A High-Order Discontinuous Galerkin Method with Unstructured Space-Time Meshes for Two-Dimensional Compressible Flows on Domains with Large Deformations

Luming Wang[a], Per-Olof Persson[a,*]

[a]*Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720-3840, USA*

Abstract

We present a high-order accurate space-time discontinuous Galerkin method for solving two-dimensional compressible flow with fully unstructured space-time meshes. The discretization is based on a nodal formulation, with appropriate numerical fluxes for the first and the second-order terms, respectively. The scheme is implicit, and we solve the resulting non-linear systems using a parallel Newton-Krylov solver. The meshes are produced by a mesh moving technique with element connectivity updates, and the corresponding space-time elements are produced directly based on these local operations. To obtain globally conforming tetrahedral meshes, we first derive the required conditions on a prism boundary mesh to allow for a valid local triangulation. Next, we present an efficient algorithm for finding a global mesh that satisfies these conditions. We also show how to add and remove mesh nodes, again using local constructs for the space-time mesh. Our method is demonstrated on a number of test problems, showing the high-order accuracy for model problems, and the ability to solve flow problems on domains with complex large deformations.

Keywords: discontinuous Galerkin, space-time, high-order accuracy, deformable domains, Navier-Stokes

1. 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, various solutions have been proposed. A number

*Corresponding author. Tel.: +1-510-642-6947; Fax.: +1-510-642-8204.
Email addresses: lwang@math.berkeley.edu (Luming Wang), persson@berkeley.edu (Per-Olof Persson)

of methods are based on computational meshes that do not necessarily align with the domain boundary (often Cartesian grids), and impose the boundary conditions using approaches such as the embedded boundary/cut-cell methods [3–5], the immersed boundary method [6, 7], and the fictitious domain methods [8, 9]. These popular schemes are often simple and highly efficient, but somewhat difficult to modify for high-order accuracy close to the boundaries and for thin boundary layers.

Another widely used technique is the Arbitrary Lagrangian-Eulerian (ALE) method [10–12]. These formulations are based on a time-varying mapping or a deforming grid that conforms to the domain boundary, which is compensated for by a modification of the equations. Because of this body-fitted grid motion, the method can produce high-order accurate solutions to a wide range of problems, and DG formulations have been developed for fully unstructured meshes in [12–15]. The main drawback with the ALE approach is that when the domain deformation is large and/or complex, it is difficult to deform the mesh without inverting the elements. Remeshing is then commonly employed, which introduces errors during the solution transfer between the old and the new meshes. Some topology-change strategies are introduced in [16–20], but so far it appears that satisfactory high-order accurate ALE methods for large boundary deformations have not yet been developed. In addition, the ALE formulations require special care to ensure the satisfaction of the so-called geometric conservation law (GCL) [13].

As an alternative, the so-called space-time framework are fully consistent discretizations that allow for arbitrary changes of the domain in both space and time [21]. The method essentially treats the time-dependency with the same technique as the spatial terms, but exploiting the causal nature of the equations to improve the efficiency. Some of the early work on space-time methods include [22, 23], where a Galerkin/least square finite element method was introduced for solving classical elastodynamics problems. Extensions to flow problems on moving domains, with streamline-upwind/Petrov-Galerkin (SUPG) stabilization, where developed in [24, 25]. A priori and a posteriori error estimates for a space-time finite element discretization for linear second order hyperbolic equations were presented in [26].

General formulations and analyses of space-time DG methods were presented in [27–29]. The methods have been used for a number of practical applications, for example in [30] where a space-time DG method was demonstrated on several aerodynamic applications. Formulations for the Oseen equations where developed in [31], and for two-fluid flows in [32]. Since the space-time DG formulations lead to fully implicit discretizations, many previous authors have studied specialized solution techniques. For example, [33] provided a pseudo-time stepping method to deal with the time evolution, and a h-multigrid solver was introduced in [34]. Also, space-time hybridizable discontinuous Galerkin (HDG) methods were recently proposed in [35, 36].

Most work on space-time DG methods have focused on spatial meshes that do not undergo connectivity changes throughout the time evolution. The corresponding space-time meshes are then essentially composed of extruded prism-elements, possibly deforming in time. Some previous work have explored time adaptivity, for example in [37] where a space-time finite volume method with nonuniform time stepping was developed. However, to handle moving domain problems with large deformations, fully unstructured space-time meshes are required the spatial mesh topology has to be modified during the simulation. Previous work on fully unstructured space-time meshes includes [38], which proposed a space-time finite volume method on unstructured meshes generated by a standard 3D mesh generator. A so-called tent-pitcher space-time mesh generator based on an advancing front method was introduced in [39, 40] with various extensions in [41–43]. Finally, in [44] space-time meshes were generated by connecting spatial meshes using a Delaunay approach, which required some additional techniques to eliminate sliver elements.

In this work, we present a space-time discontinuous Galerkin discretization of the compressible Navier-Stokes equations and a novel fully unstructured space-time mesh generation procedure. The resulting discretizations from the space-time DG method are higher-dimensional and fully implicit, and we solve the resulting nonlinear systems of equations using efficient parallel Newton-Krylov solvers [45, 46]. We generate high-quality moving meshes using the DistMesh algorithm [47], and construct the space-time elements for each layer of timesteps using a local construction. Unlike the previous work outline above, our procedures allow for local mesh topology changes in the spatial meshes in order to handle large deformation problems without regenerating meshes from scratch. Moreover, the space-time meshes are fully unstructured and generated in an efficient and robust way, only depending on combinatoric properties of how the vertices are connected. Since no additional nodes are inserted besides those on the spatial meshes, the computational cost for solving the discretized systems is kept at a minimum. The resulting scheme can essentially handle any type of domain deformations, even including topological changes. The order of accuracy can be arbitrarily high in both space and time, provided suitable curved meshes can be generated.

The paper is organized as follows: First, in Section 2 we introduce the governing equations and derive our space-time DG formulation. Next, in Section 3 we introduce our local mesh operations and the combinatorial algorithm for generating a globally conforming space-time mesh. In Section 4 we present a number of examples and applications using our methods. We 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. We also present three 2D laminar flow problems of a mixer with a rotating object, two pitching tandem airfoils, and an airfoil with a deploying spoiler, which show our method's capability to

handle complex deformations.

2. Governing Equations

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

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

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

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

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

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

with viscous stress tensor τ_{ij} and heat flux Θ_i 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 function, μ is the viscosity coefficient and Pr is the Prandtl number.

In the solution vector $\boldsymbol{u}$, ρ is the fluid mass density, E is the total energy, u_1 and u_2 are the components of the velocity along the x_1 and x_2 directions, 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.

Next, we define our space-time formulation of the two-dimensional spatial domain Ω^t from $t = 0$ to $t = T$ in the three-dimensional 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 write the Navier-Stokes equations in $\Omega[0, T]$ as

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

where

$$\tilde{\boldsymbol{F}}_1^{\text{inv}}(\boldsymbol{u}) = \boldsymbol{F}_1^{\text{inv}}(\boldsymbol{u}), \qquad \tilde{\boldsymbol{F}}_2^{\text{inv}}(\boldsymbol{u}) = \boldsymbol{F}_2^{\text{inv}}(\boldsymbol{u}), \qquad \tilde{\boldsymbol{F}}_3^{\text{inv}}(\boldsymbol{u}) = \boldsymbol{u}. \tag{7}$$

The boundary conditions on $\partial\Omega[0, T]$ for equations (6) are the same as the boundary conditions given for the original equations (1), and the boundary conditions on Ω^0 are the given initial conditions. 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.

2.2. Discontinuous Galerkin Discretization of Euler Equations

First, we consider the first-order space-time formulation of the Euler equations,

$$\nabla_{\boldsymbol{XT}} \cdot \tilde{\boldsymbol{F}}^{\text{inv}}(u) = \boldsymbol{0}. \tag{8}$$

We introduce a conforming triangulation $\mathcal{T}^h_{[0,T]} = \{K\}$ of the 3D space-time domain $\Omega[0, T]$ into tetrahedral elements K. On $\mathcal{T}^h_{[0,T]}$, we define the broken space $\mathcal{V}^h_T$ as the spaces of functions whose restriction to each element K are polynomial functions of degree at most $p \geq 1$ [48]:

$$\mathcal{V}^h_T = \{\boldsymbol{v} \in [L^2(\Omega[0, T])]^4 \mid \boldsymbol{v}|_K \in [\mathcal{P}_p(K)]^4 \ \ \forall K \in \mathcal{T}^h_{[0,T]}\}, \tag{9}$$

where $\mathcal{P}_p(K)$ denotes the space of polynomials of degree at most $p \geq 1$ on K. Our DG formulation for equation (8) then becomes: find $\boldsymbol{u}^h \in \mathcal{V}_T^h$ such that for each $K \in \mathcal{T}_{[0,T]}^h$, we have

$$\int_K \left(\nabla_{\boldsymbol{XT}} \cdot \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}^h) \right) \cdot \boldsymbol{v}^h \, dV = - \int_K \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}^h) : \nabla_{\boldsymbol{XT}} \boldsymbol{v}^h \, dV + \oint_{\partial K} (\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}}) \cdot \boldsymbol{v}^h \, dA = 0, \quad \forall \boldsymbol{v}^h \in \mathcal{V}_T^h. \tag{10}$$

Here, $\boldsymbol{n} = (n_1, n_1, n_3)$ is the outward unit normal to the boundary ∂K of the space-time tetrahedron K. The numerical flux $\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}}$ is an approximation to $\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}$ on the face of element K, which is specified in terms of $\boldsymbol{u}^h$ on the two sides of the element boundary and by the boundary conditions. More precisely, if we define $\boldsymbol{n}_s = (n_1, n_2)$ and $\tilde{\boldsymbol{F}}_s^{\text{inv}} = (\tilde{\boldsymbol{F}}_1^{\text{inv}}, \tilde{\boldsymbol{F}}_2^{\text{inv}})$, and normalize $\tilde{\boldsymbol{n}}_s = \boldsymbol{n}_s / |\boldsymbol{n}_s|$ and $\tilde{n}_3 = n_3 / |n_3|$, we can decompose the numerical flux into two parts as

$$\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}} = |\boldsymbol{n}_s| \left[\widehat{\tilde{\boldsymbol{F}}_s^{\text{inv}} \cdot \tilde{\boldsymbol{n}}_s} \right] + |n_3| \left[\widehat{\tilde{\boldsymbol{F}}_3^{\text{inv}} \tilde{n}_3} \right] = |\boldsymbol{n}_s| \boldsymbol{\mathcal{F}}^s + |n_3| \boldsymbol{\mathcal{F}}^t. \tag{11}$$

For the first term, 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 [49]. For the second term, we define 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 equation (8).

Note that on the boundaries Ω^0 and Ω^T, the boundary conditions are indirectly incorporated by the temporal numerical fluxes $\boldsymbol{\mathcal{F}}^{\boldsymbol{t}}$. 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 non-linear system of equations of the form

$$\boldsymbol{R}(\boldsymbol{u}^h) = 0, \tag{12}$$

which is an algebraic system without explicit time dependence. Note that unlike method-of-lines discretizations, the time discretization is part of the DG formulation and thus it is impossible to choose other methods, such as multistep or explicit time integrators. We solve the system (12) using Newton's method and an efficient parallel block-ILU(0) preconditioned GMRES method [45, 46].

2.3. Discontinuous Galerkin Discretization of Viscosity Terms

Next we describe the discretization of the viscous terms in the compressible Navier-Stokes equations. We apply a standard procedure for second-order terms [50], where the system (6) is split into a first-order system of equations:

$$\nabla_{\boldsymbol{XT}} \cdot \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}) = \nabla_{\boldsymbol{X}} \cdot \boldsymbol{F}^{\text{vis}}(\boldsymbol{u}, \boldsymbol{q}), \tag{13}$$

$$\nabla_{\boldsymbol{X}} \boldsymbol{u} = \boldsymbol{q}. \tag{14}$$

This system if discretized using a standard Galerkin procedure. We introduce the broken space Σ_T^h as

$$\Sigma_T^h = \{\boldsymbol{\sigma} \in [L^2(\Omega[0,T])]^{4\times 2} \mid \boldsymbol{\sigma}|_K \in [\mathcal{P}_p(K)]^{4\times 2} \ \ \forall K \in \mathcal{T}_{[0,T]}^h\}, \tag{15}$$

and the DG formulation for equations (13) and (14) 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

$$\begin{aligned} -\int_K \tilde{\boldsymbol{F}}^{\text{inv}}(\boldsymbol{u}^h) : \nabla_{\boldsymbol{XT}} \boldsymbol{v}^h \, dx + \oint_{\partial K} \widehat{(\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n})} \cdot \boldsymbol{v}^h \, ds & \\ = -\int_K \boldsymbol{F}^{\text{vis}}(\boldsymbol{u}^h, \boldsymbol{q}^h) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^h \, dx + \oint_{\partial K} \widehat{(\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s)} \cdot \boldsymbol{v}^h \, ds, & \quad \forall \boldsymbol{v}^h \in \mathcal{V}_T^h \end{aligned} \tag{16}$$

$$\int_K \boldsymbol{q}^h : \boldsymbol{\sigma}^h \, dx = -\int_K \boldsymbol{u}^h \cdot (\nabla_{\boldsymbol{X}} \cdot \boldsymbol{\sigma}^h) \, dx + \oint_{\partial K} (\widehat{\boldsymbol{u}^h} \otimes \boldsymbol{n}_s) : \boldsymbol{\sigma}^h \, ds, \quad \forall \boldsymbol{\sigma}^h \in \Sigma_T^h. \tag{17}$$

Recall that $\boldsymbol{n_s} = (n_1, n_2)$ are the first two components (that is, the spatial component) of the outward unit normal $\boldsymbol{n}$ at the boundary ∂K.

The numerical fluxes $\widehat{\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s}$ and $\widehat{\boldsymbol{u}^h}$ are approximations to $\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s$ and $\boldsymbol{u}$ on the boundary of element K. For the inviscid part $\widehat{\tilde{\boldsymbol{F}}^{\text{inv}} \cdot \boldsymbol{n}}$, we use (11) as previously described. For the viscous numerical flux, we first perform a similar normalization:

$$\widehat{\boldsymbol{F}^{\text{vis}} \cdot \boldsymbol{n}_s} = |\boldsymbol{n}_s| \widehat{\boldsymbol{F}^{\text{vis}} \cdot \tilde{\boldsymbol{n}}_s}. \tag{18}$$

Since the viscous terms in equations (13) as well as all of equations (14) 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 modifications. Here, we choose the numerical fluxes according to the Compact Discontinuous Galerkin (CDG) method [51]. By equations (16) and (17), a non-linear discrete system is formulated which we again solve using a parallel Newton-GMRES solver [45].

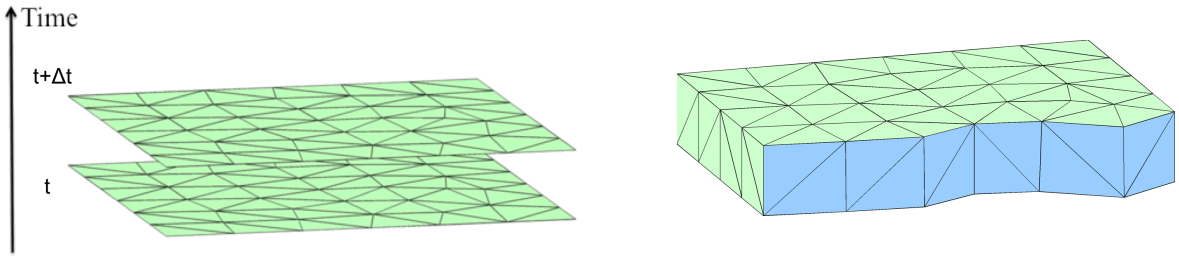


Figure 1: Space-Time Mesh Generation. The left figure illustrates two mesh layers at time t and $t + \Delta t$, and the right figure shows a corresponding 3D space-time mesh between the two layers. The blue faces show a cross-section of the tetrahedral mesh.

3. Space-Time Mesh Generation

Our mesh generator is based on the DistMesh algorithm [47], which iteratively improves a triangular mesh using only node movements and element connectivity updates [18]. To reduce the computational cost, we generate tetrahedral space-time meshes for each slab $\Omega[t, t + \Delta t]$ separately, see figure 1. Since the DistMesh algorithm is entirely based on local mesh modifications, the space-time mesh can be both efficiently and robustly generated directly from these operations.

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 for Ω^0 is needed, which is then improved at each subsequent time step. More importantly, all the mesh improvement techniques are performed on the 2D spatial mesh, and the tetrahedral triangulation is entirely based on local combinatorial connections.

3.1. Mesh Motion and Edge Flipping

At time $t = 0$, 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 operations on the previous triangulation $\mathcal{T}_0^h$. 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 use the DistMesh scheme [47] to improve the mesh.

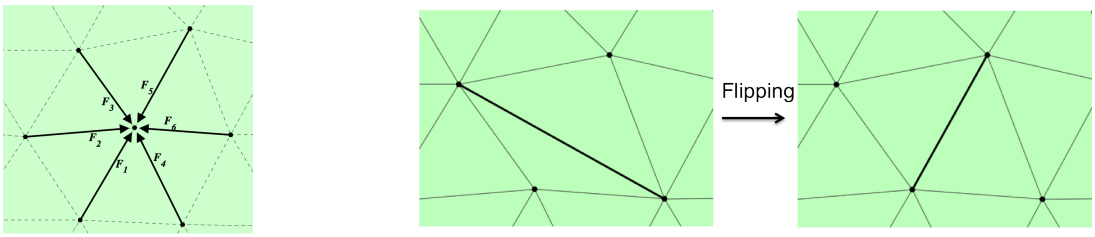


Figure 2: Force-based smoothing and edge flipping. The left plot shows the net force exerted on one node, and the right plot gives an example of edge flipping for improving the triangle qualities.

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

$$|\boldsymbol{F}(\boldsymbol{l})| = \begin{cases} k(l - l_0) & \text{if } l \geq l_0, \\ 0 & \text{if } l < l_0, \end{cases} \tag{19}$$

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 [47].

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

$$\boldsymbol{p}^{(n+1)} = \boldsymbol{p}^{(n)} + \delta \boldsymbol{F}(\boldsymbol{p}^{(n)}) \tag{20}$$

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

When the time-dependent domain undergoes large deformations, node movements are usually not sufficient to obtain high-quality elements and avoid element inversion. For our space-time meshes, however, we can perform local connectivity changes to improve the mesh qualities [18]. For a triangular mesh, this can be done using edge swapping operations [52] as shown in figure 2, where two adjacent triangles flip their shared edge and produce two new triangles sharing the new edge.

To simplify the tetrahedral triangulation algorithm in the later sections, we require that each element is flipped at most once during each time step, which means that all the flipped elements come in pairs. This simplification does not appear to impose a severe limitation in any of our numerical tests, which

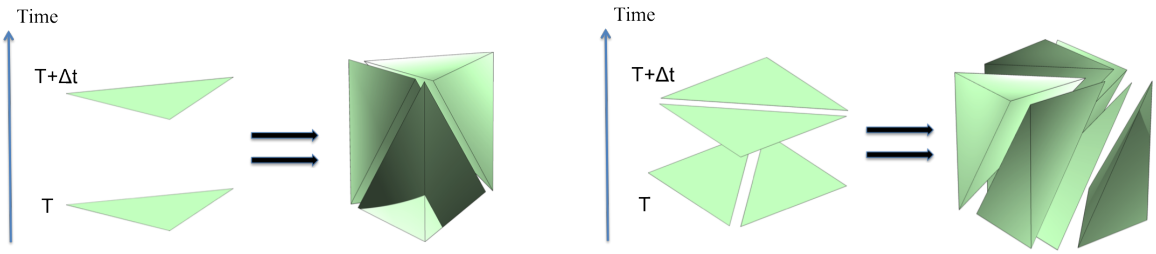


Figure 3: Tetrahedral Triangulation. The left plot illustrates a valid triangulation for an element without edge flips, and the right plot shows a triangulation for a pair of elements with a flipped edge.

include highly complex domain motions and deformations. In principle it is possible that multiple flips may be required to obtain high element qualities, in which case we simply reduce the time step. Another effective way to further alleviate this limitation is to restrict the mesh topology changes to certain regions of the domain and keep other parts of the mesh either fixed or rigidly following the boundary motion. By performing local mesh adaption only on the regions which allow for mesh topology changes, we obtain a sufficient number of flipped pairs to keep a high mesh quality.

Note that during the process described, the number of nodes, edges and elements remain unchanged (we discuss density control for adding or removing mesh nodes below). 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. In summary, the process will generate a sequence of two-dimensional meshes $\{\mathcal{T}_0^h, \mathcal{T}_{\Delta t}^h, \mathcal{T}_{2\Delta t}^h, \ldots, \mathcal{T}_T^h\}$ using only local mesh operations.

3.2. Tetrahedral Triangulation of the 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 $\Omega^{t+\Delta t}$.

First, consider an element of Ω^t without edge flipping (figure 3, left). This element can be 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). Next, for elements involved in an edge flip during the interval $[t, t + \Delta t]$, each can be extruded together with the paired element it flipped an edge with, which locally forms a quadrangular prism with two reverse diagonals on the top and the bottom faces (figure 3, right).

Again, similar to the unflipped case, the edges at Ω^t are not necessarily parallel to those at $\Omega^{t+\Delta t}$. However, for convenience in our notation, we will still refer to these vertically skew quadrilaterals as 'lateral faces' of the 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.

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 valid local triangulation of a prism, and second how to globally ensure that two adjacent prisms respect the same diagonal on their shared lateral face.

3.2.1. 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 counter-clockwise order. As shown in figure 4, 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 $1 \leq i \leq n$.

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

$$S_i^V(p_i^{V,t}, p_i^{V,t+\Delta t}, p_j^{V,t}, p_j^{V,t+\Delta t}) = \begin{cases} -1 & \text{if the diagonal edge is } \overline{p_i^{V,t} p_j^{V,t+\Delta t}} \\ +1 & \text{if the diagonal edge is } \overline{p_i^{V,t+\Delta t} p_j^{V,t}} \end{cases} \tag{21}$$

for $1 \leq i \leq n$.

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 (illustrated in figure 4, left). 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.

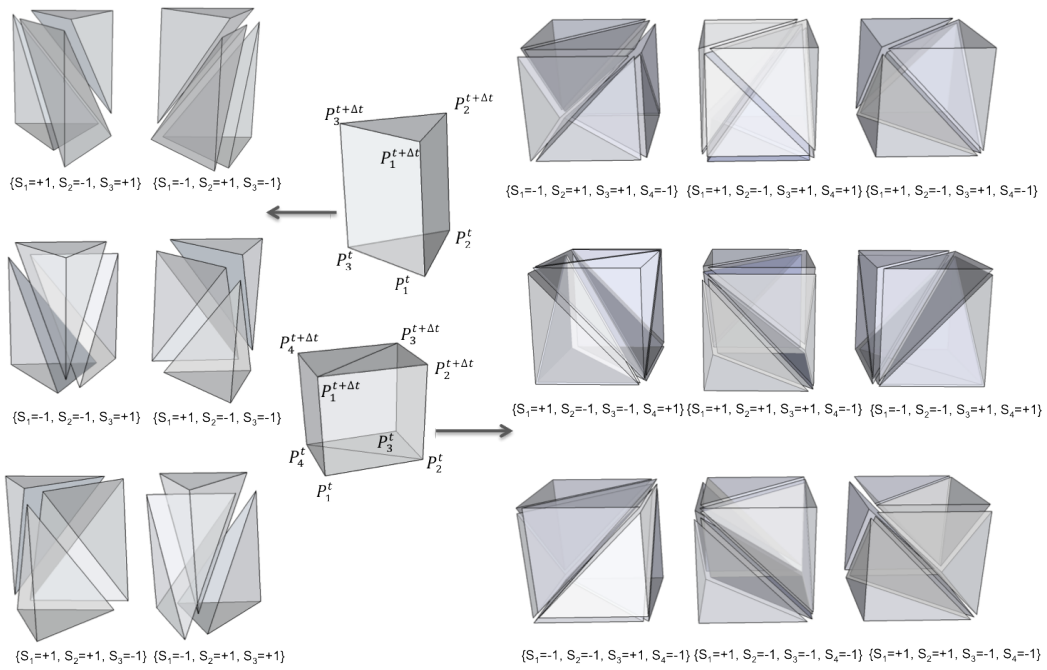


Figure 4: All the valid triangulations of a triangular prism (left) and of a quadrangular prism (right). The sets below each triangulation show the values of the corresponding sign functions.

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 and -1 if i is even.*

Since a quadrangular prism V has 4 lateral faces, a triangulation is determined 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 at least two consecutive sign functions S_i^V and $S_{mod(i,4)+1}^V$ which are set to their standard values.*

Geometrically, this condition means at least one of the 4 tetrahedra $\{p_1^{V,t}, p_2^{V,t}, p_4^{V,t}, p_1^{V,t+\Delta t}\}$, $\{p_2^{V,t}, p_3^{V,t}, p_4^{V,t}, p_3^{V,t+\Delta t}\}$, $\{p_2^{V,t}, p_1^{V,t+\Delta t}, p_2^{V,t+\Delta t}, p_3^{V,t+\Delta t}\}$ or $\{p_4^{V,t}, p_1^{V,t+\Delta t}, p_3^{V,t+\Delta t}, p_4^{V,t+\Delta t}\}$ must be formed and included in the final triangulation. This results in 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 (illustrated in figure 4, right).

3.2.2. Global Space-Time Mesh Generation

The last step is to obtain a global tetrahedral triangulation from the extruded prism elements between Ω^t to $\Omega^{t+\Delta t}$. This is nontrivial as the local triangulations are not independent, because each prism should match

the diagonals of shared lateral faces with their neighbor prisms. Here we describe an efficient depth-first algorithm which finds a global triangulation that satisfies these restrictions.

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.*

During the algorithm, we will make the assumption that each prism V only has three possible states as follows,

Triangular Prism at State 1. *$\{F_1^V, F_2^V, F_3^V\}$ are all accessible;*

Triangular Prism at State 2. *Exactly one of $\{F_1^V, F_2^V, F_3^V\}$ has become a wall and the other two are accessible;*

Triangular Prism at State 3. *$\{F_1^V, F_2^V, F_3^V\}$ have all become walls and $\{S_1^V, S_2^V, S_3^V\}$ can make a valid triangulation of V.*

Quadrangular Prism at State 1. *$\{F_1^V, F_2^V, F_3^V, F_4^V\}$ are all accessible;*

Quadrangular Prism at State 2. *Exactly one of the pair faces $\{F_1^V, F_3^V\}$ and $\{F_2^V, F_4^V\}$ have both become walls, at least one of two corresponding S_i^V was set to its standard value, and both F_i^V in the other pair are accessible.*

Quadrangular Prism at State 3. *$\{F_1^V, F_2^V, F_3^V, F_4^V\}$ have all become walls and $\{S_1^V, S_2^V, S_3^V, S_4^V\}$ can make a valid triangulation of V.*

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

Operation 1: Optimal Local Triangulation of Prisms.

Based on the assumption, throughout the algorithm, if V has not been triangulated it must be at state 1 or 2. We then choose an optimal local triangulation of V by

$$\arg\max_{\mathcal{T}^V} \min_{K \in \mathcal{T}^V} Q(K) \tag{22}$$

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 measure proposed by Field [53]

$$Q(K) = 72\sqrt{3}\frac{\mathrm{Vol}(K)}{(\sum_{i=1}^{6} l_i(K)^2)^{3/2}}, \tag{23}$$

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

From the local triangulations in figure 4, it can easily be verified that $\mathcal{T}^V$ is nonempty when V is at state 1 or 2. In other words, we can always find a valid triangulation of V.

Operation 2: Sign Function Synchronization of Neighbor Prisms.

When a prism V is triangulated by operation 1, in order to not violate the assumption made at the beginning, we have to transfer V to state 3. Therefore, as 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^*. Again, according to the assumption, V^* must be in state 1 or 2 since $F_{i^*}^{V^*}$ was accessible before the triangulation of V. So in total, there are 4 possible cases for V^*:

Case 1. *If V^* is triangular at state 1, we simply set $S_{i^*}^{V^*} = S_i^V$, which makes F_i^V and $F_{i^*}^{V^*}$ walls and transfers V^* to state 2;*

Case 2. *If V^* is quadrangular at state 1, we set $S_{i^*}^{V'} = S_i^V$ and $S_{mod(i^*+1,4)+1}^{V^*}$ to their standard values. This transfers V^* into state 2. 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 at state 2 with a wall $F_{j^*}^{V^*}$ for some $j^* \neq i^*$. We then 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 $F_{k^*}^{V^*}$. To transfer V^* to state 3, we have to continue updating sign functions of V^{**} recursively using operation 2;*

Case 4. *Suppose V^* is quadrangular at state 2 with a pair of opposite walls, say, $S_{j^*}^{V^*}$ and $S_{k^*}^{V^*}$ (where $i^* \neq j^*$ and $i^* \neq k^*$). 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 $S_{i^*}^{V^*} = S_i^V$. In spite of having three restrictions, we claim that the equation (22) is always solvable for V^*. To prove this, we first note that either $S_{j^*}^{V^*}$ or $S_{k^*}^{V^*}$ has been set to their standard values since V^* is at state 2. Therefore, as long as $S_{l^*}^{V'}$ is set to the standard value, the combination of sign functions must satisfy Condition 1 and thus gives a valid local triangulation. 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, in order to turn $F_{l^*}^{V^*}$ into a wall and thereby transfer V^* to state 3.*

Operation 3: Triangulation Adjustment of Root Prism.

As shown in figure 5, 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 5, right); 2. a domain boundary; 3. back to the root prism V from a face which is not yet a wall (figure 5, 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 $F_{i'}^{V'} = F_i^V$ 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.

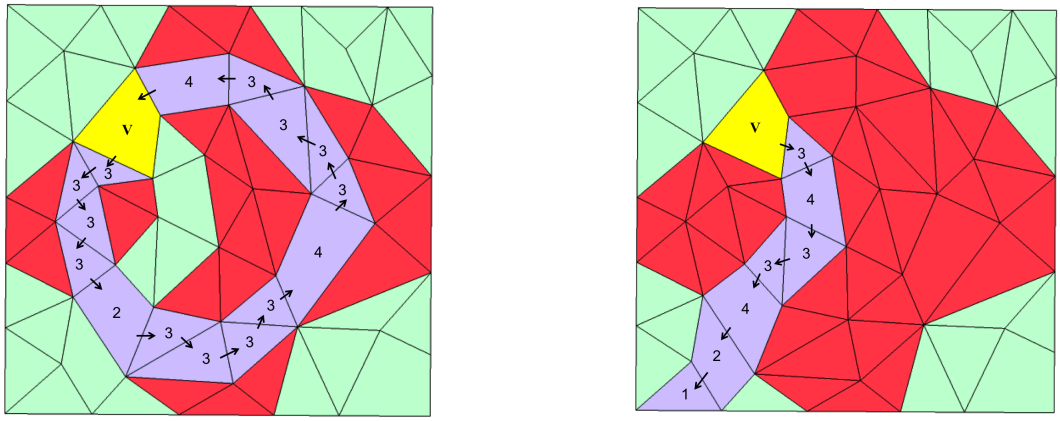


Figure 5: The examples of updating paths. Each triangle represents a triangular prism and each quadrilateral represents a quadrangular prism. The yellow element is the root prism V, the green elements are prisms at state either 1 or 2, and the red elements are already triangulated, i.e., at state 3. 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.

It is clear from the local triangulations derived in Section 3.2.1 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. We will avoid this situation by arranging the order by which new updating paths are launched.

First, we consider a triangular root prism V. If V was at state 1 before its triangulation, at least two updating paths will be launched from V. If the first two updating paths are launched from faces with different values of their sign functions, then there will always be a valid triangulation of V regardless of whether any path will return to V or the value of the third sign function will be changed. In fact, the condition above can always be satisfied if the first updating path is launched from the face with a sign function value different from the other two. Similarly, if V was at state 2 with a wall F_i^V before its triangulation, operation 3 will not destroy the validity of the local triangulation provided that the first updating path is launched from a face with sign function value different from S_i^V.

Next we consider the case that the root prism V is quadrangular. Recall that for a valid triangulation of V to exist, it is required that there are two consecutive sign functions set to their standard values. If V was at state 2 before its triangulation, it must have a wall, say F_i^V, whose sign function was set to the standard value. After triangulation by operation 1, there will be another face adjacent to F_i^V, say F_j^V, whose sign function is also set to the standard value. When the first updating path from F_j^V is launched, Condition 1 will then not be violated even if operation 3 is applied. Finally, if V was in state 1, to respect Condition 1 we have to launch the first two updating paths from faces with standard sign function values. This can always be done, since no face of V is a wall and thus any one can be changed to the standard value if necessary.

Based on the three operations described above, the full algorithm is summarized in algorithm 1. Recall that we described the operations based on the assumption that throughout the algorithm, each prism only has three possible states. In fact, initially all prisms faces are set to state 1; and as the algorithm progresses, the operations in the algorithm can only change a prism into state 2 or 3. Finally, the global tetrahedral triangulation is complete if and only if all the prism faces have become walls. Therefore, by induction, it is clear that the assumption holds for all prisms and that the algorithm will return a global triangulation of the space-time domain $\Omega[t, t + \Delta t]$.

Algorithm 1 Space-Time Mesh Generation

Require: A spatial mesh $MESH1$ of Ω^t and $MESH2$ of $\Omega^{t+\Delta t}$
Ensure: 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 is not at state 3 **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 discussed 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 resultant triangulation of V into $STMESH$
  end while
  return $STMESH$

3.3. Density Control and Other Local Mesh Operations

In many applications with large domain deformations, a dynamically changing mesh size functions is needed as the mesh changes. To incorporate this into our space-time meshes, we present two more local mesh operations – edge splitting and edge collapsing [54], for locally increasing or reducing the number of elements when mesh density control is needed.

For edge splitting (see figure 6, top), we can locally refine the mesh by adding the middle point of an edge and then splitting two adjacent elements into four smaller ones. On the other hand, if a coarser local mesh is needed, we consider two cases for edge collapsing. If a node is shared by three elements, we simply

remove it and substitute a new larger element (figure 6, middle). If a node is shared by four elements, we remove it and locally re-triangulate the resulting quadrilateral into two triangular elements (figure 6, bottom). Besides coarsening of the mesh, edge collapsing can also be used to improve the mesh quality by removing thin elements with big obtuse angles. As before, in order to avoid the possible conflict between the mesh operations, we require that within a time step any element can be involved in at most one edge flip, split, or collapse operation.

One advantage of these three local mesh operations is that like the edge flipping, they do not change any mesh connectivities except for the elements involved. Locally, in the setting of the space-time framework, these elements can be extruded together along the temporal dimension. For the first case of edge collapsing, it forms a triangular prism. For edge splitting and the second case of edge collapsing, a quadrangular prism is created. It can easily be verified that the local triangulations of these new types of prisms respect the same conditions as in Section 3.2.1. More precisely, using the terminology in the previous sections, if $\{S_1^V, S_2^V, S_3^V\}$ of a triangular prism have different values or $\{S_1^V, S_2^V, S_3^V, S_4^V\}$ of a quadrangular prism satisfy Condition 1, a valid triangulation of these new prisms can be obtained by adding one or two interior edges. Three examples of these triangulations are illustrated in figure 6, right. In summary, this allows us to use our global space-time mesh generation algorithm as described in Section 3.2.2 without any modifications.

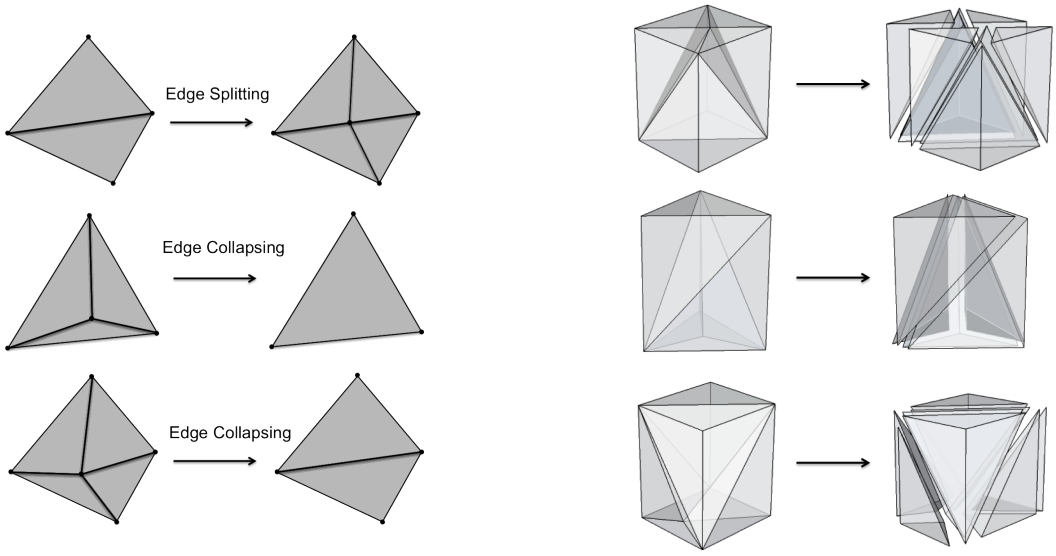


Figure 6: Edge Splitting and Edge collapsing. These operations are used to control the local mesh density by adding or removing nodes. The right figures show three examples of local space-time triangulations, one for each case.

4. Numerical Results

4.1. 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 analytical solution at (x, y, t) is given by

$$u = u_\infty(\cos\theta - \frac{\epsilon((y-y_0)-\bar{v}t)}{2\pi r_c}\exp(\frac{f(x,y,t)}{2})) \tag{24}$$

$$v = u_\infty(\sin\theta + \frac{\epsilon((x-x_0)-\bar{u}t)}{2\pi r_c}\exp(\frac{f(x,y,t)}{2})) \tag{25}$$

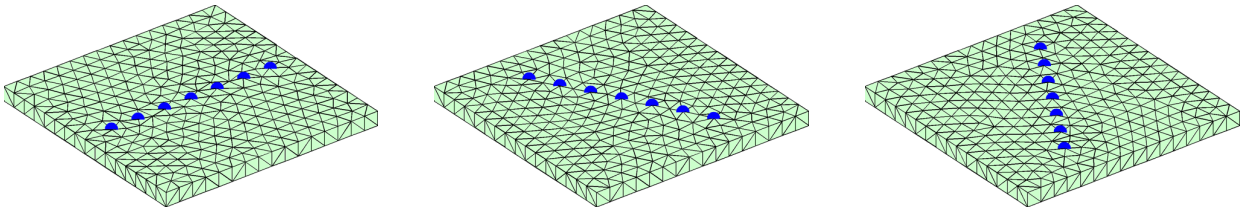
$$\rho = \rho_\infty(1 - \frac{\epsilon^2(\gamma-1)M_\infty^2}{8\pi^2}\exp(f(x,y,t)))^{\frac{1}{\gamma-1}} \tag{26}$$

$$p = p_\infty(1 - \frac{\epsilon^2(\gamma-1)M_\infty^2}{8\pi^2}\exp(f(x,y,t)))^{\frac{\gamma}{\gamma-1}} \tag{27}$$

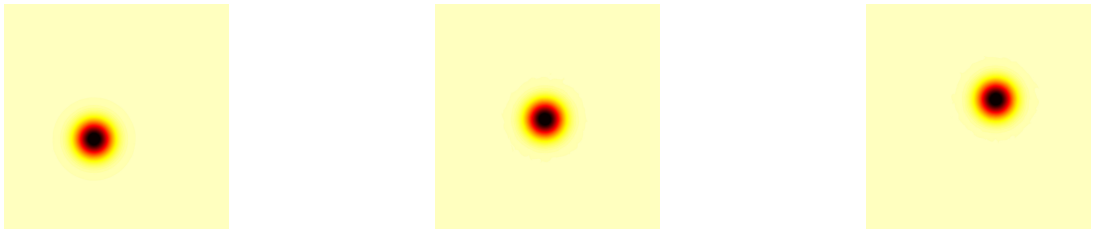
where $f(x,y,t) = (1 - ((x-x_0)-\bar{u}t)^2 - ((y-y_0)-\bar{v}t)^2)/r_c^2$, $M_\infty = 0.5$ is the Mach number, $\gamma = c_p/c_v$, and u_∞, p_∞, ρ_∞ are free-stream velocity, pressure and density, respectively. Moreover, $\bar{u}$ and $\bar{v}$ are the Cartesian components of the free-stream velocity 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 a starting point, an unstructured mesh of the domain is created with element size h by DistMesh [47]. In order to show that our method remains high-order accurate even for large mesh deformations, we rotate some of the vertices (the blue nodes showed in figure 7) about the center of the domain with the angular velocity $\omega = \frac{2}{3}\pi$, which induces large mesh deformations. To avoid inverted and low-quality elements, we improve the elements 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. 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 spatial discretization. Note that this choice of small Δt is not due to the limitations of the edge flips or due to any stability restrictions from the discretization, we could have chosen a much larger Δt if we allowed for temporal errors.

We carry out the convergence test for a range of spatial mesh sizes h and polynomial degrees p. For a comparison, we also solve the problem on a fixed mesh, where no vertices are rotated and thus the initial



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 $t = 5.0$

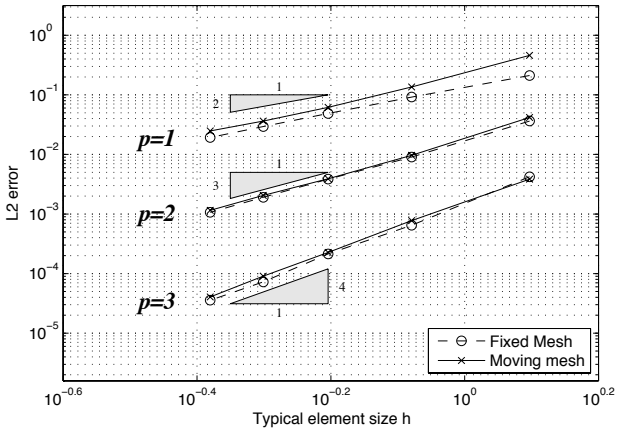


Figure 7: Convergence Test for the Euler Vortex Problem. The top three plots are samples of space-time meshes of $\Omega[0, \Delta t]$, $\Omega[2.5, 2.5 + \Delta t]$ and $\Omega[5, 5 + \Delta t]$ with $\Delta t = 5 \times 10^{-3}$, $h = 1.25$ and $p = 3$. In these plots, the thickness of each space-time 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 a rigid manner, which induces other vertex movements and local element flipping. Below each mesh, the corresponding solutions are shown (pressure fields). The bottom plot shows the convergence results for $p = 1$, 2 and 3.

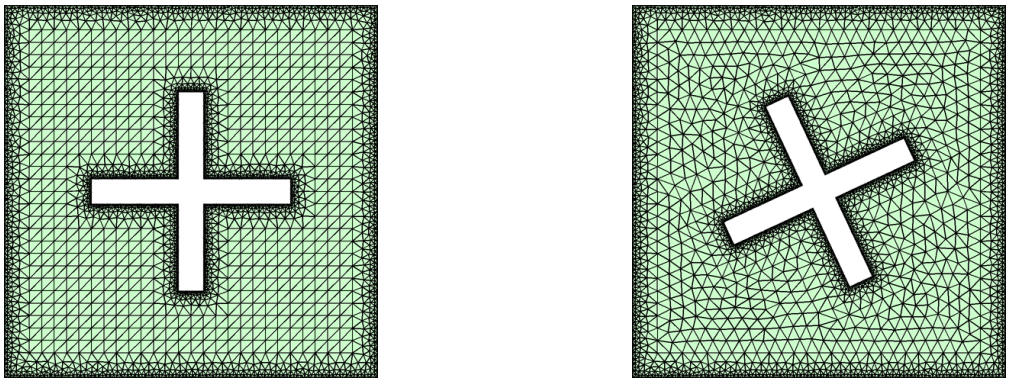


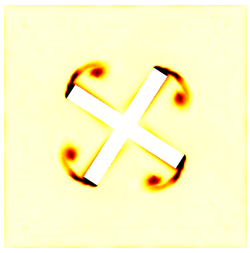
Figure 8: Spatial Meshes of a Spinning Cross. The left plot shows the initial mesh of the spatial domain Ω^0, the right plot shows the mesh of the spatial domain $\Omega^{2.0}$

unstructured mesh remains unchanged for all time steps. In figure 7, three sample space-time meshes and solutions of the pressure field are given, and the bottom convergence plot compares the errors in the L^2-norm of our space-time DG method for the moving and the fixed mesh, respectively. It can be seen that the solutions from our moving meshes have essentially the same accuracy as the ones using a fixed mesh. This is different from the mapping-based ALE schemes commonly used with high-order methods, as shown in [13] where a similar Euler vortex problem gave less accurate results on a moving mesh than on a fixed mesh, since the ALE mapping between the physical and the reference domain introduced undesired variations in the mesh resolution. In our space-time framework, although the mesh is moving in the spatial domain, the tetrahedral mesh is fixed for each space-time domain $\Omega[t, t + \Delta t]$ and no mapping is employed. From the convergence plot, the results clearly show that the orders of convergence are approximately $O(h^{p+1})$.

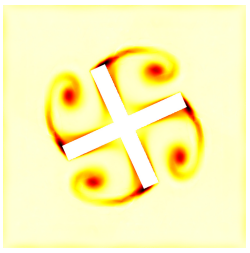
4.2. Spinning Cross

Next, we consider a 3-by-3 square domain where a cross is spinning counterclockwise at the center, which is similar to the example problem in [38]. We solve the compressible Navier-Stokes equations starting from a zero-velocity initial condition, and set the Mach number 0.2 and Reynolds number 1500, based on the cross diameter 1.5 and the angular velocity $\omega = 1$.

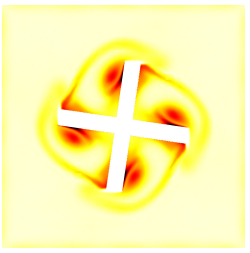
We initialize the mesh by gluing three parts together: two graded meshes around the fixed outside walls and the spinning cross walls, and a Cartesian grid in the middle (as shown in figure 8). To simplify the mesh movement, we move the graded mesh around the cross rigidly with the geometry movement, fix the graded mesh around the outside walls, and only apply our moving mesh and flipping techniques to the uniform mesh in the middle. Figure 8 shows that our method can retain the quality of the elements even with the large deformations induced by the spinning cross.



Entropy Plot at $t = 1.0$

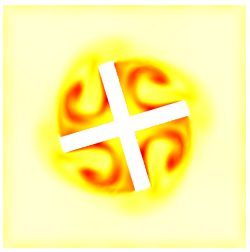


Entropy Plot at $t = 2.0$

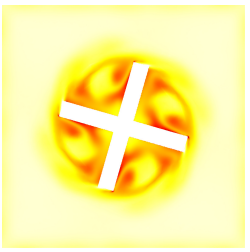


Entropy Plot at $t = 3.0$

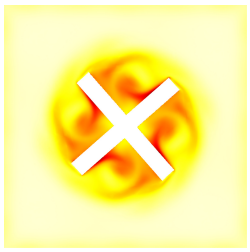
Entropy Plot at $t = 4.0$



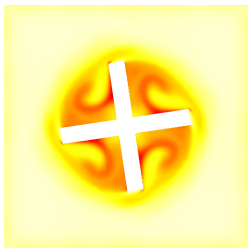
Entropy Plot at $t = 5.0$



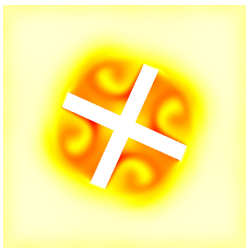
Entropy Plot at $t = 6.0$



Entropy Plot at $t = 7.0$



Entropy Plot at $t = 8.0$



Entropy Plot at $t = 9.0$

Figure 9: Compressible Navier-Stokes flow in a domain with a spinning cross (entropy).

Finally, to better resolve the solution, we implement our space-time DG method with polynomial orders $p = 2$ with linear element geometries. Entropy plots of the flow solutions at a few time steps are shown in figure 9. Note that the optimal order of accuracy $O(h^{p+1})$ will not be achieved unless the boundary of the cross is also curved, which is an important future improvement.

4.3. Pitching Tandem Airfoils

We next consider a Navier-Stokes simulation similar to the one studied in [55]. It consists of two pitching NACA 0012 airfoils with chord length $c = 1$ in a 6×2 rectangular domain. As shown in figure 10, 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

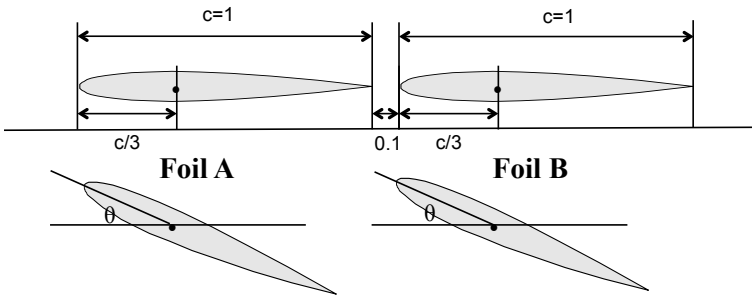


Figure 10: The pitching tandem airfoil model.

of their leading edges. The rotation follows a prescribed harmonic function as

$$\theta = A\sin(-2\pi f t) \tag{28}$$

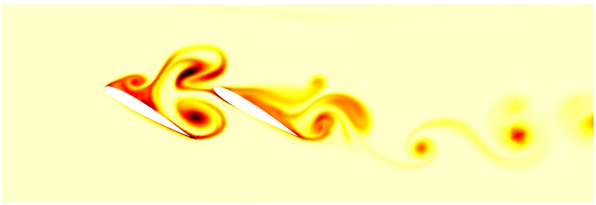
where $A = \pi/6$ and $f = 0.05$. The flow has Mach number 0.2 and Reynolds number 3000.

As the two foils are placed very close and rotated based on the same harmonic function, 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. We again implement our space-time DG method with polynomial order $p = 2$ with linear element geometries. Three sample meshes and the corresponding entropy plots are given in figure 11. In figure 12, plots of drag and lift coefficients are shown.

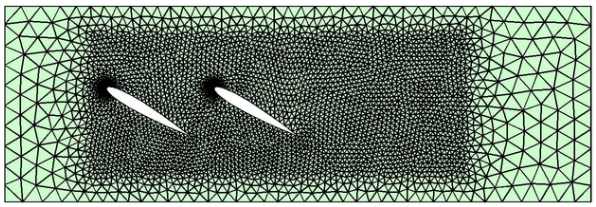
4.4. Airfoil with a Deploying Spoiler

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, similar to the problem introduced in [56]. As illustrated in figure 13, 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. 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.

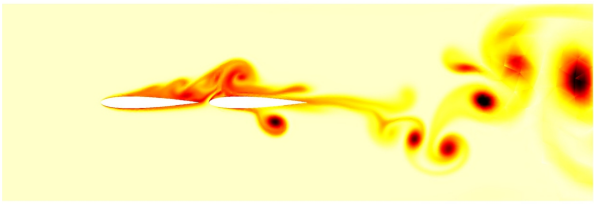
The numerical simulation starts with a steady flow around the flat foil with a closed spoiler, at Mach



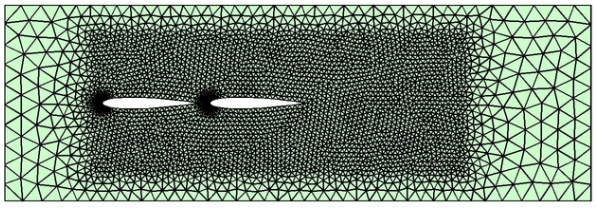
Entropy Plot at $t = 5.0$



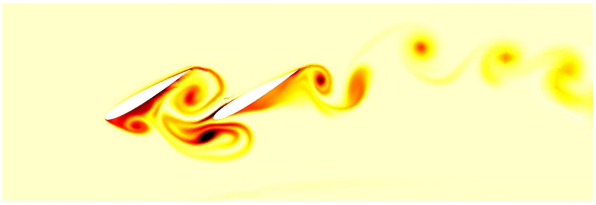
Unstructured Mesh of the spatial domain at $t = 5.0$



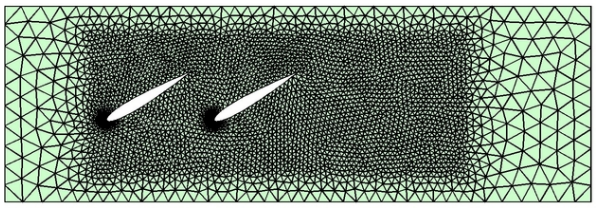
Entropy Plot at $t = 10.0$



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

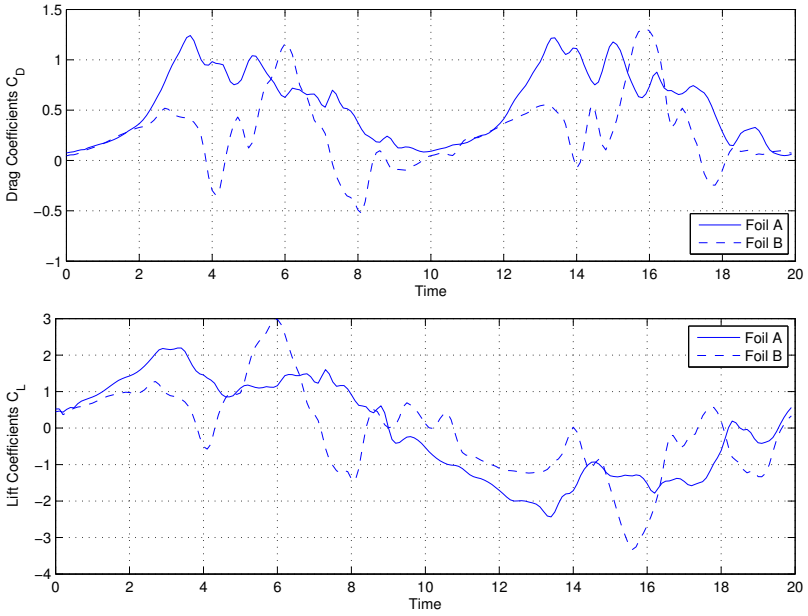
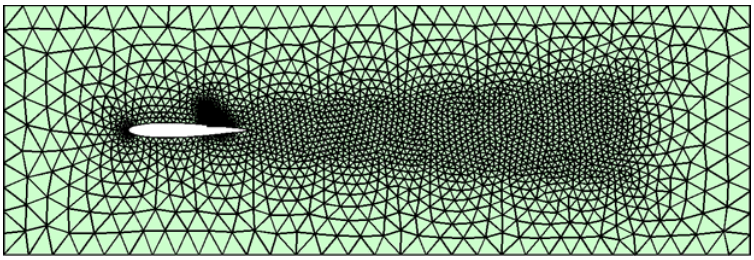
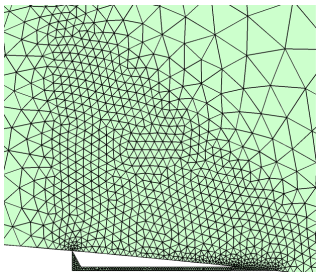


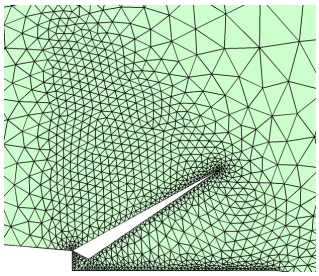
Figure 12: Drag and lift coefficients around the pitching tandem NACA0012 airfoils as a function of time.



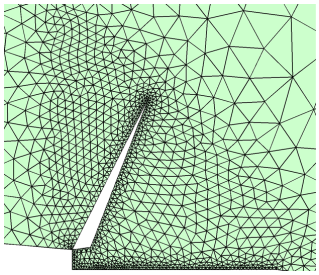
The initial spatial mesh at $t = 0.0$



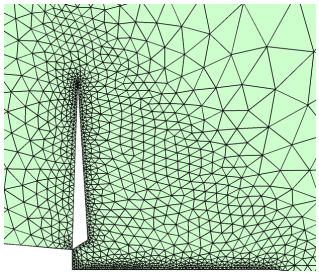
Zoom-in spatial mesh around spoiler at $t = 0.0$



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



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

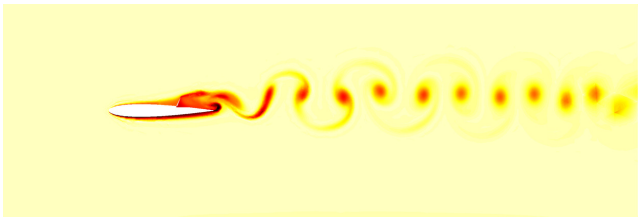


Zoom-in spatial mesh around spoiler at $t = 18.0$

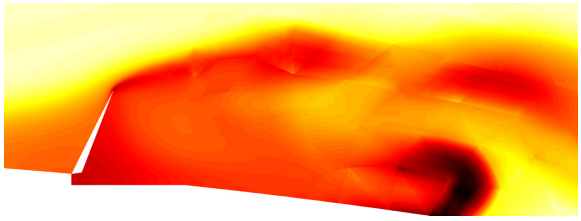
Figure 13: Spatial Meshes of Airfoil with a Deploying Spoiler.

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 tests, we use polynomial orders $p = 2$ with linear element geometries, in order to better resolve the solution fields.

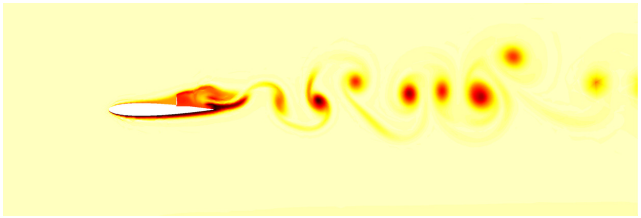
In figure 13, some mesh plots are given to show how our local mesh operations improve the spatial mesh as the spoiler is raised, and three samples of entropy plots are shown in figure 14. In the zoom-in plots, we can confirm that our space-time DG method retains the high quality of the solutions even for the large



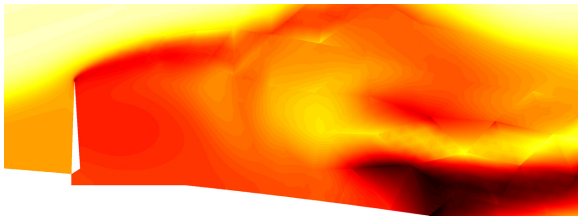
Entropy Plot at $t = 12.0$



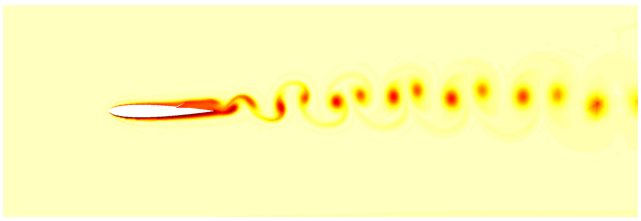
Zoom-in Entropy Plot around spoiler at $t = 12.0$



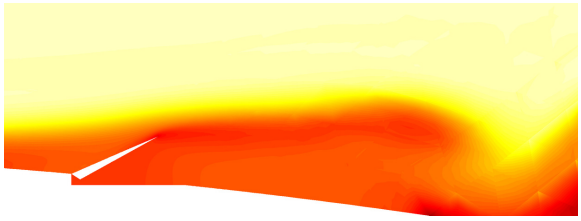
Entropy Plot at $t = 28.0$



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 14: Compressible Navier-Stokes flow around an airfoil with a deploying spoiler .

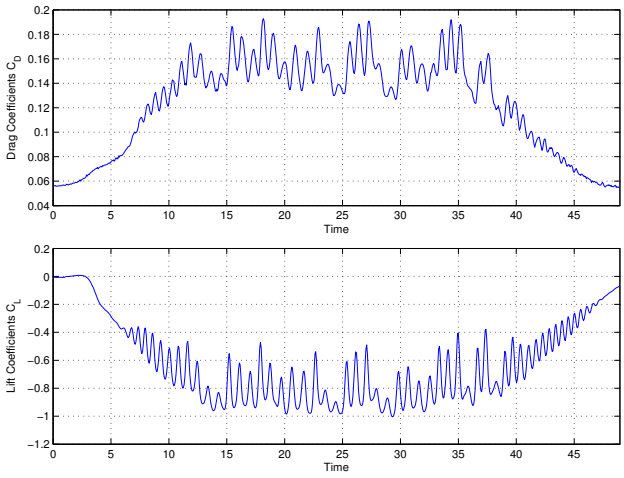


Figure 15: Drag and lift coefficients around the NACA0012 airfoil with a deploying spoiler as a function of time.

deformation between the foil and the spoiler. The lift and the drag coefficients during the entire process are shown in figure 15.

5. Conclusions and Future Work

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 on the spatial mesh only, we generate simplex elements for slabs of space-time domains within each timestep separately and use implicit solvers to advance the solution in time. Our method can handle complex domain changes and mesh reconfigurations, without reduced accuracy or conservation problems. We demonstrated the scheme on a model test problem as well as applications involving complex mesh motions.

One of the main goals of our future work is the extension of the space-time mesh generation procedure to three spatial dimensions plus time. The moving mesh generation extends to 3D in a straight-forward way [47], although the corresponding element topology operations (edge flips and face swaps) are more complicated [18]. However, they are still local in the sense that they only replace a number of neighboring elements by an alternate set of elements. Therefore, given two layers of 3D tetrahedral meshes, the structures of the corresponding 4D prismatic elements are given by similar point-wise connections between the two meshes. The local triangulations of these 4D prisms can be analyzed using the process described in this paper. For unflipped tetrahedra, their triangulation depends entirely on the combinatorial properties of the prism vertices. For the tetrahedra involved in element flips, their triangulation becomes more complicated. One option is to allow for insertion of an additional vertex within each 4D prism corresponding to the flipped tetrahedra, which can always achieve a valid triangulation for any combination of connections between the vertices. The robustness of this approach and the resulting mesh quality has to been further investigated. Finally, another important extension for high-order accurate methods is the generation of curved space-time elements.

Acknowledgments

We would like to acknowledge the generous support from the AFOSR Computational Mathematics program under grant FA9550-10-1-0229, the Alfred P. Sloan foundation, and the Director, Office of Science, Computational and Technology Research, U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] B. Cockburn, C.-W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (2001) 173–261.

[2] J. Peraire, P.-O. Persson, Adaptive High-Order Methods in Computational Fluid Dynamics, volume 2 of *Advances in CFD*, World Scientific Publishing Co.

[3] P. Colella, D. T. Graves, B. J. Keen, D. Modiano, A Cartesian grid embedded boundary method for hyperbolic conservation laws, J. Comput. Phys. 211 (2006) 347–366.

[4] M. J. Aftosmis, M. J. Berger, G. Adomavicius, A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries, in: 38th AIAA Aerospace Sciences Meeting and Exhibit. AIAA-2000-0808.

[5] M. J. Aftosmis, M. J. Berger, J. E. Melton, Robust and efficient cartesian mesh generation for component-based geometry, in: 35th AIAA Aerospace Sciences Meeting and Exhibit. AIAA-97-0196.

[6] C. S. Peskin, The immersed boundary method, Acta numerica 11 (2002) 479–517.

[7] D. Kim, H. Choi, Immersed boundary method for flow around an arbitrarily moving body, Journal of Computational Physics 212 (2006) 662–680.

[8] R. Glowinski, T. Pan, T. Hesla, D. Joseph, J. Periaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow, Journal of Computational Physics 169 (2001) 363–426.

[9] D. Wan, S. Turek, Fictitious boundary and moving mesh methods for the numerical simulation of rigid particulate flows, Journal of Computational Physics 222 (2007) 28–56.

[10] J. Donea, Arbitrary lagrangian-eulerian finite element methods, Computational methods for transient analysis(A 84-29160 12-64). Amsterdam, North-Holland, 1983, (1983) 473–516.

[11] C. Farhat, P. Geuzaine, Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids, Comput. Methods Appl. Mech. Engrg. 193 (2004) 4073–4095.

[12] I. Lomtev, R. M. Kirby, G. E. Karniadakis, A discontinuous Galerkin ALE method for compressible viscous flows in moving domains, J. Comput. Phys. 155 (1999) 128–159.

[13] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1585–1595.

[14] C. A. A. Minoli, D. A. Kopriva, Discontinuous galerkin spectral element approximations on moving meshes, Journal of Computational Physics 230 (2011) 1876–1902.

[15] D. J. Mavriplis, C. R. Nastase, On the geometric conservation law for high-order discontinuous galerkin discretizations on dynamically deforming meshes, Journal of Computational Physics 230 (2011) 4285–4300.

[16] G. Compere, J.-F. Remacle, J. Jansson, J. Hoffman, A mesh adaptation framework for dealing with large deforming meshes, International journal for numerical methods in engineering 82 (2010) 843–867.

[17] G. Olivier, F. Alauzet, A new changing-topology ale scheme for moving mesh unsteady simulations, in: 49th AIAA Aerospace Sciences Meeting, AIAA Paper, volume 474, pp. 252–271.

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

[19] D. Isola, A. Guardone, Simulation of flows with strong shocks with an adaptive conservative scheme, J. Comput. Appl. Math. 236 (2012) 4660–4670.

[20] A. Guardone, D. Isola, G. Quaranta, Arbitrary Lagrangian Eulerian formulation for two-dimensional flows using dynamic meshes with edge swapping, J. Comput. Phys. 230 (2011) 7706–7722.

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

[22] T. J. Hughes, G. M. Hulbert, Space-time finite element methods for elastodynamics: formulations and error estimates, Computer methods in applied mechanics and engineering 66 (1988) 339–363.

[23] G. M. Hulbert, T. J. Hughes, Space-time finite element methods for second-order hyperbolic equations, Computer Methods in Applied Mechanics and Engineering 84 (1990) 327–348.

[24] S. Aliabadi, T. Tezduyar, Space-time finite element computation of compressible flows involving moving boundaries and interfaces, Computer Methods in Applied Mechanics and Engineering 107 (1993) 209–223.

[25] A. Masud, T. J. Hughes, A space-time galerkin/least-squares finite element formulation of the navier-stokes equations for moving domain problems, Computer Methods in Applied Mechanics and Engineering 146 (1997) 91–126.

[26] C. Johnson, Discontinuous galerkin finite element methods for second order hyperbolic problems, Computer Methods in Applied Mechanics and Engineering 107 (1993) 117–129.

[27] J. Sudirham, J. van der Vegt, R. van Damme, Space–time discontinuous galerkin method for advection–diffusion problems on time-dependent domains, Applied numerical mathematics 56 (2006) 1491–1518.

[28] J. J. W. van der Vegt, H. van der Ven, Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation, J. Comput. Phys. 182 (2002) 546–585.

[29] H. Van der Ven, J. Van der Vegt, Space–time discontinuous galerkin finite element method with dynamic grid motion for inviscid compressible flows: Ii. efficient flux quadrature, Computer methods in applied mechanics and engineering 191 (2002) 4747–4780.

[30] C. M. Klaij, J. J. W. van der Vegt, H. van der Ven, Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations, J. Comput. Phys. 217 (2006) 589–611.

[31] J. J. W. van der Vegt, J. J. Sudirham, A space-time discontinuous Galerkin method for the time-dependent Oseen equations, Appl. Numer. Math. 58 (2008) 1892–1917.

[32] W. E. H. Sollie, O. Bokhove, J. Van der Vegt, Space–time discontinuous galerkin finite element method for two-fluid flows, Journal of Computational Physics 230 (2011) 789–817.

[33] C. M. Klaij, J. J. van der Vegt, H. van der Ven, Pseudo-time stepping methods for space–time discontinuous galerkin discretizations of the compressible navier–stokes equations, Journal of Computational Physics 219 (2006) 622–643.

[34] C. M. Klaij, M. H. van Raalte, H. van der Ven, J. J. van der Vegt, ¡ i¿ h¡/i¿-multigrid for space-time discontinuous galerkin discretizations of the compressible navier–stokes equations, Journal of Computational Physics 227 (2007) 1024–1045.

[35] S. Rhebergen, B. Cockburn, A space-time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains, J. Comput. Phys. 231 (2012) 4185–4204.

[36] S. Rhebergen, B. Cockburn, J. J. W. van der Vegt, A space-time discontinuous Galerkin method for the incompressible Navier-Stokes equations, J. Comput. Phys. 233 (2013) 339–358.

[37] K. Mani, D. Mavriplis, Efficient solutions of the euler equations in a time-adaptive space-time framework, in: 49th AIAA Aerospace Sciences Meeting and Exhibit. AIAA-2011-774.

[38] T. C. S. Rendall, C. B. Allen, E. D. C. Power, Conservative unsteady aerodynamic simulation of arbitrary boundary motion using structured and unstructured meshes in time, Internat. J. Numer. Methods Fluids 70 (2012) 1518–1542.

[39] A. Üngör, A. Sheffer, Pitching tents in space-time: mesh generation for discontinuous Galerkin method, Internat. J. Found. Comput. Sci. 13 (2002) 201–221. Volume and surface triangulations.

[40] A. Üngör, A. Sheffer, R. B. Haber, S.-H. Teng, Layer based solutions for constrained space-time meshing, Appl. Numer. Math. 46 (2003) 425–443. Applied numerical computing: grid generation and solution methods for advanced simulations.

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

[42] J. Erickson, D. Guoy, J. M. Sullivan, A. Üngör, Building spacetime meshes over arbitrary spatial domains, Engineering with Computers 20 (2005) 342–353.

[43] S. Thite, Adaptive spacetime meshing for discontinuous galerkin methods, Computational Geometry 42 (2009) 20–44.

[44] M. Behr, Simplex space–time meshes in finite element simulations, International journal for numerical methods in fluids 57 (2008) 1421–1434.

[45] P.-O. Persson, J. Peraire, Newton-GMRES preconditioning for discontinuous Galerkin discretizations of the Navier-Stokes equations, SIAM J. Sci. Comput. 30 (2008) 2709–2733.

[46] P.-O. Persson, Scalable parallel Newton-Krylov solvers for discontinuous Galerkin discretizations, in: 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida. AIAA-2009-606.

[47] P.-O. Persson, G. Strang, A simple mesh generator in matlab, SIAM Review 46 (2004).

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

[49] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.

[50] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM J. Numer. Anal. 39 (2001/02) 1749–1779.

[51] J. Peraire, P.-O. Persson, The compact discontinuous Galerkin (CDG) method for elliptic problems, SIAM J. Sci. Comput. 30 (2008) 1806–1824.

[52] S. K. Lahiri, J. Bonet, J. Peraire, A variationally consistent mesh adaptation method for triangular elements in explicit Lagrangian dynamics, Internat. J. Numer. Methods Engrg. 82 (2010) 1073–1113.

[53] D. A. Field, Qualitative measures for initial meshes, Internat. J. Numer. Methods Engrg. 47 (2000) 887–906.

[54] P.-O. Persson, Mesh Generation for Implicit Geometries, Ph.D. thesis, M.I.T., 2005.

[55] R. A. Shirsath, R. Mukherjee, Unsteady aerodynamics of tandem airfoils pitching in phase, in: 2nd International Conference on Mechanical, Production and Automobile Engineering (ICMPAE'2012).

[56] G. Quaranta, D. Isola, A. Guardone, Numerical simulation of the opening of aerodynamic control surfaces with two-dimensional unstructured adaptive meshes, in: 5th European Conference on Computational Fluid Dynamics - ECCOMAS CFD 2010, volume 236.