Fluid Dynamics and Co-located Conferences
June 24-27, 2013, San Diego, CA
21st AIAA Computational Fluid Dynamics Conference

A Discontinuous Galerkin Method for the Navier-Stokes Equations on Deforming Domains using Unstructured Moving Space-Time Meshes

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

We describe a high-order accurate space-time discontinuous Galerkin (DG) method for solving compressible flow problems on two-dimensional moving domains with large deformations. The DG discretization and space-time numerical fluxes are formulated on a three-dimensional space-time domain. The scheme is implicit, and we solve the resulting non-linear systems using a parallel Newton-Krylov solver. Instead of remeshing when the mesh elements are deformed, we use local mesh operations such as node movement and edge flips to improve the mesh at each time step. We then produce a globally conforming space-time mesh using an efficient algorithm based on element extrusions between two consecutive spatial meshes. In this way, no additional nodes are inserted for each space-time mesh slab except for those on the spatial meshes. We show various numerical examples with complex domain deformations to illustrate both the accuracy and the capabilities of our method.

I. Introduction

Discontinuous Galerkin (DG) methods have received much attention during the last decade due to their ability to produce stable and high-order accurate discretizations of conservation laws on fully unstructured meshes.[1,2] In particular for challenging fluid problems, it is widely believed that the low dissipation of the DG schemes make them ideal for the simulation of turbulent flows with complex vortical structures and non-linear interactions.

Many practical applications involve time-varying geometries, such as rotor-stator flows, flapping flight or fluid-structure interactions. For these deforming domains, a number of solutions have been proposed. The Arbitrary Lagrangian-Eulerian (ALE) formulation allows for the computational mesh to deform in time and compensates for this by modifying the equations.[3,4,5] The method is widely used, but when the domain deformation is large and/or complex it is difficult to deform the mesh without element inversion. Remeshing is then commonly employed, which introduces errors during the solution transfer between the old and the new meshes. In addition, special care needs to be taken to ensure the satisfaction of the so-called geometric conservation law (GCL).

As an alternative, the so-called space-time DG methods are fully consistent discretizations that allow for arbitrary changes of the domain in both space and time.[6,7,8,9,10] The method essentially treats the time-dependency with the same technique as the spatial terms, but exploiting the causal nature to improve the efficiency. Much of the previous work on space-time DG methods is based on meshes with structured prismatic extrusions of the spatial elements (e.g. in refs. 11, 12, 13, 14, 15, 16). These schemes have many attractive properties, but suffer from similar limitations as the ALE method with element inversion for large domain deformations. However, the space-time formulations do allow for fully unstructured meshes in both space and time, and provided that appropriate meshes can be generated this is a competitive approach for problems with large domain deformations. Some previous work on this include refs. 17, 18, 19, 20, where

*Ph.D. Candidate, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: lwang@math.berkeley.edu. AIAA student Member
†Assistant Professor, Department of Mathematics, University of California, Berkeley, Berkeley CA 94720-3840. E-mail: persson@berkeley.edu. AIAA Member.

several unstructured space-time approaches are presented. However, these results largely rely on remeshing for each spatial domain and an ideal tetrahedral mesh generator. Work based on local mesh modifications include ref. 21, where an efficient moving-mesh technique based on face swapping was developed, and ref. 22, where a changing-topology finite-volume based ALE schemes was presented.

In this work, we demonstrate a fully unstructured space-time mesh generation procedure and the solution of the Navier-Stokes equations using an implicit space-time discontinuous Galerkin method. We use the DistMesh algorithm for the mesh motion and deformation,[23] and construct the space-time elements for each layer of timesteps using a local construction.[24] The resulting scheme can essentially handle any type of domain deformations, even with topological changes. The order of accuracy can be arbitrarily high in both space and time, provided suitable curved meshes can be generated.

This paper is organized as follows: First, we derive our space-time DG formulation for the compressible Navier-Stokes equations. Next we introduce our local mesh operations and the combinatorial algorithm for generation of a globally consistent space-time mesh. In Section IV, three numerical tests are presented. We first demonstrate our framework on a model 2D problem of an inviscid Euler vortex, where we show that the scheme remains high-order accurate even for complex mesh reconfigurations. Finally, we present two 2D laminar flow problems, where we demonstrate the capability of handling complex deformations that cannot be solved using standard ALE techniques without remeshing.

II. Space-Time Discontinuous Galerkin Scheme

II.A. The Compressible Navier-Stokes Equations and its Space-Time Formulation

Consider the conservation form of the compressible Navier-Stokes equations[2] on a time-dependent domain from time $t = 0$ to $t = T$ for some fixed time $T > 0$. Let (x_1, x_2) be the spatial variables. We define $\Omega^t \in \mathbb{R}^2$ as this flow domain at time t and when $t_1 < t_2$, $\Omega[t_1, t_2] = \{(x_1, x_2, t) \mid t_1 \le t \le t_2, \ \ (x_1, x_2) \in \Omega^t\}$. Denote $\nabla_{\boldsymbol{X}} = (\partial_{x_1}, \partial_{x_2})$ as standard 2D spatial gradient operator and then we write the system as:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}^{\boldsymbol{inv}}(\boldsymbol{u}) = \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}^{\boldsymbol{vis}}(\boldsymbol{u}, \nabla_{\boldsymbol{X}} \boldsymbol{u}), \tag{1}$$

with some appropriate boundary conditions imposed on the domain boundary $\partial\Omega[0, T]$ and initial condition on Ω^0. Here,

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{pmatrix}, \qquad \boldsymbol{F}_1^{\text{inv}}(u) = \begin{pmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ u_1(\rho E + p) \end{pmatrix}, \qquad \boldsymbol{F}_1^{\text{vis}}(u, \nabla_{\boldsymbol{X}} \boldsymbol{u}) = \begin{pmatrix} 0 \\ \tau_{11} \\ \tau_{12} \\ \tau_{11} u_1 + \tau_{12} u_2 - \Theta_1 \end{pmatrix}$$

$$\boldsymbol{F}_2^{\text{inv}}(u) = \begin{pmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ u_2(\rho E + p) \end{pmatrix}, \qquad \boldsymbol{F}_2^{\text{vis}}(u, \nabla_{\boldsymbol{X}} \boldsymbol{u}) = \begin{pmatrix} 0 \\ \tau_{21} \\ \tau_{22} \\ \tau_{21} u_1 + \tau_{22} u_2 - \Theta_2 \end{pmatrix}, \tag{2}$$

where the viscous stress tensor τ_{ij} and heat flux Θ_i are given by:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right)\right) \tag{3}$$

and

$$\Theta_i = -\frac{\mu}{Pr}\frac{\partial}{\partial x_i}\left(E + \frac{p}{\rho} - \frac{1}{2}(u_1^2 + u_2^2)\right) \tag{4}$$

where δ_{ij} is the Kronecker delta, μ is the viscosity coefficient and $Pr = 0.72$ is the Prandtl number.

In the solution $\boldsymbol{u}$, ρ is the fluid mass density, E is the total energy, u_1 and u_2 are components of velocity along x_1 and x_2 direction, respectively. The quantity p is the pressure which has the form

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}(u_1^2 + u_2^2)\right) \tag{5}$$

where γ is the adiabatic gas constant.

We next define the space-time formulation of equation (1) by treating the temporal dimension as an additional spatial dimension. In this way, the time-dependent problem on a 2D domain Ω^t from $t = 0$ to $t = T$ is then transformed into a time-independent problem on 3D space-time domain $\Omega[0, T]$. We introduce a new space-time gradient operator $\nabla_{\boldsymbol{XT}} = (\partial_{x_1}, \partial_{x_2}, \partial_t)$ and rewrite the Navier-Stokes equations in $\Omega[0, T]$ as

$$\nabla_{\boldsymbol{XT}} \cdot \tilde{\boldsymbol{F}}^{\boldsymbol{inv}}(\boldsymbol{u}) = \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}^{vis}(\boldsymbol{u}, \nabla_{\boldsymbol{X}} \boldsymbol{u}), \tag{6}$$

where

$$\tilde{\boldsymbol{F}}_{\boldsymbol{1}}^{\boldsymbol{inv}}(\boldsymbol{u}) = \boldsymbol{F}_{\boldsymbol{1}}^{\boldsymbol{inv}}(\boldsymbol{u}) \ \ , \ \ \tilde{\boldsymbol{F}}_{\boldsymbol{2}}^{\boldsymbol{inv}}(\boldsymbol{u}) = \boldsymbol{F}_{\boldsymbol{2}}^{\boldsymbol{inv}}(\boldsymbol{u}) \ \ , \ \ \tilde{\boldsymbol{F}}_{\boldsymbol{3}}^{\boldsymbol{inv}}(\boldsymbol{u}) = \boldsymbol{u}. \tag{7}$$

For equations (6), the boundary conditions on $\partial\Omega[0, T]$ are exactly the same as those of original equations (1); the boundary conditions on Ω^0 are the initial condition of equations (1); on the boundary Ω^T of $\Omega[0, T]$, no boundary conditions are needed for the space-time formulation, since the characteristics move in the positive time-direction.

II.B. Discontinuous Galerkin Discretization

Next we describe the discontinuous Galerkin discretization of equations (6). An LDG-type approach[25] is applied to the second-order terms , where the system (6) is split into a new first-order system of equations.

$$\nabla_{\boldsymbol{XT}} \cdot \tilde{\boldsymbol{F}}^{\boldsymbol{inv}}(\boldsymbol{u}) = \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}^{vis}(\boldsymbol{u}, \boldsymbol{q}), \tag{8}$$

$$\nabla_{\boldsymbol{X}} \boldsymbol{u} = \boldsymbol{q} \tag{9}$$

We introduce discontinuous Galerkin (DG) broken spaces $\mathcal{V}_T^h$ and Σ_T^h associated a triangulation $\mathcal{T}_{[0,T]}^h = \{K\}$ of 3D space-time domain $\Omega[0, T]$ as the spaces of functions whose restriction to each element K are polynomial functions of degree at most $p \geq 1$:[26]

$$\mathcal{V}_T^h = \{\boldsymbol{v} \in [L^2(\Omega[0, T])]^4 \mid \ \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^4 \ \ \forall K \in \mathcal{T}_{[0,T]}^h\}, \tag{10}$$

$$\Sigma_T^h = \{\boldsymbol{\sigma} \in [L^2(\Omega[0, T])]^{4\times 2} \mid \ \boldsymbol{\sigma}|_K \in [\mathcal{P}_p(K)]^{4\times 2} \ \ \forall K \in \mathcal{T}_{[0.T]}^h\}, \tag{11}$$

where $\mathcal{P}_p(K)$ denotes the space of polynomials of degree at most $p \geq 1$ on K. Then our space-time DG formulation of equation (6) becomes: find $\boldsymbol{u}^h \in \mathcal{V}_T^h$ and $\boldsymbol{q}^h \in \Sigma_T^h$ such that for each $K \in \mathcal{T}_{[0,T]}^h$, we have

$$\begin{aligned} -\int_K \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}^h) : \nabla_{\boldsymbol{XT}} \boldsymbol{v}^h \, dx + \oint_{\partial K} (\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}}) \cdot \boldsymbol{v}^h \, ds \\ = -\int_K \boldsymbol{F}^{\text{vis}}(\boldsymbol{u}^h, \boldsymbol{q}^h) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^h \, dx + \oint_{\partial K} (\widehat{\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s}) \cdot \boldsymbol{v}^h \, ds, \qquad \forall \boldsymbol{v}^h \in \mathcal{V}_T^h \end{aligned} \tag{12}$$

$$\int_K \boldsymbol{q}^h : \boldsymbol{\sigma}^h \, dx = -\int_K \boldsymbol{u}^h \cdot (\nabla_{\boldsymbol{X}} \cdot \boldsymbol{\sigma}^h) \, dx + \oint_{\partial K} (\widehat{\boldsymbol{u}^h} \otimes \boldsymbol{n}_s) : \boldsymbol{\sigma}^h \, ds, \qquad \forall \boldsymbol{\sigma}^h \in \Sigma_T^h. \tag{13}$$

Here, for the space-time domain $\Omega[0, T]$, $\boldsymbol{n} = (n_1, n_1, n_3)$ is the outward unit normal to the boundary ∂K, and the numerical fluxes $\widehat{\tilde{\boldsymbol{F}}^{\boldsymbol{inv}} \cdot \boldsymbol{n}}$, $\widehat{\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s}$ and $\widehat{\boldsymbol{u}^h}$ are the approximations to $\tilde{\boldsymbol{F}}^{\boldsymbol{inv}} \cdot \boldsymbol{n}$, $\boldsymbol{F}^{\boldsymbol{vis}} \cdot \boldsymbol{n}_{\boldsymbol{s}}$ and $\boldsymbol{u}$ on the face of element K, respectively, which are specified in terms of $\boldsymbol{u}^h$ on two sides of the face of element K, and boundary conditions. More precisely, if we define $\boldsymbol{n}_{\boldsymbol{s}} = (n_1, n_2)$ and $\tilde{\boldsymbol{F}}_{\boldsymbol{s}}^{\boldsymbol{inv}} = (\tilde{\boldsymbol{F}}_{\boldsymbol{1}}^{\boldsymbol{inv}}, \tilde{\boldsymbol{F}}_{\boldsymbol{2}}^{\boldsymbol{inv}})$, and normalize $\tilde{\boldsymbol{n}}_{\boldsymbol{s}} = \boldsymbol{n}_{\boldsymbol{s}}/|\boldsymbol{n}_{\boldsymbol{s}}|$ and $\tilde{n}_3 = n_3/|n_3|$, we then decompose the numerical fluxes as

$$\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}} = |\boldsymbol{n}_s| \left[\widehat{\tilde{\boldsymbol{F}}_s^{\text{inv}} \cdot \tilde{\boldsymbol{n}}_s}\right] + |n_3| \left[\widehat{\tilde{\boldsymbol{F}}_3^{\text{inv}} \tilde{n}_3}\right] = |\boldsymbol{n}_s| \boldsymbol{\mathcal{F}}^s + |n_3| \boldsymbol{\mathcal{F}}^t. \tag{14}$$

$$\widehat{\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s} = |\boldsymbol{n}_s| \left[\widehat{\boldsymbol{F}^{\text{vis}} \cdot \tilde{\boldsymbol{n}}_s}\right] \tag{15}$$

For the inviscid numerical flux, we define the spatial numerical flux $\boldsymbol{\mathcal{F}}^s = \widehat{\tilde{\boldsymbol{F}}_s^{\text{inv}} \cdot \tilde{\boldsymbol{n}}_s}$ as the standard approximate Riemann solver proposed by Roe,[27] and the temporal numerical flux $\boldsymbol{\mathcal{F}}^t = \widehat{\tilde{\boldsymbol{F}}_3^{\text{inv}} \tilde{n}_3}$ by standard upwinding of the corresponding linear time-derivative term $\boldsymbol{u}_t$ in equations (8).

Since the viscous terms in equations (8) as well as all of equations (9) only involve derivatives with respect to the two spatial variables, x_1 and x_2, we can approximate $\widehat{\boldsymbol{F}^{\text{vis}} \cdot \tilde{\boldsymbol{n}}_s}$ and $\widehat{\boldsymbol{u}^h}$ using standard schemes without modification. Here, we choose these numerical fluxes according to the Compact Discontinuous Galerkin (CDG) method proposed by Peraire and Persson.[28]

Note that on the boundaries Ω^0 and Ω^T, the boundary conditions are indirectly incorporated by the temporal numerical fluxes $\boldsymbol{\mathcal{F}^t}$. In particular, since these are defined by upwinding, the initial conditions of equations (1) are used on Ω^0 and the interior solutions on Ω^T. This property makes it possible to advance the solution for a single interval Δt at a time, without connecting the entire space-time solution domain. In this sense, the space-time DG formulation is similar to a standard implicit method of lines formulation.

The discretization above results in a final formulation without time evolution which we solve using Newton's method and a block-ILU(0) preconditioned GMRES method.[29]

III. Moving Space-Time Mesh Generation

We present our space-time mesh generator based on the DistMesh algorithm,[23] which iteratively improves a triangular mesh using only node movements and element connectivity updates.[21] To reduce the computational cost, tetrahedral space-time meshes for each slab $\Omega[t, t+\Delta t]$ are generated separately, see figure 1. More specifically, given an unstructured mesh of $\Omega^t \in \mathbb{R}^2$ at time t, we first generate a unstructured mesh of $\Omega^{t+\Delta t}$ as the time-dependent flow domain is deforming, using only node movements and local edge flips. Based on the resulting two layers of triangular meshes, we apply an efficient combinatorial tetrahedral triangulation method to generate the space-time mesh of $\Omega[t, t+\Delta t]$. We then solve the compressible Navier-Stokes equations in this space-time mesh using the DG scheme described in the previous section, and repeat the procedure for the next space-time slab $\Omega[t+\Delta t, t+2\Delta t]$, etc. The domains Ω^t are never re-meshed from scratch, instead only one initial mesh generation of Ω^0 is needed which is improved at each subsequent time step. More importantly, all the mesh improvement techniques are performed on the 2D mesh, and the tetrahedral triangulation is entirely based on local combinatorial connections. This simplifies the process considerably, and is likely the preferred way to generalize our scheme to generate 4-dimensional space-time simplex meshes.

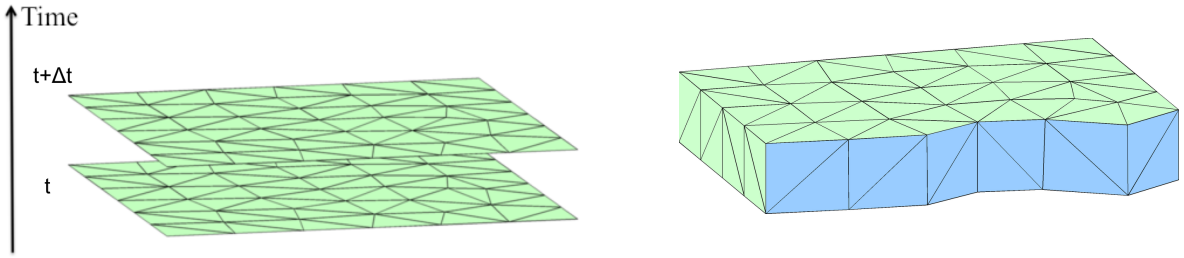


Figure 1. Space-Time Mesh Generation. The left figure illustrates two 2D-mesh layers at time t and $t+\Delta t$, and the right figure shows the corresponding slab of the 3D space-time mesh generated based on the left two mesh layers. The blue faces show the cross-sections.

III.A. Mesh Motion and Edge Flipping

As a starting point, an initial triangulation $\mathcal{T}_0^h$ of Ω^0 is generated using any standard spatial mesh generation technique. At the next time step Δt, as the domain deforms, the triangulation $\mathcal{T}_{\Delta t}^h$ of $\Omega^{\Delta t}$ is obtained by performing local update operations on the previous triangulation $\mathcal{T}_0^h$. First, the boundary nodes are moved rigidly according to the prescribed geometry movement. The element qualities generally decrease after the boundary nodes are moved, so next we improve the mesh using the DistMesh scheme[23] and optimize the locations of the interior nodes.

As illustrated in figure 2, the movement of interior nodes is driven by repulsive forces from each attached edge, which depends on the edge length l and an equilibrium length l_0:

$$|\boldsymbol{F}(\boldsymbol{l})| = \begin{cases} k(l-l_0) & \text{if } l \geq l_0 \\ 0 & \text{if } l < l_0 \end{cases} \tag{16}$$

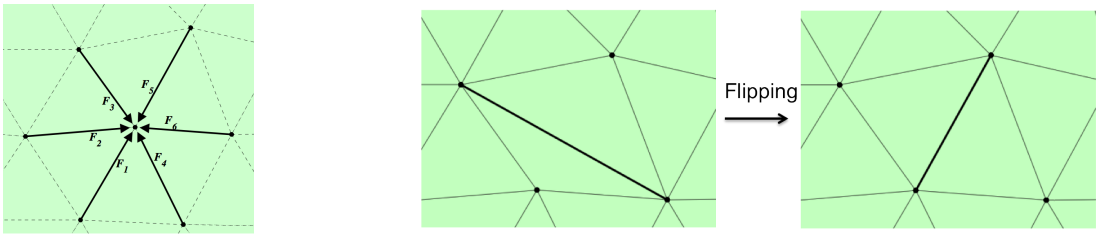


Figure 2. Force-based Smoothing and Edge Flipping. The left plot shows the net force exerted on one node, and the right one gives an example of edge flipping to improve the triangle qualities.

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.[23]

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

$$\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} + \delta \boldsymbol{F}(\boldsymbol{p}^{(n)}) \tag{17}$$

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

With large deformations of the time-dependent domain, node movements are usually not sufficient to produce high-quality elements and avoid element inversion, which is also the main drawback of ALE-based techniques for deformable domain problems.[5] For our space-time meshes, however, we can perform local connectivity changes to improve the mesh qualities.[21] For a triangular mesh, this can be done simply by edge swapping operations[30] as shown in figure 2, where two adjacent triangles flip their shared edge and produce two new triangles sharing the new flipped edge. Finally, to simplify the tetrahedra triangulation algorithm, we require that each element can be flipped at most once during each time step. Therefore, based on this restriction, all the flipped elements come in pairs.

Note that during the process above, the number of nodes, edges and elements remain unchanged. In fact, all elements have the same edge connections from $\mathcal{T}_0^h$ to $\mathcal{T}_{\Delta t}^h$ except those involved in the edge swapping. If we improve the mesh at each time step, a sequence of 2D meshes $\{\mathcal{T}_0^h, \mathcal{T}_{\Delta t}^h, \mathcal{T}_{2\Delta t}^h, \ldots, \mathcal{T}_T^h\}$ is created only using the local mesh operations.

III.B. Tetrahedra Triangulation of Space-Time Domain $\Omega[t, t+\Delta t]$

The next step is to efficiently generate a space-time mesh $\Omega[t, t+\Delta t]$ for each time step based on the initial mesh of the spatial domain Ω^t and the deformed and improved mesh of $\Omega^{t+\Delta t}$. Recall that our mesh moving and edge flipping algorithm is able to keep the same number of nodes on $\Omega^{t+\Delta t}$ as that of Ω^t, so we can simply connect each node of Ω^t with its corresponding node of $\Omega^{t+\Delta t}$, as the first step of our space-time mesh generation. This point-wise connection will ensure that the space-time mesh respects the moving boundary, due to the rigid motion of boundary nodes from Ω^t to $\Omega^{t+\Delta t}$.

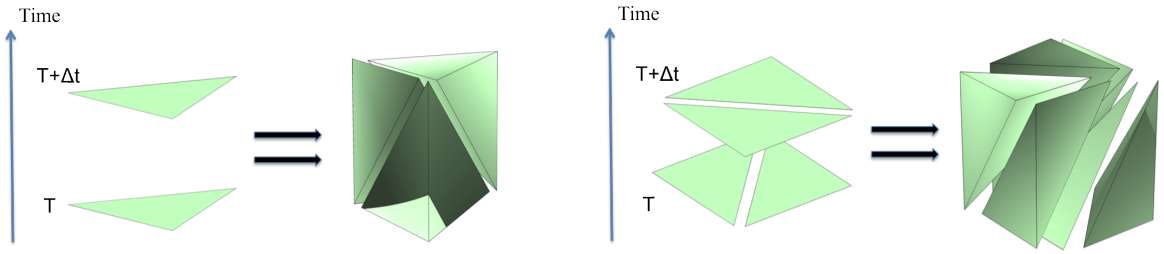


Figure 3. Tetrahedra Triangulation. The left plot illustrates a valid triangulation for a non-edge-flipped element, and the right plots shows a triangulation for a pair of elements with a flipped edge

As a result shown in figure 3, each element of Ω^t without edge flipping is extruded to $\Omega^{t+\Delta t}$ and form an irregular triangular prism, where 'irregular' means that the edge on the bottom face is not necessarily

parallel to its corresponding edge on the top face (due to different node displacements during the force-based smoothing procedure); for those elements involved in an edge flip during period $[t, t+\Delta t]$, each can be extruded together with the paired element it flipped edge with, and then locally form a quadrangular prism with two reverse diagonals on the top and bottom faces. Again, similar to the unflipped case, the edges at Ω^t are not necessarily parallel to those at $\Omega^{t+\Delta t}$. Nevertheless, for convenience in our notation, we will still refer to these vertically skew quadrilaterals as 'lateral faces' of prisms. Finally, it is clear that the amount of node displacement during a time step must be limited to ensure sufficiently high element qualities. We control this dynamically by adjusting the size of the time step Δt and the pseudo time step δ in order to avoid inverted prisms and reverse orientation of vertices.

The point-wise connection strategy described above produces a mesh of triangular and quadrangular prisms. Next we will consider how to split these into a conforming mesh of tetrahedra, by first describing how to perform a local triangulation of a prism, and second how to globally ensure that two adjacent prisms respect the same diagonal on their shared lateral face.

III.B.1. Local Triangulation of Prisms

We will study local triangulations that are entirely based on the nodes in the given spatial meshes, that is, no additional nodes are inserted. First of all, we locally index the nodes of each prism in a counterclockwise order. For each prism V between Ω^t and $\Omega^{t+\Delta t}$, if V a triangular prism, we locally number the vertices on the bottom face as $\{p_1^{V,t}, p_2^{V,t}, p_3^{V,t}\}$ and the vertices on its top face as $\{p_1^{V,t+\Delta t}, p_2^{V,t+\Delta t}, p_3^{V,t+\Delta t}\}$. Similarly, vertices of a quadrangular prism V on the bottom and top face are locally numbered as $\{p_1^{V,t}, p_2^{V,t}, p_3^{V,t}, p_4^{V,t}\}$ and $\{p_1^{V,t+\Delta t}, p_2^{V,t+\Delta t}, p_3^{V,t+\Delta t}, p_4^{V,t+\Delta t}\}$, respectively. In addition, without loss of generality, we require that the original shared edge on Ω^t is the line segment $\overline{p_2^{V,t} p_4^{V,t}}$ and the new shared edge on $\Omega^{t+\Delta t}$ is the line segment $\overline{p_1^{V,t+\Delta t} p_3^{V,t+\Delta t}}$. We will denote by F_i^V the lateral face with vertices at $p_i^{V,t}$, $p_i^{V,t+\Delta t}$, $p_j^{V,t}$ and $p_j^{V,t+\Delta t}$, where $j = (i \mod n) + 1$, n is the number of lateral faces of V, and $1 \le i \le n$.

For each lateral face F_i^V, there are two possible face diagonals which we define using a sign function $S_i^V(p_i^{V,t}, p_i^{V,t+\Delta t}, p_j^{V,t}, p_j^{V,t+\Delta t})$ for each F_i^V according to

$$S_i^V(p_i^{V,t}, p_i^{V,t+\Delta t}, p_j^{V,t}, p_j^{V,t+\Delta t}) = \begin{cases} -1 & \text{if the diagonal edge is } \overline{p_i^{V,t} p_j^{V,t+\Delta t}} \\ +1 & \text{if the diagonal edge is } \overline{p_i^{V,t+\Delta t} p_j^{V,t}} \end{cases} \tag{18}$$

for $1 \le i \le n$.

Now, a triangulation of a triangular prism V is completely determined by the values of its 3 sign functions S_1^V, S_2^V and S_3^V. Combinatorially, it is easy to see that there are $2^3 = 8$ different combinations, but only 6 of these give valid triangulations. Note that the two uniform cases $\{S_1^V = +1, S_2^V = +1, S_3^V = +1\}$ and $\{S_1^V = -1, S_2^V = -1, S_3^V = -1\}$ cannot be used for valid triangulations.

For the quadrangular case, we first make the following definition:

Definition 1 *For a quadrangular prism V, we define the* standard value *of the sign function S_i^V as $+1$ if i is odd while -1 if i is even.*

Since a quadrangular prism V has 4 lateral faces, a triangulation is defined by the values of the 4 corresponding sign functions S_1^V, S_2^V, S_3^V and S_4^V, for a total of $2^4 = 16$ different combinations. However, in order allow for a valid triangulation of V, a combination of sign functions must satisfy the following condition:

Condition 1 *There are two consecutive sign functions S_i^V and $S_{mod(i,4)+1}^V$ which are set to their standard values.*

To meet the condition above, there is a total of 9 possible combinations of S_1^V, S_2^V, S_3^V and S_4^V that correspond to valid triangulations of V.

III.B.2. Algorithms for the Global Space-Time Mesh Generation

The last step is to obtain a global tetrahedral triangulation based on local triangulations of prisms. These prism triangulations are not independent since each prism should match the diagonals at shared lateral

faces with their neighbor prisms. With this restriction, we next introduce a depth-first algorithm which can efficiently find a global triangulation of $\Omega[t, t+\Delta t]$.

Before describing the algorithm, we introduce the following definitions:

Definition 2 *For a prism V, let V^* be the adjacent prism of V with $F_{i^*}^{V^*} = F_i^V$ for an index i^*. We say F_i^V is a* wall *if the values of S_i^V and $S_{i^*}^{V^*}$ are both set and $S_i^V = S_{i^*}^{V^*}$. We say F_i^V is* accessible *if the values of S_i^V and $S_{i^*}^{V^*}$ are both unset.*

With this definition, we now introduce the algorithm by its three main operations.

Operation 1: Optimal Local Triangulation of Prisms

Throughout the algorithm, if V has not been triangulated, we optimize the local triangulation of V by

$$\arg\max_{\mathcal{T}^V} \min_{K\in\mathcal{T}^V} Q(K) \tag{19}$$

where $\mathcal{T}^V$ denotes the set of all the possible valid triangulations of V, whose sign functions respect the ones prescribed on the walls. $Q(K)$ represents the quality of each tetrahedron K of $\mathcal{T}^V$, which is calculated by the following formula proposed by Field[31]

$$Q(K) = 72\sqrt{3}\frac{\mathrm{Vol}(K)}{(\sum_{i=1}^{6} l_i(K)^2)^{3/2}} \tag{20}$$

where $\mathrm{Vol}(K)$ is the volume of K and l_i is the length of each edge $i = 1, \ldots, 6$.

Operation 2: Sign Function Synchronization of Neighbor Prisms

Once a prism V is triangulated by operation 1, in operation 2, we start from each accessible F_i^V, and update the sign functions of the corresponding neighbor prisms to make F_i^V a wall. For instance, suppose F_i^V was accessible before the local triangulation of V, and V^* is the adjacent prism with $F_{i^*}^{V^*} = F_i^V$ for some index i^*. There are 4 different possible cases for V^*:

Case 1 *If V^* is triangular with all faces accessible, we simply set $S_{i^*}^{V^*} = S_i^V$, which makes F_i^V and $F_{i^*}^{V^*}$ walls;*

Case 2 *If V^* is quadrangular with all faces accessible, we set $S_{i^*}^{V'} = S_i^V$ and $S_{mod(i^*+1,4)+1}^{V^*}$ to their standard values. If V^{**} is the adjacent prism of V^* with shared face $F_{mod(i^*+1,4)+1}^{V^*}$, then we continue to update sign functions of V^{**} recursively using operation 2;*

Case 3 *Suppose V^* is triangular resulting from Case 1, then it has a wall $F_{j^*}^{V^*}$ for some $j^* \neq i^*$. For this case, we use operation 1 to triangulate V^* immediately under the restrictions imposed by the prescribed values of $S_{j^*}^{V^*}$ and $S_{i^*}^{V^*} = S_i^V$. Let k^* be the third index other than i^* and j^* and V^{**} be the adjacent prism of V^* with shared face $F_{k^*}^{V^*}$. We continue updating sign functions of V^{**} recursively using operation 2;*

Case 4 *Suppose V^* is quadrangular resulting from Case 2, then it has a pair of opposite walls, say, $S_{j^*}^{V^*}$ and $S_{k^*}^{V^*}$ (where $i^* \neq j^*$ and $i^* \neq k^*$ since $S_{i^*}^{V^*}$ has not been a wall). Let l^* be the fourth index other than i^*, j^* and k^*. Again, we use operation 1 to triangulate V^*, under the restrictions imposed by the prescribed values of $S_{j^*}^{V^*}$ and $S_{k^*}^{V^*}$, as well as $S_{i^*}^{V^*} = S_i^V$. Finally, similarly to the previous case, if V^{**} is the adjacent prism of V^* with the shared face $F_{l^*}^{V^*}$, we continue to update sign functions of V^{**} recursively using operation 2.*

Operation 3: Triangulation Adjustment of Root Prism

As shown in figure 4, if we triangulate a prism V by operation 1 and repeatedly encounter the cases $2-4$ when synchronizing sign functions of neighbors by operation 2, then a path will be made which we will refer to as an 'updating path'. In fact, every updating path will eventually end with one of three possibilities: 1. a prism belonging to Case 1 (figure 4, right); 2. a domain boundary; 3. back to the root prism V from a face which is not yet a wall (figure 4, left). The third case is the only potentially difficult one, since the last prism of an updating path is a neighbor of the root prism V, but they may have inconsistent values of the sign functions corresponding to their shared face. Suppose the last prism is V' with the shared face

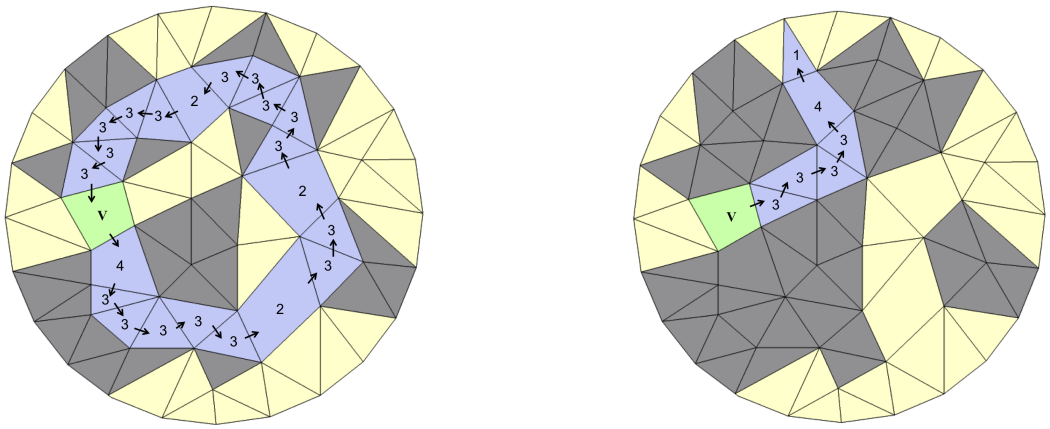


Figure 4. An example of a triangulation path. Each triangle represents a triangular prism and each quadrilateral represents a quadrangular prism. The green element is the root prism V, yellow ones are untriangulated prisms and gray ones are already triangulated. The purple elements denote an updating path directed by the black arrows. The corresponding case number that each purple element belongs to is also shown. The example path on the left ends when it returns to V, and the example path on the right ends with a triangular prism belonging to Case 1.

$F_{i'}^{V'} = F_i^V$ but $S_{i'}^{V'} \neq S_i^V$. Operation 3 is to change the value of S_i^V to that of $S_{i'}^{V'}$ and thus make both $F_{i'}^{V'}$ and F_i^V into walls.

It is clear from the local triangulations derived in Section III.B.1 that changing values of S_i^V might result in a new combination of sign functions the does not correspond to a valid local triangulation of the root prism V. However, with a further investigation, we found out that this situation can be avoided by arranging the order by which new updating paths are launched.

Based on the three operations described above, the full algorithm is summarized in algorithm 1. In summary, the global tetrahedral triangulation is complete if and only if for all prisms, all their faces become walls. For more details and analysis of the algorithm, see ref. 24.

Algorithm 1 Space-Time Mesh Generation

Input: A spatial mesh $MESH1$ of Ω^t and $MESH2$ of $\Omega^{t+\Delta t}$
Output: A space-time mesh $STMESH$ of $\Omega[t, t+\Delta t]$
Create prisms by extruding elements from $MESH1$ to $MESH2$ and make a list of those prisms called $PList$
Initialize an empty list $STMESH$ for storing the elements of the space-time mesh
while $PList$ is non-empty **do**
  Pop a prism V from $PList$
  if V has not been triangulated **then**
    Triangulate V by operation 1
    Make a list of F_i^V which has not been a wall, called $FList$
    Sort $FList$ by the order of launching updating paths mentioned for operation 3
    for F_i^V in $FList$ **do**
      Find the neighbor prism $NBPrism$ adjacent to V by F_i^V ▷ Initialization of an updating path
      while $NBPrism$ exists (not exist if encountering domain boundary) and is not V **do**
        Synchronize sign functions of $NBPrism$ by operation 2
        if $NBPrism$ belongs to Case 2-4 **then**
          Update $NBPrism$ by operation 2 and continue the updating path
        else
          Break ▷ The updating path ends with a $NBPrism$ of Case 1
        end if
      end while
      if $NBPrism$ is V **then** ▷ The updating path back to the root prism
        Adjust the values of sign functions of V by operation 3 if necessary
      end if
    end for
  end if
  Push all the elements from the resulting triangulation of V into $STMESH$
end while
return $STMESH$

IV. Numerical Results

IV.A. Euler Vortex

First, we solve the Euler equations for a model problem of a compressible vortex in a 20-by-20 square domain and make a convergence test to demonstrate the high order accuracy of our space-time discontinuous Galerkin Method.

The vortex is initially centered at $(x_0, y_0) = (8, 8)$ and moves with the free-stream at an angle $\theta = \pi/4$ with respect to the x-axis. The analytic solution at (x, y, t) is as follows

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

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

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

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

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. Moreover, $\bar{u}$ and $\bar{v}$ are 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 a starting point, an unstructured mesh of the domain is created with element size h by DistMesh.[23] In practice, in order to show the high-order accuracy of this method even with large mesh deformation, at each time step we rotate some of the vertices (the blue nodes showed in figure 5) about the center of the domain with angular velocity $\omega = \frac{2}{3}\pi$, such that large mesh deformations are generated immediately. To avoid inverted and low-quality elements, we continue to improve the mesh by our mesh moving and element flipping techniques. Then we use our space-time DG method to solve the Euler equations based on this moving mesh until time $T = \sqrt{4^2+4^2}$ and compare the numerical results with the analytical solutions above. Note that the time step Δt (i.e. the thickness of each space-time mesh) is chosen as $\Delta t << h$ to ensure that truncation errors are dominated by the part introduced from the spatial discretization of h. We give the convergence test for a variety of spatial mesh sizes h and polynomial orders p. As a benchmark, we also apply our space-time DG method for the same convergence test but based on the fixed mesh, where no vertices are rotated and thus the initial unstructured mesh remains unchanged for all time steps.

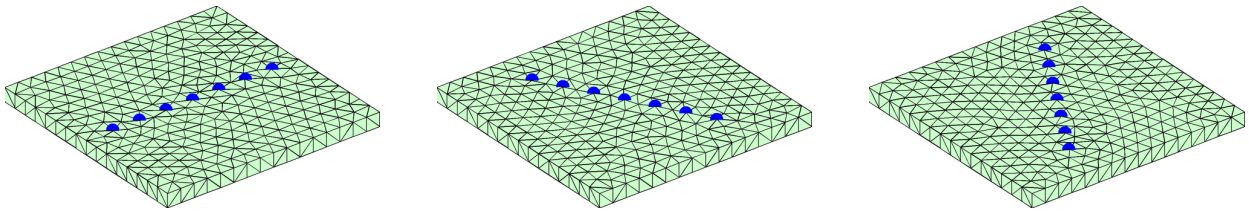
In figure 5, three sample space-time meshes and solutions of the pressure field are given, and the bottom plot compares the errors in the discrete L^2-norm of our space-time DG method for the moving and the fixed mesh, respectively. It can be seen that unlike the ALE method, the solutions from our moving meshes have essentially the same accuracy as those from the fixed mesh. This is expected because although the mesh is moving in the spatial domain, in the space-time framework, no mapping is employed and the tetrahedral mesh is fixed for each space-time domain $\Omega[t, t+\Delta t]$. This avoids large variations in the resolution of the solution that would have been introduced by the ALE mapping. From the convergence plot, the results clearly show that the orders of convergence are approximately $O(h^{p+1})$.

IV.B. Pitching Tandem Airfoils

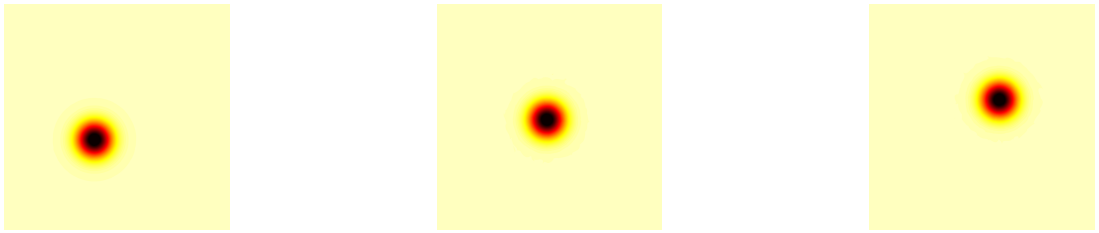
We next consider a Navier-Stokes simulation similar to the one studied in ref. 32. It consists of two pitching NACA 0012 airfoils with chord length $c = 1$ in a 6×2 rectangular domain. As shown in figure 6, at the initial time $t = 0$ the two foils have a zero pitching angle and are aligned on the horizontal 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.



Space-Time Mesh of $\Omega[0, \Delta t]$ Space-Time Mesh of $\Omega[2.5, 2.5 + \Delta t]$ Space-Time Mesh of $\Omega[5, 5 + \Delta t]$



Pressure Field at $t = 0$ Pressure Field at $t = 2.5$ Pressure Field at $t = 5.0$

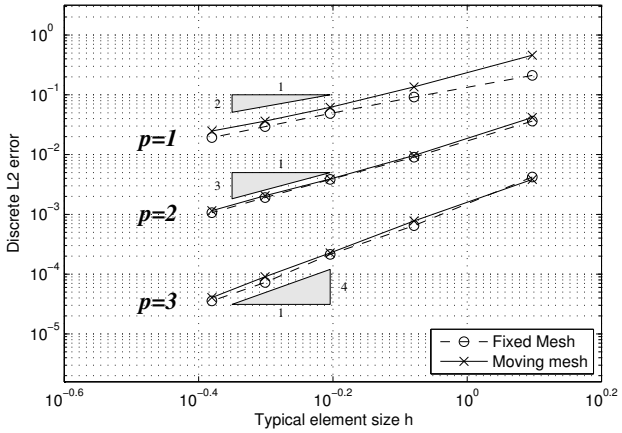


Figure 5. Convergence Test for the Euler Vortex Problem. The top three plots are samples of space-time meshes of $\Omega[0, \Delta t]$, $\Omega[2.5, 2.5 + \Delta t]$ and $\Omega[5, 5 + \Delta t]$ with $\Delta t = 5 \times 10^{-3}$, $h = 1.25$ and $p = 3$. In these plots, the thickness of each space-time mesh is rescaled to 1.25 to better illustrate the mesh structure. The seven blues nodes are rotated about the center of the domain in rigid manner, which induces other vertex movements and local element flipping. Below each mesh, the sample solutions of pressure field are given at $t = 0$, $t = 2.5$ and $t = 5.0$. The bottom plot shows the convergence results for $p = 1$, 2 and 3.

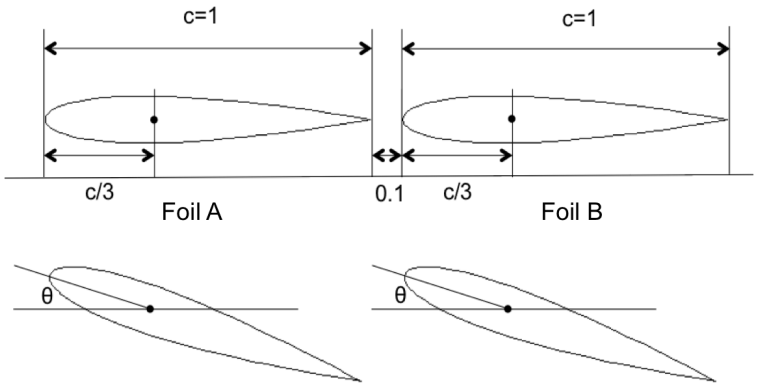
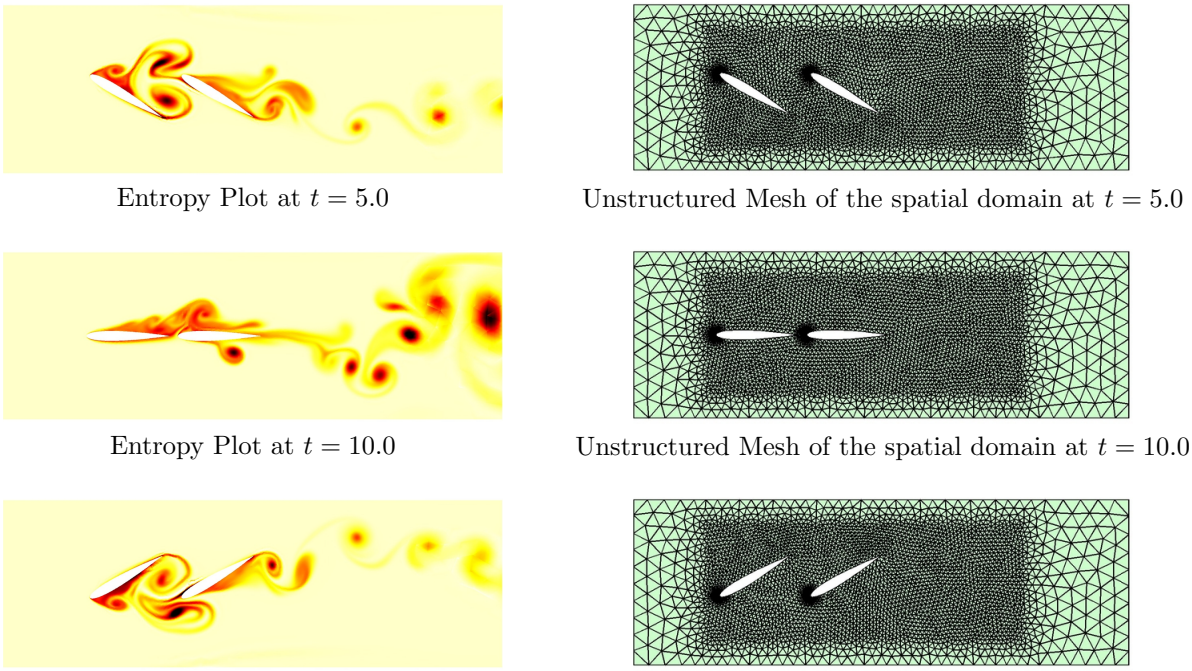


Figure 6. The pitching tandem airfoil model.

As the two foils are placed very close and rotated based on the same harmonic function, an ALE method would have a hard time to solve this deformable domain problem since finding a smooth mapping on the small gap between two foils is difficult. Instead, our space-time formulation only requires an unstructured two-dimensional mesh of the initial domain Ω^0 and is able to improve the mesh automatically by local mesh operations. To better resolve the solution field, we implement our space-time DG method with polynomial order $p = 2$ but with linear element geometries (future work includes the generation of appropriate curved space-time meshes). Three sample meshes and the corresponding entropy plots are given in figure 7, and plots of drag and lift coefficients are shown in figure 8.



Entropy Plot at $t = 5.0$ | Unstructured Mesh of the spatial domain at $t = 5.0$

Entropy Plot at $t = 10.0$ | Unstructured Mesh of the spatial domain at $t = 10.0$

Entropy Plot at $t = 15.0$ | Unstructured Mesh of the spatial domain at $t = 15.0$

Figure 7. Compressible Navier-Stokes flow around two pitching tandem airfoils, entropy of the solutions (left) and the spatial meshes (right) at 3 time instances.

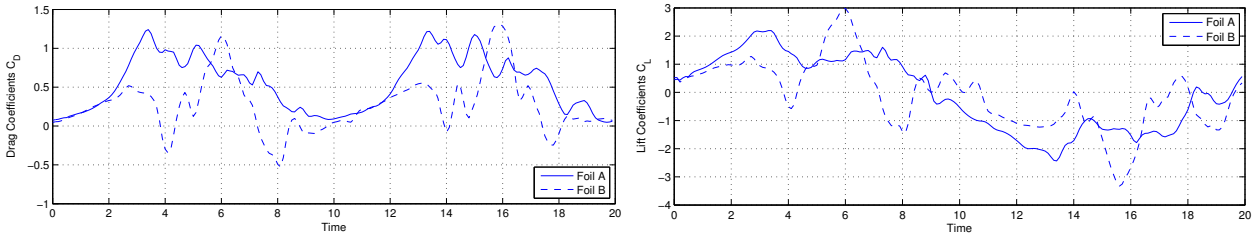


Figure 8. Drag and lift coefficients around the pitching tandem NACA0012 airfoils as a function of time.

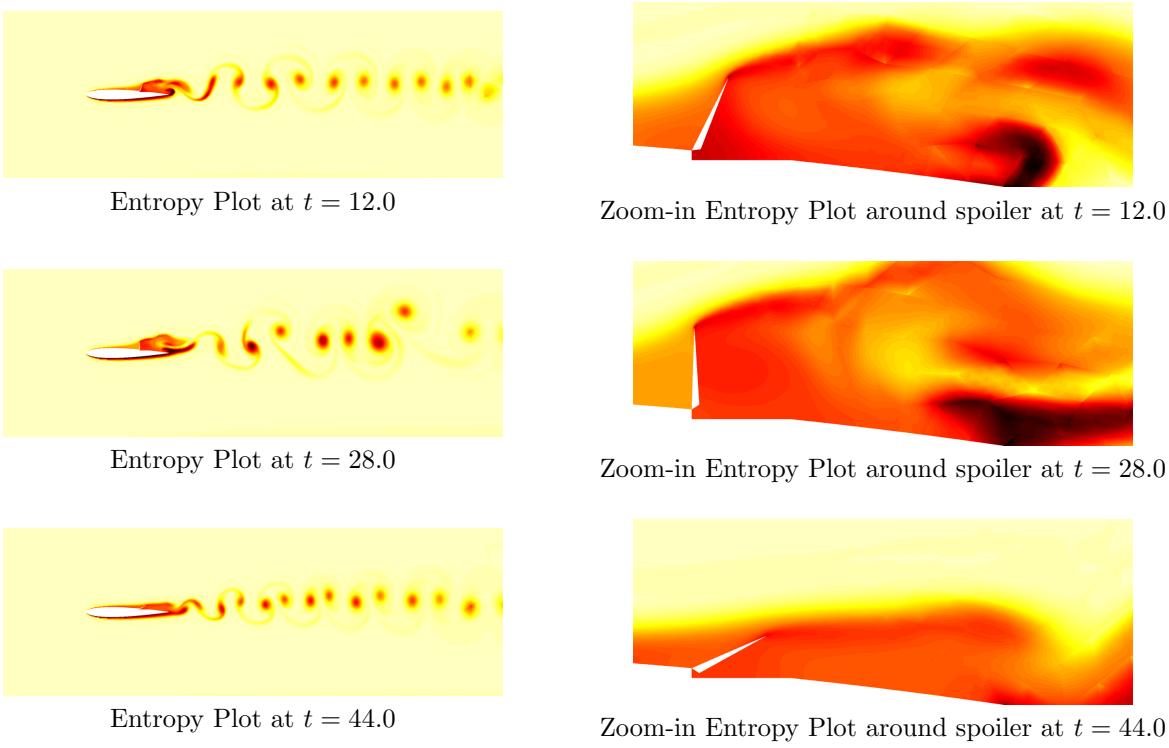
IV.C. Airfoil with a Deploying Spoiler

As an example of more complicated domain deformation, we solve for the flow around a NACA0012 airfoil with chord length 1, in a 6×2 rectangular domain. As illustrated in figure 10, the foil is located between $x = 0$ to $x = 1$ with axis of symmetry $y = 0$. We then remove a right triangle with curved hypotenuse from $x = 0.6383$ to $x = 0.7534$ and replace it by a thin spoiler of length 0.1. To avoid topology changes, we keep a horizontal gap of width 2×10^{-3} between the foil and the spoiler, which are only connected at the point $(0.6383, 0.0422)$. An adaptive mesh is applied with refined elements around the spoiler. When the spoiler is deployed, it rotates about the connecting point with the foil with angular velocity 0.1, which generates a

large domain deformation around the spoiler. To address this, we update the adaptive mesh size function at each timestep and improve the mesh quality by our local mesh operations.

The numerical simulation starts with a steady flow around the flat foil with a closed spoiler, at Mach number 0.2 and Reynolds number 5000, based on the airfoil chord length 1 and the free-stream velocity 1. Next, we fix the foil but raise the spoiler gradually up to a 90 degrees angle, which results in massive flow separation behind the foil. We keep the spoiler at the vertical state for a short time period, and then close it again by reversing the motion. During this entire process, we use our space-time DG method to solve for the compressible viscous flow during the raising and closing part, and a regular two-dimensional method-of-lines DG method for the time period when the spoiler position is fixed. Again, as in the previous test, we use polynomial orders $p = 2$ with linear element geometries, in order to better resolve the solution fields.

In figure 10, some mesh plots are given to show how our local mesh operations improve the spatial mesh as the spoiler is raised, and three samples of entropy plots are shown in figure 9. In the zoom-in plots, we can confirm that our space-time DG method retains the high quality of the solutions even for the large deformation between the foil and the spoiler. Finally, the lift and the drag coefficients during the entire process are shown in figure 11.



Entropy Plot at $t = 12.0$ | Zoom-in Entropy Plot around spoiler at $t = 12.0$

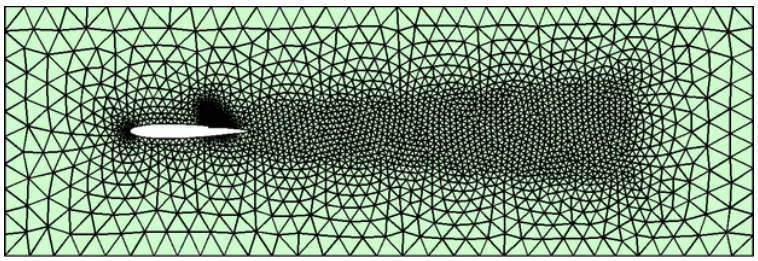
Entropy Plot at $t = 28.0$ | Zoom-in Entropy Plot around spoiler at $t = 28.0$

Entropy Plot at $t = 44.0$ | Zoom-in Entropy Plot around spoiler at $t = 44.0$

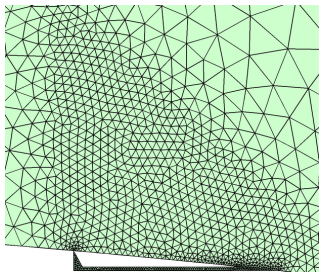
Figure 9. Compressible Navier-Stokes flow around an airfoil with a deploying spoiler .

V. Conclusions

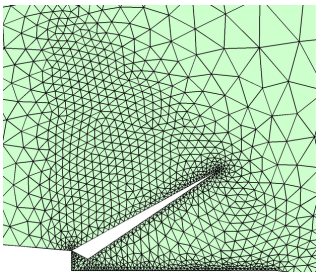
We have presented a fully unstructured space-time mesh generator and a high-order accurate discontinuous Galerkin discretization of the space-time Navier-Stokes equations. Using local mesh operations, we generate simplex elements for each layer of timesteps separately and use implicit solvers to advance the solution in time. Unlike the ALE schemes, our method can handle complex domain changes and mesh reconfigurations, without reduced accuracy or conservation problems. We demonstrated the scheme on model test problems as well as applications involving complex mesh motions. Future work includes the extension of the space-time mesh generation procedure to three spatial dimensions plus time, and the generation of curved high-order space-time elements.



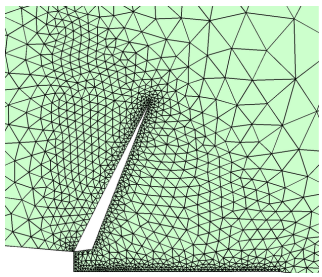
The initial spatial mesh at $t = 0.0$



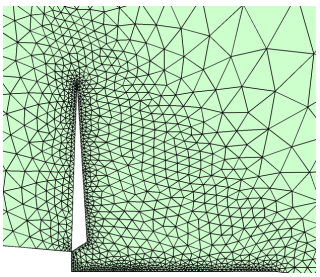
Zoom-in spatial mesh around spoiler at $t = 0.0$



Zoom-in spatial mesh around spoiler at $t = 6.0$



Zoom-in spatial mesh around spoiler at $t = 12.0$



Zoom-in spatial mesh around spoiler at $t = 18.0$

Figure 10. Spatial Meshes of Airfoil with a Deploying Spoiler.

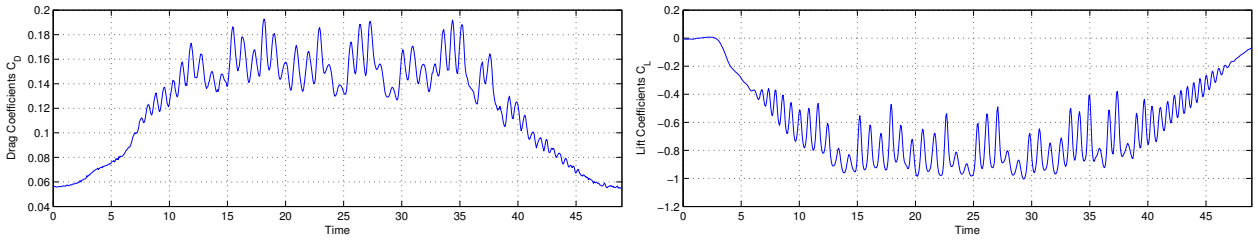


Figure 11. Drag and lift coefficients around the NACA0012 airfoil with a deploying spoiler as a function of time.

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]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.

[4]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.

[5]Persson, P.-O., Bonet, J., and Peraire, J., "Discontinuous Galerkin Solution of the Navier-Stokes Equations on Deformable Domains," *Comp. meth. in Appl. Mech and Engnr., to appear*, 2009.

[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]Johnson, C., "Discontinuous Galerkin finite element methods for second order hyperbolic problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 107, No. 1, 1993, pp. 117–129.

[9]Lowrie, R. B., Roe, P. L., and Van Leer, B., "Space-time methods for hyperbolic conservation laws," *Barriers and Challenges in Computational Fluid Dynamics*, Springer, 1998, pp. 79–98.

[10]Aliabadi, S. and Tezduyar, T., "Space-time finite element computation of compressible flows involving moving boundaries and interfaces," *Computer Methods in Applied Mechanics and Engineering*, Vol. 107, No. 1, 1993, pp. 209–223.

[11]Klaij, C. M., van der Vegt, J. J. W., and van der Ven, H., "Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 217, No. 2, 2006, pp. 589–611.

[12]Sudirham, J., van der Vegt, J., and van Damme, R., "Space–time discontinuous Galerkin method for advection–diffusion problems on time-dependent domains," *Applied numerical mathematics*, Vol. 56, No. 12, 2006, pp. 1491–1518.

[13]van der Vegt, J. J. W. and van der Ven, H., "Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation," *J. Comput. Phys.*, Vol. 182, No. 2, 2002, pp. 546–585.

[14]Rhebergen, S. and Cockburn, B., "A space-time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains," *J. Comput. Phys.*, Vol. 231, No. 11, 2012, pp. 4185–4204.

[15]Rhebergen, S., Cockburn, B., and van der Vegt, J. J. W., "A space-time discontinuous Galerkin method for the incompressible Navier-Stokes equations," *J. Comput. Phys.*, Vol. 233, 2013, pp. 339–358.

[16]Mani, K. and Mavriplis, D., "Efficient Solutions of the Euler Equations in a Time-Adaptive Space-Time Framework," *49th AIAA Aerospace Sciences Meeting and Exhibit*, 2011, AIAA-2011-774.

[17]Rendall, T. C. S., Allen, C. B., and Power, E. D. C., "Conservative unsteady aerodynamic simulation of arbitrary boundary motion using structured and unstructured meshes in time," *Internat. J. Numer. Methods Fluids*, Vol. 70, No. 12, 2012, pp. 1518–1542.

[18]Üngör, A. and Sheffer, A., "Tent-Pitcher: A Meshing Algorithm For Space-Time Discontinuous Galerkin Methods," *Proceedings of the 9th International Meshing Roundtable*, Sandia Nat. Lab., 2000, pp. 111–122.

[19]Abedi, R., Chung, S.-H., Erickson, J., Fan, Y., Haber, R., Sullivan, J., Thite, S., and Zhou, Y., "Spacetime meshing with adaptive coarsening and refinement," *4th Symposium on Trends in Unstructured Mesh Generation, 7th US National Congress on Computational Mechanics*, Citeseer, 2003.

[20]Behr, M., "Simplex space–time meshes in finite element simulations," *International journal for numerical methods in fluids*, Vol. 57, No. 9, 2008, pp. 1421–1434.

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

[22]Olivier, G. and Alauzet, F., "A new changing-topology ALE scheme for moving mesh unsteady simulations," *49th AIAA Aerospace Sciences Meeting, AIAA Paper*, Vol. 474, 2011, pp. 252–271.

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

[24]Wang, L. and Persson, P.-O., "A High-Order Discontinuous Galerkin Method with Unstructured Space-Time Meshes for Domains with Large Deformations," in review.

[25]Cockburn, B. and Shu, C.-W., "The local discontinuous Galerkin method for time-dependent convection-diffusion systems," *SIAM J. Numer. Anal.*, Vol. 35, No. 6, 1998, pp. 2440–2463.

[26]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.

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

[28]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.

[29]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.

[30]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.

[31]Field, D. A., "Qualitative measures for initial meshes," *Internat. J. Numer. Methods Engrg.*, Vol. 47, 2000, pp. 887–906.

[32]Shirsath, R. A. and Mukherjee, R., "Unsteady Aerodynamics of Tandem Airfoils Pitching in Phase," *2nd International Conference on Mechanical, Production and Automobile Engineering (ICMPAE'2012)*, 2012.