A Robust Mesh Moving Solver for Moving-Boundary Problems

Shuang Z. Tu* and Chao Jiang†
Jackson State University, Jackson, MS, 39217, USA

This paper presents a robust mesh moving solver developed to address moving boundary problems. Crucially, the resulting deformed mesh retains the same topology as the original mesh without being overly distorted. The mesh is treated as an elastic material, and the deformation of the computational domain resulting from moving boundaries is determined by solving the equilibrium linear elasticity equations. The linear elasticity equations are discretized by the classic Galerkin finite element method and solved by the block conjugate gradient iterative method. To maintain the quality of the mesh after motion, the Young's modulus of each element is weighted by the reciprocal of the distance between the element center and the moving boundaries. The effectiveness of this approach is demonstrated through a set of 2D and 3D test cases featuring prescribed translational and/or rotational motion of the embedded object. The method is now ready for integration into our existing in-house CFD solvers.

I. Introduction

Mesh moving in Computational Fluid Dynamics (CFD) can occur in two particular situations. One is when the r-typed mesh adaptation is employed in the solution process and the other is when the computational domain involves moving boundaries. In this paper, we focus on the mesh moving scenario due to moving boundaries.

Many practical applications (aeroelasticity, store separation, turbomachinery, rotors, etc.) involve moving boundaries due to the body deformation caused by the fluid-structure interaction or the prescribed boundary motion or a combination of both. Moving boundaries embedded in the flow field generate a new challenge for numerical algorithms. There are many approaches in the literature to handling moving objects in the flow field depending on the employed grid methodology.

The most common approach is probably the *dynamic or moving mesh method*.[2,14] The dynamic mesh method moves the interior mesh nodes in response to the boundary node motion. Usually, the mesh topology does not change when the mesh is moved. This restriction limits the use of the moving-mesh method when the deformation is large because the mesh can become overly distorted and even entangled. Mesh moving methods can be supplemented by the local mesh optimization via local reconnection and node insertion/deletion to allow topology changes.[6] Allowing topology changes improves the method's flexibility and robustness at the price of the added complexity. In space-time methods, mesh topology changes introduce new space-time slabs of special shapes. Constructing space-time slabs is a complicated process.[17] The mesh vertex displacements are often obtained by solving a global equilibrium equation system derived from the linear spring analogy,[1]

*Professor, AIAA Associate Fellow, (shuang.z.tu@jsums.edu).
†PhD candidate, (j00744382@students.jsums.edu).

torsional spring analogy[3,5] or other variants.[10] By treating the mesh as a piece of elastic material, one can also solve the equilibrium linear elasticity equations[13,14] for node displacements.

Another approach is the *overset or chimera grid method* which was originally proposed to eliminate the difficulty with meshing complex geometries. It can also be used to handle moving rigid bodies. The overset grid method uses a set of overlapping structured or unstructured grids to discretize the domain and individual moving components. Interpolation is required to couple the solution on different grids. PEGASUS[12] and SUGGAR[11] are two examples of general-purpose overset grid packages. The solution transfer between the donor grid and the receptor grid is often one of the primary sources of error and degrades the accuracy of the underlying discretization method.

Keeping the topology of the mesh unchanged is a desired feature for existing CFD solvers. The main focus of this paper is to ensure that the deformed mesh closely preserves the quality of the original mesh, thereby minimizing excessive distortion and preventing negative volumes, while preserving the original mesh topology.

In this paper, we treat the computational domain as a piece of elastic material. The linear elasticity equations are solved for the displacement of mesh vertices. The classic finite element method is used to discretize the equations. The block conjugate gradient iterative method is used to solve the resulting discretized linear equations. Besides, parallelization based on the Message Passing Interface (MPI) library is implemented to speed up the solution process.

The paper is organized as follows. Section II briefly reviews the equilibrium linear elasticity equations. Section III presents the Galerkin finite element method used to solve the linear elasticity equations. Section V describes the current approach to improve the quality of the deformed mesh by scaling Young's modulus by the reciprocal of the distance of the element from the moving boundary. Section VI presents several test cases to demonstrate the effectiveness of the current mesh moving approach. Finally, Sec. VII concludes this paper.

II. Linear Elasticity Equations

We assume the computation domain Ω is made of fictitious elastic material free of body forces and the initial state of the domain is in stress equilibrium. When the boundary nodes are moved due to body deformation or the prescribed body motion, the interior nodes must adjust to new locations to reach the new stress equilibrium. We can then solve the equilibrium linear elasticity equations for the displacements of all mesh nodes. The equation in vector form is given as

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}_m(\mathbf{d}) = 0 \tag{1}$$

where $\mathbf{d}$ is the displacement of the mesh node and $\boldsymbol{\sigma}_m$ is the stress tensor. The following constitutive equation is a result of Hooke's law that defines the relationship between the stress and the strain:

$$\boldsymbol{\sigma}_m = \lambda_m(\boldsymbol{\nabla} \cdot \mathbf{d})\mathbf{I} + 2\mu_m\boldsymbol{\varepsilon}_m(\mathbf{d}) \tag{2}$$

where $\boldsymbol{\varepsilon}_m$ is the strain tensor, and λ_m and μ_m are the Lamé constants. The subscript 'm' is used here to avoid confusion with similar notations for viscous stresses. The strain tensor satisfies the following strain-displacement equation:

$$\boldsymbol{\varepsilon}_m(\mathbf{d}) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{d} + (\boldsymbol{\nabla}\mathbf{d})^T\right) \tag{3}$$

Substituting Eq. (3) into Eq. (2) to obtain the components of the stress tensor $\boldsymbol{\sigma}_m$ as follows:

$$\begin{aligned}
\sigma_{xx} &= \lambda_m \left(\frac{\partial d_1}{\partial x} + \frac{\partial d_2}{\partial y} + \frac{\partial d_3}{\partial z} \right) + 2\mu_m \frac{\partial d_1}{\partial x} \\
\sigma_{xy} = \sigma_{yz} &= \mu_m \left(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x} \right) \\
\sigma_{xz} = \sigma_{zx} &= \mu_m \left(\frac{\partial d_1}{\partial z} + \frac{\partial d_3}{\partial x} \right) \\
\sigma_{yy} &= \lambda_m \left(\frac{\partial d_1}{\partial x} + \frac{\partial d_2}{\partial y} + \frac{\partial d_3}{\partial z} \right) + 2\mu_m \frac{\partial d_2}{\partial y} \\
\sigma_{yz} = \sigma_{zy} &= \mu_m \left(\frac{\partial d_2}{\partial z} + \frac{\partial d_3}{\partial y} \right) \\
\sigma_{zz} &= \lambda_m \left(\frac{\partial d_1}{\partial x} + \frac{\partial d_2}{\partial y} + \frac{\partial d_3}{\partial z} \right) + 2\mu_m \frac{\partial d_3}{\partial z}
\end{aligned}$$

where d_i's ($i = 1, 2, 3$) is the three component of $\mathbf{d}$ in x-, y- and z-direction, respectively.

The Lamé constants can be related to Young's modulus, E, and the Poisson ratio, ν_m.

$$\lambda_m = \frac{\nu_m E}{(1+\nu_m)(1-2\nu_m)}, \quad \mu_m = \frac{E}{2(1+\nu_m)} \tag{4}$$

Young's modulus, E, describes the elastic properties of a solid undergoing tension or compression. In a real solid material, E is a fundamental property of the material. Higher Young's modulus indicates stiffer material. To allow better control of the element quality after mesh moving, E is allowed to vary element by element. The Poisson ration ν_m is assumed to be uniform across the domain. Since the right hand side of Eq. (1) is zero, one can modify the Lamé parameters by multiplying both sides with $2(1+\nu_m)$ to have

$$\bar{\lambda}_m = \frac{2\nu_m E}{1-2\nu_m}, \quad \bar{\mu}_m = E. \tag{5}$$

Since E and ν_m are the only two parameters in the above equations, it can be speculated that different values of these two parameters may affect the quality of the deformed mesh.

III. Galerkin Finite Element Discretization of Linear Elasticity Equations

The continuous Galerkin finite element method is used to solve the equilibrium linear elasticity equations which are of elliptic type. Multiply Eq. (1) by some test function v and the resulting Galerkin formulation can be written as:

$$\int_\Omega v \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}_m \mathrm{d}\Omega = 0. \tag{6}$$

Applying integration by parts on the above equation yields

$$-\int_\Omega \nabla v \cdot \boldsymbol{\sigma}_m \mathrm{d}\Omega + \int_{\partial\Omega} v \boldsymbol{\sigma}_m \cdot \mathbf{n} \mathrm{d}\Gamma = 0.$$

To solve the discretized equations, Dirichlet or Neumann boundary conditions for the displacements of boundary nodes must be given.

The resulting linear equation system is solved by a block conjugate gradient solver. The block conjugate gradient solver is adapted from our implementation of the scalar conjugate gradient

method for solving the pressure Poisson equation in our incompressible flow solver.[15] The Eisenstat[4] trick is modified to work with the block conjugate gradient solver as a preconditioner.

After the displacement of each node is obtained, the coordinates of each mesh node can be simply updated via

$$\mathbf{x}(t^{n+1}) = \mathbf{x}(t^n) + \mathbf{d}. \tag{7}$$

IV. Parallelization of the Numerical Solver

To solve large scale problems, the current numerical solver is parallelized using the Message Passing Interface (MPI) library to access multiple processors simultaneously and speed up computation. The choice of the MPI parallel paradigm is due to its standardization, excellent platform independent portability and flexibility on both distributed memory and shared memory machines. The parallelism is achieved via the Single Program Multiple Data (SPMD) principle. The computational mesh is first partitioned across certain number of processes using the ParMETIS library[8] (cf. Fig. 1). The partitioning ensures the number of elements is roughly the same on each of the processes for the load balancing purpose. In addition, ParMETIS also minimizes the inter-process communication overhead. The same numerical solver program is then executed on each of the processes on its portion of the mesh simultaneously. Inter-process communication occurs to synchronize the computation. Since the current cell-centered finite volume and nodal finite element solvers are constructed on compact computational stencils, the inter-processor communication involves only nodes, faces and elements on the partition boundaries (cf. Fig. 2). This compactness makes it trivial to attain high parallelizability using MPI for fixed-topology meshes. Very efficient non-blocking MPI functions can be called to set up the inter-processor "gather" and "scatter" routines in the pre-processing stage.[7,16] The communication overhead has been minimized thanks to these routines.

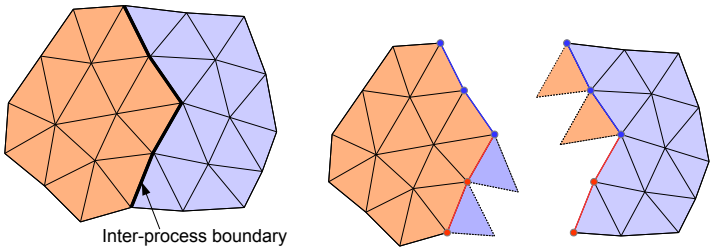


Figure 1: Mesh partitioning.

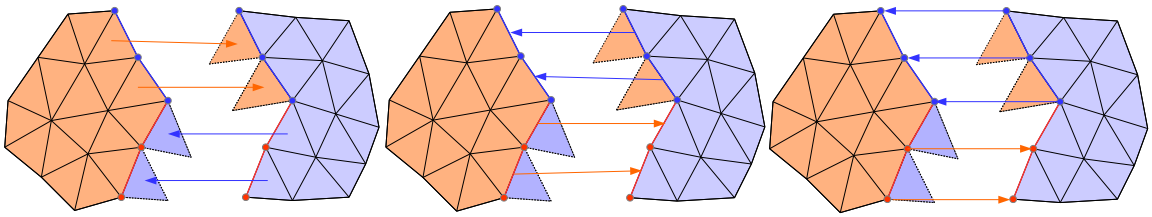


Figure 2: Inter-process communication. Left: element communication. Middle: face communication. Right: vertex communication.

V. Mesh Quality Control

As stated in Sec. II, Young's modulus, E, and Poisson's ratio, ν_m, are the two parameters that can possibly be used to control the mesh quality during mesh deformation.

The Poisson ratio ν_m is a measure of the simultaneous change in elongation and in cross-sectional area within the elastic range when stretched. Most practical engineering materials have positive Poisson ratios which means when the material is stretched, the reduction in cross-sectional area is proportional to the increase in length by a factor, the Poisson ratio. The typical value is between 0.0 and 0.5. For example, cork is close to 0.0, most steels are around 0.3, and rubber is almost 0.5. However, some new materials, mostly polymer foams, have a negative Poisson ratio, which means these materials get thicker when stretched. In our numerical tests, changing the value of Poisson's ratio has little or no effect on the quality of the deformed mesh. Therefore, to simplify things, Poisson's ratio is kept a fixed value across the entire domain during the mesh moving process, i.e. $\nu_m = 0.0$. We focus on manipulating the distribution of Young's modulus.

V.A. Young's Modulus Based on the Inverse Distance to the Moving Boundaries

Larger E indicates stiffer material. Therefore, an appropriate distribution of E can be used to control the mesh quality during mesh moving. For regions close to the moving boundaries, we desire to retain the original mesh quality as much as possible. In this study, we found the distribution of E weighed by the inverse of the distance to the moving boundaries greatly improve the mesh quality. The shortest distance of the mesh vertices from the moving boundaries can be efficiently calculated using the KDTREE2 package.[9]

V.B. Metric for Evaluating Mesh Quality

To measure the resulting mesh quality, elements with equal edges are considered standard reference elements.

V.B.1. Two Dimensional Elements

For 2D elements (triangles and quadrilaterals), the interior angle of each element will be compared with the corresponding reference element. The standard reference interior angle of triangles and quadrilaterals is 60° and 90°, respectively, as listed in Table 1.

Table 1: Reference interior angles for 2-D triangles and quadrilaterals

Element Type	Standard Reference Interior Angle
Triangle	60°
Quadrilateral	90°

V.B.2. Three Dimensional Elements

For 3D elements (tetrahedra, hexahedra, prisms, and pyramids), the dihedral angle of each element is an appropriate measure to evaluate the quality of such elements. A dihedral angle is defined as the angle between two faces sharing a common edge. Table 2 lists the dihedral angles for each type of elements. Note that prisms and pyramids have two different dihedral angles.

Table 2: Reference dihedral angles for 3-D elements

Element Type		Standard Reference Dihedral Angle
Tetrahedron		70.53°
Hexahedron		90°
Prism	Angle of Type 1	60°
	Angle of Type 2	90°
Pyramid	Angle of Type 1	54.74°
	Angle of Type 2	109.47°

VI. Numerical Tests

In this section, several test cases involving moving boundaries are used to demonstrate the effectiveness of current mesh moving solver in preserving the mesh quality while keeping the original mesh topology.

VI.A. Test 1: Movement of a Square Object inside a 2-D Domain

We first use a simple 2D case to demonstrate the effectiveness of current mesh moving method. In this case, a moving square object of size 0.2×0.2 is originally sitting in the middle of a unit square domain. The domain is discretized by 2278 unstructured triangles. The motion of the object is prescribed as a combination of both translation along the diagonal line $y = x$ and rotation around its own center. The translation distance as a function of time is given by

$$d = 0.2 \sin(2\pi t/T_0)$$

where the period $T_0 = 2.0$. Similarly, the rotation angle (in degree) as a function of time is given by

$$\theta = -45^\circ \sin(2\pi t/T_0).$$

The object starts to translate and rotate at $t = 0.0$ and continues for one period until $t = 2.0$. The time step for the motion is chosen as $\Delta t = 0.025$. Figure 3 shows the initial mesh. Figures 4 and 5 compares the deformed meshes at different moments for uniform Young's modulus and Young's modulus weighted by inverse of the element-to-object distance. As can be visually seen, when Young's modulus is weighted by the inverse of the element-to-object distance, the mesh quality is significantly improved near the moving object. Table 3 shows the comparison of minimal and maximal interior angles of the deformed mesh at different time moments. Uniform Young's modulus leads to overly distortion of elements quickly while inverse distance weighted Young's modulus preserves the mesh quality in a much more acceptable way. Note that Δt larger than 0.025 will produce negative volume elements for the case of uniform Young's modulus.

VI.B. Test 2: Movement of an Airfoil inside a 2-D Domain

Next we consider the NACA0012 airfoil rotating about its quarter-chord center inside the computational domain according to the rotation angle (in degree) as a function of time given by

$$\theta = -90^\circ \sin(2\pi t/T_0) \tag{8}$$

where the period $T_0 = 8$. The airfoil starts to rotate at $t = 0.0$ until $t = 2.0$. At $t = 0.0$, the angle of attach of the airfoil is 0 degree. At $t = 2.0$, the angle of attach becomes 90°.

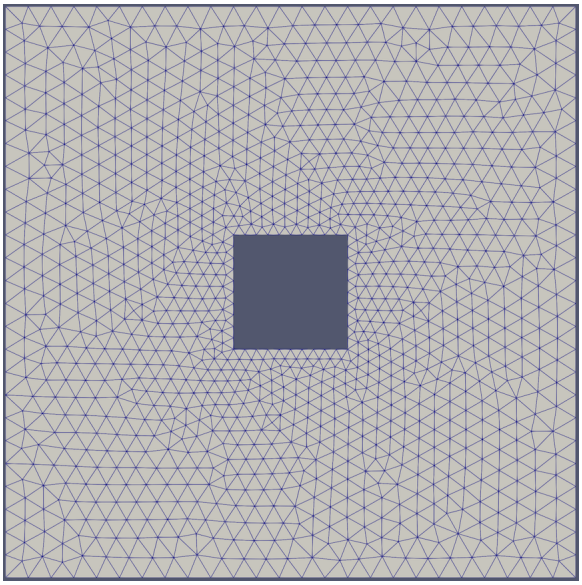
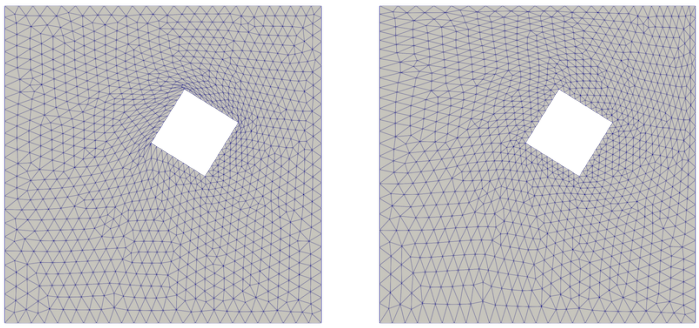


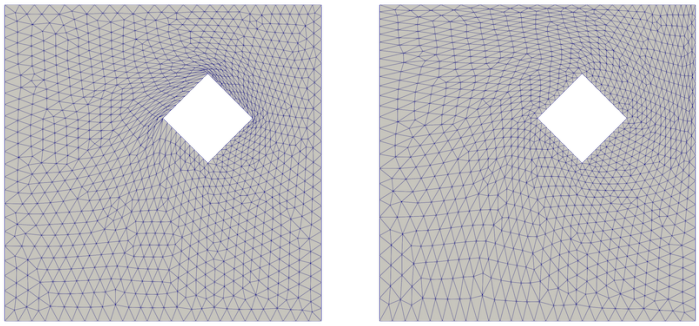
Figure 3: Initial mesh at $t = 0.0$.

Table 3: Mesh quality comparison.

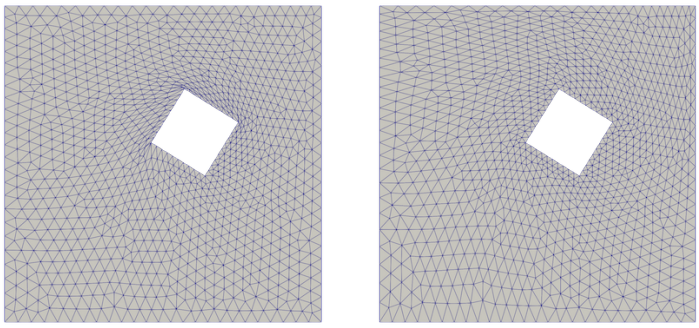
	Uniform Young's modulus		Inverse distance weighted Young's modulus	
t	min angle	max angle	min angle	max angle
0.0	37.04°	99.51°	37.04°	99.51°
0.25	1.61°	174.06°	19.93°	126.84°
0.5	0.00°	180.00°	13.10°	146.61°
0.75	0.06°	179.80°	19.84°	126.87°
1.0	10.76°	128.67°	36.52°	100.51°
1.25	5.67°	161.44°	22.24°	127.72°
1.5	1.24°	176.64°	15.48°	146.37°
1.75	5.46°	161.53°	22.29°	127.77°
2.0	9.89°	127.28°	36.27°	101.11°



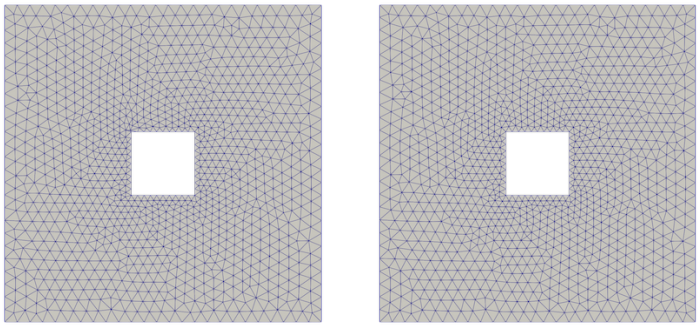
(a) $t = 0.25$



(b) $t = 0.5$

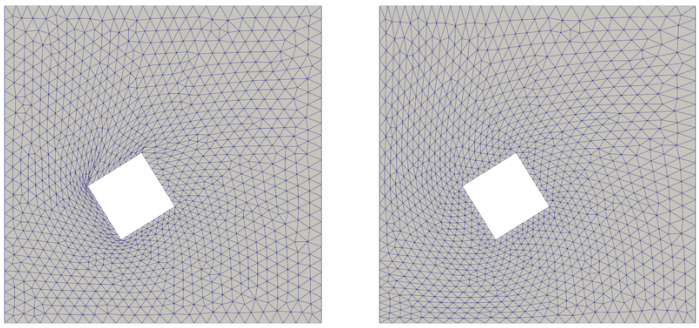


(c) $t = 0.75$

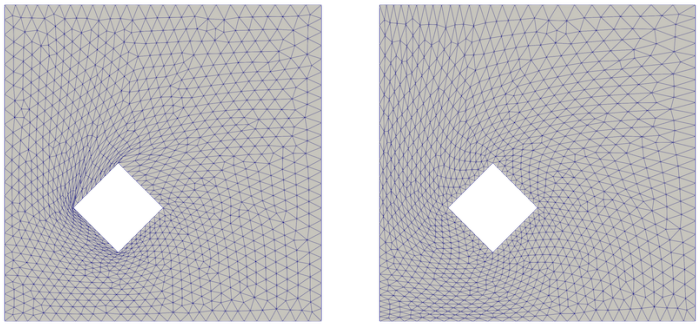


(d) $t = 1.0$

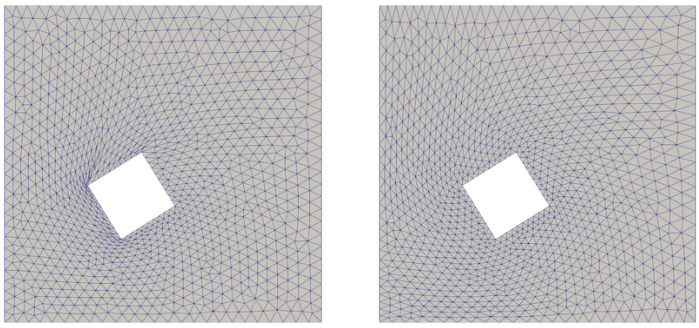
Figure 4: Deformed mesh at $t = 0.25, 0.5, 0.75, 1.0$. Left column: uniform E. Right column: E weighted by inverse distance from the moving object.



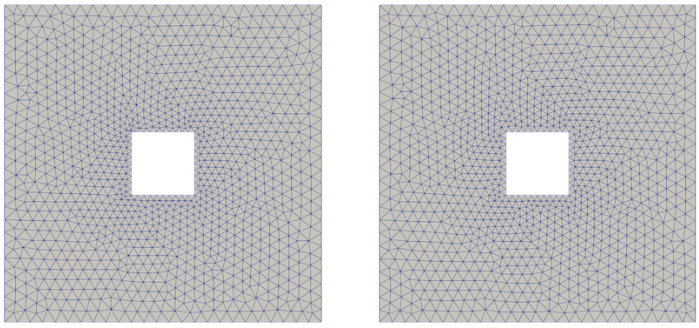
(a) $t = 1.25$



(b) $t = 1.5$



(c) $t = 1.75$



(d) $t = 2.0$

Figure 5: Deformed mesh at $t = 1.25, 1.5, 1.75, 2.0$. Left column: uniform Young's modulus. Right column: Young's modulus weighted by inverse distance from the moving object.

Figure 6 is a triangular mesh around the NACA0012 airfoil for an inviscid flow simulation. The mesh contains 5093 triangles. Here only the mesh moved with inverse-distance-weighted Young's modulus is shown. The mesh moved with uniform Young's modulus is overly distorted. Figure 7 shows the close-up views of the mesh at $t = 0.0$ and $t = 2.0$. As can be seen, only slight deformation of the elements can be seen near the leading edge and the trailing edge of the airfoil after 90° rotation.

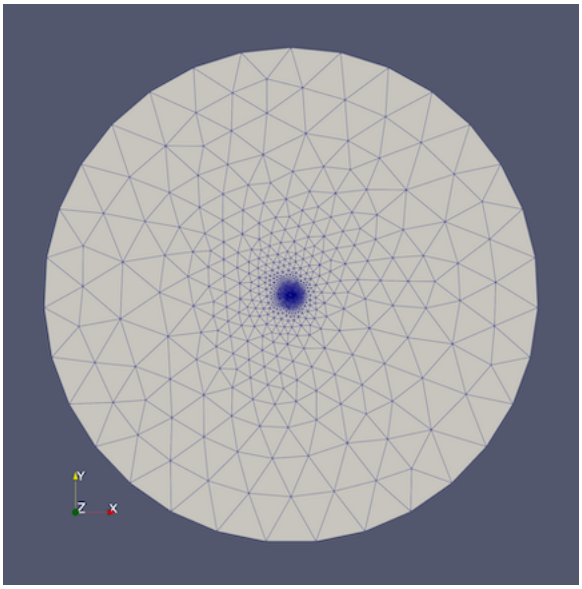


Figure 6: A triangular mesh around the NACA0012 airfoil for an inviscid flow simulation.

Figure 8 is a hybrid quadrilateral/triangular mesh around the NACA0012 airfoil for a viscous flow simulation. The mesh contains 5772 quadrilaterals near the airfoil and in the wake region and 9877 triangles elsewhere. The airfoil rotates around its quarter-chord center according to the motion specified in (8). The high aspect ratio elements near the airfoil can easily get overly distorted if uniform Young's modulus is used. Figure 9 shows the close-up view of the mesh near the airfoil when the Young's modulus is weighted by the inverse distance to the airfoil. As can be seen, the quality of the elements near the leading edge and the trailing edge of the airfoil has been preserved to a very satisfactory level after 90° rotation.

VI.C. Test 3: Movement of a Cuboid Wing inside a 3-D Domain

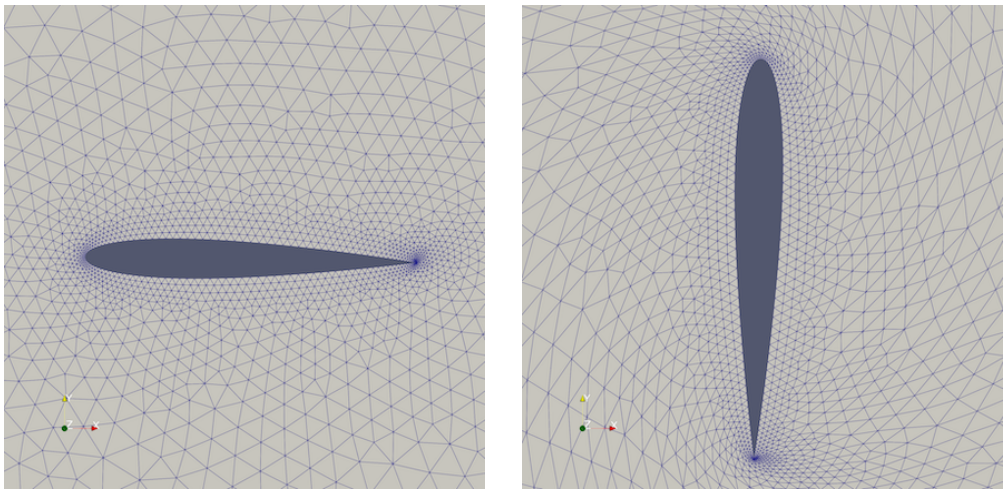
Finally, we consider a moving cuboid wing embedded in a 3-D domain as shown in Fig. 10 (a). The computational domain is a cuboid box defined by $[-10, 13] \times [0, 10] \times [-10, 10]$. The cuboid wing is attached to the symmetry plane at $y = 0.0$. The mesh contains 313,760 prisms and 1,043,025 tetrahedra. The motion of the wing is prescribed as a combination of translation

$$d = 2\sin(2\pi t/T_0) \tag{9}$$

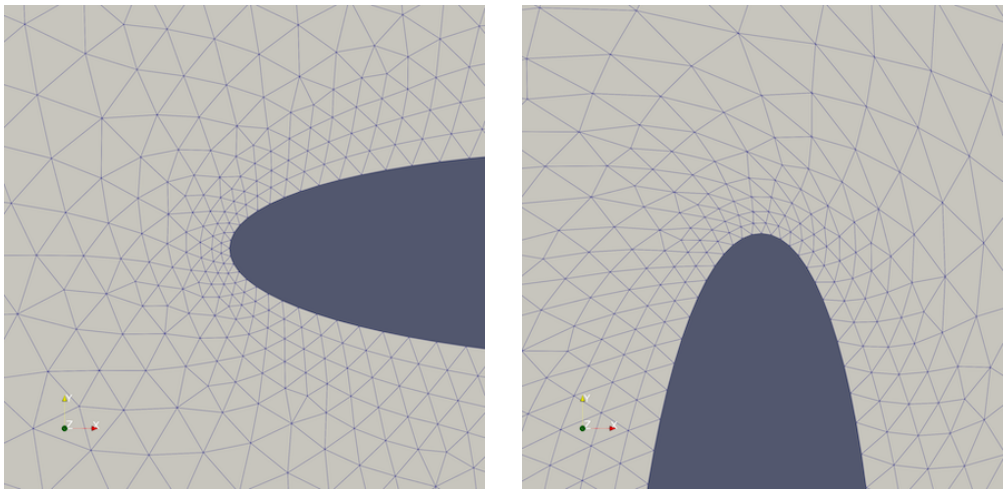
along the direction indicated by the vector $(\sqrt{2}/2, 0, \sqrt{2}/2)$ and rotation

$$\theta = -90^\circ \sin(2\pi t/T_0) \tag{10}$$

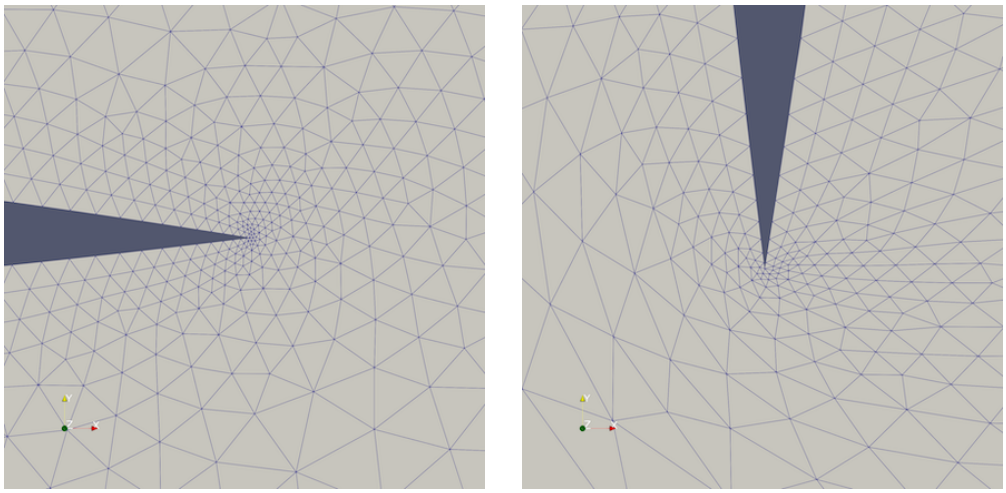
about the rotating axis defined by the line connecting points $(0, 0, 0)$ and $(0, 10, 0)$. Here $T_0 = 8.0$. The wing starts to move at $t = 0.0$ when the angle of attack is 0° and stops at $t = 2.0$ when the angle of attack is 90°. Figure 10 (b) shows the mesh in the symmetry plane. Figure 11 shows some close-up views of the mesh at $t = 0.0$ and $t = 2.0$. As can be seen, the mesh near the leading edge and the trailing edge remains high quality after combined translation and 90° rotation.



(a) Near-field close-up view of the mesh



(b) Close-up view of the mesh near the leading edge



(c) Close-up view of the mesh near the trailing edge

Figure 7: A triangular mesh around the NACA0012 airfoil for an inviscid flow simulation. Left column: mesh at $t = 0$. Right column: mesh at $t = 2.0$.

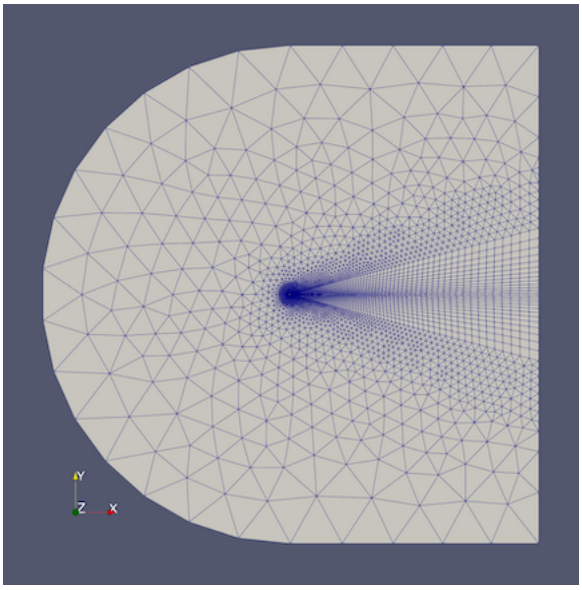


Figure 8: A hybrid quadrilateral/triangular mesh around the NACA0012 airfoil for a viscous flow simulation.

VII. Conclusions

By treating the mesh as a linear elastic material and solving the linear elasticity equation to determine the displacement of the deformed mesh vertices, a significant enhancement in the quality of the deformed mesh can be achieved when Young's modulus is weighted based on the reciprocal of the element's distance from the moving object. As a result, this approach offers a reliable solution for addressing mesh displacement caused by moderate interior object motion, all while preserving the mesh's topology. The integration of this mesh moving technique into existing CFD solvers is anticipated to be a straightforward process.

Acknowledgments

This work is partially supported by U.S. Department of Energy Office of Science Award No. DE-SC0023125, NSF Award No. 2219542 and NASA Award No. 80NSSC21M0332.

References

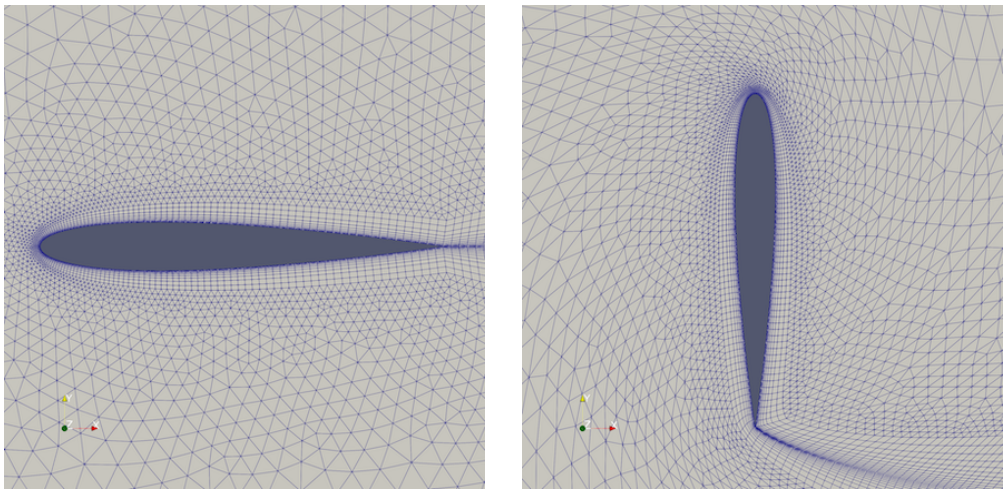
[1]J. Batina. Unsteady Euler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysisuler algorithm with unstructured dynamic mesh for complex-aircraft aerodynamic analysis. *AIAA Journal*, 29(3):327–333, 1991.

[2]C. Burg. A robust unstructured grid movement strategy using three-dimensional torsional springs, 2004. AIAA Paper 2004-2529.

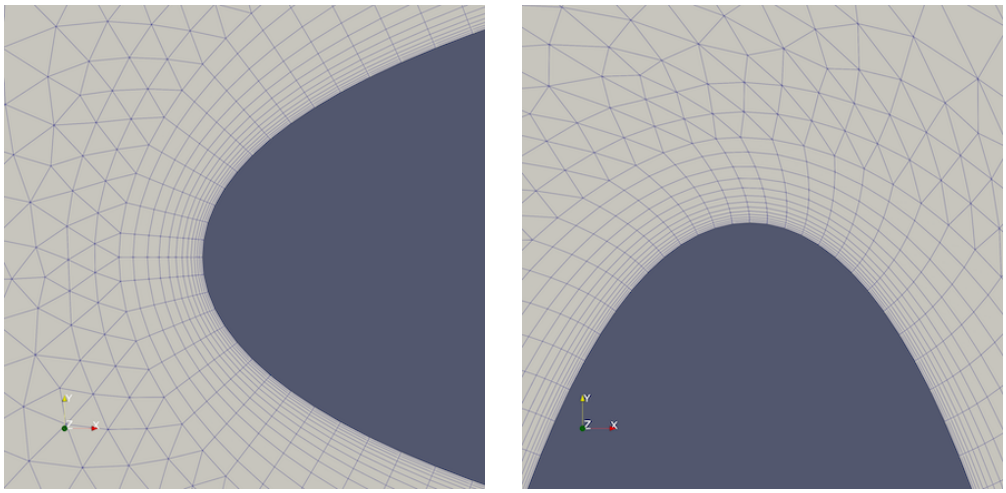
[3]C. Degand and C. Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures*, 80:305–316, 2002.

[4]S. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Stat. Comp.*, 2:1–4, 1981.

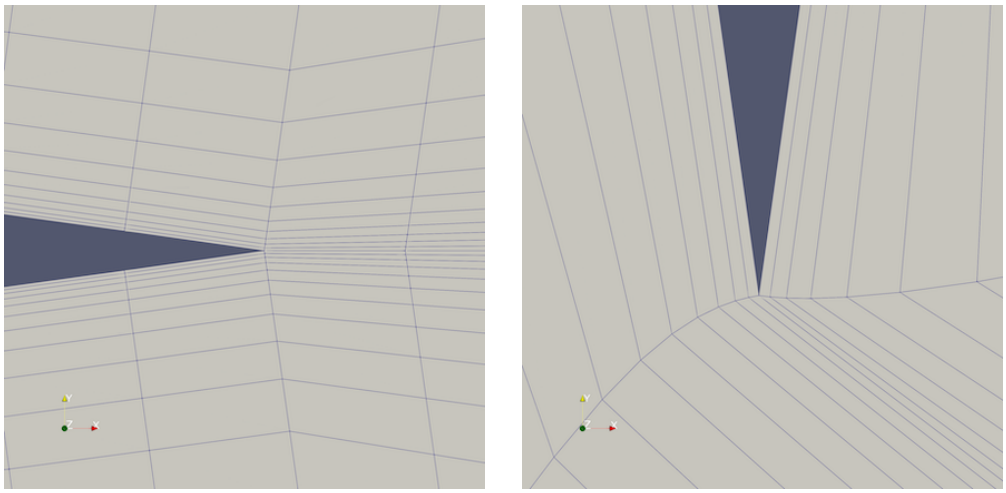
[5]C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Comput. Methods Appl. Mech. Engrg.*, 163:231–245, 1998.



(a) Near-field close-up view of the mesh

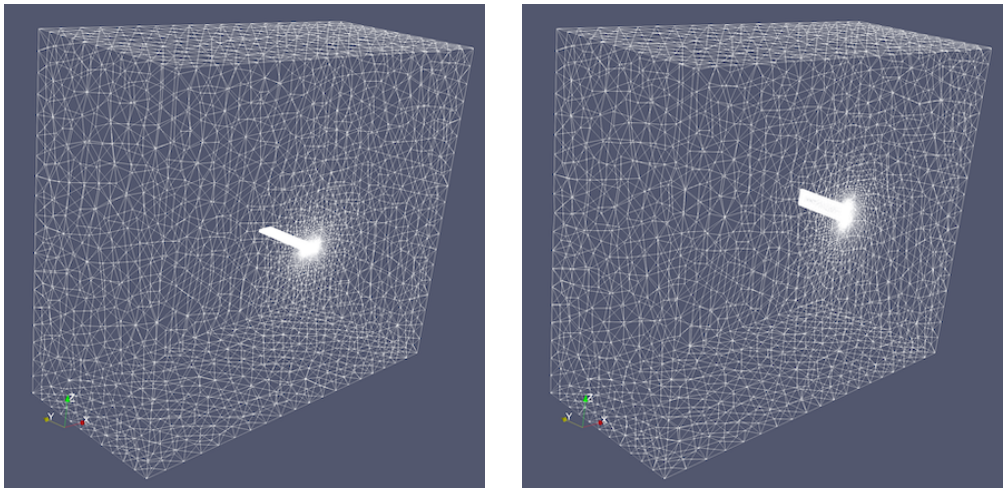


(b) Close-up view of the mesh near the leading edge

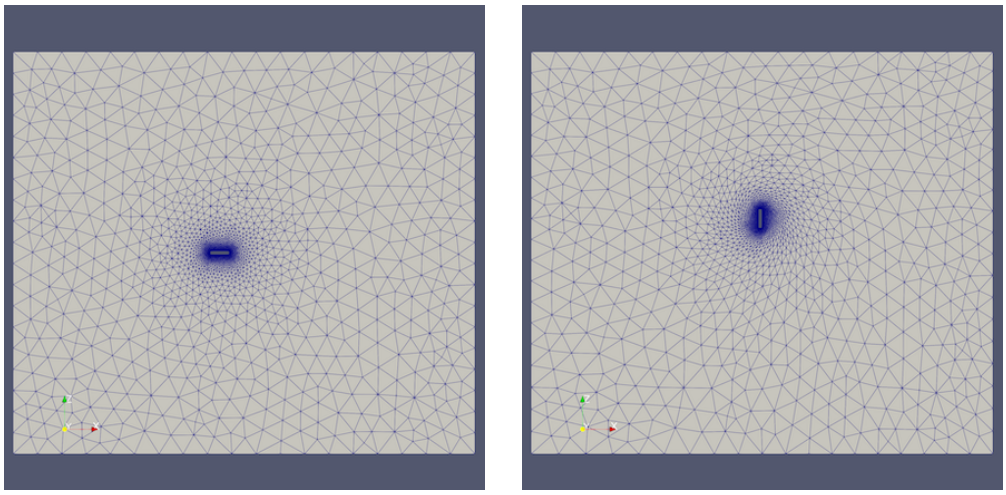


(c) Close-up view of the mesh near the trailing edge

Figure 9: A hybrid quadrilateral/triangular mesh around the NACA0012 airfoil for a viscous flow simulation. Left column: mesh at $t = 0$. Right column: mesh at $t = 2.0$.

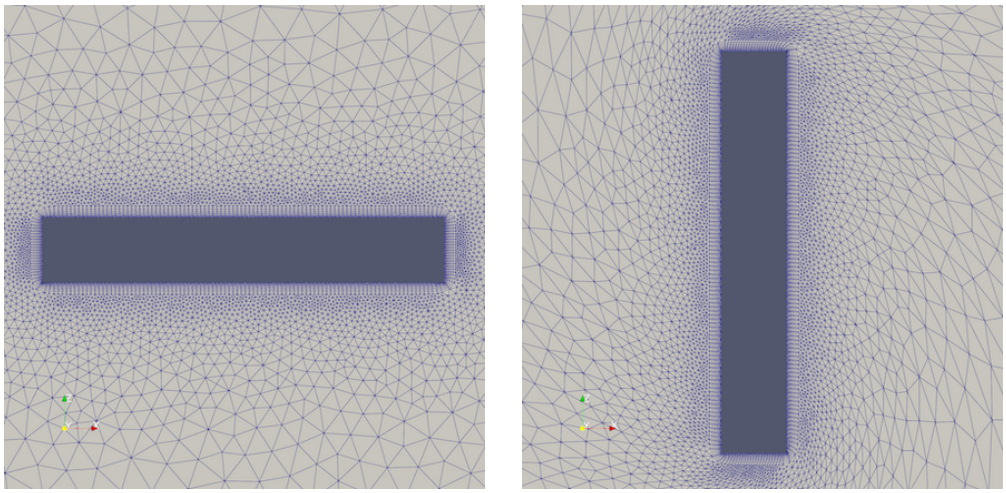


(a) Global view of the 3D mesh.

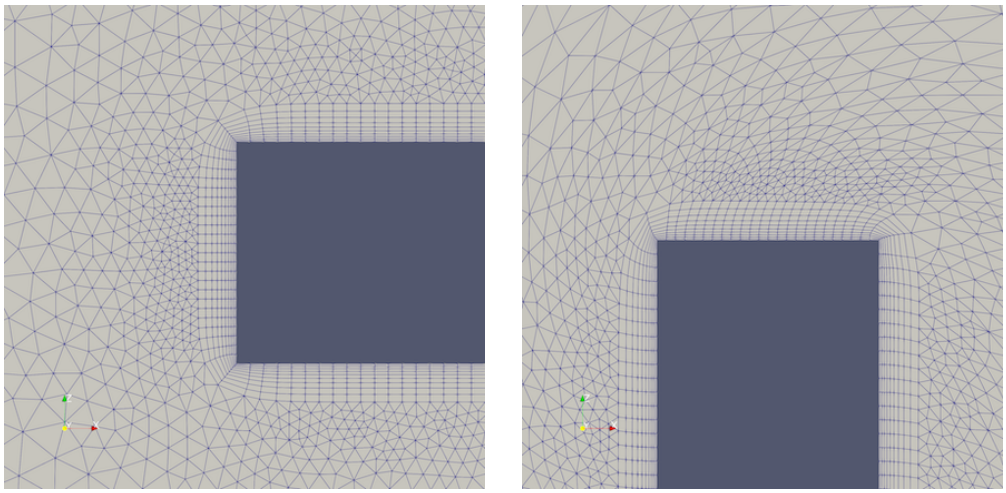


(b) Global view of the mesh in the symmetry plane ($y = 0.0$)

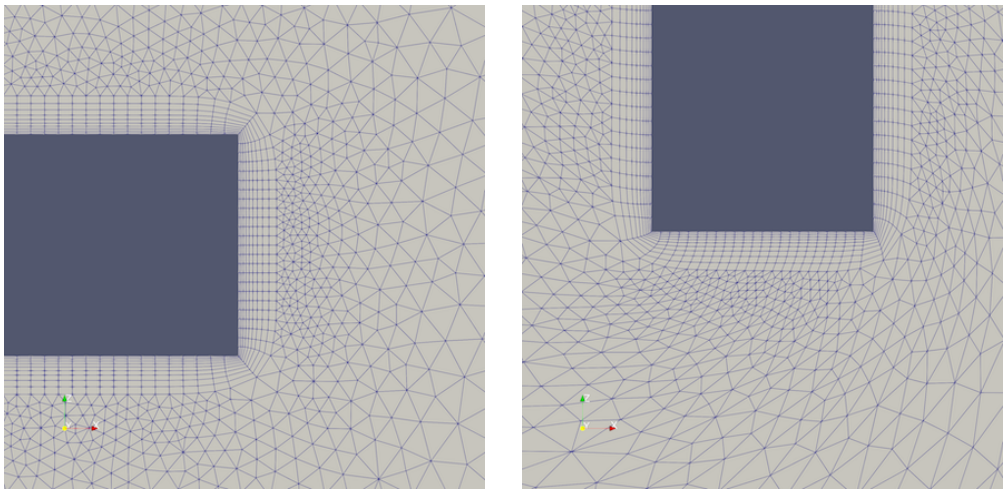
Figure 10: A hybrid prismatic/tetrahedral mesh around a cuboid wing. Left: $t = 0.0$ and right: $t = 2.0$.



(a) Close-up view of the mesh in the symmetry plane



(b) Close-up view of the mesh near the leading edge in the symmetry plane



(c) Close-up view of the mesh near the trailing edge in the symmetry plane

Figure 11: A hybrid prismatic/tetrahedral mesh around a cuboid wing. Left column: $t = 0.0$ and right column: $t = 2.0$.

[6]A. Johnson. Dynamic-mesh CFD and its application to flapping-wing micro-air vehicles. In *Proceedings of the 25th Army Science Conference*, Orlando, FL, 2006.

[7]A. Johnson and T. Tezduyar. Parallel computation of incompressible flows with complex geometries. *Int. J. Numer. Meth. Fluids*, 24:1321–1340, 1997.

[8]G. Karypis. ParMETIS - parallel graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview.

[9]M. Kennel. KDTREE2: Fortran 95 and C++ software to efficiently search for near neighbors in a multi-dimensional Euclidean space, 2004.

[10]G. A. Markou, Z. S. Mouroutis, D. C. Charmpis, and M. Papadrakakis. The ortho-semi-torsional (ost) spring analogy method for 3D mesh moving boundary problems. *Computer Methods in Applied Mechanics and Engineering*, 196(4-6):747–765, 2007.

[11]R. Noack. SUGGAR: a general capability for moving body overset grid assembly, 2006. AIAA Paper 2005-5117.

[12]S. Rogers, N. Suhs, and W. Dietz. PEGASUS 5: an automated preprocessor for overset-grid computational fluid dynamics. *AIAA Journal*, 41(6):1037–1045, 2003.

[13]K. Stein, T. Tezduyar, and R. Benney. Computational methods for modeling parachute systems. *Computing in Science and Engg.*, 5(1):39–46, 2003.

[14]T. Tezduyar, M. Behr, S. Mittal, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces–the DSD/ST procedure: II. computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Comput. Methods Appl. Mech. Eng.*, 94:353–371, 1992.

[15]S. Tu and S. Aliabadi. Development of a hybrid finite volume/element solver for incompressible flows on unstructured meshes. *Int. J. Numer. Meth. Fluids*, 55(2):177–203, 2007.

[16]S. Tu, S. Aliabadi, A. Johnson, and M. Watts. A robust parallel implicit finite volume solver for high-speed compressible flows, January 2005. AIAA Paper 2005-1396.

[17]P. J. Zwart, G. D. Raithby, and M. J. Raw. The integrated space-time finite volume method and its application to moving boundary problems. *J. Comput. Phys.*, 154(2):497–519, 1999.