SAKARYA ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ DERGİSİ
SAKARYA UNIVERSITY JOURNAL OF SCIENCE

e-ISSN: 2147-835X
Dergi sayfası: http://dergipark.gov.tr/saufenbilder

Geliş/Received
12.12.2016
Kabul/Accepted
20.04.2017

Doi
10.16984/saufenbilder.309643

Mesh stitching method for moving and stationary bodies

Erdal Yılmaz[*1]

ABSTRACT

An efficient and versatile mesh method is suggested and implemented for problems involving individual overlapping mesh blocks around moving objects. The overlapped mesh blocks are combined using a stitching stripe, which considers smooth transition of the mesh element size between the blocks. Interpolation of the flow values between the mesh blocks is not needed. This is considered as a relative advantage of this method compared to the overset mesh method. Unsteady flow around an oscillating body test case is used to demonstrate applicability of this method. Other test cases include hypothetical multiple flapping airfoils in relative motion and store separation from a store bay.

Keywords: unstructured mesh, mesh stitching, moving mesh, grid adaptation, grid stitching

Hareketli ve sabit cisimler için ağ yama yöntemi

ÖZ

Bu çalışmada hareketli nesneler etrafında birbiri ile örtüşen çözüm ağlarını içeren problemler için az çaba ile üretilebilen ve yaygın olabilecek bir yöntem önerilmiş ve uygulanmıştır. Örtüşen çözüm ağ blokları dikiş atma yöntemi kullanılarak, ağ hücre büyüklerini dikkate almak suretiyle yumuşak yama bandı oluşturularak birleştirilmiştir. Çözüm değerlerini aktarırken interpolasyon yapmaya ihtiyaç duyulmamaktadır. Bu sebeple, mevcut yöntem benzer chimera yöntemine göreceli bir avantaj sağlamaktadır. Yunuslama hareketi yapan bir kanat profile etrafındaki akış, bu yöntemi göstermek için kullanılmıştır. Diğer test problemleri ise bu çalışmaya özel olarak deneme amacıyla oluşturulmuş birbirine göre hareket eden çoklu kanat profile ile uçaktan ayrılan bir nesneye benzetim yapmaktadır.

Anahtar Kelimeler: yapısal olmayan çözüm ağı, ağ dikme, ağ uyumlandırma, hareketli ağlar, ağ adaptasyonu

[*] TÜBİTAK-Space Tehnology Research Institute / TÜBİTAK Uzay Teknolojileri Araştırma Enstitüsü
1 Adjunct Faculty, Aerospace Engineering Dept, METU/ Ziyaretçi Öğretim Üyesi, Havacılık Müh. Blm. ODTÜ

1. INTRODUCTION

When multiple mesh blocks are used to define computational domain composed of several static and moving objects, an approach to combine mesh and solution for these blocks are needed. In many engineering systems, moving components such as pintles, balls, disks, and flapper valves, are essential parts of the systems to be analyzed. In aerospace applications, oscillating and flapping wings, propeller analysis, store separation, and launch systems etc. all involves moving components. Therefore, flow solvers and mesh methods must have capability to analyze carefully the interaction of multiple static and moving objects.

Among several methods, multi-block mesh approach assumes matching interface boundaries between the mesh blocks. With tens of the individual mesh blocks in whole domain, forcing matching of the interface boundary for the mesh blocks is a major drawback of this method. Though non-matching block interface can also be employed, it comes with additional computational cost due to the interpolation of flow values at the interface between the blocks.

Dynamic/moving mesh methods involve relocation of grid points at every time steps in response to motion of the objects. Several algorithms such as based on the spring analogy or elasticity theory to move/deform the mesh have been implemented in the past [1] [2] [3] [4]. Mesh deformation is very costly if a single block grid is used to define the complete domain. Some approaches use different mesh blocks for the moving components to limit mesh movements in the computational domain. However, after certain time, stretched grids loose accuracy especially in the flux calculations. Regeneration of the grids is essential in such cases and can be comparatively time consuming.

Overset or Chimera grid approach allows overlapping of the mesh blocks, which are widely used in fluid-structure interaction problems [5], and communicates interpolated flow values between mesh blocks at the overlapped regions of the mesh blocks. While it is an established method for both stationary and moving objects and in use for a couple of decades, major drawback is loss of accuracy due to the interpolation and cost of overall mesh processes. The main concern in interpolation procedure is whether conservation of the fluxes is satisfied. Some of the overset grid approaches uses non-conservative tri-linear interpolation schemes [6]. On the other hand, conservative interpolation approaches proposed for patched surfaces and arbitrarily overlapped regions [7] [8] are difficult to apply in three dimensions. To overcome those difficulties another approach [9] has been proposed for hybrid grids, hexagonal and tetrahedral grids, by eliminating the interpolation procedure at the cost of filling holes or gaps at interface region of two mesh blocks with new grids. That approach relies on two existing CFD codes, combination of the Overflow and USM3D flow solvers [10] [11] to get best out of these two codes with a new flux calculation algorithm on the faces, which are not initially designed for such applications. A powerful toolset for overset grid, Suggar and Dirtlib [12], has been implemented to use with different solvers [13]

Embedded or immersed mesh is constructed by overlaying the object of interest over a standard Cartesian mesh [14] [15] [16] [17]. It is first applied in artificial heart flow modeling [18] [19]. It has been used in geophysical flow applications as well [20] [21]. It is somewhat similar to overset mesh methods. Mesh elements that intersect with the boundary are identified and appropriate boundary conditions are imposed on the cell stencil specifically modified for the immersed bodies. This method also requires interpolation for the surface definition over the Cartesian grids.

Outside Computational Fluid Dynamics, (CFD) mesh practice, mesh morphing is another approach to smoothly combine surfaces of meta-elements. It is commonly used in computer solid modeling and special effects designs. Meta-elements automatically fuse together when placed in close proximity, thereby generating a smoothly connected polygon mesh, called Metamesh [22] [23]. However, CFD meshes have different natures and purposes than existing mesh-morphing. The CFD meshes seek merging of the volume meshes as well as the surface if collapses are allowed. In fusing process, the new vertices might be introduced to meet objective of the smooth transitions. When the overlap of two mesh blocks occurs, CFD requires finest mesh elements in the overlapping domain to get better resolution and solution quality.

Different than traditional approaches such as the overset and embedded meshes, this paper considers a better approach, as far as interpolation accuracy concerned, called *mesh stitching method* for flows around moving objects. It can be applicable to several disciplines, not just CFD. The mesh stitching in general is not a newly introduced concept, yet to be studied in depth for different applications. Weber et al. [24] implemented a kind of stitching grids for multiple adaptive grid patches to extract iso-surfaces from a given set of static grid data, which includes multiple refinement level. In another study, an unstructured triangulation to stitch between the boundary layers and background mesh has been proposed [25]. However, the mesh stitching in this paper has started unaware of those studies, indeed, this study differs from them. The mesh stitching approach in this paper is to combine two or more mesh blocks using seamless stitching mesh. The stitching mesh in this study is based on optimum mesh element size of the overlapping mesh blocks. The proposed approach facilitates integration of the overlapping mesh blocks without going through the interpolation of the flow solutions between the mesh blocks as in the case of the overset mesh method or generating new vertices to force smooth transition between mesh blocks as in the metamesh approach. In the following section, detail of the mesh stitching is introduced. After that, it follows with the flow solver algorithm used in this study. Finally, applications of the stitching method are demonstrated using three test cases; oscillating single NACA 0012 airfoil, multiple airfoils configurations, and stores in motion.

2. MESH STITCHING METHOD

2.1. Overview

For the sake of convenience, in further sections of this paper, some definitions are made to present this approach better. Two kinds of mesh blocks are defined: 1) minor mesh block, which is attached to moving object(s), and 2) background mesh block, which covers the complete flow domain. The background mesh is a single block mesh, which is stationary, while the minor mesh blocks may be in relative motion within the background mesh domain. To put the stitching concept in a simpler form, a one-dimensional representation is presented in
Figure 1. The first step is to identify overlapping nodes of the mesh blocks. For better mesh resolution, finest mesh elements of all overlapping mesh blocks are maintained to construct the final mesh. Then, coarser mesh in the overlapped region of all blocks will be removed by cutting them off from the computational domain, hence generating a gap between the mesh blocks. This follows with triangulation of the gap region. The mesh blocks will be fused as seamlessly possible using the stitching mesh generated for this gap region. The final mesh is a unified single block mesh that will be used in the flow solution process. In this method, four items contribute the computational cost: identification of the overlap regions, cutting the overlapped region off, generating element connectivity in the gap region, and then fusing the mesh blocks with the stitching grid.

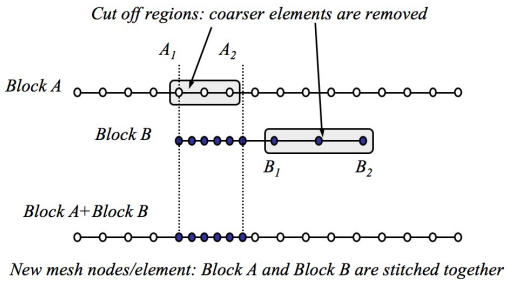


Figure 1. One-dimensional representation of the stitching method for two mesh blocks. Block A represents the *background mesh and* Block B represents the *minor mesh*.

2.2. Mesh Stitching Steps

After unstructured mesh blocks for the individual objects and background flow domain are generated, following steps are used in constructing the stitching mesh in this paper:

Step 1: Move the mesh blocks with the prescribed motion to the next position in time. First check is whether there is any solid-to-solid collision or not. Moving objects have all closed solid boundaries most of the time except zero-thickness shells. A criterion called *containment test*, detail is given in Step 3, is applied for any mesh point of a block that overlaps with given closed boundary. Solid-to-solid collision is determined using the containment test. If there is solid-to-solid collision, the stitching process stops. At this time bounce back is not implemented to handle the collision.

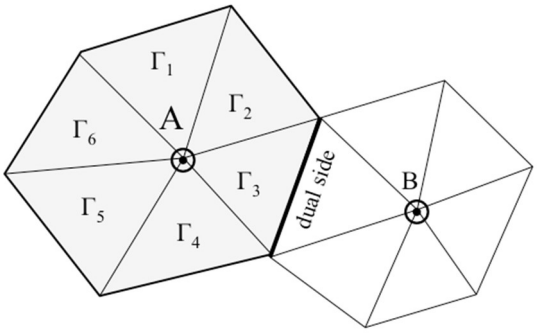


Figure 2. Finite control volume at node A and B with shared dual side referring to two elements on both sides.

Step 2: Calculate mesh spacing for the elements of all mesh blocks. The flow solver used in this study is based on the cell-vertex based finite volume discretization. Therefore, a finite control volume around a node is constructed by all triangular elements connected at that node as given in Figure 1. In the grid data structure, shared edges between two triangles forms dual edges, which address the control volume points on both sides, points A and B as in Figure 2. Corresponding cell spacing at a node is calculated as average of areas of the cells connected to that node using:

$$\bar{\Gamma}_A = \frac{1}{Nelem_A} \sum_{i=1}^{Nelem_A} \Gamma_i \qquad (1)$$

where, at node A, Γ_i is area of individual triangular elements connected to node A, $Nelem_A$ is total number triangular elements connected to node A, and $\bar{\Gamma}_A$ is average of all triangular elements connected to node A. Note that total area at node A is constructed by summing up areas of all triangular elements around node A , Ω_A. Area of a triangle is calculated using the line integral over three edges of the triangles, rather than the Heron's formula, as follows:

$$\Gamma_i = \frac{1}{2} \sum_{i=1}^{3} \{(x_a + x_b)(y_a - y_b)\}_i \qquad (2)$$

where, x_a, x_b, and y_a, y_b are coordinates of terminal points $\{a,b\}$ of edges of triangle, respectively.

Step 3: Identify nodes of the *background mesh* overlapped with the *minor mesh* and vice versa. To determine status of a given node whether it is in or out of an irregular area formed by a closed curve, a simple search criteria, *containment test*, is used. Figure 3 gives description of the containment test for a convex/concave shaped object. In Figure 3, node A is inside the object while node B is outside. Node A intersects the boundary of the object in one way either (+x or –x-direction) at odd numbers while node B intersects the object in either way (+y or –y) direction at even numbers.

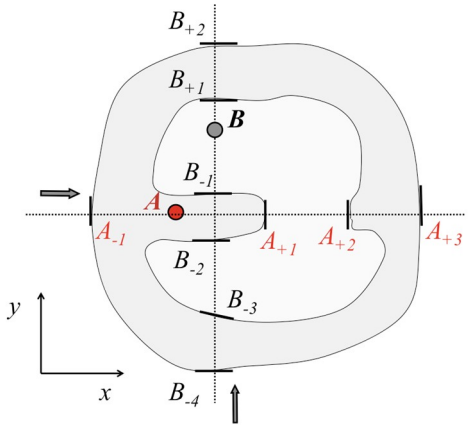


Figure 3. Description of the containment test for in-and-out criteria of a node for a convex/concave shaped object.

This process can be decomposed into following steps:

I. Sweep along x- (or y-) direction and search for the intersection of x- (or y-) coordinate of a given point with mesh edges on the surface of given closed area.
II. Count number of intersection in either + (right side) or – (left side) of the point along either x-axis (or y-axis).
III. If the number of intersections along x-direction (or y-direction) is odd, then the given point is inside the closed area. If it is even, then the point is outside of the area. Point A in Figure 3 is inside the closed curve because the intersection count is 1 and 3 along – and + direction of the x-axis, respectively. Similarly, Point B in Figure 3 is out of because intersection count is even, 2 and 4 along + and y-direction, respectively. Note that if the inner curve in Figure 3 is considered as closed object, the confinement test would yield point A as out and point B as in.

Step 4: Construct a front where element size of the minor mesh and the background mesh (or any two overlapping meshes of different blocks) are the same or within a given tolerance. In other words, if the mesh spacing of a control volume at a node in a one of the overlapping mesh blocks is greater than the mesh spacing of an element on the other overlapping block, where the node of the other block lies in, then, that node is cut out. That is if $s_A \geq s_{123}$, where s_A is mesh spacing of node A in

one block and s_{123} is mesh spacing of an overlapping triangle with edges 1, 2, and 3 in the other block, where node A of the other block lies in. The front is at inner and outer boundaries of the cut-out mesh blocks. Note that front line can be concave or/and convex. Two fronts, inner and outer boundary of the stitching stripe, form a narrow stripe, where no node is maintained in between. The stripe width is set around one-mesh size depth. However, sometimes it may be close to twice of an element size in its neighborhood. A criterion is set so that this stripe can not be thinner than a given percentage of the element size in the vicinity. The mesh spacing calculated in Step 2 is used here. The process in this step ensures that when two mesh blocks overlap, minimum mesh elements of both blocks are used to form single block mesh for the flow solution.

Step 5: All nodes of the background mesh that are inside the front line will be cut out, in a similar way but conversely for the minor mesh all the nodes that are outside of the front line are cut out.

Step 6: Once the nodes at the inner and outer fronts of the *minor mesh* and *background mesh*, are identified then triangulation process of these nodes takes place. In this paper, Delaunay triangulation algorithm is adopted [26].

Step 7: Triangulation generates all possible elements regardless of concave or convex nature of the stitching stripe. Therefore, all triangular elements outside of the stripe should be cleaned out for final stitching. A neighborhood searching method is applied for this purpose. As summarized below items and also depicted in Figure 4.

.

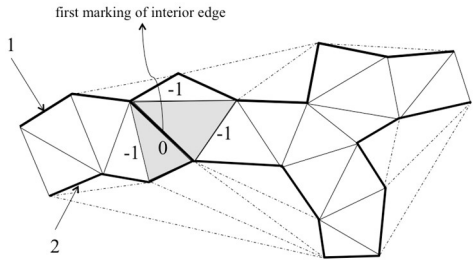


Figure 4. Cleanup of unused Delaunay triangles outside the stitching stripe.

Mark front edges of overlapping blocks with their block numbers, such as “1”, “2”, etc. All other edges will be marked as “-1” initially.

Find an edge in-between the inner and outer front lines, an edge that lies inside the stitching stripe, then mark it as “0”. This edge will be starting point to advance the search.

Loop over dual edges that are marked as “0” and check edges of the triangles on both sides of the dual edge, whether the edges of the triangles are marked as “-1”. If so, change the marker of that edge from “-1” to “0”. Looping continues several times until no such edge is found. At the end of this looping, only edges outside the stitching stripe will maintain “-1” marker.

Delete all triangular elements, which have edge marker as “-1”. Only elements inside the stitching stripe remain.

Step 8: Find pair edges on both sides of the stitching stripe, which match all edges of the *minor mesh* and the *background mesh,* respectively. This is also known as *edge-to-edge tessellation* of two mesh domains, which is sharing of full edges with the adjacent mesh blocks. Figure 5 shows two mesh blocks and stitch mesh with edge-to-edge pairing.

Step 9: Update connectivity of the elements from all three kinds of blocks: minor, major, and stitching stripe to merge all into single connectivity data set.

Finally, the merged mesh will be ready for the flow solution. Above process will be repeated every time step after objects are moved to their new positions.

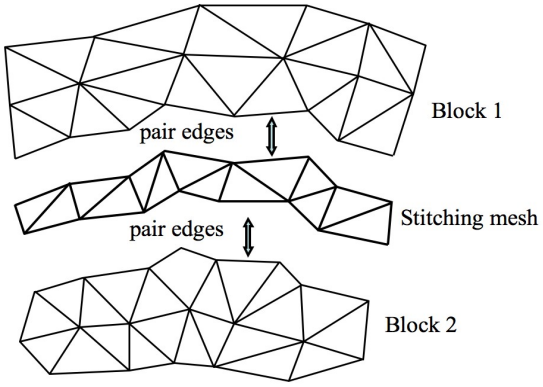


Figure 5. Edge-to-edge tessellation of two mesh blocks using stitching stripe.

3. FLOW SOLVER

Flow solver used in this paper is based on two-dimensional finite volume formulation at cell-vertex with explicit time stepping [27]. Conservations of mass, momentum, and energy lead to the following equation for two-dimensional, compressible, and inviscid flows:

$$\frac{\partial}{\partial t}\int U\,dV + \oint_{\Omega}\left(F dy - G dx\right) = 0 \qquad (3)$$

where U is called solution vector and represents the conserved quantities, density, velocity and internal energy. F and G are x and y Cartesian components of the flux vector of these quantities, respectively. Hereafter F and G are used for the convective flux terms only, which are functions of density, velocity, and internal energy. When the integral form of the Euler equation (Equation 3) is applied to a finite control volume Ω_K, one obtains the coupled ordinary differential equations given in Equation 6.

$$\frac{\partial}{\partial t}\left(U_K \Omega_K\right) + \sum_{i}^{Nedge}\left(F_i \Delta y_i - G_i \Delta x_i\right) = 0 \qquad (4)$$

where, at node K, Ω_K is the surface area of the control volume associated with node K; U_K is the solution vector; F_i and G_i are the Cartesian components of the flux vector on edge (face in 3D) i of the control volume; *Nedge* is the total number of edges which surround the control volume; Δx_i and Δy_i are the increments of x and y along edge i, respectively. The fluxes F_i and G_i are functions of flow variables at the neighboring points, and thus the system of equation is coupled in space. Further details of the flow solution can be found at [27].

4. RESULTS

The mesh stitching method was applied to three problems to demonstrate capabilities and understand challenges. First test case is an airfoil oscillating with small amplitude. This test case has experimental flow solution to compare with. Though it is not the intension of this study to demonstrate flow solver aspect, for the completeness of this study, the stitching algorithm is run together with the flow solver for the unsteady oscillation. Next two examples demonstrate hypothetical cases without any flow solution. The second test case uses the same airfoil but in biplane and tandem configuration to demonstrate stitching of three objects separated with high angles in relative motion. Third test case is a store separation problem with two stores released from a bay using a prescribed motion. All runs were performed on a high-end laptop computer since computational time was not an issue for the size of problems demonstrated in this study.

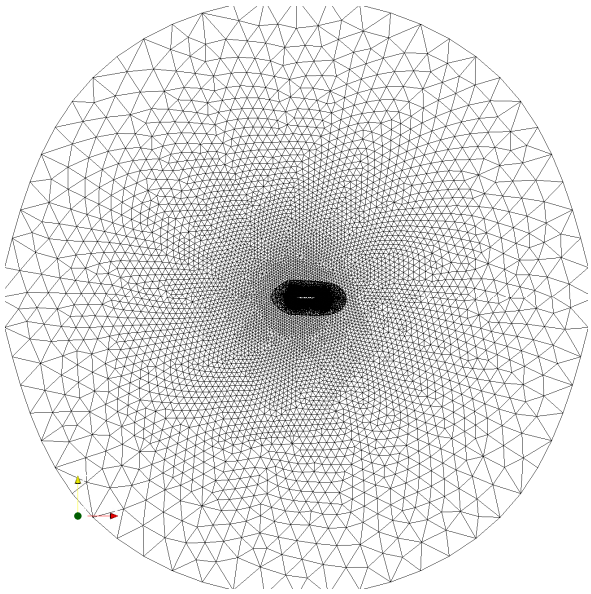
4.1. Case1: Oscillating Airfoil With Flow Solution

Flow field validation of the present method has been demonstrated using a commonly studied oscillating airfoil. The test case chosen is NACA 0012 symmetric airfoil at Mach number M_∞ =0.755, with mean free-stream angle of attack of α_m=0.016 degrees, with pitching amplitude of α_p =2.51 degrees. Airfoil goes through pitching mode of oscillation given in Equation 5.

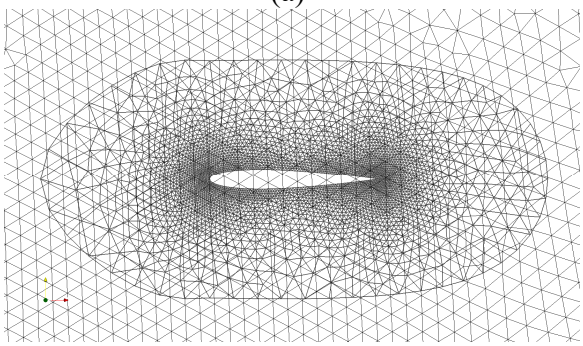
$$\alpha(t) = \alpha_m + \alpha_p \sin\left(\frac{2kV_\infty}{c}t\right) \qquad (5)$$

where k is defined as reduced frequency, V_∞ is free-stream velocity number, t is time, and c is airfoil chord length.

Figure 6 gives the overlapped *minor and background mesh* blocks for the airfoil at zero degrees angles of attack. The background mesh comprised of 10,602 triangular elements while airfoil mesh has 6,823. Note that mesh element size for the *background mesh* is almost constant, however getting denser where it overlaps, while the *minor mesh*, mesh around the airfoil, has varying mesh distribution, dense in the vicinity of the airfoil surface and gradually coarsen away from the airfoil. Once the *confinement test* marks the overlapped nodes and elements, the cut-out process identifies nodes at the cut-out fronts comprising the *stitching stripe* for triangulation.



(a)



(b)

Figure 6. Two overlapping mesh blocks for the NACA 0012 airfoil: minor mesh around the airfoil and background mesh to cover whole domain. (a) Overall mesh domain and (b) Close up view.

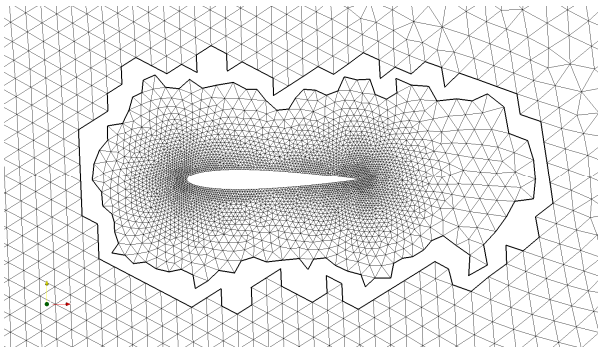


Figure 7. Mesh after cutting out of overlapping elements for the NACA 0012 airfoil.

Figure 7 shows the mesh after the cut-out process for the NACA 0012 airfoil. Nodes from both minor and background mesh blocks are cut out. Boundary of the cut-out section is set where element sizes of the overlapping mesh blocks are almost equal. The width of this stripe is about 1/3 to 2 times of the local element size. To guarantee a continuous stitching stripe, the width should not be very narrow. Since the stitching region is far from the airfoil, some mesh stretching can be tolerated inside the stitching stripe. Figure 8 shows the triangulated stitching stripe. The Delaunay triangulation generates elements outside the stitching stripe too. Therefore, these extra elements need to be cleaned before the stitching process is applied. Final merged mesh, where the flow solution takes place, with the stitching mesh highlighted is presented in Figure 9 . Note that there is not any new node added into the domain. Some of the elements are a little more stretched than original elements nearby, which are far from the solid wall and can be easily tolerated by the finite volume flow solvers.

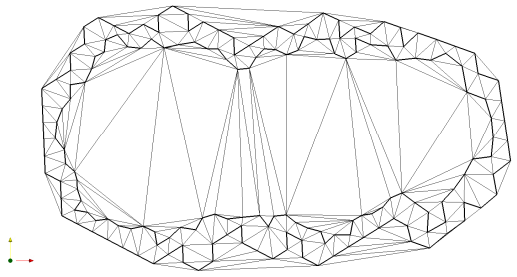


Figure 8. Delaunay triangulation of nodes at the cut-out boundary, bold edges, to generate the stitching mesh. All elements outside the hallow stitching stripe, some are in the inner closed area, are deleted.

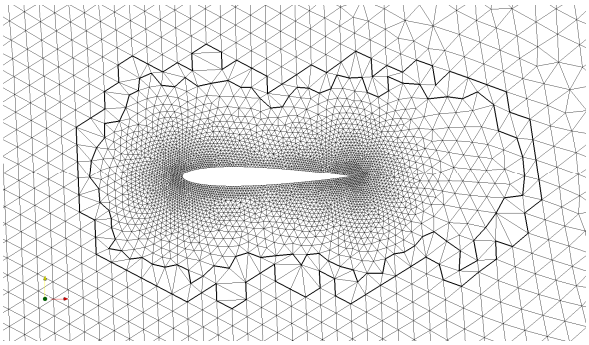


Figure 9. Final mesh after the cut-out region filled with the stitching mesh, highlighted edges of at the boundary.

For the flow solution, pitching oscillation starts with a converged initial flow solution at angle of attack equal to α_m=0.016. Therefore, initial solution has been obtained by local time stepping to have faster convergence. For this explicit solver, 3000 local time steps have been used for the initial solution. Then the pitching motion starts. The time step used for the unsteady solution is determined using the time step for the maximum amplitude of the pitching and the time step for the Courant number for the particular mesh used. Therefore, minimum of those two time steps is used to advance the solution in time. The time step from

the pitching motion depends on how many steps will be taken for one cycle. Current solution assumes about 4000 time steps for one cycle of pitching.

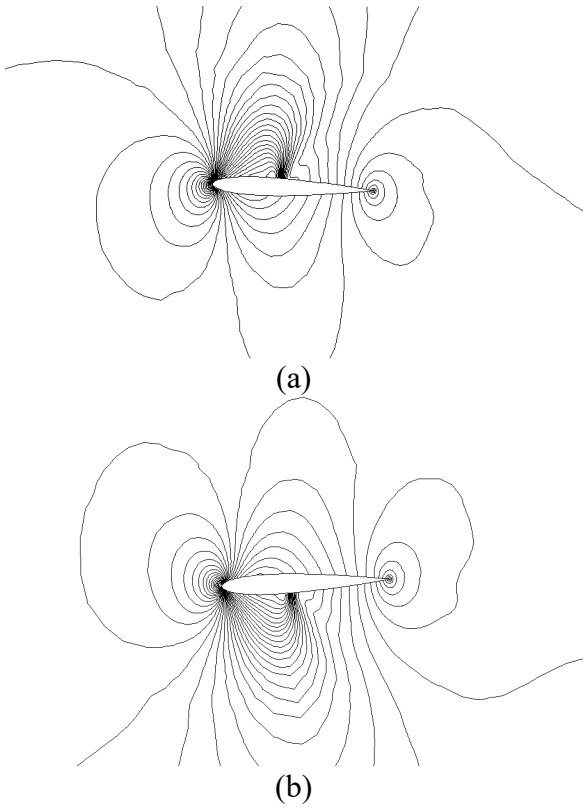


Figure 10. Flow solution at the end of third pitching oscillation cycle for the NACA 0012 airfoil for Mach number 0.755. Pressure contours at a) +2.51 degrees of angle of attack, b) -2.51 degrees of angle of attack.

Figure 10 demonstrates pressure contours at maximum and minimum angles of attacks, respectively. Pressure contours show continuity in the overall solution. The flow is transonic, therefore, shock waves are formed in the upper and lower surfaces at the maximum and minimum angles of attacks, respectively. To capture shock wave accurately it is necessary to use finer mesh in the vicinity of shock positions. The unsteady flow solution is run for three cycles of oscillations.

Figure 11 shows lift curve comparison with the experiment [28] for three cycles. There is a minor shift from the experimental result. This might be associated with the nature of the Euler solution as well as mesh element size on the airfoil where the shock wave occurs. However, main objective of this paper is to demonstrate the stitching concept rather than inquiry of the flow solver accuracy. History of the lift coefficient is given in Figure 11. Minimum and maximum values of the lift coefficient converges almost after one cycle.

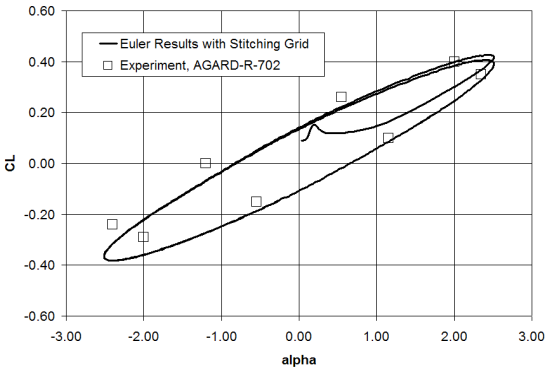
4.2. Case 2: Multiple airfoil configurations

Next, the stitching method is applied to three airfoils in relative motion such as in flapping airfoil problems. The same airfoil mesh in the previous case is duplicated, with one placed at half chord length apart in biplane configuration and the other at one and half chord upstream in tandem configuration on the same background mesh. Imposed motion on the airfoils in biplane configuration are counter clockwise and clockwise rotations around the leading edge of the upper and lower airfoils, respectively. The stitching program was run for 180 degrees with one-degree increment continuously. Stitched meshes at different positions are presented in Figure 12. The stitching method overall generates fairly acceptable meshes. The mesh cut-off process between the two airfoils rarely generates meshes size of more than one element size of the neighboring cells. This is most likely due to the cut-out decision applied individually on each block in sequential order. Another reason would be due to cut-out decision applied at nodes first instead of edges. When a node is cut-out, it naturally cuts all the elements connected to a node off, hence creating a bigger hole than cutting out a single element. Since solid-to-solid collision is not implemented yet, airfoils are not let to rotate all the way to collide. However, a stopping criterion is set to end the stitching process if solid collision occurs.

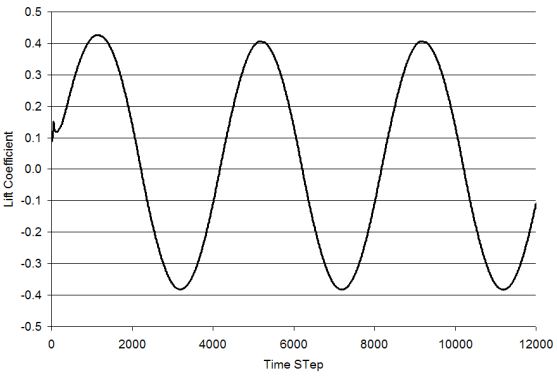
4.3. Case 3: Store separation

Last example is a hypothetical store separation problem. Two stores are placed in close proximity in a rectangular bay of an airplane and then dropped along a prescribed path. Special care should be taken when solid-to-solid collision occurs, which is not covered in this study. This example also shows how the stitching method handles objects when they are placed closely and moves through comparatively coarse background mesh. Each store comprised of 3760 triangles while the background mesh composed of 10,373 triangular elements. Imposed motions on the stores are translation along the vertical axis and one-degree rotation about their tails every motion step. Original mesh blocks for two stores, background mesh, and stitched meshes at different time instances are given in Figure 13. Outer mesh boundary of one of the stores extends out of the

background domain at the beginning. In the stitching process, this non-overlapping section of the store mesh is cut out as well. The stitching mesh boundary is given in bold lines. The stitching mesh provides fairy smooth transition between the blocks when they are at still in the store bay since local mesh size variation is not too high for all blocks. The stitching mesh is also shown when the stores are off the bay. Since the mesh elements get coarser far from the bay, stitching of fine store mesh with the coarse background mesh generates stretched mesh elements, which is unavoidable without local mesh refinement procedure, which is not the purpose of this study.

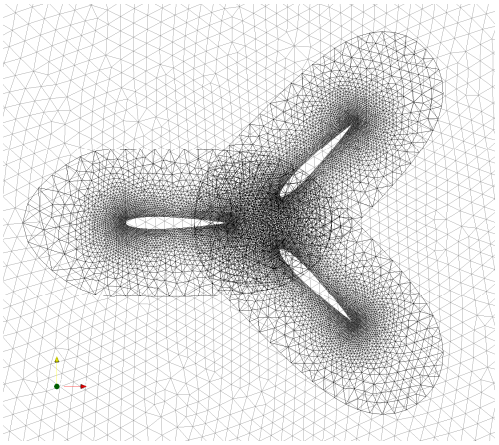


(a)

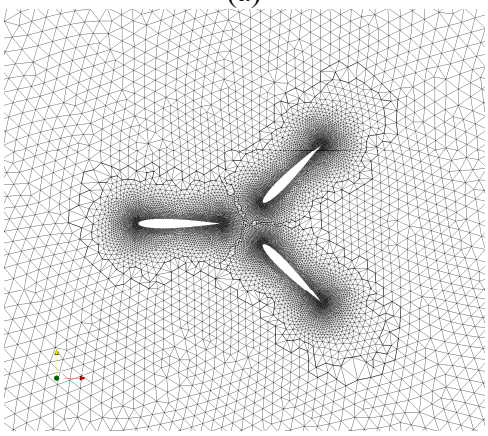


(b)

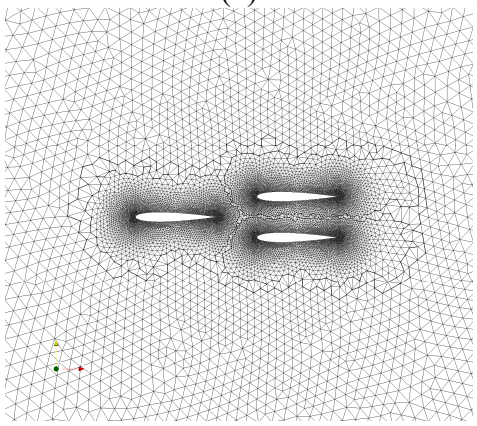
Figure 11. Lift coefficient vs angle of attack comparison of pitching NACA 0012 airfoil with the experiment [21] for Mach number of 0.755 and lift coefficient history for three cycle of pitching oscillation.



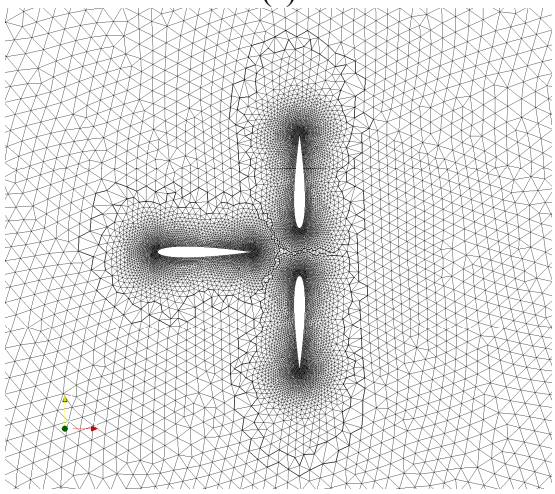
(a)



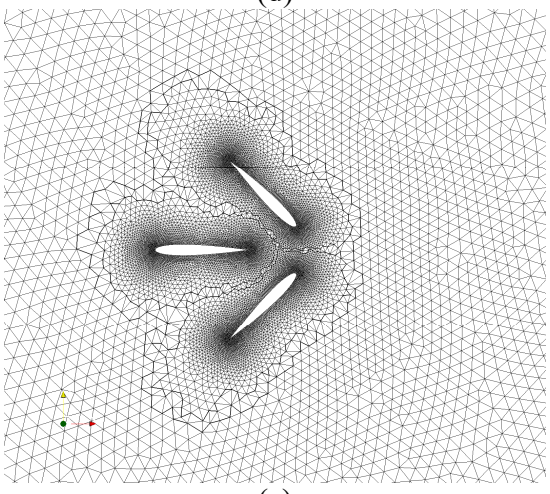
(b)



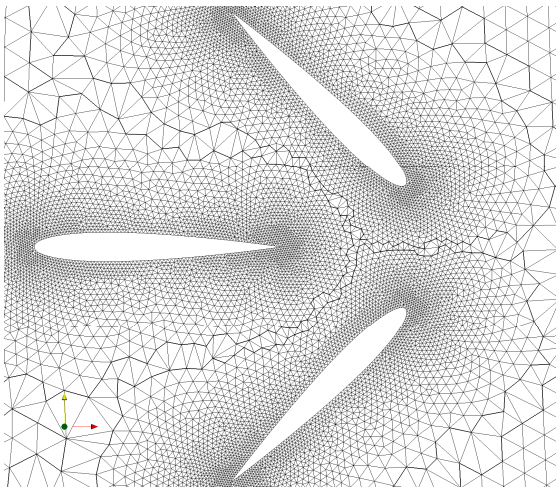
(c)



(d)

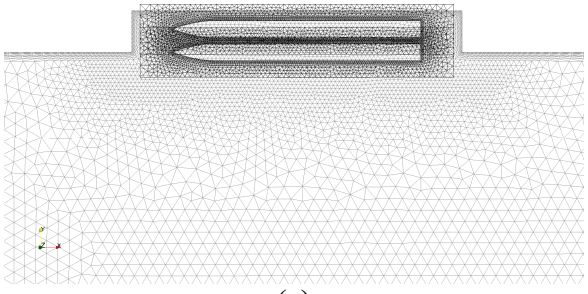


(e)

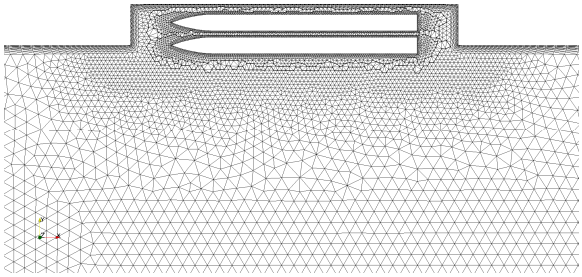


(f)

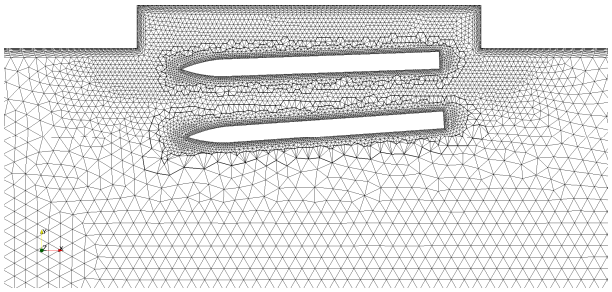
Figure 12. Stitching mesh application to multiple NACA 0012 airfoils in relative motion with respect to their leading edges. Two airfoils in tandem arrangement rotate linearly, one in counter clockwise direction and the other in clockwise direction, while one on the left is stationary. a) Original overlapped mesh blocks at 45 degrees rotation. b) Stitched mesh at 0 degrees. c) Stitched mesh at 45 degrees. d) Stitched mesh at 90 degrees. e) Stitched mesh at 135 degrees close up view with cut-out boundary highlighted.



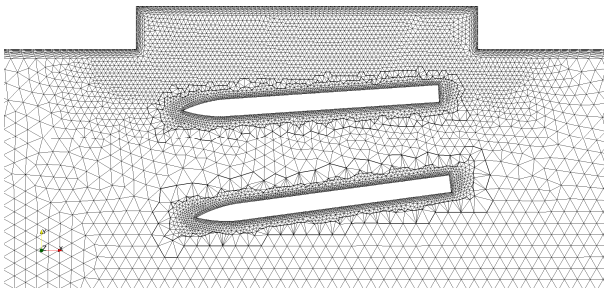
(a)



(b)



(c)



(d)

Figure 13. Stitching mesh application to two stores dropped from a rectangular weapon bay of an aerial platform. Prescribed motion is linear translation and one-degree rotation with respect to tails of the stores at each motion step. a) Original overlapped mesh blocks. b) Stitched mesh before the stores are released. c) and d) Stores at different instances of separation after released.

5. CONCLUSIONS AND DISCUSSIONS

In this paper, an alternative mesh strategy for moving objects, the mesh stitching method, is introduced and applied to some test cases. Unique part of the present method is to eliminate interpolation of the flow solution variables between the mesh blocks, as in the case of the overset mesh methods. Another advantage of the stitching method is that it requires no modification of the original flow solver code since the final mesh is unified to form a single continuous mesh block. Finally, the stitching stripe is constructed using optimum mesh size of all overlapping blocks. The only overhead would be computational cost of constructing the *stitching stripe*. However, this needs to be compared with an overset mesh implementation using the same flow solver. Since similar search for the overlapping nodes and elements needs to be done for the overset method, a fair comparison of these two methods would include interpolation cost of the overset method versus cost of the triangulation of the stitching stripe.

Three test cases are used to demonstrate applications of the stitching method. The first one is an oscillating airfoil with flow solution. It is unsteady and solution is compared with an experiment. Since the stitching region is far from the airfoil surface, some mesh stretching can be tolerated inside the stitching stripe. Second test case included multiples airfoils at relative motions such as in flapping wing problems. However, this case has no real flow solution and generated solely for the purpose of showing more complexity. Final case was a simple store separation involving two payload stores released from a hypothetical

aircraft weapon bay. This case demonstrated interaction of fine mesh around a store and coarse mesh around the bay. Mesh spacing difference for blocks seems to be significant for this case, therefore stitching stripe has stretched mesh elements as the store moves far.

Some improvements can make the stitching approach demonstrated here even more competitive such as an intelligent search of the overlapping region from one time step to the next when solving moving object problems. Such a search procedure would allow using previous locations of the stitching mesh front so that setting up the search domain from the scratch can be avoided. This can be done using nearest neighborhood search starting with the existing front. However, it comes with additional memory requirements since the previous time information needs to be stored. Instead of using standard Delaunay triangulation, a simple customized element construction can be implemented since the only two parallel layers of nodes are meshed to construct a stitching stripe. This would eliminate cleanup process after the Delaunay triangulation. Lastly, solid-to-solid collision implementation would make this method very useful for contact problems. However, the challenge in viscous flow applications for solid contact problems would require the treatment of highly stretched viscous mesh layers.

6. ACKNOWLEDGEMENT

This study was partially conducted in Indiana University-Purdue University at Indianapolis, IN, USA and supported through an SBIR program from MDA/NAVY: *SBIR 2004.4 Topic MDA04-136, Phase I*. Views expressed here reflects views of the author only.

REFERENCES

[1] L. Ding, L. Zhiliang ve G. Tongqing, «An efficient dynamic mesh generation method for complex multi-block structured grid.,» *Advances in Applied Mathematics and Mechanics,* cilt 6, no. 01, pp. 120-134, 2014.

[2] J. Batina, "Unsteady Euler airfoil solutions using unstructured dynamic meshes," *AIAA Journal 28 (8),* p. 1381–1388, 1990.

[3] A. Johnson ve T. Tezduyar, «Mesh Update Strategies in Parallel Finite Element Computations of Flow Probblems with Moving Boundaries and Interfaces,» *Comput. Methods Appl. Mech. Engrg.,,* cilt 119, pp. 73-94, 1994.

[4] A. Montorfano, F. Piscaglia ve A. Onorati, «An extension of the dynamic mesh handling with topological changes for LES of ICE in OpenFOAM,» 2015.

[5] S. T. Miller, R. L. Campbell, C. W. Elsworth, J. S. Pitt ve D. A. Boger, «An Overset Grid Method for Fluid-Structure Interaction,» *World Journal of Mechanics,* no. 4, pp. 217-237, 2014.

[6] J. Benek, P. Buning ve a. S. J.L., «A 3D Chimera Grid Embedding Technique,» 1986.

[7] M. Rai, «A conservative treatment of zonal boundaries for Euler equation calculations,» *Journal of Comp. Physics,* cilt 62, pp. 472-503, 1986.

[8] N. Suhs ve R. Tramel, «PEGSUS 4.0 User's Manual,» Arnold AFB TN, 1991.

[9] M. Liou ve Y. Zheng, «A Flow Solver for Three-dimensional DRAGON Grids,» 2002.

[10] P. G. Bunning, 23 February 2016. [Çevrimiçi]. Available: https://overflow.larc.nasa.gov/.

[11] N. Frink, «Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver,» %1 içinde *AIAA Paper*, 1994, pp. 94-0061.

[12] R. Noack ve D. A. Boger, «Improvements to Suggar and Dirtlib for Overset Store Separation Simulations.,» %1 içinde *Proceedings of the 47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Orlando, 2009.

[13] S. T. Miller, R. L. Campbell, C. W. Elsworth, J. S. Pitt ve D. A. Boger, «An Overset Grid Method for Fluid-Structure Interaction,» *World Journal of Mechanics,* cilt 4, no. 7, p. 21, 2014.

[14] E. Fadlun, R. Verzicco, P. Orlandi ve J. Mohd-Yusof, «Combined Immersed-Boundary Finite- Difference Methods for Three-Dimensional Complex Flows

Simulations,» *Journal of Computational Physics ,* cilt 161, 2000.

[15] C. Zhou ve J. Ai, «Mesh adaptation for simulation of unsteady flow with moving immersed boundaries.,» *Int. J. Numer. Meth. Fluids,* no. 72, p. 453–477, 2013.

[16] J. Mohd-Yusof, «Combined Immersed-Boundary/B-Spline Methods for Simulations of Flow in Complex Geometries,» Stanford, CA, 1997.

[17] R. Verzicco, J. Mohd-Yusof, P. Orlandi ve D. Haworth, «Large Eddy Simulation in Complex Geometric Configurations Using Boundary Body Forces,» *AIAA Journal,* cilt 38, no. 3, 2000.

[18] C. Peskin, «Flow Patterns Around Heart Valves: A Numerical Method,» *Journal of Computational Physics,* cilt 10, 1972.

[19] I. Borazjani, «Fluid–structure interaction, immersed boundary-finite element method simulations of bio-prosthetic heart valves,» *Computer Methods in Applied Mechanics and Engineering,* cilt 257, pp. 103-116, 15 April 2013.

[20] Y. Tseng ve J. Ferziger, «A ghost-cell immersed boundary method for flow in complex geometry,» *Journal of Computational Physics,* cilt 192, no. 2, pp. 593-623, 2003.

[21] A. Landsberg ve J. Boris, «The Virtual Cell Embedding Gridding Method: A Simple Approach for Complex Geometries,» %1 içinde *Proceedings of 13th AIAA CFD Conference, paper 97-1982*, 1997.

[22] F. Lazarus ve A. Verroust, «Three-dimensional Metamorphosis: A survey,» *The Visual Computer,* cilt 14, no. 8/9, p. 373–389, 1998.

[23] M. Ahn ve S. Lee, «Mesh Metamorphosis with Topological Transformations,» %1 içinde *Proceedings of Pacific Graphics 2002: IEEE Computer Society Press; p. 481–482.*, 2002.

[24] G. Weber, O. Kreylos, T. Ligocki, J. Shalf, H. Hagen, B. Hamann ve K. Joy, «Extraction Of Crack-Free Isosurfaces From Adaptive Mesh Refinement Data,» %1 içinde *Data Visualization 2001 (Proceedings of VisSym '01); p 25–34*, 2001.

[25] C. T. D. JR, «An Adaptive Hybrid Mesh Generation Method For Complex Geometries,» Chattanooga, 2011.

[26] B. Joe, «GEOMPACK--a software package for the generation of meshes using geometric algorithms,» *Advanced Engineering Software,* cilt 13, pp. 325-331, 1991.

[27] E. Yilmaz ve M. Kavsaoglu, «Euler Solution of Axisymmetric Jets in Supersonic Flow,» *Journal of Spacecraft and Rockets,* cilt 35, no. 1, 1998.

[28] R. Landon, «Compendium of Unsteady Aerodynamic Measurements,» AGARD Report 702, 1982.