High-order Discontinuous Galerkin Simulations on Moving Domains using an ALE Formulation and Local Remeshing with Projections

Luming Wang* and Per-Olof Persson†
University of California, Berkeley, Berkeley, CA 94720-3840, U.S.A.

I. Introduction

High-order methods such as the Discontinuous Galerkin (DG) method are considered promising alternatives for the numerical simulation of turbulent flows with complex vortical structures and non-linear interactions.[1,2] Being based on high-order discretization of conservation laws, the DG methods are able to obtain stable solutions with low numerical dissipation for compressible flow problems. Since the schemes can easily be implemented on fully unstructured meshes, DG methods are well-suited for problems on complex geometries, which is an essential requirement for real-world applications.

Many practical problems involve time-varying geometries, such as rotor-stator flows, flapping flight or fluid-structure interactions. For these deforming domain problems, a number of solutions have been proposed such as embedded domain methods,[3,4,5] space-time methods[6,7,8] and the Arbitrary Lagrangian-Eulerian (ALE) method.[9,10,11] The popular ALE method can be viewed as a change of variable using a smooth mapping from a fixed reference domain to the moving physical domain, which allows the mesh to change in time and leads to a set of modified equations in the reference domain. The approach is efficient and easy to implement, and is one of the most widely used techniques in particular for CFD problems discretized using high-order accurate methods.[12]

However, a limitation with the ALE method is that in order for the deformation mapping to be smooth, the mesh topology must be fixed which manes that the initial element connectivities have to be kept unchanged throughout the time evolution. This restriction can be severe for large or complex deformations, where remeshing is required to maintain well-shaped elements. In order to transfer the solutions between the original and the recomputed mesh, careful treatment is needed to obtain accuracy and stability. Many interpolation techniques have been proposed, including standard $\mathcal{L}^2$ projections, but in general the accuracy is significantly reduced when frequent remeshing is employed. The projections are straight-forward to formulate and have many desirable properties, but the implementation is complicated and costly for high dimensional unstructured meshes, which limits their practical applications.

In this work, we propose a simple combined approach for solving deforming domain problems with large deformation with high-order accuracy. The method is based on nodal discontinuous Galerkin formulation and an arbitrary Lagrangian-Eulerian framework. The mesh adjustment during the domain motion follows a spring-based technique with local element flipping. Since all the topological changes are local, the corresponding $\mathcal{L}^2$ projections are also local and can be precomputed and easily applied in both 2D and 3D.

In this paper, we first demonstrate our framework on a 2D model problem of an inviscid Euler vortex, where we show that the scheme remains high-order accurate for complex mesh reconfigurations and frequent edge connectivity changes. We also present a 2D laminar flow problem to show the ability of our method to deal with complex domain motions. Finally, we carry out a convergence test in 3D based on a similar Euler-vortex model problem, which again demonstrates the high-order accuracy of the method.

*Ph.D. Candidate, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: lwang@math.berkeley.edu. Student AIAA member.

†Associate Professor, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: persson@berkeley.edu. Senior AIAA member.

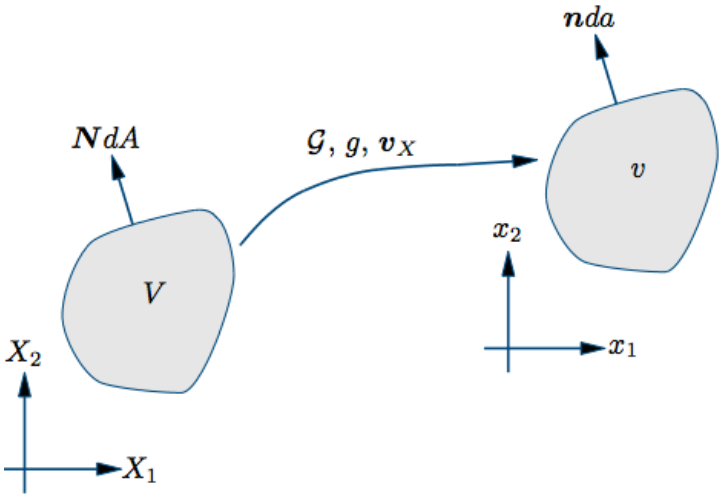


Figure 1. The mapping between the reference domain and the physical domain in the ALE framework.

II. Numerical Scheme

II.A. ALE formulation

As illustrated in Fig. 1, we denote the time-varying domain as $\boldsymbol{v}(t) \in \mathbb{R}^n$. We consider the compressible Navier-Stokes equations in $\boldsymbol{v}(t)$ as a system of conservation laws,

$$\frac{\partial \boldsymbol{u}}{\partial t} - \nabla \cdot \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = 0 \tag{1}$$

where $\boldsymbol{u}$ is the vector of conserved variables and $\boldsymbol{f}$ is the flux function. For compressible flow, $\boldsymbol{f}$ can be split into two parts: the inviscid and viscous contributions, where $\boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{f}^{\text{inv}}(\boldsymbol{u}) + \boldsymbol{f}^{\text{vis}}(\boldsymbol{u}, \nabla \boldsymbol{u})$.

Following the approach in Ref. 12, our Arbitrary Lagrangian-Eulerian (ALE) formulation chooses a reference domain as V, which is fixed at all times, and constructs a smoothly differentiable mapping $\boldsymbol{\mathcal{G}}(\boldsymbol{X}, t) : \boldsymbol{V} \to \boldsymbol{v}(t)$ from the reference domain to the moving domain, such that every point $\boldsymbol{X} \in \boldsymbol{V}$ is mapped to a point $\boldsymbol{\mathcal{G}}(\boldsymbol{X}, t) \in \boldsymbol{v}(t)$. We define the deformation gradient $\boldsymbol{G}$, mapping velocity $\boldsymbol{v_X}$ and mapping Jacobian g as

$$\boldsymbol{G} = \nabla_{\boldsymbol{X}} \boldsymbol{\mathcal{G}}, \quad \boldsymbol{v_X} = \frac{\partial \boldsymbol{\mathcal{G}}}{\partial t}, \quad g = \det \boldsymbol{G} \tag{2}$$

We can then rewrite the conservation law (1) as a new system in the domain $\bar{\mathcal{D}}$,

$$\frac{\partial \boldsymbol{U}}{\partial t} - \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}(\boldsymbol{U}, \nabla_{\boldsymbol{X}} \boldsymbol{U}) = 0 \tag{3}$$

where

$$\boldsymbol{U} = g\boldsymbol{u}, \quad \boldsymbol{F} = g\boldsymbol{G}^{-1}\boldsymbol{f} - \boldsymbol{u}\boldsymbol{G}^{-1}\boldsymbol{v}. \tag{4}$$

More specifically, the corresponding inviscid and viscous parts can be written as,

$$\boldsymbol{F} = \boldsymbol{F}^{\text{inv}} + \boldsymbol{F}^{\text{vis}}, \quad \boldsymbol{F}^{\text{inv}} = g\boldsymbol{G}^{-1}\boldsymbol{f}^{\text{inv}} - \boldsymbol{u}\boldsymbol{G}^{-1}\boldsymbol{v}, \quad \boldsymbol{F}^{\text{vis}} = g\boldsymbol{G}^{-1}\boldsymbol{f}^{\text{vis}} \tag{5}$$

and by the chain rule, we also have,

$$\nabla \boldsymbol{u} = (\nabla_{\boldsymbol{X}}(g^{-1}\boldsymbol{U}))\boldsymbol{G}^{-T} = (g^{-1}\nabla_{\boldsymbol{X}}\boldsymbol{U} - \boldsymbol{U}\nabla_{\boldsymbol{X}}(g^{-1}))\boldsymbol{G}^{-T}. \tag{6}$$

We refer to Ref. 12 for more details on the derivation of this transformation.

II.B. Discontinuous Galerkin Discretization

We apply a standard Discontinuous Galerkin (DG) method to solve the compressible Navier-Stokes equations (3) in the reference domain. A standard procedure[13] is used for the viscous terms, where the system is split into a first-order system of equations:

$$\frac{\partial \boldsymbol{U}}{\partial t} - \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}(\boldsymbol{U}, \boldsymbol{q}) = 0 \tag{7}$$

$$\nabla_{\boldsymbol{X}} \boldsymbol{U} = \boldsymbol{q}. \tag{8}$$

This system is discretized using a standard Galerkin procedure. We introduce a conforming triangulation $\mathcal{T}^h = \{K\}$ of the computational domain $\boldsymbol{V}$ into elements K. On $\mathcal{T}^h$, we define the broken space $\mathcal{V}_T^h$ and Σ_T^h as the spaces of functions whose restriction to each element K are polynomial functions of degree at most $p \geq 1$,[14]

$$\mathcal{V}_T^h = \{\boldsymbol{v} \in [L^2(\bar{\mathcal{D}})]^n \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^n \ \ \forall K \in \mathcal{T}^h\}, \tag{9}$$

$$\Sigma_T^h = \{\boldsymbol{\sigma} \in [L^2(\bar{\mathcal{D}})]^{n\times m} \mid \boldsymbol{\sigma}|_K \in [\mathcal{P}_p(K)]^{n\times m} \ \ \forall K \in \mathcal{T}^h\} \tag{10}$$

where n is the spatial dimension, m is the number of components in solution $\boldsymbol{U}$, and $\mathcal{P}_p(K)$ denotes the space of polynomials of degree at most $p \geq 1$ on K. Then the DG formulation for equations (7) and (8) becomes: find $\boldsymbol{U}^h \in \mathcal{V}_T^h$ and $\boldsymbol{q}^h \in \Sigma_T^h$ such that for each $K \in \mathcal{T}^h$, we have

$$\begin{aligned}\int_K \frac{\partial \boldsymbol{U}^h}{\partial t} \cdot \boldsymbol{v}^h dx - \int_K \boldsymbol{F}^{\text{inv}}(\boldsymbol{U}^h) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^h \, dx + \oint_{\partial K} (\boldsymbol{F}^{\text{inv}} \cdot \boldsymbol{n}) \cdot \boldsymbol{v}^h \, ds \\ = -\int_K \boldsymbol{F}^{\text{vis}}(\boldsymbol{U}^h, \boldsymbol{q}^h) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^h \, dx + \oint_{\partial K} (\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}) \cdot \boldsymbol{v}^h \, ds, \qquad \forall \boldsymbol{v}^h \in \mathcal{V}_T^h\end{aligned} \tag{11}$$

$$\int_K \boldsymbol{q}^h : \boldsymbol{\sigma}^h \, dx = -\int_K \boldsymbol{U}^h \cdot (\nabla_{\boldsymbol{X}} \cdot \boldsymbol{\sigma}^h) \, dx + \oint_{\partial K} (\boldsymbol{U}^h \otimes \boldsymbol{n}) : \boldsymbol{\sigma}^h \, ds, \qquad \forall \boldsymbol{\sigma}^h \in \Sigma_T^h. \tag{12}$$

For numerical fluxes, we use Roe's method[15] to approximate the inviscid flux and we treat the viscous flux using the Compact Discontinuous Galerkin (CDG) method proposed by Peraire and Persson,[16] which is a modified LDG method to obtain a compact and sparser stencil with improved stability properties. By equations (11) and (12), a non-linear semi-discrete system is formulated which we solve using a parallel high-order diagonally implicit Runge-Kutta (DIRK) solver. The details on this derivation can be found in Refs. 17, 18.

III. Mesh Motion and Edge Flipping

As the physical domain deforms, the mesh has to be moved accordingly. In order to find a well-conditioned smooth mapping, a couple of local mesh operations are introduced to adjust the mesh topology and maintain high element qualities. First, we move the boundary nodes rigidly according to the prescribed geometry motion. This approach is sufficient for our examples, but in general it might be necessary to, for example, redistribute the boundary nodes if the boundary deformations are large. The element qualities generally decrease after the boundary nodes are moved, and we improve the mesh by moving the interior mesh nodes using the DistMesh scheme.[19]

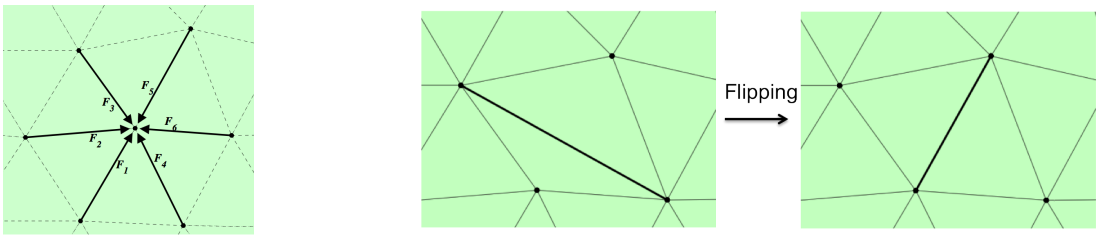


Figure 2. Force-based smoothing and edge flipping. The left plot shows the net force exerted on one node, and the right plot gives a 2D example of edge flipping for improving the triangle qualities.

As illustrated in Fig. 2 (left), the movement of the interior nodes is driven by repulsive forces from each attached edge, which depend on the edge length l and an equilibrium length l_0:

$$|\boldsymbol{F}(\boldsymbol{l})| = \begin{cases} k(l - l_0) & \text{if } l \geq l_0, \\ 0 & \text{if } l < l_0, \end{cases} \tag{13}$$

where k is a constant (corresponding to Hooke's constant for a linear elastic spring). The equilibrium length l_0 has to be set manually. For a uniform mesh it can be a constant, but for more general adaptive meshes it can be given by a specified mesh size function. In addition, a scaling is applied to ensure that most edges are under compression.[19]

For each node $\boldsymbol{p}$, denote by $\boldsymbol{F}(\boldsymbol{p})$ the sum of all forces from edges connected to $\boldsymbol{p}$. Then we iteratively update the node position by

$$\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} + \delta \boldsymbol{F}(\boldsymbol{p}^{(n)}) \tag{14}$$

where δ is an appropriate pseudo time step. The iterations are repeated until an approximate force equilibrium is obtained.

When the time-varying domain undergoes large deformations, node movements are usually not sufficient to obtain high-quality elements and avoid element inversion. For such cases, we perform local connectivity changes to improve the mesh qualities.[20] For a triangular mesh in 2D, this can be done using edge swapping operations[21] as shown in Fig. 2 (right), where two adjacent triangles flip their shared edge and produce two new triangles sharing the new edge. Note that in 2D, during the process described, the number of nodes, edges and elements remain unchanged. Furthermore, all elements have the same edge connections except for the ones involved in edge swapping.

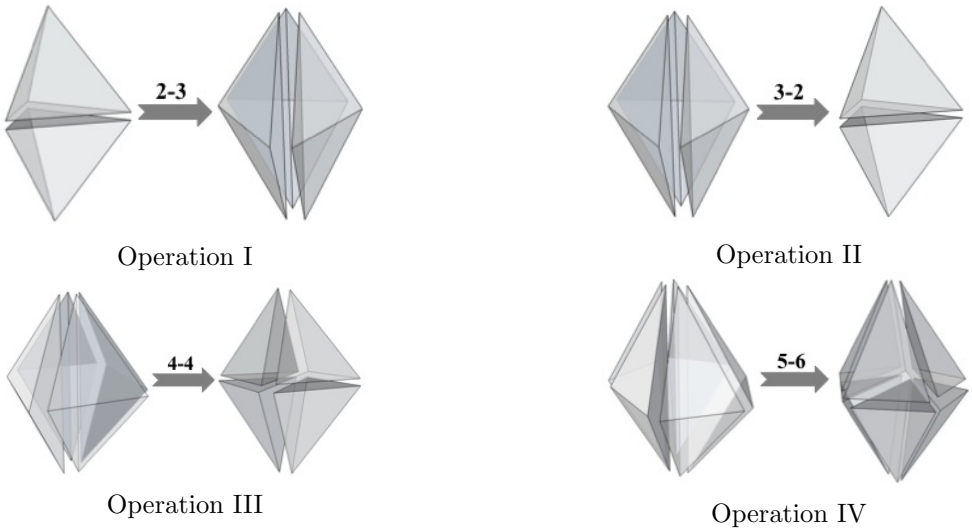


Figure 3. Change of local element connectivity in 3D. The numbers of old and new elements are shown above the arrows.

Extending this moving mesh strategy to 3D cases, we use the same force-based smoothing technique above to update the positions of interior nodes. However, the local connectivity changes need substantially more complicated operations in order to address the element distortion in 3D. For example, consider a pair of tetrahedra sharing a common surface. These can be transformed into three new tetrahedra by removing the shared surface and introducing a new interior edge (Operation I in Fig. 3). Inversely, we can also take a small group of tetrahedra sharing a common edge and transform them into tetrahedra sharing a common surface (Operation II, III, IV in Fig. 3). In this work, we use all these four local connectivity operations, which are shown in Fig. 3. Based on our results, these operations are sufficient to make the 3D mesh movement robust and keep the tetrahedra well-shaped. Note that although the number of elements might change during this process, all the connectivity operations are still local and limited to a small group of elements.

IV. $\mathcal{L}^2$ Projection

In many ALE simulations, the mesh is moved without connectivity changes for as many steps as possible and the transformed Navier-Stokes equations are solved in the reference domain based on well-conditioned smoothly differentiable mappings $\boldsymbol{\mathcal{G}}(\boldsymbol{X}, t)$. But for problems involving large domain deformation, the mesh will eventually become poorly shaped due to the appearance of nearly inverted elements. To address this, remeshing can be used to replace the old triangulation $\mathcal{T}^h = \{K\}$ by a new one $\tilde{\mathcal{T}}^h = \{\tilde{K}\}$ in the reference domain $\boldsymbol{V}$ such that the mapping image of the new mesh in the physical domain $\boldsymbol{v}(t)$ maintains high quality.

Consider the broken spaces $\mathcal{V}_{\tilde{T}}^h$ and $\Sigma_{\tilde{T}}^h$, which are similar to the spaces defined in (9) and (10) but associated with the triangulation $\tilde{\mathcal{T}}^h$. Recall that in the standard DG procedure, the numerical solution $\boldsymbol{U}^h$ is written as a linear combination of basis functions $\{\phi_1, \phi_2, \ldots, \phi_N\}$ on the broken space $\mathcal{V}_T^h$,

$$\boldsymbol{U}^h = \sum_{i=1}^{N} \boldsymbol{U_i} \phi_i. \tag{15}$$

In order to continue the time-stepping after remeshing, a new approximate solution to $\boldsymbol{U}^h \in \mathcal{V}_T^h$ is required. Denote the basis functions of $\mathcal{V}_{\tilde{T}}^h$ by $\{\tilde{\phi}_1, \tilde{\phi}_2, \ldots, \tilde{\phi}_{\tilde{N}}\}$, and the $\mathcal{L}^2$ projection of $\boldsymbol{U}^h$ onto $\mathcal{V}_{\tilde{T}}^h$ by $\tilde{\boldsymbol{U}}^h$. This projection satisfies the following property: for each $\tilde{K} \in \tilde{\mathcal{T}}^h$,

$$\int_{\tilde{K}} (\boldsymbol{U}^h - \tilde{\boldsymbol{U}}^h) \tilde{\phi}_i dx = 0, \qquad i = 1, \ldots, \tilde{N}. \tag{16}$$

More precisely, if

$$\tilde{\boldsymbol{U}}^h = \sum_{i=1}^{\tilde{N}} \tilde{\boldsymbol{U}}_{\boldsymbol{i}} \tilde{\phi}_i, \tag{17}$$

then

$$\sum_{i=1}^{\tilde{N}} \int_{\tilde{K}} \tilde{\boldsymbol{U}}_{\boldsymbol{i}} \tilde{\phi}_i \tilde{\phi}_j dx = \sum_{i=1}^{N} \int_{\tilde{K}} \boldsymbol{U_i} \phi_i \tilde{\phi}_j dx = \sum_{i=1}^{N} \sum_{K \in \mathcal{T}^h} \int_{\tilde{K} \cap K} \boldsymbol{U_i} \phi_i \tilde{\phi}_j dx, \qquad j = 1, \ldots, \tilde{N}. \tag{18}$$

The equations (18) result in a linear system,

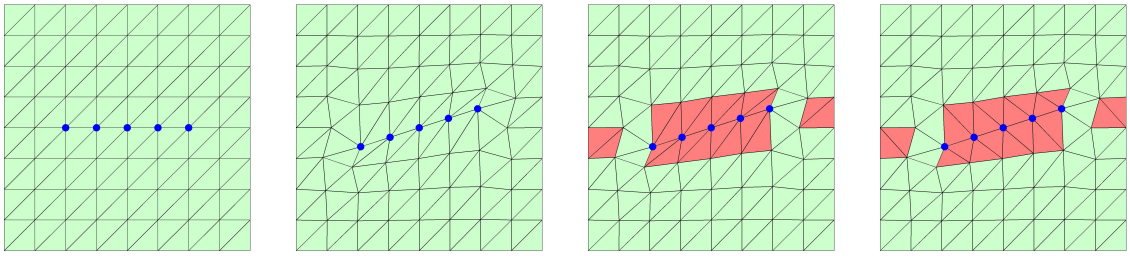
$$M \tilde{\boldsymbol{U}}^h = P \boldsymbol{U}^h \tag{19}$$

where

$$M_{j,i} = \int_{\tilde{K}} \tilde{\phi}_i \tilde{\phi}_j dx, \qquad P_{j,i} = \sum_{K \in \mathcal{T}^h} \int_{\tilde{K} \cap K} \phi_i \tilde{\phi}_j dx. \tag{20}$$

Equation 20 can be solved for $\tilde{\boldsymbol{U}}^h$ and used as the transferred solution to resume the time-stepping process on the new mesh.

However, since both ϕ_i and $\tilde{\phi}_i$ are discontinuous element-wise polynomials, the second equal sign of equation (18) indicates that each new element $\tilde{K}$ must be split into a collection of disjoint intersections $\tilde{K} \cap K$ for some $K \in \mathcal{T}^h$. Dealing with these projections for two arbitrary meshes poses several difficulties. First, an efficient and robust cut-cell algorithm is required to split the elements, which is a fairly involved procedure for 3D tetrahedral meshes; second, the intersections of simplex elements can have many possible shapes, and a sophisticated quadrature technique for arbitrary polygons or polyhedra must be incorporated for the evaluation of the volume integrals. In our proposed method, by taking advantage of our mesh moving techniques with local mesh operators and discontinuous element-wise polynomial solutions of the DG method, we can use a simpler and more efficient algorithm to handle the large deformation using local $\mathcal{L}^2$ projections.



Phase I Phase II Phase III Phase IV

Figure 4. An illustration of the process in algorithm 1. Phase I: Select blue nodes to move; Phase II: Update interior nodes by force-based smoothing; Phase III: Select pairs of elements in red for flipping; Phase IV: Change the local connectivity within each pair of red elements.

V. ALE Method with Local $\mathcal{L}^2$ Projections

The details of our method are described in algorithm 1. Recall that our mesh improvement strategy has two stages. The first stage only moves the nodes without any mesh topology changes. Therefore, it allows us to construct a smooth map and solve for one time step using a standard ALE method. Next, if needed, the second stage is to change the connectivity locally. In this stage, all the node positions are fixed before and after the connectivity change, so we can conduct our local $\mathcal{L}^2$ Projections within each group of flipped elements and transfer the solution very efficiently. The algorithm repeats the two stages above and evolves the solution throughout the whole time period.

In 2D cases, it is clear that each local flipping operator replaces two old elements by two new elements, as shown in Fig. 5. In other words, the old pair $\{K_1, K_2\}$ and the new pair $\{\tilde{K}_1, \tilde{K}_2\}$ share the same group of vertices, so 4 sub-triangles $\{\tilde{K}_1 \cap K_1, \tilde{K}_1 \cap K_2, \tilde{K}_2 \cap K_1, \tilde{K}_2 \cap K_2\}$ are always formed as their intersections. The integrals in equation (18) are straightforward to evaluate based on these four sub-triangles and thus the components of solution $\boldsymbol{U}^h$ within two old elements are transferred into those of the new solution $\tilde{\boldsymbol{U}}^h$ with respect to the two new elements. Except for the pairs of flipped elements, all other components of $\boldsymbol{U}^h$ and $\tilde{\boldsymbol{U}}^h$ are unchanged. A 2D example is shown in Fig. 4 to demonstrate how one step of our algorithm performs.

In 3D, similarly to the 2D cases, we also need to split old and new tetrahedra into sub-tetrahedra at the second stage and then implement the local $\mathcal{L}^2$ projections by evaluating the integrals of equation (18) in each of the sub-tetrahedra. This tetrahedral splitting is significantly more complicated that in 2D. However, since the connectivity changes are limited to a small group of tetrahedra, it is straightforward to analyze the resulting geometric structures and to consider all possible cases of splitting for each operation.

From Fig. 3, we can see that for any 3D operation, the old and the new groups of elements provide two different triangulations of a polyhedron with a special structure, which has a polygon (triangle, quadrilateral or pentagon) in the middle with one node on each side (top and bottom in the figure). By connecting these top and bottom nodes, we can find an intersection between the middle polygon and the connecting line (marked with a red cross in each plot of Fig. 6). We can then consider the splitting of each group of tetrahedra into sub-tetrahedra case by case, and the resulting triangulation gives all the sub-tetrahedra needed for computing the integrals in equation (18).

- For Operation I and II, connect the intersection to each vertex of the middle triangle, and connect all the middle nodes to the top and the bottom nodes.
- For Operation III, triangulate the middle quadrilateral by assigning a diagonal. The intersection of the connecting line is then located in one of the two resultant triangles. Connect this intersection point to each vertex of the middle quadrilateral, and finally connect all the middle nodes to the top and to the bottom nodes.
- For Operation IV, triangulate the middle pentagon by assigning two diagonals. The intersection of the connecting line is then in the middle triangle or in one of the two side triangles. Connect this intersection point to each vertex of the middle quadrilateral, and for the latter case, add one additional line to complete a valid triangulation of the pentagon. Finally connect all the middle nodes to the top and the bottom nodes.

Algorithm 1 Discontinuous Galerkin ALE Method with Local $\mathcal{L}^2$ Projections

Require: Triangulation $\mathcal{T}^h$ on $\bar{\mathcal{D}}$ and initial solution $\boldsymbol{U}^{h,t_0}$ at t_0
Require: Time step Δt and mesh quality threshold δ
Ensure: Solution $\boldsymbol{U}^{h,t_i}$ for each time step t_i until time T
  while $t_0 < t_i \leq T$ **do**
    Move the mesh by the DistMesh algorithm[19]
    Compute deformation gradient G, mapping velocity $\boldsymbol{v}$ and mapping Jacobian g
    Solve $\boldsymbol{U}^{h,t_i}$ by the DG method with ALE framework
    if min quality of $K \in \mathcal{T}^h < \delta$ **then**
      Create $\tilde{\mathcal{T}}^h$ by local element flipping
      Solve for $\tilde{\boldsymbol{U}}^{h,t_i}$ by local $\mathcal{L}^2$ projections
      $\mathcal{T}^h \leftarrow \tilde{\mathcal{T}}^h$
      $\boldsymbol{U}^{h,t_i} \leftarrow \tilde{\boldsymbol{U}}^{h,t_i}$
    end if
  end while

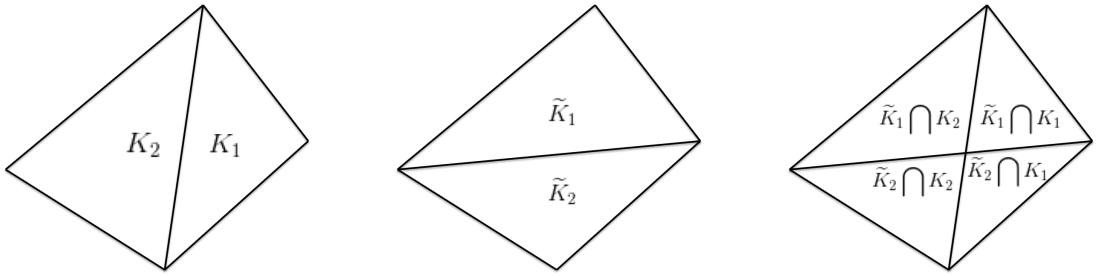


Figure 5. Local element splitting in 2D.

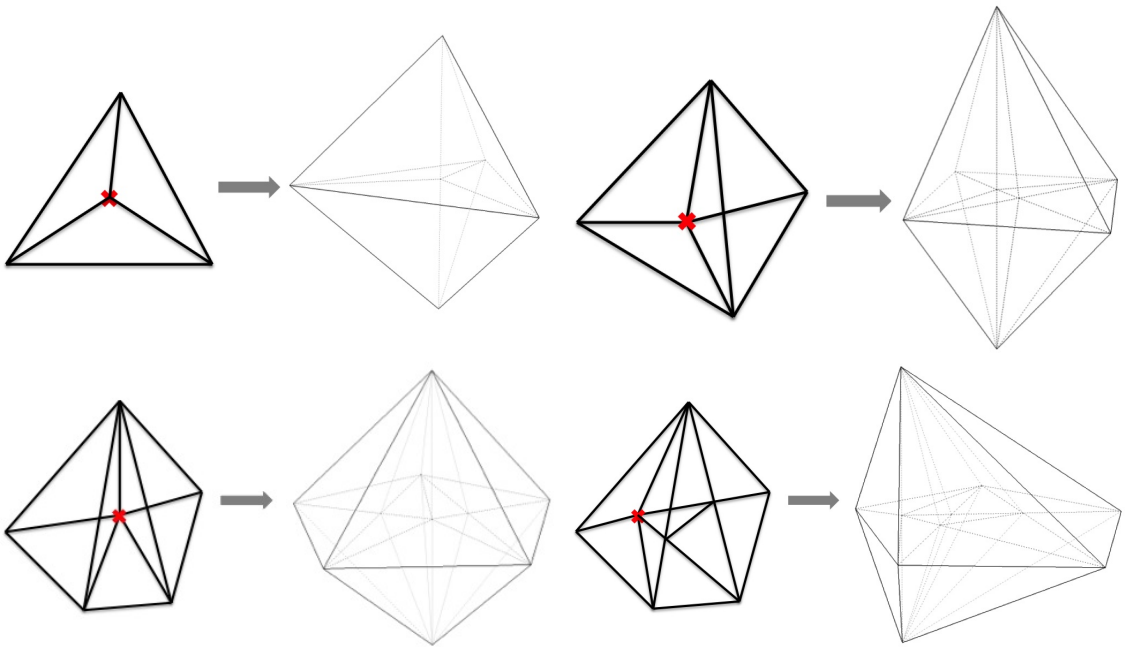


Figure 6. Tetrahedral splitting for 3D operations. In each plot, the triangulation of the middle polygon is on the left and the corresponding 3D triangulation is on the right. The red cross is the intersection between the middle polygon and the connecting line between the top and the bottom nodes. Note that in the last case, one additional edge is needed in addition to the diagonals and edges connecting intersection and vertices, in order to complete a valid triangulation of the middle polygon.

VI. Numerical Test

VI.A. Euler Vortex

In our first numerical experiment, we solve for a propagating compressible Euler vortex in a 20-by-20 square domain and make a convergence test to demonstrate the high order accuracy of our local ALE approach with frequent local $\mathcal{L}^2$ projections in 2D.

The vortex is initially centered at $(x_0, y_0) = (-4, 4)$ and moves with the free-stream at an angle $\theta = \pi/4$ with respect to the x-axis. The analytical solution at (x, y, t) is given by

$$u = u_\infty(\cos\theta - \frac{\epsilon((y-y_0)-\bar{v}t)}{2\pi r_c}\exp(\frac{f(x,y,t)}{2})) \tag{21}$$

$$v = u_\infty(\sin\theta + \frac{\epsilon((x-x_0)-\bar{u}t)}{2\pi r_c}\exp(\frac{f(x,y,t)}{2})) \tag{22}$$

$$\rho = \rho_\infty(1 - \frac{\epsilon^2(\gamma-1)M_\infty^2}{8\pi^2}\exp(f(x,y,t)))^{\frac{1}{\gamma-1}} \tag{23}$$

$$p = p_\infty(1 - \frac{\epsilon^2(\gamma-1)M_\infty^2}{8\pi^2}\exp(f(x,y,t)))^{\frac{\gamma}{\gamma-1}} \tag{24}$$

where $f(x,y,t) = (1-((x-x_0)-\bar{u}t)^2 - ((y-y_0)-\bar{v}t)^2)/r_c^2$, $M_\infty = 0.5$ is the Mach number, $\gamma = c_p/c_v$, and u_∞, p_∞, ρ_∞ are free-stream velocity, pressure and density, respectively. Moreover, $\bar{u}$ and $\bar{v}$ are the Cartesian components of the free-stream velocity with $\bar{u} = u_\infty\cos\theta$ and $\bar{v} = u_\infty\sin\theta$. The parameter $\epsilon = 3$ is the strength of the vortex and $r_c = 1.5$ is its size.

As shown in Fig. 7, we choose a centered area of an unstructured mesh (colored in yellow) and rotate this part with a constant angular velocity $\omega = \frac{\pi}{6}$. As the outside green part stays unchanged all the time, the central rotation induces large mesh deformations so that the global unstructured mesh has to be adjusted by flipping some of the elements in the layer of mesh (colored blue) between the fixed and the moving mesh.

We solve the Euler equations using our proposed algorithm on this moving mesh until time $T = 10$ with timestep $\Delta t = 0.01$. Then we compare the numerical results with the given analytical solutions. Note that Δt is chosen much smaller than the spatial discretization size h to ensure that the temporal numerical errors are negligible.

We carry out the convergence test for a range of spatial mesh sizes h and polynomial degrees p, and compute inf-norms of the errors. From the resulting convergence plot it is clear that the optimal $O(h^{p+1})$ orders of convergence are obtained.

VI.B. Pitching Tandem Airfoils

Next, we consider a compressible Navier-Stokes simulation similar to the one studied in Ref.[8] It consists of two pitching NACA0012 airfoils with chord length $c = 1$ in a rectangular domain. As shown in Fig. 8, at the initial time $t = 0$ the two foils have a zero pitching angle and are aligned on the horizontal axis close to each other. The distance between the trailing edge of the first foil and the leading edge of the second foil is $d = 0.1$. The two foils are both treated as rigid bodies and rotated around the points $p = c/3$ to the right of their leading edges. The rotation follows a prescribed harmonic function as

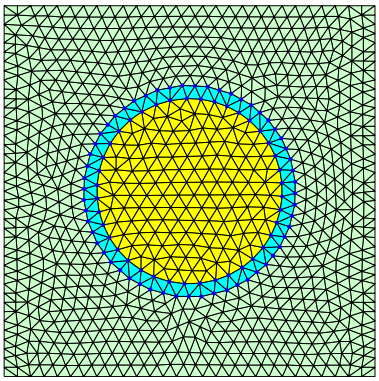
$$\theta = A\sin(-2\pi f t) \tag{25}$$

where $A = \pi/6$ and $f = 0.05$. The flow has Mach number 0.2 and Reynolds number 3000.

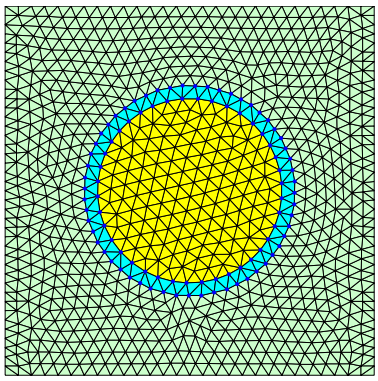
As the two foils are placed very close and rotated based on the same harmonic function, the mesh quality decays very rapidly if only the position of the nodes are updated. However, using our simple $\mathcal{L}^2$ projection strategy, the moving mesh remain well shaped and avoids inverted elements. We implement our high-order DG ALE method with polynomial degree $p = 2$ and linear element geometries. A couple of sample entropy plots are shown in Fig. 9, and a plot of the drag and the lift forces is shown in Fig. 9.

VI.C. Euler Vortex in 3D

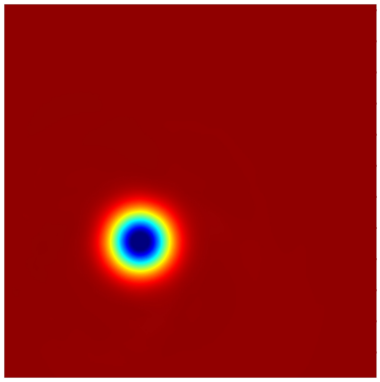
The last example is used to demonstrate the ability of our moving mesh strategy to adjust the mesh quality in 3D and achieve high-order accuracy. Similar to the first example, we propagate an Euler vortex cylinder



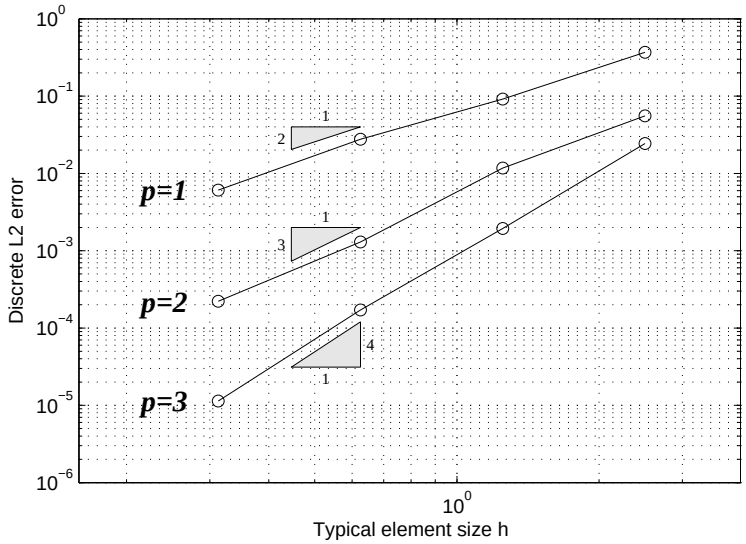
Moving Mesh at $t = 0$



Moving Mesh of $t = 1.5$



Sample pressure plot of Euler Vortex



Convergence Plot

Figure 7. Convergence Test for the Euler Vortex Problem. The upper left and middle plots are two sample meshes at $t = 0$ and $t = 1.5$. All the connectivity changes happen in the blue layer of triangles; A sample entropy plot of this Euler vortex problem is on the upper right.

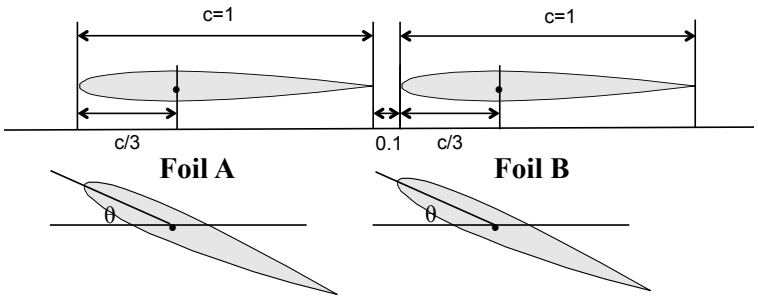
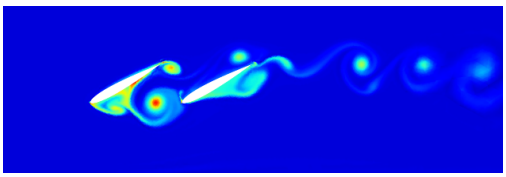
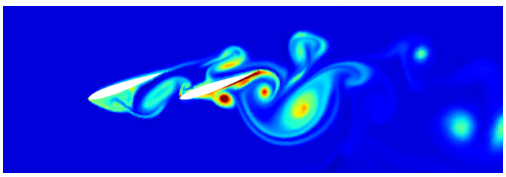


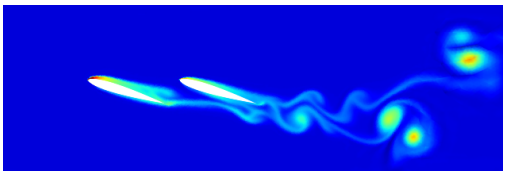
Figure 8. Schematics of the pitching tandem airfoil problem.



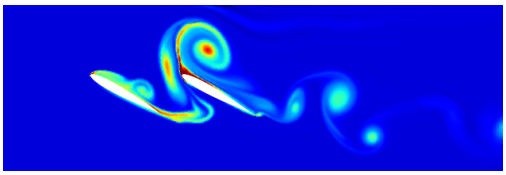
Entropy Plot at $t = 4.0$



Entropy Plot at $t = 8.0$



Entropy Plot at $t = 12.0$



Entropy Plot at $t = 16.0$

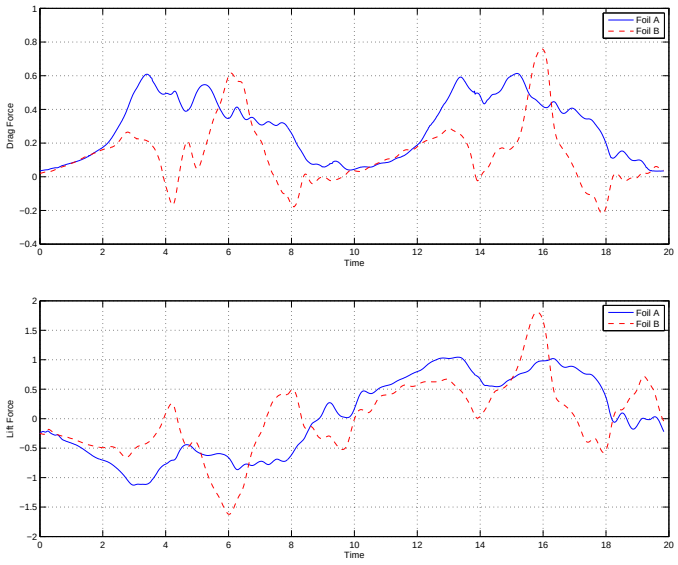
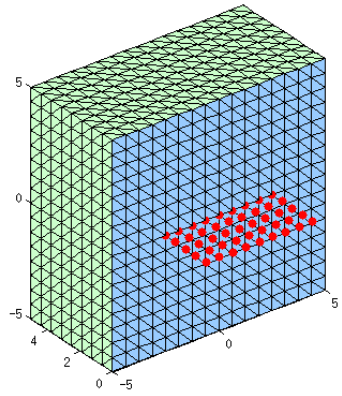


Figure 9. Numerical results for tandem foils. On the top, there are solution fields for the pitching tandem airfoils problem (entropy of the flow at 4 time instances). The drag and lift forces on the pitching tandem NACA0012 airfoils as a function of time are shown at the bottom.

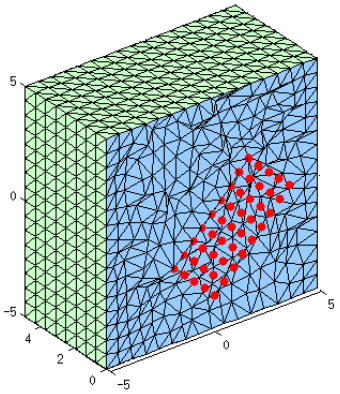
in a $10 \times 10 \times 10$ cube (as shown in Fig. 10). The analytical solutions at (x, y, z, t) are the same as in equations (21)-(24) except that we set $(x_0, y_0) = (-2, -2)$.

The mesh of the cube is initially generated as a 3D Cartesian grid. To show the effectiveness of our moving mesh approach, we induce large mesh deformations by rigidly rotating a set of interior mesh nodes with respect to the y-axis, which are showed in red in Fig. 10. These red nodes are chosen as all the mesh nodes of the initial grid with $x \in [-2.5, 2.5]$, $y \in [-2.5, 2.5]$ and $z = 0$. The Euler vortex travels from $t = 0$ to $t = T = \sqrt{3^2 + 3^2}$ and the set of red nodes are rotated about 30 degree throughout the time interval $[0, T]$. With these rigid motions, the mesh quality decays rapidly and will eventually be unacceptable without mesh adjustments. On the contrary, when our moving mesh is employed, the mesh quality can be retained by smoothing mesh nodes and locally flipping the tetrahedra. In Fig. 10, we can clearly see that the initially structured mesh quickly becomes fully unstructured and that it respects the motion of red nodes. All element qualities remain acceptably high throughout the process.

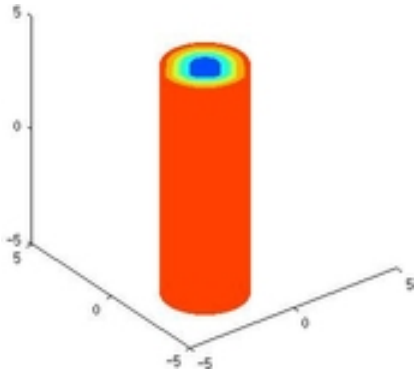
Finally, we solve the 3D Euler vortex cylinder problem with different spatial mesh sizes h and degrees of polynomial p. As a benchmark, the same test is also carried out with the initial mesh fixed for all time steps. The convergence plot in Fig. 10 illustrates that the test with moving meshes achieves the same optimal $O(h^{p+1})$ order of accuracy as that with fixed mesh, even with frequent elements flipping in 3D.



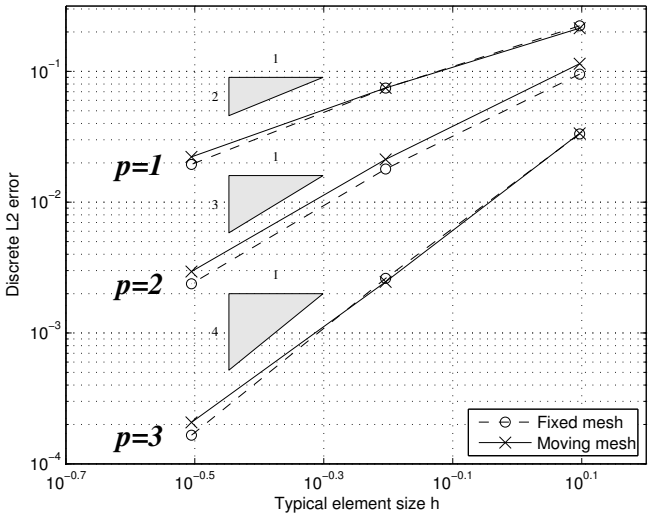
Moving Mesh at $t = 0$



Moving Mesh of $t = T$



Sample solution plot of Euler Vortex



Convergence Plot

Figure 10. Convergence Test for the 3D Euler Vortex Problem. The upper left and middle plots are two sample meshes at the initial and the final time, respectively. These meshes are generated on a $10 \times 10 \times 10$ cube. The blue faces show a cross-section of the tetrahedral mesh and the green faces are the outside surfaces of the cube. The red nodes are rotated rigidly. The right plot shows some sample pressure isosurfaces.

VII. Conclusions

We have presented a high-order Discontinuous Galerkin Method for solving compressible flows in deforming domain using an arbitrary Lagrangian-Eulerian framework. The techniques focus on large deformation problem, where a moving mesh is adjusted by local mesh operations. A local $\mathcal{L}^2$ projection is introduced to transfer the solution between the meshes, which is efficient and easy to implement due to its local character. Several numerical tests are included to show the effectiveness of the method. Our future work will mostly be focused on using the proposed method for practical simulations, in particular for 3D applications.

References

[1]Cockburn, B. and Shu, C.-W., "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems," *J. Sci. Comput.*, Vol. 16, No. 3, 2001, pp. 173–261.

[2]Peraire, J. and Persson, P.-O., *Adaptive High-Order Methods in Computational Fluid Dynamics*, Vol. 2 of *Advances in CFD*, chap. 5 – High-Order Discontinuous Galerkin Methods for CFD, World Scientific Publishing Co., 2011.

[3]Colella, P., Graves, D. T., Keen, B. J., and Modiano, D., "A Cartesian grid embedded boundary method for hyperbolic conservation laws," *J. Comput. Phys.*, Vol. 211, No. 1, 2006, pp. 347–366.

[4]Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," *38th AIAA Aerospace Sciences Meeting and Exhibit*, 2000, AIAA-2000-0808.

[5]Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *35th AIAA Aerospace Sciences Meeting and Exhibit*, 1997, AIAA-97-0196.

[6]Hughes, T. J. and Hulbert, G. M., "Space-time finite element methods for elastodynamics: formulations and error estimates," *Computer methods in applied mechanics and engineering*, Vol. 66, No. 3, 1988, pp. 339–363.

[7]Hulbert, G. M. and Hughes, T. J., "Space-time finite element methods for second-order hyperbolic equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 84, No. 3, 1990, pp. 327–348.

[8]Wang, L. and Persson, P.-O., "A High-Order Discontinuous Galerkin Method with Unstructured Space-Time Meshes for Domains with Large Deformations," *Comput. & Fluids*, to appear.

[9]Donea, J., "Arbitrary Lagrangian-Eulerian finite element methods," *Computational methods for transient analysis(A 84-29160 12-64). Amsterdam, North-Holland, 1983,*, 1983, pp. 473–516.

[10]Farhat, C. and Geuzaine, P., "Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids," *Comput. Methods Appl. Mech. Engrg.*, Vol. 193, No. 39-41, 2004, pp. 4073–4095.

[11]Lomtev, I., Kirby, R. M., and Karniadakis, G. E., "A discontinuous Galerkin ALE method for compressible viscous flows in moving domains," *J. Comput. Phys.*, Vol. 155, No. 1, 1999, pp. 128–159.

[12]Persson, P.-O., Bonet, J., and Peraire, J., "Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains," *Comput. Methods Appl. Mech. Engrg.*, Vol. 198, 2009, pp. 1585–1595.

[13]Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D., "Unified analysis of discontinuous Galerkin methods for elliptic problems," *SIAM J. Numer. Anal.*, Vol. 39, No. 5, 2001/02, pp. 1749–1779.

[14]Hesthaven, J. S. and Warburton, T., *Nodal discontinuous Galerkin methods*, Vol. 54 of *Texts in Applied Mathematics*, Springer, New York, 2008, Algorithms, analysis, and applications.

[15]Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Comput. Phys.*, Vol. 43, No. 2, 1981, pp. 357–372.

[16]Peraire, J. and Persson, P.-O., "The compact discontinuous Galerkin (CDG) method for elliptic problems," *SIAM J. Sci. Comput.*, Vol. 30, No. 4, 2008, pp. 1806–1824.

[17]Persson, P.-O. and Peraire, J., "Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations," *SIAM J. Sci. Comput.*, Vol. 30, No. 6, 2008, pp. 2709–2733.

[18]Persson, P.-O., "Scalable Parallel Newton-Krylov Solvers for Discontinuous Galerkin Discretizations," *47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida*, 2009, AIAA-2009-606.

[19]Persson, P.-O. and Strang, G., "A Simple Mesh Generator in MATLAB," *SIAM Review*, Vol. 46, 2004.

[20]Alauzet, F., "Efficient moving mesh technique using generalized swapping," *Proceedings of the 21th International Meshing Roundtable*, Sandia Nat. Lab., 2012, pp. 17–37.

[21]Lahiri, S. K., Bonet, J., and Peraire, J., "A variationally consistent mesh adaptation method for triangular elements in explicit Lagrangian dynamics," *Internat. J. Numer. Methods Engrg.*, Vol. 82, No. 9, 2010, pp. 1073–1113.