Journal of Computational Physics **155,** 128–159 (1999)
Article ID jcph.1999.6331, available online at http://www.idealibrary.com on IDEAL®

A Discontinuous Galerkin ALE Method for Compressible Viscous Flows in Moving Domains

I. Lomtev, R. M. Kirby, and G. E. Karniadakis[1]

Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912
E-mail: gk@cfm.brown.edu

Received February 10, 1999; revised June 10, 1999

We present a matrix-free discontinuous Galerkin method for simulating compressible viscous flows in two- and three-dimensional moving domains. To this end, we solve the Navier–Stokes equations in an arbitrary Lagrangian Eulerian (ALE) framework. Spatial discretization is based on standard structured and unstructured grids but using an orthogonal hierarchical spectral basis. The method is third-order accurate in time and converges exponentially fast in space for smooth solutions. A novelty of the method is the use of a force-directed algorithm from graph theory that requires no matrix inversion to efficiently update the grid while minimizing distortions. We present several simulations using the new method, including validation with published results from a pitching airfoil, and new results for flow past a three-dimensional wing subject to large flapping insect-like motion.

1. INTRODUCTION

Despite great research efforts in designing good unstructured grids for aerodynamic flows, especially for three-dimensional simulations, most finite element and finite volume solutions depend strongly on the quality of the grid. For highly distorted grids convergence is questionable, and in most cases convergence rates are typically less than second-order. Moreover, efforts to increase the accuracy of finite volume methods to higher than second-order have not been very successful as *conservativity* in the formulation or *monotonicity* of the solution has to be compromised. These difficulties are particularly pronounced for simulations in moving computational domains involving aeroelastic motion and other flow-structure interaction problems [1, 2].

In the current work we develop algorithms for the *compressible* Navier–Stokes in moving domains employing high-order spectral/*hp* element discretizations. Unlike our previous

[1] To whom correspondence should be addressed.

0021-9991/99 $30.00

work for incompressible flows [3] or the mixed formulation approach for two-dimensional compressible flows in [4], here we use a *discontinuous Galerkin* approach that allows the use of an *orthogonal* polynomial basis of different order in each element. In particular, we develop a discontinuous Galerkin formulation for *both* the advective as well as the diffusive components of the Navier–Stokes equations. This allows multi-domain representation with a discontinuous (i.e., globally L^2) trial basis. This discontinuous basis is orthogonal, hierarchical, and maintains a tensor-product property even for non-separable domains [5, 6]. Moreover, in the proposed method the *conservativity* property is maintained automatically in the element-wise sense by the discontinuous Galerkin formulation, while *monotonicity* is controlled by varying the order of the spectral expansion and by performing h-refinement around discontinuities.

The work presented here was motivated by the work of Cockburn and Shu on discontinuous finite elements for hyperbolic problems presented in a series of papers (see [7–12]). An implementation of these ideas for hyperbolic systems using quadrilateral Legendre spectral elements was developed in [13]. Theoretical work on discontinuous Galerkin methods for diffusion is more recent [7] and has provided a justification of the application of the discontinuous Galerkin method to compressible Navier–Stokes equations done earlier in [14, 15]. A different approach has been independently developed by Oden and collaborators for the *hp* version of finite elements (see [16–20]). Discontinuous Galerkin methods use concepts both from finite volume and finite element methodology. In the current paper, we also add *high-order accuracy* using spectral/*hp* expansions on standard unstructured and structured grids.

We have followed the arbitrary Lagrangian Eulerian (ALE) framework as in previous works, e.g., [21–27], but with an important difference on computing the grid velocity. Specifically, we developed a modified version of the force-directed method [28] to compute the grid velocity via incomplete iteration. We then update the location of the vertices of the elements using the known grid velocity. In addition to the ALE treatment, the proposed method is new both in the formulation (e.g., construction of inviscid and viscous fluxes, use of characteristic variables, no need for limiters) as well as in the discretization as it uses *polymorphic* subdomains. We will demonstrate this flexibility in the context of simulating viscous flows that require accurate boundary layer resolution.

The paper is organized as follows: We first present the ALE discontinuous Galerkin formulation for advection and diffusion scalar equations separately for clarity, and subsequently we discuss the algorithm for the grid velocity as well as other implementation issues. We then present convergence results (in time and in space), a 3D simulation using hybrid discretization, and two simulations in moving computational domains. In the Appendixes, we briefly review the spectral basis we use in conjunction with the polymorphic subdomains in two and three dimensions and provide details of the numerical quadrature used.

2. NUMERICAL FORMULATION

We consider the non-dimensional compressible Navier–Stokes equations, which we write in compact form in an Eulerian reference frame as

$$\vec{U}_t + \nabla \cdot \mathbf{F} = Re_\infty^{-1} \nabla \cdot \mathbf{F}^v \qquad \text{in } \Omega, \tag{1}$$

where $\mathbf{F}$ and $\mathbf{F}^v$ correspond to inviscid and viscous flux contributions, respectively, and Re_∞ is the reference Reynolds number. Here the vector $\vec{U} = [\rho, \rho u_1, \rho u_2, \rho u_3, \rho e]^t$ with $\mathbf{u} = (u_1, u_2, u_3)$ the local fluid velocity, ρ the fluid density, and e the total energy. Splitting the Navier–Stokes operator in this form allows for a separate treatment of the inviscid and viscous contributions, which, in general, exhibit different mathematical properties.

In the following, we will solve the Navier–Stokes equations in a time-dependent domain $\Omega(t)$ by discretizing on a grid whose points may be moving with velocity $\mathbf{U}^g$, which is, in general, *different* than the local fluid velocity. This is the so-called arbitrary Lagrangian Eulerian (or ALE) formulation which reduces to the familiar Eulerian and Lagrangian forms by setting $\mathbf{U}^g = 0$ and $\mathbf{U}^g = \mathbf{u}$, respectively [21–26]. In this context, we will review the discontinuous Galerkin formulation employed in the proposed method.

As regards the spatial representation, we will use a discretization similar to finite elements but with an orthogonal spectral basis consisting of Jacobi polynomials (see Appendix and also [29]). A rigorous analysis of the discrete advection operator was presented in [4]. In the proposed formulation, no flux limiters are used but instead we reduce the order of polynomial around discontinuities and perform h-refinement with zero-order elements. Also, Cockburn and Shu [7] did not use any limiters for the convection-diffusion system in their version of the discontinuous Galerkin method with lower order elements. This is justified in part by theoretical work on scalar nonlinear equations in [30] that shows stability in the L_2-norm and convergence of the method to the entropy solution, assuming a strictly convex or concave nonlinearity. This result holds for any value of degree of approximating polynomials. Similar results were proved by Johnson and co-workers in earlier work [31, 32] but for nonlinear conservation laws containing an additional term that is responsible for shock-capturing. At this stage, no stability theory exists for a system of nonlinear conservation laws.

In the following, we discuss separately the formulation for the advection and diffusion terms, and subsequently we present a heuristic algorithm to update the grid velocity, $\mathbf{U}^g$.

2.1. *Discontinuous Galerkin for Advection*

Using the Reynolds transport theorem we can write the Euler equations in the ALE framework following the formulation proposed in [26] as

$$\vec{U}_t + G_{i,i} = -U^g_{i,i}\vec{U}, \tag{2}$$

where the ALE flux term is defined as

$$G_i = \left(u_i - U^g_i\right)\vec{U} + p[0, \delta_{1i}, \delta_{2i}, \delta_{3i}, u_i], \qquad i = 1, 2, 3.$$

We can recover the *Euler flux* $\mathbf{F}$ (see Eq. (1)) by simply setting $\mathbf{U}^g = 0$, and in general we have that $G_i = F_i - U^g_i\vec{U}$. Now if we write the ALE Euler equations in terms of the *Euler flux* then the source term on the right-hand-side of Eq. (2) is eliminated and we obtain

$$\vec{U}_t + F_{i,i} - U^g_i\vec{U}_{,i} = 0, \tag{3}$$

which can then be recast in the standard quasi-linear form

$$\vec{U}_t + \left[A_i - U^g_i I\right]\vec{U}_{,i} = 0,$$

where $A_i = \partial F_i / \partial U \, (i = 1, 2, 3)$ is the flux Jacobian and I is the unit matrix. In this form it is straightforward to obtain the corresponding characteristic variables since the ALE Jacobian matrix can be written

$$A_i^{ALE} \equiv \left[A_i - U_i^g I \right] = R_i \cdot \left[\Lambda_i - U_i^g I \right] \cdot L_i,$$

where brackets denote matrices. Here the diagonal matrix Λ contains the eigenvalues of the original Euler Jacobian matrix A, and R and L are the right- and left-eigenvector matrices, respectively, containing the corresponding eigenvectors of A. Notice that the shifted eigenvalues of the ALE Jacobian matrix do not change the corresponding eigenvectors in this characteristic decomposition.

To explain the discontinuous Galerkin ALE formulation we consider the two-dimensional equation for advection of a conserved scalar q in a region $\Omega(t)$

$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{F}(q) - \mathbf{U^g} \cdot \nabla q = 0.$$

In the discontinuous Galerkin framework, we test the equation above with discontinuous test functions v separately on each element (e) (see also [4, 17]) to obtain

$$(v, \partial_t q)_e + (v, \nabla \cdot \mathbf{F}(q))_e - (v, \mathbf{U^g} \cdot \nabla q)_e + \int_{\partial T_e} v \left[\tilde{f}(q_i, q_e) - \mathbf{F}(q) - (q_{up} - q_i) \cdot \mathbf{U^g} \right] \cdot \hat{n} \, ds = 0. \tag{4}$$

Here $(\cdot, \cdot)$ denotes inner product evaluated over each element, and $\tilde{f}$ is a numerical boundary flux [4]; the notation is explained in Fig. 1. Notice that this form is different than the form used in the work of [25, 33] where the time derivative is applied to the inner product,

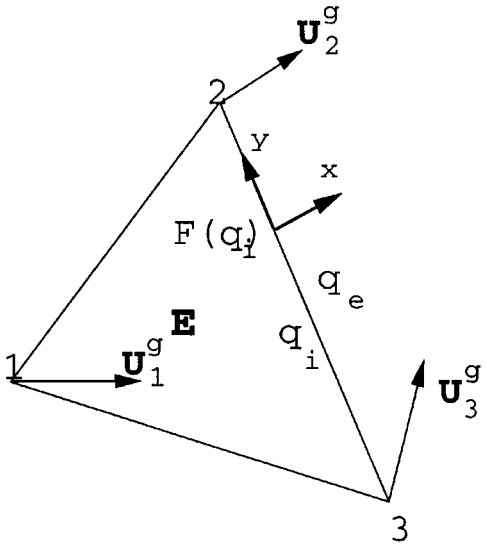

FIG. 1. Notation for a triangular element. The subscript (i) denotes interior quantities and (e) exterior quantities. Also, $\mathbf{U}^g$ is the grid velocity.

i.e.,

$$\partial_t(v,q)_e + (v, \nabla\cdot\mathbf{F}(q))_e - (v, \mathbf{U^g}\cdot\nabla q)_e - (v, q\nabla\cdot\mathbf{U^g})_e + \int_{\partial T_e} v\left[\tilde{f}(q_i,q_e) - \mathbf{F}(q) - (q_{up}-q_i)\cdot\mathbf{U^g}\right]\cdot\hat{n}\,ds = 0. \tag{5}$$

Note that from the Reynolds transport theorem we have that

$$\partial_t \int_{\Omega(t)} q\,d\Omega = \int_{\Omega(t)} \left(q_t + q\nabla\cdot\mathbf{U}^g\right) d\Omega,$$

where the partial time derivative on the right-hand side is with respect to the moving ALE grid. The difference between the forms in Eqs. (4) and (5) is that the different treatment of the time derivative introduces a term in the second case (Eq. (5)) that involves the divergence of the grid velocity. While the two forms are equivalent in the continuous case, they are not necessarily equivalent in the discrete case as we will discuss further in Section 3 in the context of a geometric conservation law [34].

To compute the boundary terms, we follow an upwind treatment based on *characteristics* similar to the work in [4], including here the term representing the grid motion. To this end, we need to linearize the ALE Jacobian *normal to the surface*, i.e., $[A - U_n^g I] = R[\Lambda - U_n^g I]L$, where U_n^g is the velocity of the grid in the $\mathbf{n}$ direction. The term $(q_{up} - q_i)$ expresses a jump in the variable at inflow edges of the element resulting from an upwind treatment. In the case of a system of conservation laws the numerical flux $\tilde{f}$ is computed from an approximate Riemann solver [4].

In this formulation, the space of test functions may contain formally discontinuous functions, and thus the corresponding discrete space contains polynomials within each "element" but zero outside the element. Here the "element" is, for example, an individual triangular region T_i in the computational mesh applied to the problem. Thus, the computational domain $\Omega = \cup_i T_i$, and T_i, T_j overlap only on edges.

2.2. *Discontinuous Galerkin for Diffusion*

The viscous contributions do not depend on the grid velocity $\mathbf{U}^g$, and therefore we can apply the following discontinuous Galerkin formulation. Let us consider as a model problem the parabolic equation with variable coefficient $\nu(\mathbf{x})$ to demonstrate the treatment of the viscous contributions:

$$u_t = \nabla\cdot(\nu\nabla u) + f, \qquad \text{in } \Omega,\ u \in L^2(\Omega)$$
$$u = g(\mathbf{x},t), \qquad \text{on } \partial\Omega.$$

We then introduce the flux variable

$$\mathbf{q} = -\nu\nabla u \tag{6}$$

with $\mathbf{q}(\mathbf{x},t) \in \mathbf{L}^2(\Omega)$, and re-write the parabolic equation

$$u_t = -\nabla\cdot\mathbf{q} + f, \qquad \text{in } \Omega$$
$$1/\nu\mathbf{q} = -\nabla u, \qquad \text{in } \Omega$$
$$u = g(\mathbf{x},t), \qquad \text{on } \partial\Omega.$$

The weak formulation of the problem is then as follows: *Find* $(\mathbf{q}, u) \in \mathbf{L^2}(\mathbf{\Omega}) \times L^2(\Omega)$ *such that*

$$(u_t, w)_e = (\mathbf{q}, \nabla w)_e - \langle w, \mathbf{q_b} \cdot \mathbf{n} \rangle_e + (f, w)_e, \qquad \forall w \in L^2(\Omega)$$
$$1/\nu(\mathbf{q}, \mathbf{v})_e = (u, \nabla \cdot \mathbf{v})_e - \langle u_b, \mathbf{v} \cdot \mathbf{n} \rangle_e, \qquad \forall \mathbf{v} \in \mathbf{L^2}(\mathbf{\Omega})$$
$$u = g(\mathbf{x}, t), \qquad \text{on } \partial\Omega,$$

where the parentheses denote standard inner product in an element (e), as before and the angle brackets denote boundary terms on each element, with $\mathbf{n}$ denoting the outward facing unit normal. The boundary terms contain weighted boundary values of u_b, q_b which can be chosen as the arithmetic mean of values from the two sides of the boundary, i.e., $u_b = 0.5(u_i + u_e)$, and $q_b = 0.5(q_i + q_e)$ [35, 14] where the subscript (i) denotes contributions evaluated at the interior side of the element boundary, and (e) on the exterior side of the element boundary (see Fig. 1).

Upon integrating by parts once more, we obtain an equivalent formulation which is easier to implement, and hence is used in the computer code. The new variational problem is

$$(u_t, w)_e = (-\nabla \cdot \mathbf{q}, w)_e - \langle w, (\mathbf{q_b} - \mathbf{q_i}) \cdot \mathbf{n} \rangle_e + (f, w)_e, \qquad \forall w \in L^2(\Omega)$$
$$1/\nu(\mathbf{q}, \mathbf{v})_e = (-\nabla u, \mathbf{v})_e - \langle u_b - u_i, \mathbf{v} \cdot \mathbf{n} \rangle_E, \qquad \forall \mathbf{v} \in \mathbf{L^2}(\mathbf{\Omega})$$
$$u = g(\mathbf{x}, t), \qquad \text{in } \partial\Omega.$$

This method is element-wise conservative, a property which is particularly difficult to preserve in high-order finite elements. A similar conservative discontinuous Galerkin method for diffusion problems but using a *single* variational statement, i.e., without the introduction of the auxiliary flux variable (Eq. (6)), has been developed by Oden *et al.* [18] (see also [19, 20]). We refer the interested reader to these works and to [18] in particular for a theoretical treatment of the diffusion problem, including a more rigorous definition of discrete spaces (the so-called broken Sobolev spaces) as well as derivation of *a priori* error estimates.

2.3. *Grid Velocity Algorithm*

The grid velocity is arbitrary in the ALE formulation and therefore great latitude exists in the choice of technique for updating it. Mesh constraints such as smoothness, consistency, and lack of edge crossover, combined with computational constraints such as memory use and efficiency dictate the update algorithm used. Two broad classifications of algorithms exist for updating the mesh: Velocity smoothing methods and coordinate smoothing methods.

Typically, in *velocity smoothing* methods the grid velocity $\mathbf{U^g}$ is updated by solving

$$\nabla \cdot (k(\mathbf{x}) \nabla \mathbf{U^g}) = 0,$$

with Dirichlet conditions for $\mathbf{U^g}$ on both the moving wall boundary and on the outer boundary of the computational domain. The choice $k = 1$ leads to the classic elliptic velocity smoothing which produces the most uniform deformation of the elements. Since in most computational fluid applications Poisson solvers are necessary, the choice of a Laplacian velocity smoother is natural due to its straightforward implementation. Though this method produces the most uniform deformation of elements, even small body motions can lead to

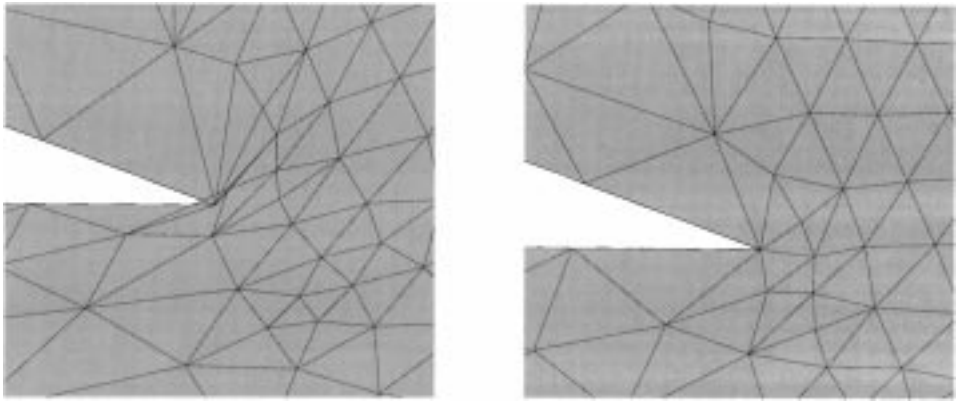

FIG. 2. The Laplacian velocity smoothing (left) leads to cross edging unlike the modified barycenter method (right) introduced here. The turn of the airfoil is about 8 degrees, while the original grid configuration is identical in both cases. (Only the trailing edge of the airfoil is shown.)

edge crossover as demonstrated in Fig. 2. Specifically, we consider the pitching airfoil (see example in Subsection 4.2) and starting from the same original unstructured grid we see after 8 degrees of rotation how the grid is deformed. We see in Fig. 2 (left) that using the standard Laplacian smoothing leads to edge crossing which will render the computation unstable.

Modifications to this approach were presented in [27] where a variable diffusivity ($k(\mathbf{x})$ being a function of position within the mesh) was introduced to help avoid edge crossing. Contrastingly, other researchers have attempted to calculate the mesh deformation using coordinate smoothing methods [36–38]. Mesh positions are obtained using methods based on a graph theory analogy to the spring problem. Verticies are treated as *nodes*, while edges are treated as *springs* of varying length and tension. At each time step, the mesh coordinate positions are updated by equilibration of the spring network. Once the new vertex positions are calculated, the mesh velocity is obtained through differences between the original and equilibrated mesh vertex positions.

In the current work, for updating the grid velocity we combined the two concepts mentioned above by formulating the problem of solving for the mesh velocity in terms of its graph theory equivalent problem. Specifically, we incorporate the idea of variable diffusivity as in [27] while maintaining the computational efficiency of the methods used in [36–38]. The combination of these two methodologies provides a computationally efficient way of minimizing edge crossover in situations where Laplacian smoothing fails (as demonstrated in Fig. 2, right).

The method we use for updating the mesh velocity is a variation of the barycenter method [28] and relies on graph theory. Given the graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ of element vertices V and connecting edges E, we define a partition $V = V_0 \cup V_1 \cup V_2$ of V such that V_0 contains all vertices affixed to the moving boundary, V_1 contains all vertices on the outer boundary of the computational domain, and V_2 contains all remaining interior vertices. To create the effect of variable diffusivity, we use the *concept of layers*. As is pointed in [27], it is desirable for the vertices very close to the moving boundary to have a grid velocity almost equivalent to that of the boundary. Hence, locally the mesh appears to move with solid movement, whereas far away from the moving boundary the velocity must gradually go to zero. To accomplish this in our formulation, we use the concept of local tension within

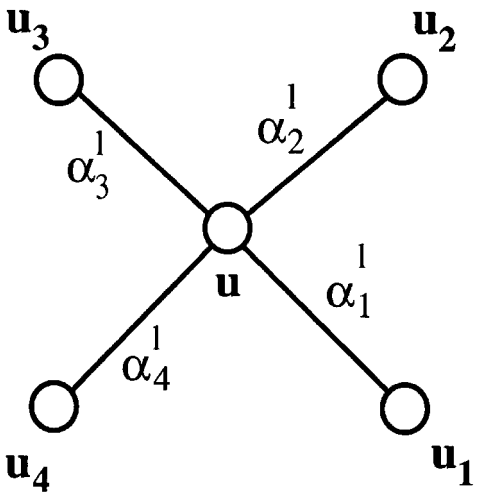

FIG. 3. Graph showing vertices with associated velocities and edges with associated weights.

layers to allow us to prescribe the rigidity of our system. Each vertex is assigned to a layer value which heuristically denotes its distance from the moving boundary. Weights are chosen such that vertices closer to the moving boundary have a higher influence on the updated velocity value. To find the updated grid velocity u^g, at a vertex $v \in V_2$, we use a force-directed method. Given a configuration as in Fig. 3, the grid velocity at the center vertex is given by

$$u^g = \sum_{i=1}^{\deg(v)} \alpha_i^l u_i, \qquad \sum_{i=1}^{\deg(v)} \alpha_i^l = 1,$$

where $\deg(v)$ is the number of edges meeting at the vertex v and α_i^l is the lth layer weight associated with the ith edge. This is subjected to the following constraints: $u^g = 0\ (\forall v \in V_1)$ and $u^g(\forall v \in V_0)$ is prescribed to be the wall velocity. This procedure is repeated for a few cycles following an *incomplete* iteration algorithm, over all $v \in V_2$. (Here by incomplete we mean that only a few sweeps are performed and not full convergence is sought.) Once the grid velocity is known at every vertex, the updated vertex positions are determined using explicit time-integration of the newly found grid velocities.

To demonstrate the flexibility provided by having *variable stiffness* in the proposed grid velocity algorithm, we plot in Fig. 4 the grid after the airfoil shown in Fig. 2 is rotated by about 8 degrees. We have changed the discretization to a hybrid one by employing quadrilateral elements around the airfoil and triangular elements in the outer layers (see flow results in Subsection 4.2). On the left of Fig. 4 we plot the grid obtained with the uniform stiffness for all elements. On the right, we plot again the grid but with the inner layers biased to have stiffness 20 times higher than the outer layers. We see that the distortion of the quadrilaterals is clearly smaller in the latter case.

3. IMPLEMENTATION ISSUES

We present here some details on the implementation of the method in the context of spectral/*hp* element methods (see also Appendixes I and II). First, we summarize the overall algorithm:

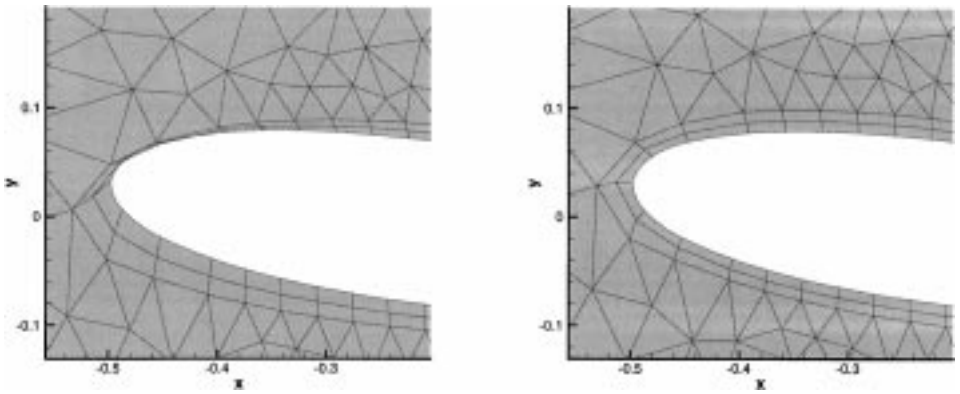

FIG. 4. Hybrid discretization. The "stiffer" grid on the right is less distorted around the airfoil than the one on the left. The turn of the airfoil is about 8 degrees, while the original grid configuration is identical in both cases.

DISCONTINUOUS GALERKIN ALGORITHM.

- Step 0. Read the initial conditions for the state vector $\vec{U}^0$ and project all fields to polynomial space.
- Step n. Begin time loop.
  - Compute the Navier–Stokes operator $\mathbf{U}_f^n$ (see below).
  - Advance in time: $\vec{U}^{n+1} = \vec{U}^n + \Delta t \mathbf{U}_f^n$. Time $t^{n+1} = (n+1)\Delta t$.
  - Overwrite the wall boundary values: $\vec{U}_w^{n+1} = \vec{U}_w$.
  - If $(n+1)\Delta t$ less than specified time continue Else exit.
- Print output and store the results in polynomial space.

Next we present the algorithm we employ to compute the Navier–Stokes operator $\mathbf{U}_f^n$ and impose the boundary conditions using characteristic treatment.

NAVIER–STOKES OPERATOR.

- Compute the convection contribution: $\mathbf{U}_{f,conv}$.
  - Compute characteristic boundary values: $\mathbf{U}_{f,conv}^{bn}$.
  - Compute divergence of the interior Euler flux: $\mathbf{U}_{f,conv}^{int}$.
  - $\mathbf{U}_{f,conv} = \mathbf{U}_{f,conv}^{bn} \mathbf{U}_{f,conv}^{int}$.
- Save the values at the outer boundaries: $\mathbf{U}_{f,conv}^{bn}$.
- Compute the viscous contributions: $\mathbf{U}_{f,visc}$.
- $\mathbf{U}_f = \mathbf{U}_{f,conv} + \mathbf{U}_{f,visc}$.
- Overwrite the outer boundary values: $\mathbf{U}_f^{bn} = \mathbf{U}_{f,conv}^{bn}$.

All operations in the above procedure are performed on the quadrature points (in "physical" space). However, as the initial conditions are in polynomial space, these operations do not take the function $\mathbf{U}$ out of the polynomial space. The only operation that can do that is the overwriting of the boundary values (this is done by setting the functional values at the quadrature points on the boundary). In this case, we should perform the projection of the field back to polynomial space. This is done locally, only in the elements which are next to the boundary. This point is explained further in Subsection 3.2.

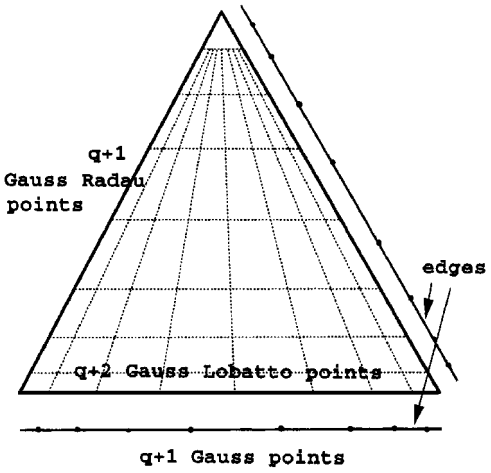

FIG. 5. Quadrature points used in a triangular element with $q = 7$. This quadrature is exact for standard inner products in the space of polynomials up to $p = 7$. The top corner is the singular corner as described in Appendix I.

3.1. *Quadrature Rules*

Of particular importance for accuracy and computational efficiency is the evaluation of the integrals involved in the discontinuous Galerkin formulation, i.e., the numerical quadrature used. Because of the special basis we use here, we employ a *mixed numerical quadrature* based on Gauss, Gauss–Lobatto, and Gauss–Radau integration. To illustrate the main points in two-dimensions, we consider a triangular element as shown in Fig. 5. The top vertex is the singular corner as defined in Appendix I and no quadrature point is assigned to it.

Consider a space of polynomials of degree up to p used in an element (e). Then we *define* the quadrature order $q = p$ (with p denoting also the spectral order). With this definition, we mean that we have $(q + 2)$ Gauss–Lobatto points in the direction across the singular corner and $(q + 1)$ Gauss–Radau points in the other direction (see Fig. 5). In this case, the quadrature rule is exact for polynomials of degree $2q$ in the interior of the elements (in non-curvilinear geometries).

All the boundary terms, i.e., boundary integrals and boundary fluxes, are computed by the interpolation of the interior values to $(q + 1)$ Gauss points on each edge. This is how we match the points of boundary flux computations between the adjacent elements. If the orders in the elements are different, then the *maximum* number of edge quadrature points should be taken for stability. The edge fluxes need to be projected to the smaller polynomial space (between the two adjacent elements) in order to preserve conservativity. We also note that on the edges the quadrature is exact for polynomials of degree $(2q + 1)$. These are conditions necessary to guarantee the maximum possible accuracy of $(p + 1)$ as proven in [11] for the linear case.

3.2. *Boundary Conditions*

Let us consider in more detail how we handle the boundary conditions for the Navier–Stokes operator. First, we compute the convection contribution $\mathbf{U}_{f,conv}$. On the outer boundaries the interior values (U_i) of the conservative variables $(\rho, \rho u_1, \rho u_2, \rho_3, \rho e)$ are taken from the previous time step (U^n). The specified reference "∞" values are taken as the

exterior values (U_e). The characteristic boundary contribution $U^{bn}_{f,conv}$ is computed by an approximate Riemann solver and is stored. At the walls, the interior values are taken as before, and the exterior values are found as follows: $\rho_e = \rho_i$, $\vec{v}_e = 0$, $T_e = T_w$ for the case of no-slip, isothermal wall.

We use the stored boundary values for the outer boundaries and wall boundary conditions as Dirichlet boundary conditions for the viscous step. In this algorithm, they are implemented in the following way: when the viscous contribution $\mathbf{U}_{f,visc}$ is computed, we add up the two contributions $\mathbf{U}_{f,conv}$ and $\mathbf{U}_{f,visc}$, and we assign the $\mathbf{U}^{bn}_{f,conv}$ for the boundary values. Then, after we advance in time, we overwrite the wall boundary values: $\vec{U}^{n+1}_w = \vec{U}_w$. Here, overwriting the boundary conditions implies changing the values at the quadrature points on the boundary; subsequently we project the function in this element back to polynomial space.

3.3. *Computation of Derivatives*

All the derivatives are computed in a weak discontinuous Galerkin sense as follows: Let $u \in L^2(\Omega)$ then $q \in L^2(\Omega)$ is x-derivative of u, $q = u_x$, if

$$(q, w)_e = (u_x, w)_e + \langle w, (u_b - u_i)n_x \rangle_e \qquad \forall w \in L^2(\Omega),$$

where $u_b = \frac{1}{2}(u_i + u_e)$. The derivatives are computed in element-wise sense and $(\cdot, \cdot)_e$ denotes standard inner product in an element (e). The angle brackets denote the inner product over elemental boundary, with n_x being the x component of the outwards unit normal. All the operations are performed in "physical" space: u_x is computed on the quadrature points, and all integrals are computed using the aforementioned quadrature rules. In our computations we use the same spaces for the functions and their derivatives. These spaces consist of functions which are polynomials of degree up to p inside the elements, and the quadrature rules used are exact in this case.

In order to demonstrate how we preserve the *diagonal mass* matrix even for non-curvilinear geometries, we consider a two-dimensional scalar equation for advection of a conserved quantity u in a curved element which we denote by e of area T and boundary ∂T,

$$\frac{\partial u}{\partial t} + \nabla \cdot \mathbf{F}(u) = 0. \tag{7}$$

The element jacobian $J(x, y)$ is variable inside the element, and we used it to multiply Eq. (7) by a *modified* test function $\frac{v}{J(x,y)}$. Then integrating by parts, we obtain

$$\int_T \frac{\partial u}{\partial t} \frac{v}{J(x,y)}\, dx + \int_{\partial T} \frac{v}{J(x,y)} \mathbf{n} \cdot \tilde{\mathbf{f}}(u)\, ds - \int_T \nabla \frac{v}{J(x,y)} \cdot \mathbf{F}(u)\, dx = 0. \tag{8}$$

We integrate by parts once more, this time taking the flux of the interior values $\mathbf{F}(u_i)$ in the boundary integral, and arrive at the formulation

$$\int_T \frac{\partial u}{\partial t} \frac{v}{J(x,y)}\, dx + \int_{\partial T} \frac{v}{J(x,y)} (\tilde{\mathbf{f}}(u_i, u_e) - \mathbf{F}(u_i)) \cdot \mathbf{n}\, ds + \int_T \frac{v}{J(x,y)} \nabla \mathbf{F}(u)\, dx, \tag{9}$$

where $\tilde{\mathbf{f}}$ is the numerical flux as before. If we define the inner product in this element e as

$(u, v)_e = \int_T \frac{uv}{J(x,y)}\, dx$, we can rewrite Eq. (9) in the form

$$\frac{\partial}{\partial t}(u, v)_e + \int_{\partial T} \frac{v}{J(x, y)}(\tilde{\mathbf{f}}(u_i, u_e) - \mathbf{F}(u_i)) \cdot \mathbf{n}\, ds + (\nabla \cdot \mathbf{F}(u), v)_e = 0. \qquad (10)$$

With this rearrangement and given that we employ an explicit time-stepping algorithm, we preserve the diagonality of the mass matrix even for curved elements, although "quadrature crimes" are introduced. The main source of these errors is due to the *modified* test function, which is not in polynomial space anymore, and also due to the collocation-type evaluation of the boundary integral in Eq. (10). However, extensive numerical tests with curvilinear 2D and 3D geometries have shown that this quadrature error is bounded and that indeed it does not affect exponential convergence [39].

3.4. *Geometric Conservation Law*

The procedure in Section 3 describes a first-order time accurate algorithm as first-order explicit integration is employed. However, as we will show in the next section second- and third-order schemes can easily be constructed using explicit multi-step time-integration. It is interesting to note that the *discrete geometric conservation law* (DGCL) [33, 34] is automatically satisfied in the ALE formulation implemented here (see Eq. (4)) since the term associated with the *grid-velocity divergence* is eliminated, unlike the form of Eq. (5). Therefore, the geometric conservation law does not pose any constraints, and thus the temporal accuracy of the scheme is determined solely by the time-integration rule employed. If Eq. (5) is employed instead to obtain the discrete system, then new time-stepping algorithms need to be constructed as in the work of [33] that honor the DCGL constraint in order to guarantee high-order time accuracy.

3.5. *Stability Issues and Over-integration*

As regards *spatial discretization* we employ a finite element mesh with an orthogonal (for non-curvilinear edges) spectral basis (see Appendix I) that leads to a diagonal mass matrix and thus no matrix inversions are necessary. It has been reported, however (F. Bassi and S. Rebay, personal communications) that isoparametric representation of geometry may lead to a weak instability. Although we could not exactly confirm that observation in our numerical experiments even with very low-order discretizations similar to the ones used in [14], we have found that *over-integration* in computing inner products in the weak formulation is important in obtaining asymptotically stable results, i.e., after long-time integration.

In particular, we use Gauss–Jacobi quadrature, which is exact for standard inner products in non-curvilinear geometries if the quadrature order q is equal to the order of the spectral basis p (see Subsection 3.1, Appendix II, and also [29]). In Fig. 6 we show simulations of compressible viscous flow at Reynolds number (based on the chord-length) $Re = 1{,}000$ past a NACA0012 airfoil for the case of $p = q = 3$ on the left, and $p = 3$ and $q = 4$ on the left. We see that the results in both cases are visually indistinguishable although there is some small quantitative difference. This is documented in Fig. 7 which shows the corresponding histories of the modal advection contributions from the boundary and the interior of the element computed at a point close to the leading edge of the airfoil. If we now increase the Reynolds number to $Re = 10{,}000$ and compare the simulations in the two cases we see significant

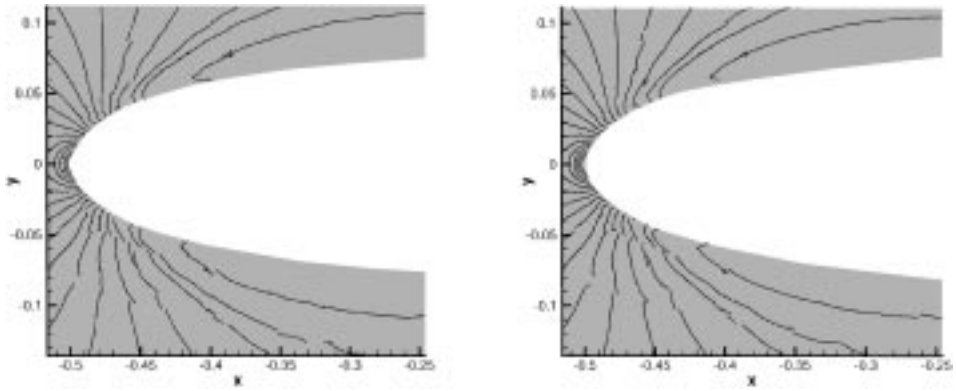

FIG. 6. Density contours for $Re = 1{,}000$ and $M = 0.2$. The simulation on the left was performed with $p = q = 3$ and on the right with $p = 3$; $q = 4$. (Note that p is the order of trial basis per element and q is the quadrature order.)

differences. This is shown in Fig. 8 where we plot the case $p = q = 3$ on the left and the case ($p = 3$; $q = 4$) on the right. We see that the latter is stable, but the former develops very steep gradient very close to the leading edge that renders the computation eventually unstable. This is documented more clearly in Fig. 9 where we plot the corresponding histories of the same quantities as before for the two computations.

If we simply increase both the interpolation order and the quadrature order so that $p = q = 4$ the method still diverges, which reinforces further the aforementioned finding on over-integration. It is also of interest to determine if the source of instability comes from the treatment of the advection terms or the diffusion terms. To this end, we performed two Euler simulations for the same problem: The first one for $p = q = 3$ and the second one with $p = 3$ and $q = 4$. As initial conditions we used a Navier–Stokes solution ($Re = 10{,}000$) in both cases, as it had more complex initial structure than a uniform state and thus instabilities were triggered earlier. We obtained an unstable computation in the former case but a stable one in the latter case as shown in Fig. 10 that plots the histories of the same corresponding modal boundary and interior contributions, as before.

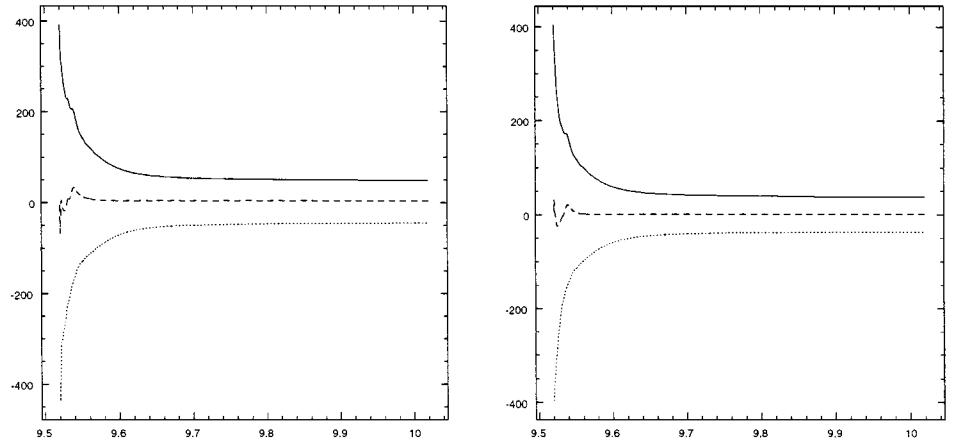

FIG. 7. Time history of modal advection boundary (dot) and interior (solid) contributions of an element close to leading edge for $Re = 1{,}000$ and $M = 0.2$. The simulation on the left was performed with $p = q = 3$ and on the right with $p = 3$; $q = 4$.

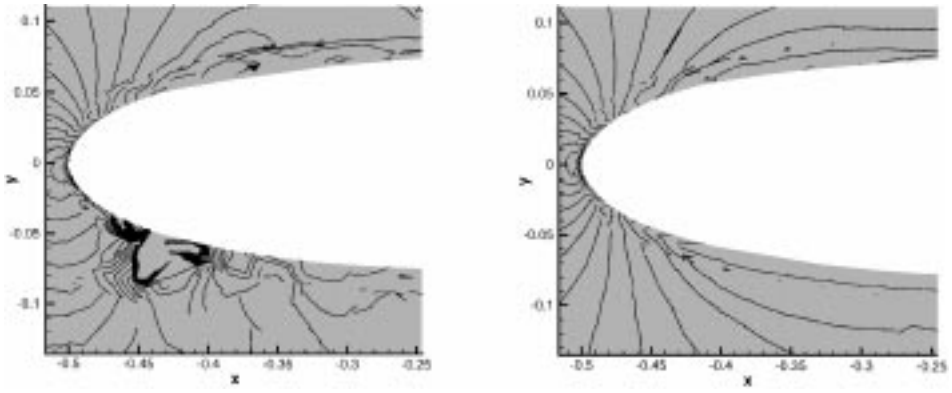

FIG. 8. Density contours for $Re = 10{,}000$ and $M = 0.2$. The simulation on the left was performed with $p = q = 3$ and on the right with $p = 3$; $q = 4$.

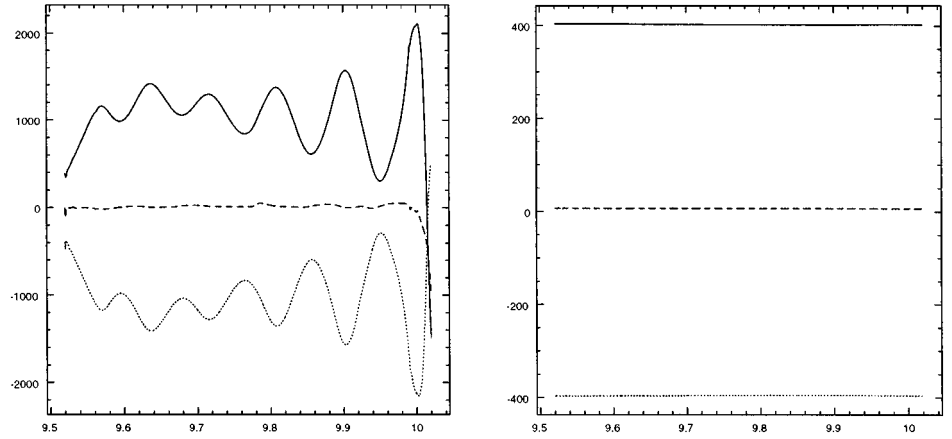

FIG. 9. Time history of modal advection boundary (dot) and interior (solid) contributions of an element close to leading edge for $Re = 10{,}000$ and $M = 0.2$. The simulation on the left was performed with $p = q = 3$ and on the right with $p = 3$; $q = 4$.

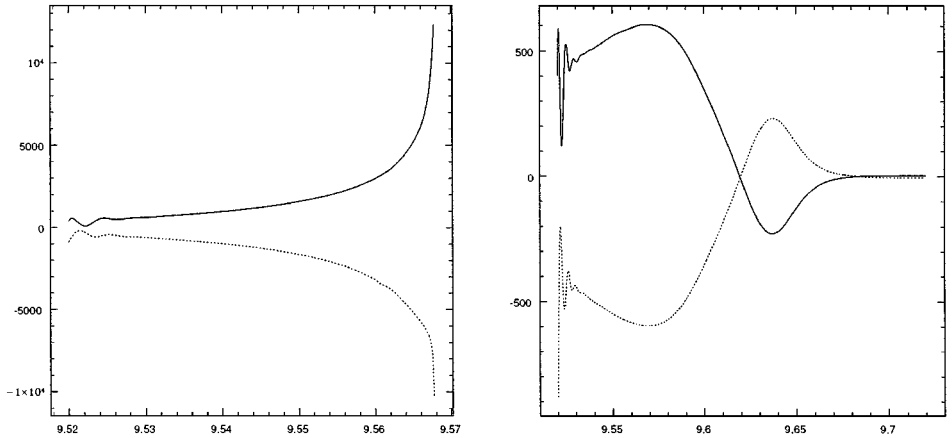

FIG. 10. Time history of advection boundary (dot) and interior (solid) contributions of an element close to leading edge for Euler simulations at $M = 0.2$. The simulation on the left was performed with $p = q = 3$ and on the right with $p = 3$; $q = 4$.

4. CONVERGENCE AND SIMULATION RESULTS

In this section we demonstrate how the proposed discontinuous Galerkin ALE method works in conjunction with the spectral basis that we employ for discretization. First, we examine how distorted three-dimensional grids affect the spatial convergence of the method, we examine its temporal accuracy and also demonstrate the flexibility of hybrid discretization. Subsequently, we focus on the flow around a rapidly pitching airfoil for validation against other established methods and on a three-dimensional computation of a flapping wing that simulates insect flight.

4.1. *Convergence and Hybrid Discretization*

4.1.1. Convergence for skewed elements. In previous work we have demonstrated the exponential convergence of the spectral discontinuous Galerkin method for the Euler and Navier–Stokes equations in two- and three-dimensional benchmark problems in stationary domains [4, 39]. Here we demonstrate that the spectral discretization we employ leads to a robust method that does not suffer from appreciable numerical errors due to grid deformation. This aspect is very important for the proposed method as high sensitivity of the solution on the grid would require frequent h-refinement taxing flow simulations in moving domains heavily.

To this end, we solve the parabolic equation $u_t = \nabla^2 u$ for an analytical solution of the form

$$u(x, y, z, t) = e^{\pi^2 t/12} \sin \frac{\pi x}{6} \sin \frac{\pi y}{6} \sin \frac{\pi z}{6}$$

with exact boundary conditions prescribed at all boundaries. The integration is for 1000 time steps with $\Delta t = 10^{-5}$ to eliminate any temporal errors. We consider four different meshes consisting of 12 tetrahedra as shown in Fig. 11. All tetrahedra share a common vertex at the

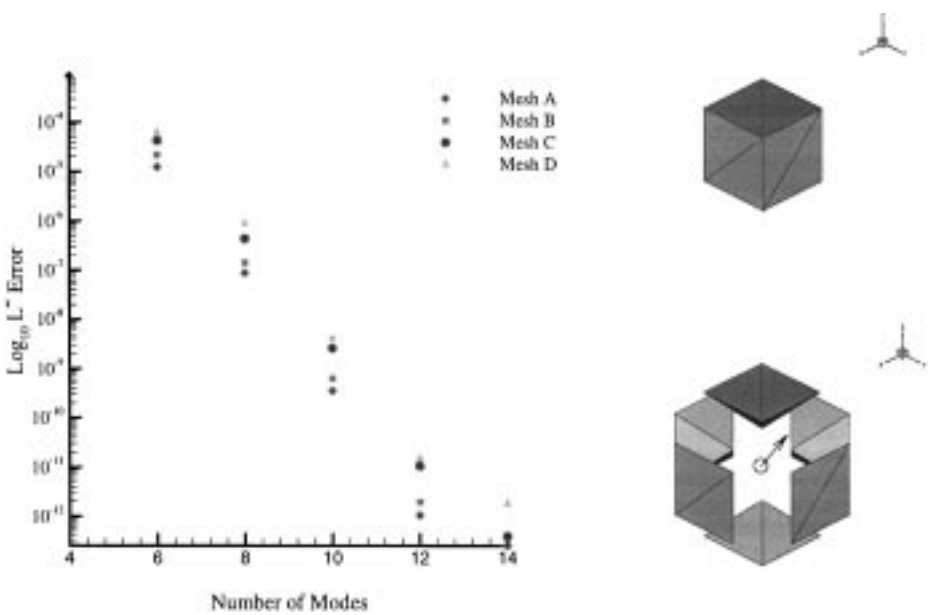

FIG. 11. L_∞ error of a 3D parabolic problem as a function of the number of modes. Domain D has elements with aspect ratio of 20. Exponential convergence is maintained even for very distorted elements.

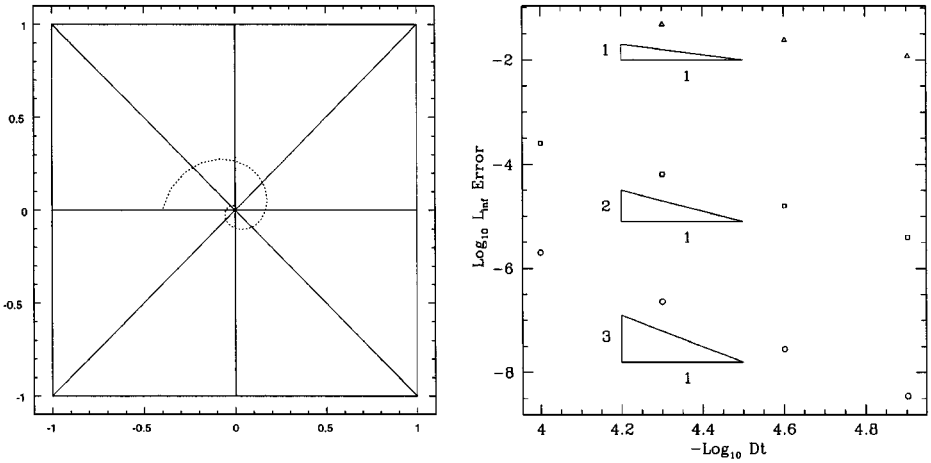

FIG. 12. Left, domain for the temporal accuracy tests. Right, L_∞ error as a function of the time step for an analytical solution of the unsteady Navier–Stokes equations obtained in the time-dependent domain shown on left.

center of the box, which we move as shown in Fig. 11 to cause distortion of the tetrahedra. In the domain D elements with aspect ratio of 20 are obtained. The error plot of Fig. 11 shows that there is a relatively small effect of distortion and that exponential convergence is maintained.

4.1.2. Temporal accuracy. In this example we use a square domain consisting of 8 triangles in order to test the time accuracy of the ALE code. Very high spectral order (p between 13 and 18) was employed to eliminate the spatial discretization errors. We used an analytical solution as described below. The grid is time-dependent; it is changing as we move the central vertex along the path shown in Fig. 12. Specifically, the coordinates of the central vertex are

$$X_v = X_0 - \cos(20\pi t)t^3 R; \qquad Y_v = Y_0 - \sin(20\pi t)t^3 R,$$

where $R = 50$, and the final time of integration is $t = 0.2$; here (X_0, Y_0) are the coordinates of the initial position of the central vertex, which are (0, 0) as shown in Fig. 12.

On the left and right sides of the domain periodic boundary conditions are assumed and on the top and bottom Dirichlet boundary conditions are prescribed. The analytical solution has the form

$$\begin{aligned} \rho &= A + B\sin(\omega x)\sin(10\pi t) \\ u &= C + D\cos(\omega x)\sin(\omega y)\cos(10\pi t) \\ T &= E + Fy\sin(10\pi t), \end{aligned}$$

where $\omega = 8\pi$, $A = 1$, $B = 0.1$, $C = 1$, $D = 0.04$, $E = 84$, and $F = 28$. The Navier–Stokes equations are then integrated using a forcing term consistent with the above solution. Numerical solutions were obtained for different sizes of time step, and the results are summarized in Fig. 12 (right) for first-, second-, and third-order Adams–Bashforth integration. Correspondingly, first-, second-, and third-order accuracy is achieved.

4.1.3. Hybrid discretization. To demonstrate the use of polymorphic domains, e.g., triangular prisms, hexahedra, etc., we consider flow past a 3D wing formed from a NACA0012

TABLE I
Simulation Parameters for Compressible Flow past a NACA0012 Wing with Endplates

Parameter	Value
Dimension	3d
Re	2000 based on chord length
Mach	0.5
Δt	le-4
P-range	1 to 4
Number of prisms	1960
Number of hexahedra	2095
Method	Discontinuous Galerkin

airfoil with plates attached to each end as a simple model of a wing between an engine and fusalage. We impose uniform upwind boundary conditions at inflow and outflow, and the domain is periodic from one end of the airfoil to the other. A thin layer of hexahedra was used on the surface of the wing, and a combination of both hexahedra and prisms was used in the remainder of the computational domain. The simulation was run with up to order $p = 4$ at $Re = 2000$ (based on chord length). A summary of the simulation parameters is given in Table I. The hybrid discretization and some representative results are shown in Fig. 13.

At low-order of interpolation, the simulation ran to steady state. This is due to a reduced effective Reynolds number achieved because of numerical dissipation. As we increased the p-order we observed unsteadiness developing in the wake of the wing, and what appears to be oblique shedding. This is only a marginally three-dimensional domain but it does demonstrate the flexibility of hybrid discretization to direct resolution into boundary layers and to fill out a domain with larger elements.

4.2. *Flow around a Pitching Airfoil*

We first validate the proposed method against established computational results [40] for a laminar flow around a rapidly pitching airfoil. In particular, we consider a NACA 0015 airfoil pitching upwards about a fixed axis at a constant rate from zero incidence to a maximum angle of attack of approximately 60 degrees. The pivot axis location is at $1/4$ of the chord measured from the leading edge. The temporal variation of the pitch given in [40] is

$$\Omega(t) = \Omega_0[1 - e^{-4.6t/t_0}], \qquad t \geq 0,$$

where t_0 denotes the time elapsed for the airfoil to reach 99% of its final pitch rate Ω_0. Here the non-dimensional values are $t_0^* = 1.0$ and $\Omega_0^* = 0.6$ based on the chord length and free stream velocity. As initial condition the computed field at 0 degrees angle of attack is used. The Mach number is $M = 0.2$ and the chord Reynolds number is $Re = 10{,}000$.

In paper [40] a similar simulation was obtained using a grid fixed to the airfoil by employing an appropriate transformation and discretizing the modified compressible Navier–Stokes equations using the implicit approximate factorization of Beam and Warming [41]. A typical grid used in [40] involved 203×101 points. Although accurate, this approach is not

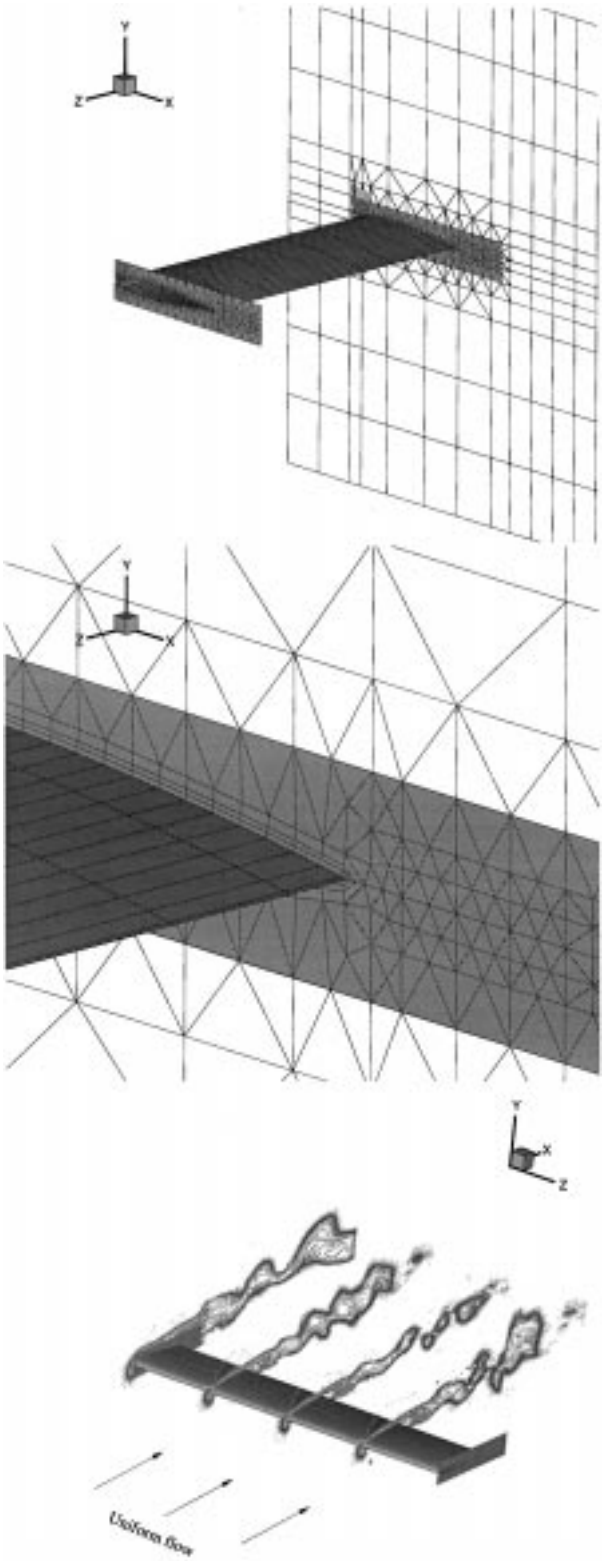

FIG. 13. Hybrid mesh for flow past a three-dimensional NACA0012 wing with endplates (top and middle). Iso-contours for x-component of momentum for $M = 0.5$ (bottom).

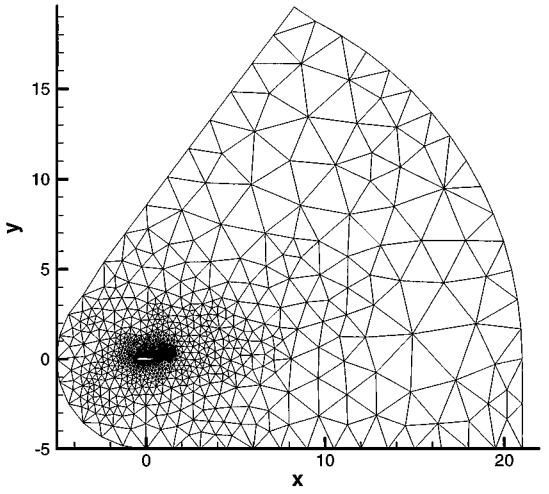

FIG. 14. Domain and triangulization for the simulation around the pitching airfoil NACA 0015.

general for moving domains and cannot be used, for example, in simulating multi-body dynamics.

In the present study, we employ the ALE formulation on the domain shown in Fig. 14. We performed two different sets of simulations, first with unstructured discretization around the airfoil (see Fig. 15; total of 3,838 triangular elements), and subsequently with hybrid discretization with quadrilateral elements around the airfoil for better resolution of boundary layers (total of 116 quadrilateral and 2167 triangular elements). We demonstrate how the hybrid discretization combined with *variable* p-order per element allows accurate resolution of boundary elements without the need for remeshing. We first performed simulations with constant p-order on all elements and subsequently with higher p-order in the inner layers of elements as shown in Fig. 16. We contrast the results in Fig. 17 for p-order $p = 3$ on the left, and p varying from 10 in the innermost layer to 2 in the far field. We see that the boundary layer is unresolved as indicated by the discontinuities at the element interfaces but it is accurately resolved in the second simulation.

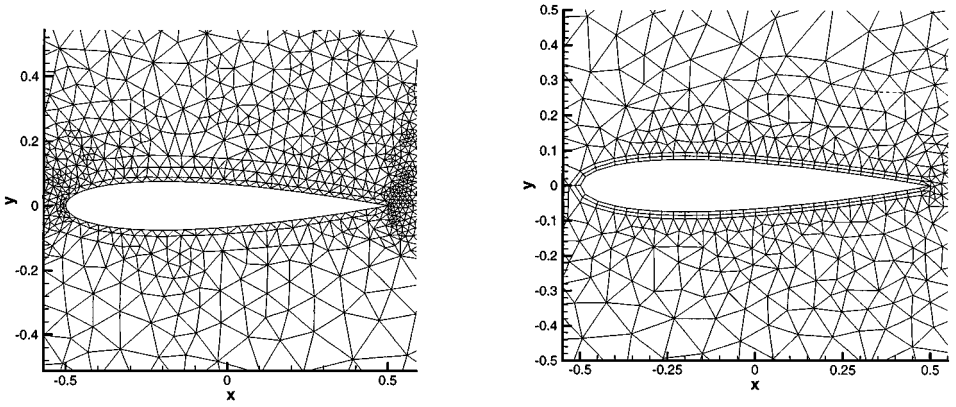

FIG. 15. Left, unstructured discretization consisting of triangles only. Right, hybrid discretizations consisting of triangles and quadrilaterals. All dimensions are in units of chord length.

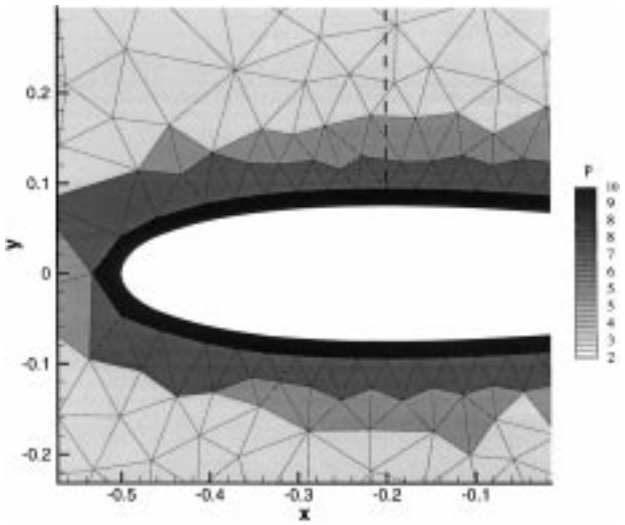

FIG. 16. Hybrid discretization showing the variable p-order on a gray-scale map around the airfoil. The dash vertical line indicates the location where boundary layer profiles are taken (see Fig. 17).

Returning now to the unstructured grid, we test convergence by also performing p-refinement on the same triangulization but with three different values of spectral order p corresponding to 2nd, 3rd, and 4th order polynomial interpolation. In Fig. 18 we plot the computed lift and drag coefficients versus the angle of attack for grids corresponding to $p = 2, 3$ and $p = 4$. We also include (with symbols) the computational results of [40], and we see that in general there is very good agreement except at the large angles of attack close to 50 degrees. This difference is due to qualitative difference in flow structure at small scales, which are only resolved with the higher-order simulations.

The above results were obtained by prescribing the grid velocity $\mathbf{U}^g$ so that the entire grid moves with the airfoil in a rigid body rotation. In this case, there is no grid distortion as in the method in [40]. In order to examine the accuracy of the proposed method in the presence of significant distortions we repeated the simulations with a grid velocity computed as described in Subsection 2.3. The results in this case were identical with the

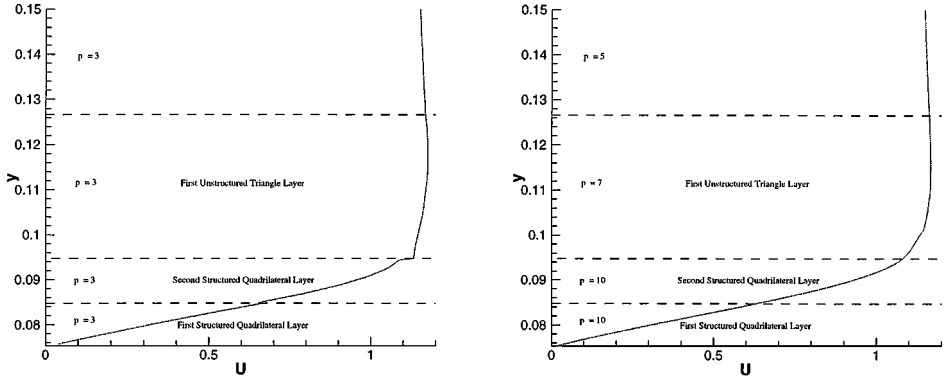

FIG. 17. Boundary layer profiles for a simulation with uniform p-resolution (left) and variable p-resolution (right, as shown in Fig. 16).

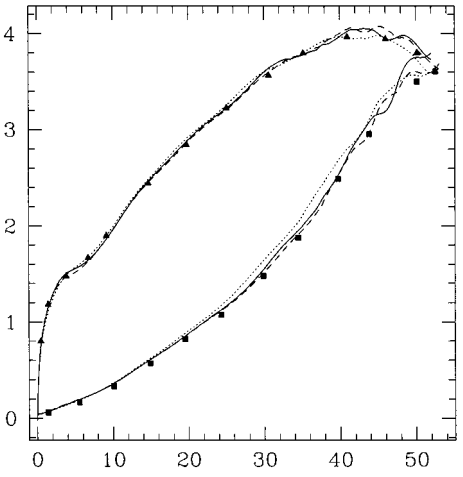

FIG. 18. Lift (upper curve) and drag (lower curve) coefficients versus angle of attack in degrees. The symbols correspond to computations of [40], the dotted line corresponds to our simulation at $p = 2$, the solid line to $p = 3$, and the dashed line to $p = 4$.

results plotted in Fig. 18. However, above an angle of about 30 degrees there is an edge crossing that produces negative Jacobian somewhere in the domain around the airfoil and the computation cannot continue with the same grid so a new h-refinement is needed.

To examine differences in the flow field due to spatial resolution we plot in Fig. 19 density contours for the cases $p = 2$ and $p = 3$ at non-dimensional time $t = 0.75$ corresponding to an angle of attack 18.55 degrees. We see that the higher resolution simulation provides a more detailed picture of the vortex shedding in the near-wake, but the contours around the airfoil are very similar. These results correspond to the ALE computation with the grid velocity computed as in Subsection 2.3 but comparison with the rigid body rotation simulation revealed identical results [39]. Similarly, at a later time $t = 1.5$ corresponding to an angle of attack of 44.1 degrees there are differences between the computations at resolution $p = 2$ and $p = 3$ and these differences are now extended to the upper surface of the airfoil where an interaction between the trailing edge vortex and the upstream propagating shed vortex

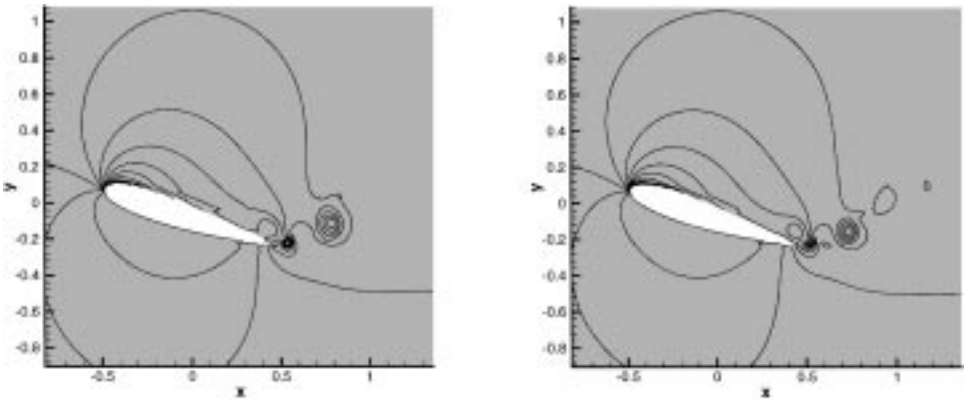

FIG. 19. Density contours of the pitching airfoil at non-dimensional time $t = 0.75$ corresponding to 18.55 degrees angle of attack. Shown on the left are contours at spectral order $p = 2$ and on the right at $p = 3$.

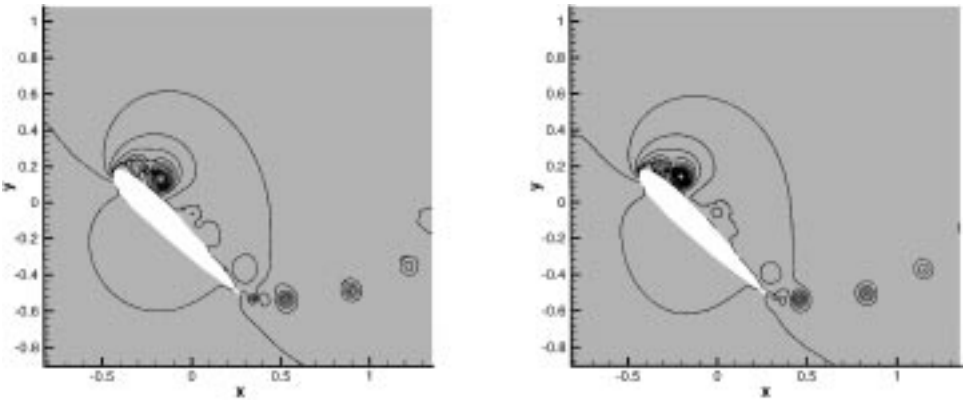

FIG. 20. Density contours of the pitching airfoil at non-dimensional time $t = 1.5$ corresponding to 44.1 degrees angle of attack. Shown on the left are contours at spectral order $p = 2$ and on the right at $p = 3$.

takes place, as shown in Fig. 20. These flow pattern differences are responsible for the aforementioned differences in the lift and drag coefficient at large angle of attack as shown in Fig. 18.

4.3. *Flow around a 3D Flapping Wing*

The next example demonstrates that the proposed discretization can sustain large grid distortions in three-dimensions without the need for h-remeshing. We consider the flow past a three-dimensional wing formed by a prismatic NACA 4420 airfoil placed at 20 degrees angle of attack. Two-dimensional subsonic and supersonic simulations were presented in [4] so here we will examine the effect of a prescribed flapping motion on the lift and drag forces. In particular, we consider the wing moving according to

$$u = 0; \qquad v = A\cos(2\pi f t)H(z - z_0)2(z - z_0)/(L_z/2); \qquad w = 0,$$

where z runs along the span of the airfoil, $z_0 = 2.5$ is the reference point, $L_z = 5$ is the spanwise length of the airfoil, $A = 0.5$ is the amplitude of the motion, and f is the frequency with $2\pi f = 1.57$; also $H(z)$ is the Heaviside function. The motion we simulate resembles in some general way the flapping motion characteristic of insect flight [42].

Here we have performed simulations at chord Reynolds number $Re = 680$ and Mach number $M = 0.3$. The discretization consists of 15,870 tetrahedra of $p = 3$ polynomial order and the time step was taken $\Delta t = 0.00025$. A typical "slice" of the discretized domain around the airfoil is shown in Fig. 21. The origin of the reference frame is at the midpoint of the airfoil and the domain extends from $x = -2.5$ at the inflow to $x = 7.5$ at the outflow and from $y = -2.5$ to $y = 2.5$ at the sides. Here the non-dimensionalization is with respect to the chord length ($C = 2$ in our computations) and the freestream velocity ($U_\infty = 1.75$ in our computations).

In Fig. 22 we plot the drag and lift coefficients for two cycles of the flapping motion. Note that these coefficients are defined by dividing the corresponding force with the area of the wing (i.e., $L_z \times C$). In the plot we present separately the forces due to pressure and due to viscosity, and we see that the viscous forces contribute a non-negligible amount to the drag force unlike the lift force. In Fig. 23 we plot again the lift and drag coefficients versus time, and we compare them with the corresponding coefficients from exactly the

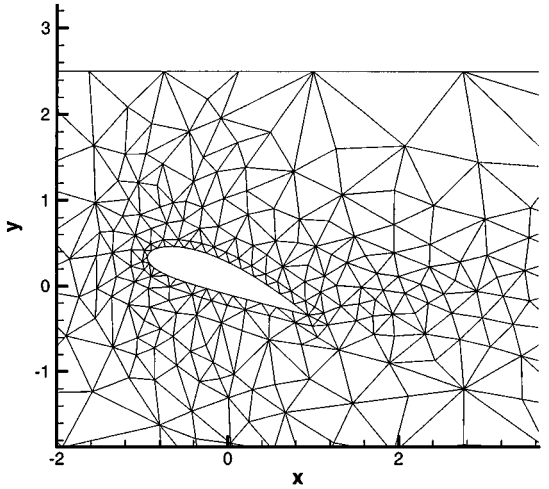

FIG. 21. A "slice" of the computational 3D domain around the airfoil; 15,870 tetrahedra are employed in the discretization.

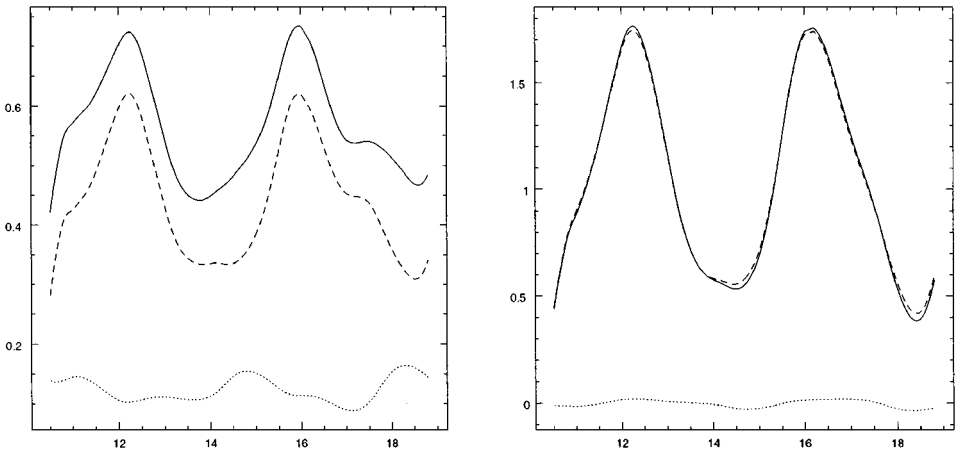

FIG. 22. Drag (left) and lift (right) coefficients versus time for the 3D flapping wing. The dashed line denotes contribution from pressure forces, and the dotted line denotes contributions from viscous forces; the solid line is the total force.

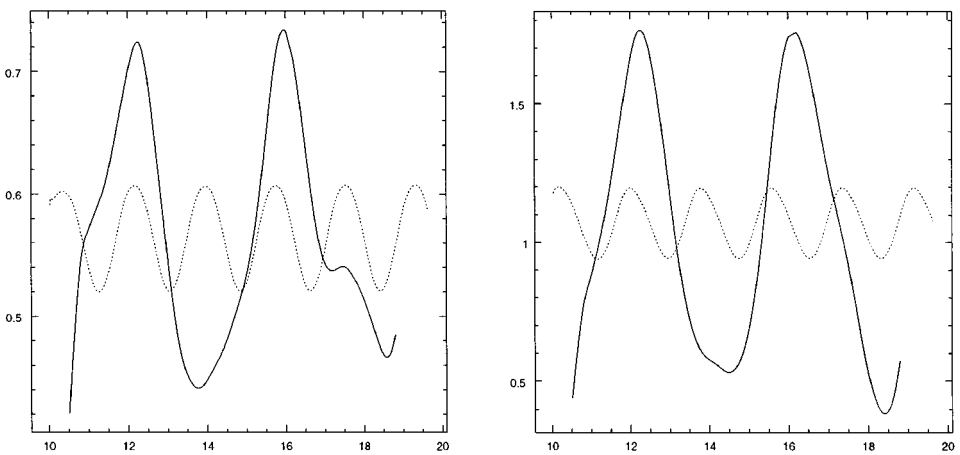

FIG. 23. Drag (left) and lift (right) coefficients versus time for the 3D flapping wing. The dotted line denotes force of the corresponding two-dimensional simulation and the solid line shows the simulation results of the 3D flapping wing.

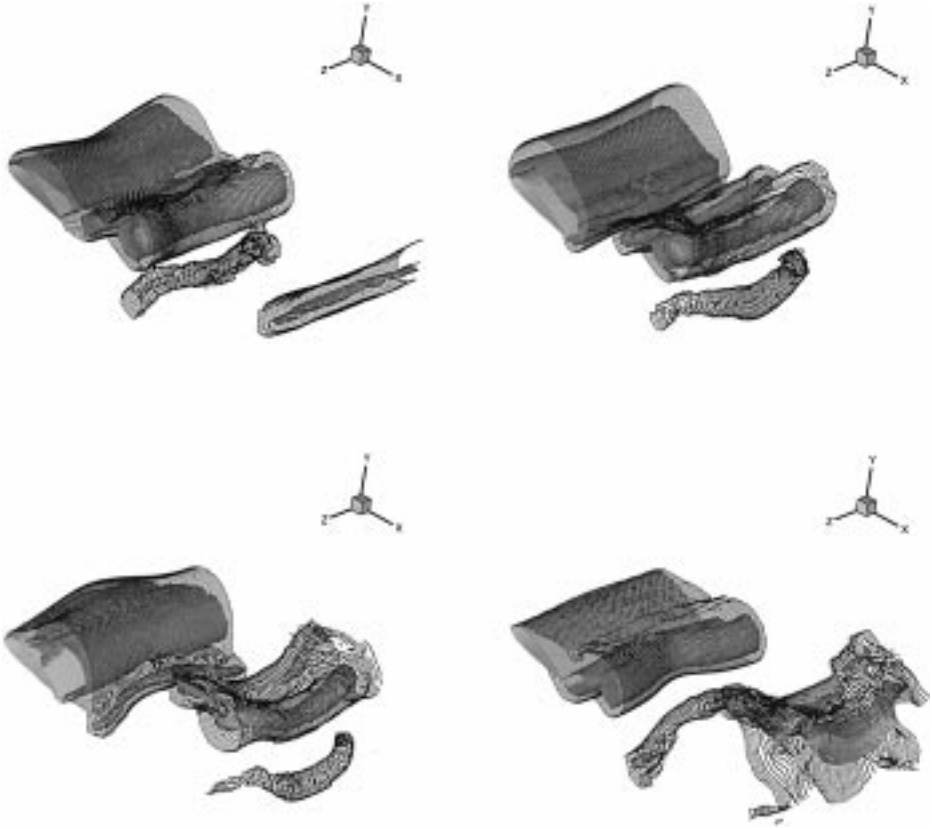

FIG. 24. Density contours from the 3D flapping wing simulation corresponding to time $t = 15.5$ (top left); 16.3 (top right); 17.5 (bottom left); 18.4 (bottom right). The simulation with flapping started at $t = 10.36$ from a simulation with the stationary configuration and the flapping period is 4 in non-dimensional units.

same airfoil but in two-dimensional flow at the same angle of attack. The time variation is due to natural vortex shedding in this case. It is interesting to note that instantaneous values of the lift-over-drag ratio can be increased by about 25% compared to the time-averaged valued, with the latter close to the value obtained from the corresponding two-dimensional simulations.

Next we present a sequence of flow visualizations during one flapping cycle in Fig. 24. We use minima of density contours to capture the vortex tubes that are shed off the flapping wing. We see that there seems to be a clear lag between the motion of the flapping wing and the visualized vortex tubes. The flapping motion essentially rearranges the vortex street resulting in a very different lift and drag force distribution. To examine qualitatively this difference we plot in Fig. 25 instantaneous contours of density first from the two-dimensional simulation (left) and also from the three-dimensional simulation at the mid-plane (right). We see that in the former case a regular von Karman vortex street is formed, but in the latter an irregular secondary vortex street is developed downstream.

4.3.1. Computational cost. Finally, we conclude this section by commenting on the computational cost of the simulations. Both the two-dimensional and the three-dimensional simulations were run using an MPI-based parallel version of the method presented here with the partitioning based on a multi-level graph approach provided by the METIS software

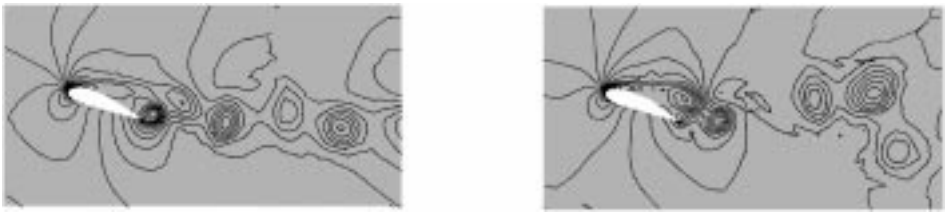

FIG. 25. Instantaneous density contours comparing the structure of vortex street for the two-dimensional simulation on the left and a mid-plane "slice" of the three-dimensional simulation on the right. For the three-dimensional simulation the time instance is the same as the bottom left image in Fig. 24.

[43]. Specifically, the three-dimensional simulation was run for 33,000 time steps for a total of 50 CPU hours on 32 processors of the IBM SP2/P2SC system.

5. SUMMARY AND CONCLUSIONS

We have developed a new method for solving the compressible Navier–Stokes equations in moving computational domains using the arbitrary Lagrangian Eulerian (ALE) formulation. Although high-order, this method employs standard unstructured and hybrid grids consisting of arbitrary triangles and quadrilaterals in two dimensions, and tetrahedra, hexahedra, prisms, and pyramids in three dimensions. The equations are solved in the weak form using a discontinuous Galerkin approach both for the advective and diffusive components. Unlike the standard Galerkin treatment [3], the new formulation relaxes continuity constraints across subdomains and thus allows the use of any convenient trial basis. In particular, in the current work we employed an orthogonal tensor-product spectral basis in the non-orthogonal subdomains, as it results in high computational efficiency. Specifically, the computational cost is $n_{el}p^{d+1}$ (where $d = 2$ or 3 in 2D and 3D, respectively) with n_{el} the number of elements and p the polynomial order in an element. This cost corresponds to differentiation and integration cost on the entire domain and is similar to the cost of such operations in standard global methods in simple separable domains [44]. The only matrix inversion required is that of a *local* mass matrix, which is diagonal, and thus trivial to invert.

As regards the grid movement, we have developed a new algorithm based on graph theory and a modification of the barycenter version of the forced-directed method. This algorithm avoids the solution of a computationally expensive Poisson equation for the grid velocity, which is typically required in ALE formulations. Moreover, because the proposed method is not particularly sensitive to grid distortions, incomplete convergence is sufficient to update the location of grid points.

We have demonstrated through examples, first the fast *convergence* of the method in distorted grids, and second that *time accuracy* of third-order can be obtained in a straight forward manner. We validated the method against published results for a pitching airfoil using unstructured and hybrid grids and demonstrated the ability to verify the solution without remeshing in physical space but rather refining hierachically in modal space. We also included a three-dimensional simulation of low subsonic flow past a three-dimensional flapping wing in insect-like motion to demonstrate the flexibility and efficiency of the method. Our experience so far with the discontinuous Galerkin method is that it is a robust method appropriate for high Reynolds number flow simulation; similar conclusions have been reached independently by other groups [15, 45]. The method is element-wise conservative and satisfies monotonicity without the need for flux limiters at subsonic, transonic, or

supersonic speeds although in the latter case a "constant element" ($p = 0$) and h-refinement around strong discontinuities is required. Several other recent supersonic simulations with moving domains and large deformations we have completed also point to the robustness of the discontinuous Galerkin ALE approach [39].

As regards the computational cost of the overall method, in our version of the method (see also [15]) the use of auxiliary variables introduces three new variables for each original one in three-dimensions, unlike the variational approach proposed by Oden *et al.* [18]. This is an important issue that we are currently investigating. Although more efficient, the difficulty with the Oden *et al.* formulation is that, as reported, constant elements ($p = 0$) lead to instabilities, and we need to have such elements in supersonic flows for shock capturing without flux limiters. A compromising solution will be to use the Oden *et al.* formulation and employ first-order ($p = 1$) elements in conjunction with some (relatively simple) flux limiters for shock capturing or other penalty-type terms [46]. The alternative approach of using high-order limiters proposed in [11, 13], which we have also used in previous work [4], is not as robust, and although it works for simple model problems it does not improve the quality of results in more realistic flow simulations.

The intended primary use of the proposed method is for *direct numerical simulation* of compressible turbulent flow past flexible structures avoiding the currently used *ad hoc* turbulence transport modeling [47], an erroneous approach for non-equilibrium turbulence.

APPENDIX I

Hierarchical Spectral Basis on Hybrid Domains

We include here for completeness the spectral basis we use as trial basis in the discontinuous Galerkin formulation presented in this paper. This basis is suitable for triangles, quadrilaterals, tetrahedra, hexahedra, prisms, and pyramids. It is orthogonal, hierarchical, and it has a tensor product form if the appropriate coordinate system is used as shown below. This basis was developed by Sherwin in [6], and more details can be found in [29].

Local Coordinate Systems

We start by defining a convenient set of local coordinates upon which we can construct the expansions. Moving away from the use of barycentric coordinates, which are typically applied to unstructured domains, we define a set of *collapsed Cartesian* coordinates in non-rectangular domains. These coordinates will form the foundation of the polynomial expansions. The advantage of this system is that every domain can be bounded by constant limits of the new local coordinates; accordingly operations such as integration and differentiation can be performed using standard one-dimensional techniques.

The new coordinate systems are based upon the transformation of a triangular region to a rectangular domain (and vice versa) as shown in Fig. 26. The main effect of the

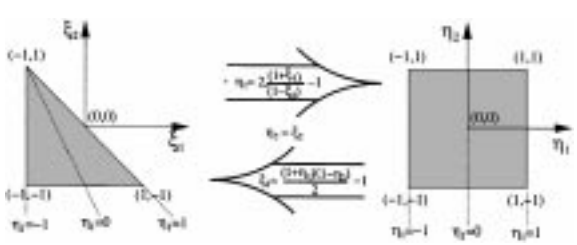

FIG. 26. Triangle to rectangle transformation.

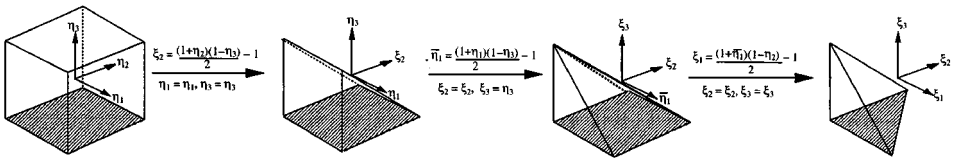

FIG. 27. Hexahedron to tetrahedron transformation.

transformation is to map the vertical lines in the rectangular domain (i.e., lines of constant η_1) onto lines radiating out of the point $(\xi_1 = -1, \xi_2 = 1)$ in the triangular domain. The triangular region can now be described using the "ray" coordinate (η_1) and the standard horizontal coordinate $(\xi_2 = \eta_2)$. The triangular domain is therefore defined by $(-1 \leq \eta_1, \eta_2 \leq 1)$ rather than the Cartesian description $(-1 \leq \xi_1, \xi_2; \xi_1 + \xi_2 \leq 0)$ where the upper bound couples the two coordinates. The "ray" coordinate (η_1) is multi-valued at $(\xi_1 = -1, \xi_2 = 1)$. Nevertheless, we note that the use of singular coordinate systems is very common arising in both cylindrical and spherical coordinate systems.

As illustrated in Fig. 27, the same transformation can be repeatedly applied to generate new coordinate systems in three dimensions. Here, we start from the bi-unit hexahedral domain and apply the triangle to rectangle transformation in the vertical plane to generate a prismatic region. The transformation is then used in the second vertical plane to generate the pyramidic region. Finally, the rectangle to triangle transformation is applied to every square cross section parallel to the base of the pyramidic region to arrive at the tetrahedral domain.

By determining the hexahedral coordinates (η_1, η_2, η_3) in terms of the Cartesian coordinates of the tetrahedral region (ξ_1, ξ_2, ξ_3) we can generate a new coordinate system for the tetrahedron. This new system and the planes described by fixing the local coordinates are shown in Fig. 28. Also shown are the new systems for the intermediate domains which are generated in the same fashion. Here we have assumed that the local Cartesian coordinates for every domain are (ξ_1, ξ_2, ξ_3).

Spectral Hierarchical Expansions

For each of the hybrid domains we can develop a polynomial expansion based upon the local coordinate system derived in Section 5. These expansions will be polynomials in terms of the local coordinates as well as the Cartesian coordinates (ξ_1, ξ_2, ξ_3). This is

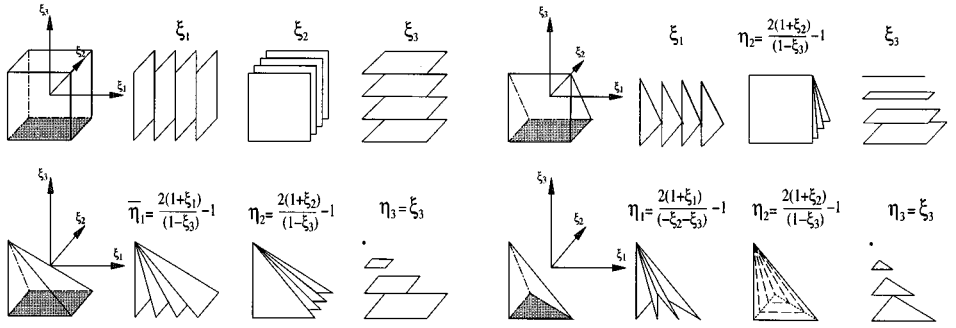

FIG. 28. The local coordinate systems used in each of the hybrid elements and the planes described by fixing each local coordinate.

a significant property as primary operations such as integration and differentiation can be performed with respect to the local coordinates but the expansion may still be considered as a polynomial expansion in terms of the Cartesian system.

We shall initially consider expansions which are orthogonal in the Legendre inner product. We define three principle functions $\phi_i^a(z)$, $\phi_{ij}^b(z)$, and $\phi_{ijk}^c(z)$, in terms of the Jacobi polynomial, $P_p^{\alpha,\beta}(z)$, as

$$\phi_i^a(z) = P_i^{0,0}(z), \qquad \phi_{ij}^b(z) = \left(\frac{1-z}{2}\right)^i P_j^{2i+1,0}(z),$$

$$\phi_{ijk}^c(z) = \left(\frac{1-z}{2}\right)^{i+j} P_k^{2i+2j+2,0}(z).$$

Using these functions we can construct the orthogonal polynomial expansions:

hexahedral expansion	prismatic expansion
$\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \phi_p^a(\xi_1)\phi_q^a(\xi_2)\phi_r^a(\xi_3)$	$\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \phi_p^a(\xi_1)\phi_q^a(\eta_2)\phi_{qr}^b(\xi_3)$
pyramidic expansion	**tetrahedral expansion**
$\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \phi_p^a(\overline{\eta_1})\phi_q^a(\eta_2)\phi_{pqr}^c(\eta_3)$	$\phi_{pqr}(\xi_1,\xi_2,\xi_3) = \phi_p^a(\eta_1)\phi_{pq}^b(\eta_2)\phi_{pqr}^c(\eta_3),$

where

$$\eta_1 = \frac{2(1+\xi_1)}{(-\xi_2-\xi_3)} - 1, \qquad \overline{\eta_1} = \frac{2(1+\xi_1)}{(1-\xi_3)} - 1, \qquad \eta_2 = \frac{2(1+\xi_2)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3,$$

are the local coordinates illustrated in Fig. 28.

The hexahedral expansion is simply a standard tensor product of Legendre polynomials (since $P_p^{0,0}(z) = L_p(z)$). In the other expansions the introduction of the degenerate local coordinate systems is linked to the use of the more unusual functions $\phi_{ij}^b(z)$ and $\phi_{ijk}^c(z)$. These functions both contain factors of the form $(\frac{1-z}{2})^p$ which is necessary to keep the expansion as a polynomial of the Cartesian coordinates (ξ_1,ξ_2,ξ_3). For example, the coordinate η_2 in the prismatic expansion necessitates the use of the function $\phi_{qr}^b(\xi_3)$ which introduces a factor of $((1-\xi_3)/2)^q$. The product of this factor with $\phi_q^a(\eta_2)$ is a polynomial function in ξ_2 and ξ_3. Since the remaining part of the prismatic expansion, $\phi_p^a(\xi_1)$, is already in terms of a Cartesian coordinate the whole expansion is a polynomial in terms of the Cartesian system.

The polynomial space, in Cartesian coordinates, for each expansion is

$$\mathcal{P} = \mathrm{Span}\left\{\xi_1^p \quad \xi_2^q \quad \xi_3^r\right\}, \tag{11}$$

where pqr for each domain is

Hexahedron	$0 \le p \le P_1$	$0 \le q \le P_2$	$0 \le r \le P_3$
Prism	$0 \le p \le P_1$	$0 \le q \le P_2$	$0 \le q+r \le P_3$
Pyramidic	$0 \le p \le P_1$	$0 \le q \le P_2$	$0 \le p+q+r \le P_3$
Tetrahedron	$0 \le p \le P_1$	$0 \le p+q \le P_2$	$0 \le p+q+r \le P_3.$

(12)

The range of the p, q, and r indices indicates how the expansions should be expanded to generate a complete polynomial space. We note that if $P_1 = P_2 = P_3$ then the tetrahedral and pyramidic expansions span the same space and are in a subspace of the prismatic expansion which is in turn a subspace of the hexahedral expansion.

An important property of the hybrid spectral basis is that it is orthogonal in the new coordinate system that we introduce. This simplifies greatly the discontinuous Galerkin formulation, since all mass matrices are diagonal and their inversion is trivial.

APPENDIX II

Numerical Quadrature and Flux Computation

Here we provide some details on how the numerical quadrature is performed on polymorphic elements, elaborating on what has already been presented in Section 3. Specifically, we examine how the flux terms involved in the discontinuous Galerkin formulation are computed. To this end, we take advantage of the tensor product in the transformed space (η_1, η_2, η_3) to perform integration. The integrations over each element can be performed as a set of one-dimensional integrals using variants of Gauss quadrature. If we used the reference coordinate systems this would be very difficult since the limits of the "collapsed" elements are not constant.

We first describe the choice of quadrature type for integrating each direction. We will then motivate the inclusion of quadrature with non-constant weights in order to reduce the number of points we use. For example, in two dimensions we consider integrals of the form

$$\begin{aligned}\int_{Physical} f(\mathbf{x})\,dx\,dy &= \int_{Reference} f(\mathbf{x}(\xi))\frac{\partial(\mathbf{x})}{\partial(\xi)}\,d\xi_1\,d\xi_2 \\ &= \int_{Tensor} f(\mathbf{x}(\xi(\eta)))\frac{\partial(\mathbf{x})\partial(\xi)}{\partial(\xi)\partial(\eta)}\,d\eta_1\,d\eta_2.\end{aligned}$$

We use the Gauss weights that will perform the discrete integral of a function as a sum

$$\int_{-1}^{1}(1-z)^{\alpha}(1+z)^{\beta} f(z)\,dz = \sum_{i=0}^{N-1} f\left(z_i^{\alpha,\beta}\right) w_i^{\alpha,\beta}.$$

This will be used in each of the d directions in the d-dimensional elements. In Table II we show the type of Gaussian quadrature we use in each of the ξ_1, ξ_2, ξ_3 directions.

For the Discontinuous Galerkin formulation it is necessary to evaluate terms of the form

$$\int_{\partial\Omega} f\phi_n + \int_{\Omega} F\phi_n,$$

TABLE II

Element	η_1	η_2	η_3
Triangle	*GLL*	$GRJ_{0,0}$	—
Quadrilateral	*GLL*	*GLL*	—
Tetrahedron	*GLL*	$GRJ_{0,0}$	$GRJ_{0,0}$
Pyramid	*GLL*	*GLL*	$GRJ_{0,0}$
Prism	*GLL*	*GLL*	$GRJ_{0,0}$
Hexahedron	*GLL*	*GLL*	*GLL*

Note. *GLL* implies Gauss–Lobatto–Legendre which is the Gauss quadrature for a constant weight function with both $x = \pm 1$ points endpoints included . $GRJ_{\alpha,\beta}$ implies Gauss–Radaus–Jacobi quadrature with (α, β) weights and one of the endpoints included.

where $\partial\Omega$ is the boundary of an element Ω, for all the ϕ_n test functions in the elemental basis. There are $N\frac{(N+1)}{2}$ test functions for a triangle so the boundary integral is an $O(N^3)$ operation. This means that the flux integration is as expensive as the volume integral. We can reduce the cost of this integral by examining the discrete sum form

$$\int_{\partial\Omega} f\phi_n = \sum_{i=0}^{N} \phi_n(\eta_{1i}, 0)\, f^1(\eta_{1i}) w_i^1 J^1(\eta_{1i}) + \sum_{i=0}^{N} \phi_n(1, \eta_{2i})\, f^2(\eta_{2i}) w_i^2 J^2(\eta_{2i}) + \sum_{i=0}^{N} \phi_n(-1, \eta_{2i})\, f^3(\eta_{2i}) w_i^2 J^3(\eta_{2i}),$$

where J^n and f^n are the Jacobian and flux function for the nth edge.

We can rewrite the $edge_1$ flux as

$$\int_{edge_1} f\phi_n = \sum_{j=0}^{N}\sum_{i=0}^{N} \left(\frac{J^1(\eta_{1i})}{w_0^b}\right) f^e(\eta_{1i})\delta_{j0}\phi_n(\eta_{1i}, \eta_{2j}) w_i^1 w_j^2,$$

where

$$\delta_{ij} = \begin{matrix} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{matrix}$$

The fluxes for the other edges can be constructed in similar ways. Using this summation representation we can now evaluate the surface flux integral by adding the edge fluxes scaled by weight and Jacobians to the F field and then evaluating one volume integral.

ACKNOWLEDGMENTS

We thank Dr. T. Warburton for his help with the hybrid discretization. This work was supported by AFOSR and DARPA and computations were done on the IBM SP2 at CFM and NPACI, and on the SGI O 2000 at NCSA.

REFERENCES

1. C. Farhat, M. Lesoinne, and P. LeTallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity, *Comput. Methods Appl. Mech. Eng.* **157**, 95 (1998).
2. T. Tezduyar, M. Behr, and J. Liu, A new strategy for finite element computations involving moving boundaries and interfaces—The deforming spatial domain/space-time procedure. I. The concept and the preliminary numerical tests, *Comput. Methods Appl. Mech. Eng.* **94**, 339 (1992).
3. S. J. Sherwin and G. E. Karniadakis, Tetrahedral hp finite elements: Algorithms and flow simulations, *J. Comput. Phys.* **122**, 191 (1996).
4. I. Lomtev, C. Quillen, and G. E. Karniadakis, Spectral/hp methods for viscous compressible flows on unstructured 2D meshes, *J. Comput. Phys.* **144**, 325 (1998).
5. M. Dubiner, Spectral methods on triangles and other domains, *J. Sci. Comput.* **6**, 345 (1991).
6. S. J. Sherwin, Hierarchical *hp* finite elements in hybrid domains, *Finite Element Anal. Design* **27**, 109 (1997).
7. B. Cockburn and C.-W. Shu, The local discontinuous Galerkin for time dependent convection-diffusion systems, *SIAM J. Numer. Anal.* **35**, 2440 (1998).
8. B. Cockburn and C.-W. Shu, TVB Runge–Kutta discontinuous method for conservation laws, V, *J. Comput. Phys.* **141**, 199 (1998).

9. B. Cockburn and C.-W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework, *Math. Comp.* **52**, 411 (1989).
10. B. Cockburn, S.-Y. Lin, and C.-W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. III. One-dimensional systems, *J. Comput. Phys.* **84**, 90 (1989).
11. B. Cockburn, S. Hou, and C.-W. Shu, The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. The multi-dimensional case, *J. Comput. Phys.* **54**, 545 (1990).
12. B. Cockburn and C.-W. Shu, *P^1-RKDG method for two-dimensional Euler equations of gas dynamics*, in *Proc. Fourth Int. Symp. on CFD, UC Davis, 1991*.
13. R. Biswas, K. Devine, and J. Flaherty, Parallel, adaptive finite element methods for conservation laws, *Appl. Numer. Math.* **14**, 255 (1994).
14. F. Bassi and S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, *J. Comput. Phys.* **131**, 267 (1997).
15. F. Bassi, S. Rebay, M. Savini, G. Marioti, and S. Pedinotti, A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows, in *Proceedings of the Second European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Antwerpen, Belgium, March 5–7, 1997.*
16. K. S. Bey, A. Patra, and J. T. Oden, hp version discontinuous Galerkin methods for hyperbolic conservation laws: A parallel adaptive strategy, *Int. J. Numer. Methods Eng.* **38**, 3889 (1995).
17. K. S. Bey, A. Patra, and J. T. Oden, hp version discontinuous Galerkin methods for hyperbolic conservation laws, *Comput. Methods Appl. Mech. Eng.* **133**, 259 (1996).
18. J. T. Oden, I. Babuska, and C. E. Baumann, A discontinuous hp finite element method for diffusion problems, *J. Comput. Phys.* **146**, 491 (1998).
19. C. E. Baumann and J. T. Oden, A discontinuous hp finite element method for the solution of the Euler and Navier–Stokes equations, *Int. J. Numer. Methods Fluids*, in press.
20. C. E. Baumann and J. T. Oden, A discontinuous hp finite element method for convection-diffusion problems, *Comput. Methods Appl. Mech. Eng.* **175**, 311 (1999).
21. T. J. R. Hughes, W. K. Liu, and T. K. Zimmerman, Lagrangian–Eulerian finite element formulation for incompressible viscous flows, *Comput. Methods Appl. Mech. Eng.* **29**, 329 (1981).
22. T. Nomura and T. J. R. Hughes, An arbitrary Lagrangian–Eulerian finite element method for interaction of fluid and a rigid body, *Comput. Methods Appl. Mech. Eng.* **95**, 115 (1992).
23. C. W. Hirt, A. A. Amsden, and H. K. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, *J. Comput. Phys.* **14**, 27 (1974).
24. J. Donea, S. Giuliani, and J. P. Halleux, An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid-structure interactions, *Comput. Methods Appl. Mech. Eng.* **33**, 689 (1982).
25. L.-W. Ho, *A Legendre Spectral Element Method for Simulation of Incompressible Unsteady Free-Surface Flows*, Ph.D. thesis, Massachustts Institute of Technology, 1989.
26. C. S. Venkatasubban, A new finite element formulation for ALE Arbitrary Lagrangian Eulerian compressible fluid mechanics, *Int. J. Eng. Sci.* **33**(12), 1743 (1995).
27. R. Lohner and C. Yang, Improved ale mesh velocities for moving bodies, *Comm. Numer. Methods Eng. Phys.* **12**, 599 (1996).
28. G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing* (Prentice Hall, Englewood Cliffs, NJ, 1998).
29. G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for CFD* (Oxford Univ. Press, London, 1999).
30. G. Jiang and C. W. Shu, On a cell entropy inequality for discontinuous Galerkin methods, *Math. Comp.* **62**, 531 (1994).
31. C. Johnson and J. Pitkaranta, An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation, *Math. Comp.* **46**, 1 (1986).
32. J. Jaffre, C. Johnson, and A. Szepessy, Convergence of the discontinuous Galerkin finite element method for hyperbolic conservation laws, *Math. Models Methods Appl. Sci.* **5**, 367 (1995).
33. B. Koobus and C. Farhat, Second-order schemes that satisfy GCL for flow computations on dynamic grids, in *AIAA 98-0113, 36th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 12–15, 1998.*

34. H. Guillard and C. Farhat, On the significance of the GCL for flow computations on moving meshes, in *AIAA 99-0793, 37th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 11–14, 1999.*
35. I. G. Giannakouros, *Spectral Element/Flux-Corrected Methods for Unsteady Compressible Viscous Flows*, Ph.D. thesis, Department of Mechanical and Aerospace Engineering, Princeton University, 1994.
36. J. T. Batina, Unsteady Euler airfoil solutions using unstructed dynamic meshes, *AIAA J.* **28**(8), 1381 (1990).
37. R. D. Rausch, J. T. Batina, and H. T. Y. Yang, Three-dimensional time-marching aeroelastic analyses using an unstructured-grid Euler method, *AIAA J.* **31**(9), 1626 (1994).
38. F. J. Blom and P. Leyland, Analysis of fluid-structure interaction by means of dynamic unstructured meshes, *Trans. ASME* **120**, 792 (1998).
39. I. Lomtev, *Discontinuous Spectral/hp Element Methods for High Speed Flows*, Ph.D. thesis, Applied Mathematics, Brown University, 1999.
40. M. R. Visbal and J. S. Shang, Investigation of the flow structure around a rapidly pitching airfoil, *AIAA J.* **27**(8), 1044 (1989).
41. R. Beam and R. Warming, An implicit factored scheme for the compressible Navier–Stokes equations, *AIAA J.* **16**, 393 (1978).
42. H. Liu and K. Kawachi, A numerical study of insect flight, *J. Comput. Phys.* **146**, 124 (1998).
43. G. Karypis and V. Kumar, *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System Version 2.0*, Technical Report, Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, 1995.
44. D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods* (Soc. for Indust. & Appl. Math., Philadelphia, 1977).
45. C. E. Baumann, *An hp-Adaptive Discontinuous Finite Element Method for Computational Fluid Dynamics*, Ph.D. thesis, The University of Texas at Austin, 1997.
46. C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method* (Cambridge Univ. Press, Cambridge, UK, 1994).
47. H. Tran, B. Koobus, and C. Farhat, Numerical solution of vortex dominated flow problems with moving grids, in *AIAA 98-0766, 36th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 12–15, 1998.*