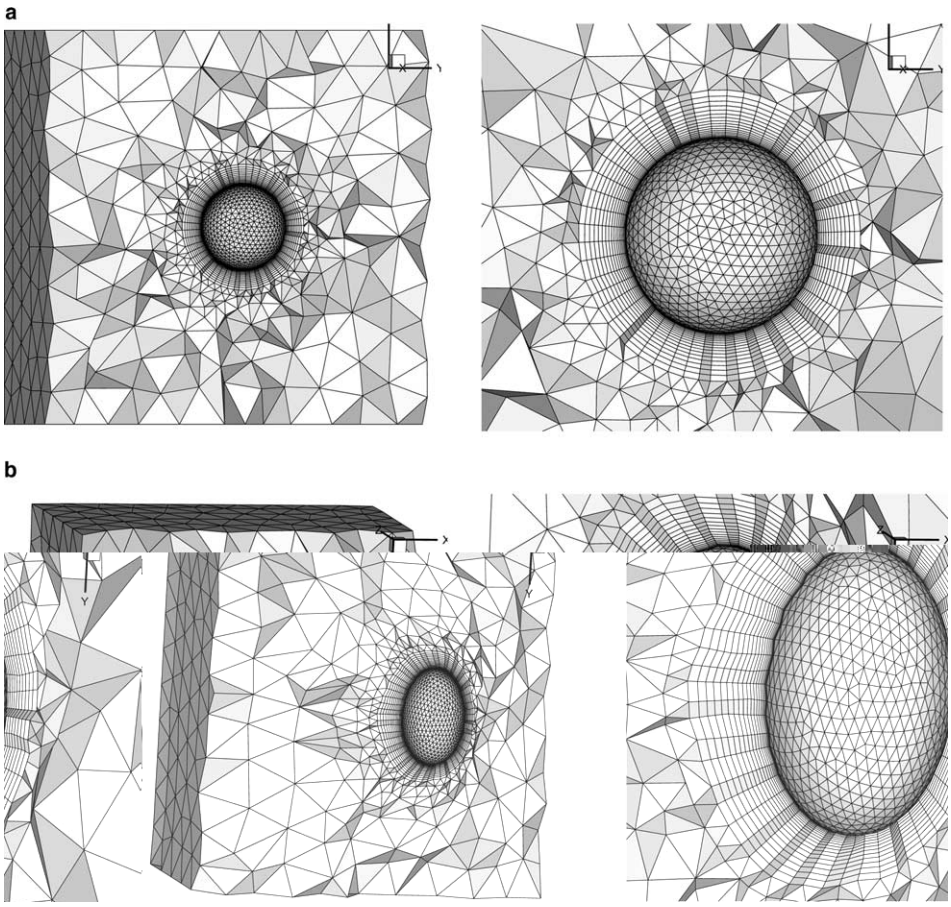


Fig. 13. Moving hybrid mesh around a sphere in a cube: (a) original hybrid grids, (b) moved hybrid grids ($dx = 8.0$).

after its movement and deformation. The Delauney graph is generated by all the surface pints and eight corner points on the cubical outer boundary.

### 3.4. Deformation of a flexible three-dimensional wing

For aero-elastic studies, it is crucial to have an efficient moving grid method for the fluid–structure interaction of flexible wing simulation. Limited oscillation (buffet) can lead to structure vibration and fatigue and large deformation may rise from the study of flutter boundaries. In this demonstration case, a three-dimensional wing is generated using the NACA0012 sections. A good quality structured viscous grid around the aerofoil is combined with the unstructured grid outside. A two-step deformation of the wing is specified, which is propagated from the surface grid to the field grid through the Delaunay graph mapping in two steps. Fig. 14(a) demonstrates the deformation process of the wing, which is twisted and bended. The twist angles are 20° and 40° and the wing tip displacements are 2 and 4 chord lengths, respectively. Fig. 14(b) demonstrates the original hybrid mesh for this wing without deformation. Fig. 14(c) and (d) illustrates the meshes after the first and second step of deformation, respectively.

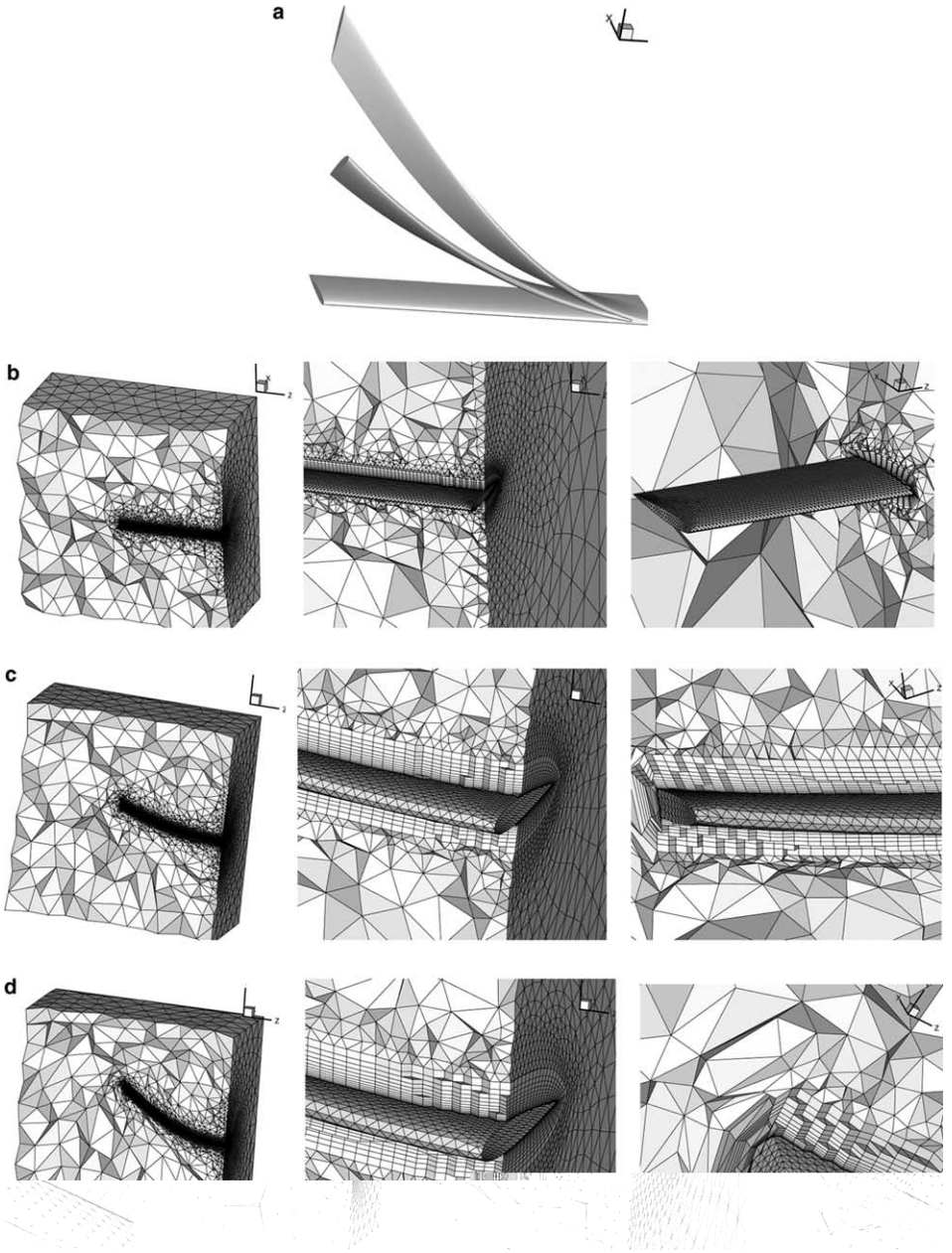


Fig. 14. Large deformation of a hybrid viscous grid around a flexible 3D wing: (a) wing twist and bending, (b) original hybrid grids, wing root and wing tip views, (c) deformed hybrid grids around wing with 20° and 2 chord length tip displacement, (d) deformed hybrid grids around wing with 40° twist and 4 chord length tip displacement.

### 3.5. Comparison with spring analogy for wing-body combination

For this last test case, the performance of the proposed dynamic grid method is compared with the spring analogy method regarding its performance in CPU time, memory requirement and grid quality.

The geometry is a wing-body combination available from [28]. A fully unstructured Euler grid is initially generated for the original geometry with 349,038 cells and 72,474 nodes. The commonly used spring analogy method of Batina [5] is used for this comparison although more sophisticated methods including torsional springs is necessary for viscous grids.

An artificial wing bending of 20° is specified. This large deformation is broken into a sequence of 20 steps of 1° bending so that the spring analogy method works. Note that for the Delaunay graph moving grid method, it is not necessary to break this deformation into 20 steps and the method works well for a single step move from 0° to 20°. However, for the comparison purpose, we have also broken the deformation into 1° steps in the Delaunay graph method.

Table 1 compares the CPU time and the memory required on a single 2.4 GHz Xeon processor using a Linux operating system for moving the original grid to its final position. For this comparison, the number of iterations in each step for the spring analogy is limited to 50. The table clearly shows that the Delaunay graph method is about an order faster than the spring analogy method for this large 20° deformation. In

Table 1
Comparison of spring analogy and Delauney graph on CPU and memory

| Dynamic grid method | Spring analogy | Delaunay graph |
|---|---|---|
| CPU (s) | 312 | 30 |
| Memory (Mb) | 72 | 114 |

Table 2
Breakdown of CPU for the Delaunay graph method

| | Generating graph | Locating element | Mapping points | Total |
|---|---|---|---|---|
| CPU (s) | 4 | 15 | 11 | 30 |

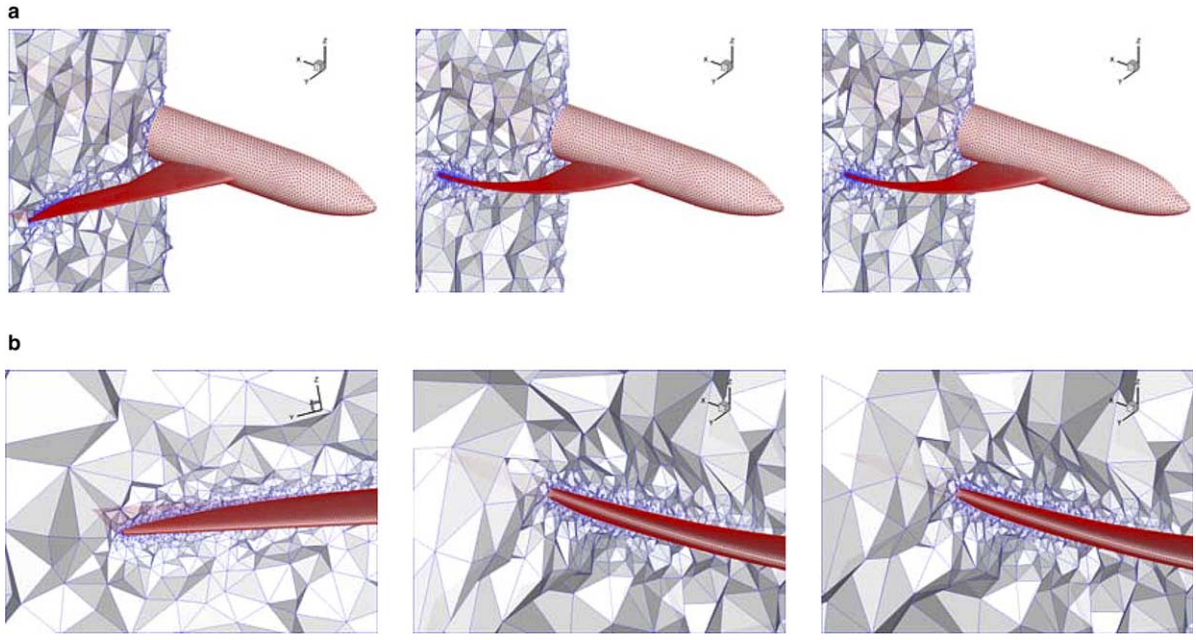


Fig. 15. Meshes before and after 20° deformation: (a) overall view, (b) local view, Original grid – Spring analog – Delaunay graph.

comparison, the Delaunay graph method requires more memory as compared with the spring analogy method, which is affordable considering the substantially improved speed. The extra memory requirement is primarily due to the storage of the Delaunay graph, the Delaunay element neighbour information and the cell volume ratio coefficients.

From Table 2, in the Delaunay graph method, about 50% percent of the time is spent on locating the Delaunay element for all the mesh points. To map the points, including the calculation of $e_i$ and the new coordinates of the points, takes about 37% of the time but the graph generation only 13%.

The final grids by the two different dynamic grid methods are compared in Fig. 15. The local grid quality around the wing tip is maintained after the large deformation for both methods in this case. The efficiency and robustness of the Delaunay graph method is expected to perform even better for hybrid viscous grids with very high aspect ratio cells near the surface, for example for the case in 3.4. This is because that the spring analogy will need (1) additional torsional springs or similar modifications for robustness, (2) smaller deformations to avoid mesh crossover, and (3) more iteration at each step of movement due to the high aspect ratio.

## 4. Conclusion

A fast dynamic grid deformation method based on the Delaunay graph is presented, which is simple to implement for arbitrary mesh topologies. In comparison with other dynamic grid generation methods based on the spring analogy deformation or grid regeneration, it is much more efficient, as it requires only non-iterative algebraic calculations.

The mapping procedure is robust without the common robustness problem for highly refined viscous flow grid, since the relative position of a grid point in the corresponding Delaunay element is maintained. No mesh crossing or negative area/volume can be generated in the grid relocation process. During the movement, the topology of the original grid is strictly carried over and the grid density distribution is maintained. Another advantage of the method is that it is independent of the grid topology and can be applied to arbitrary types of meshes or a mixture of different types. Grid type dependency is often a limitation of the spring analogy approaches and the TFI techniques for moving meshes.

The robustness of the method has been shown through a series of test cases including inviscid and viscous flow grids with large deformations. The test cases show that these desirable grid qualities in the original grid are carried over in the moving meshes. For the test cases, the viscous grids after the movement are satisfactory without the requirement of specific control over the mesh angles. The efficiency of the method has been demonstrated in the wing-body test case for a large deformation in comparison with a standard spring analogy method. An order of magnitude improvement in CPU has been achieved, accompanied by a modest increase in memory.

The method is suited for local movement with large deformation, oscillatory motions resulting from unsteady aerodynamics, fluid/structure interaction and design optimisation. For long distance relative motions, such as fuel tank or weapon release, or ejection seat movement, the efficiency of the Delaunay graph mapping method can be exploited by incorporating local grid enrichment and coarsening.

## References

[1] C. Byun, G.P. Guruswamy, A parallel, multi-block, moving grid method for aeroelastic applications on full aircraft, AIAA Paper 98-4782, September 1998.

[2] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, D. Saunders, Aerodynamics shape optimization of complex aircraft configurations via an adjoint formulation, AIAA Paper 96-0094, January 1996.

[3] W.T. Jones, J. Samareh-Abolhassani, A grid generation system for multi-disciplinary design optimization, AIAA Paper 95-1689, June 1995.
[4] A.L. Gaitonde, S.P. Fiddes, Three-dimensional moving mesh method for the calculation of unsteady transonic flows, Aeronautical J. 99 (984) (1995) 150–160.
[5] J.T. Batina, Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis, AIAA J. 29 (3) (1991) 327–333.
[6] V. Venkatakrishnan, D.J. Mavriplis, Implicit method for the computation of unsteady flows on unstructured grids, J. Comp. Phys. 127 (2) (1996) 380–397.
[7] E.J. Nielsen, W.K. Anderson, Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations, AIAA J. 37 (11) (1999) 1411–1419.
[8] C. Farhat, C. Degand, B. Koobus, M. Lesoinne, Torsional springs for two-dimensional dynamic unstructured fluid meshes, Comput. Methods Appl. Mech. Engrg. 163 (1998) 231–245.
[9] F.J. Blom, Considerations on the spring analogy, Int. J. Numer. Meth. Fluids 32 (2000) 647–668.
[10] M. Murayama, K. Nakahashi, K. Matsushima, Unstructured dynamic mesh for large movement and deformation, AIAA Paper 2002-0122, 2002.
[11] P.C. Chen, L.R. Hill, A three-dimensional boundary element method for CFD/CSD grid interfacing, AIAA Paper 99-1213, April 1999.
[12] A.A. Johnson, T.E. Tezduyar, Simulation of multiple spheres falling in a liquid-filled tube, Comp. Meth. Appl. Mech. Eng. 134 (4) (1996) 351–373.
[13] T. Baker, Mesh deformation and reconstruction for time evolving domains, AIAA Paper 2001-2535, June 2001.
[14] E.J. Nielsen, W.K. Anderson, Resent improvements in aerodynamic design optimization on unstructured meshes, AIAA J. 40 (6) (2002) 1153–1163.
[15] P. Fast, W.D. Henshaw, Time accurate simulation of viscous flow around deformation bodies using overset grids, AIAA 2001-2604, June 2001.
[16] L.P. Zhang, Z.J. Wang, A block LU-SGS implicit dual time-stepping algorithm for hybrid dynamic meshes, Comp. Fluids 33 (7) (2004) 891–916.
[17] A. Bowyer, Computing Dirichlet tessalations, Comput. J. 24 (2) (1981) 162–166.
[18] D.F. Watson, Computing the *n*-dimensional Delaunay tessalation with application to Voronoi polytopes, Comput. J. 24 (2) (1981) 167–171.
[19] D.J. Mavriplis, Unstructured grid techniques, Ann. Rev. Fluid Mech. 29 (1997) 473–514.
[20] L.P. Chew, Constrained Delaunay triangulations, Algorithmica 4 (1989) 97–108.
[21] Baker TJ. 1991. Unstructured meshes and surface fidelity for complex shapes. Proc. AIAA CFD Conf., 10th, Honolulu, AIAA Pap. 91-1591-CP, pp. 714–25.
[22] P.L. George, F. Hecht, E. Saltel, Automatic mesh generator with specified boundary, Comput. Methods Appl. Mech. Eng. 33 (1991) 975–995.
[23] N.P. Weatherill, O. Hassan, D.L. Marcum, Calculation of steady compressible flowfields with the finite-element method, AIAA Paper 93-0341.1993.
[24] L. Guibas, J. Stolfi, Randomized incremental construction of Delaunay and Voronoi diagrams, Algorithmica 7 (1992) 381–413.
[25] L. Devroye, E. Mucke, B. Zhu, A note on point location of Delaunay triangulation of random points, Algorithmica 22 (4) (1998) 477–482.
[26] D.W. Pepper, J.C. Heinrich, The finite element method: basic concepts and applications, Hemisphere Publishing Corporation, 1992.
[27] M. Leatham, S. Stokes, J.A. Shaw, J. Cooper, J. Appa, T.A. Blaylock, Automatic mesh generation for rapid-response Navier–Stokes calculations, AIAA Paper 2000-2247, June 2000.
[28] G. Redecker, DLR-F4 wing body configuration, A selection of experimental test cases for validation of CFD codes, AGARD AR-303, Paper B4,1994.