49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2011, Orlando, Florida
AIAA 2011-45

Efficient Solution Techniques for High-Order Methods on 3D Anisotropic Hybrid Meshes

Takanori Haga[1], Haiyang Gao[2] and Z.J. Wang[3]
Department of Aerospace Engineering and CFD Center, Iowa State University, Ames, IA 50011

Recently a differential discontinuous formulation for conservation laws named the Correction Procedure via Reconstruction (CPR) was developed. CPR is inspired by several other discontinuous methods such as the discontinuous Galerkin, spectral volume and spectral difference methods. In fact, all of them can be unified under the CPR framework, which is relatively simple to implement. The aim of the present work is to investigate and develop an efficient solution algorithm for high speed viscous flows using the high-order CPR discretization. A *p*-multigrid solver and an implicit line solver are developed and tested on several test cases to show the performance.

I. Introduction

Advantages of high-order methods are well recognized in the computational fluid dynamics (CFD) community especially for aeroacoustic noise predictions, vortex dominated flows, large eddy simulation and direct numerical simulation (DNS) of turbulent flows. Since the truncation error of a high-order method decreases more rapidly than that of a lower order method if the flow field is sufficiently smooth, the more stringent the accuracy requirement is, the more efficient a high-order method becomes in computational cost. For the practical use in industries, lower order (1st or 2nd) unstructured grid methods are usually employed for the reason of superior geometrical flexibility and robustness. However, these methods are likely too dissipative to capture small vortex structures in turbulent flows and are often not capable of obtaining grid converged solutions.

In the past decades, there has been significant progress in developing high-order methods capable of solving the Navier-Stokes (NS) equations on unstructured grids. For compressible flow computations in aerospace applications, the discontinuous Galerkin (DG) method [20, 5, 1] has attracted intensive interest. One particular feature of the DG method is the discontinuous solution space of high-order approximations for each element, which allows the scheme to be very flexible in dealing with complex configuration and in accommodating solution based adaptations. Other methods assuming element-wise discontinuous solution are staggered-grid (SG) multi-domain spectral method [14], spectral volume (SV) [31, 27] and spectral difference (SD) [16] methods. Another notable feature that is common among these methods is the use of one of the Riemann solvers to compute unique fluxes at element interfaces to incorporate "upwinding" characteristics of wave propagation, similar to the Godunov type finite volume method. The main difference among these methods lies in how the degrees-of-freedom (DOFs) are chosen and how the governing equations are discretized. Comprehensive reviews of these methods can be found in [29].

Recently, a novel formulation named CPR (correction procedure via reconstruction) was developed by Huynh [11, 12] for 1D conservation laws, and extended to simplex and hybrid meshes by Wang and Gao [30]. The CPR method is based on a nodal differential form, with an element-wise discontinuous polynomial solution space. The solution polynomial is interpolated from the solutions at a set of solution points. This formulation has some remarkable properties. The framework is easy to understand, efficient to implement and recovers several known methods such as the DG, SG or the SV/SD methods. Furthermore, by choosing the solution points to coincide with the flux points, the reconstruction of solution polynomials to calculate the residual can be completely avoided. The DG scheme derived through the CPR framework is probably the simplest and most efficient amongst all DG formulations since explicit integrations are avoided. In the recent study [7, 8], the CPR method has been extended to the Navier-Stokes equations on 2D and 3D mixed meshes.

One of the critical issues of those high-order methods is the large computational cost. The total cost is sharply increased as the number of DOFs within elements is increased. There have been many research efforts to develop

[1] Post-doc Research Associate, Department of Aerospace Engineering, 2271 Howe Hall, AIAA Member.
[2] Graduate Research Assistant, Department of Aerospace Engineering, 2271 Howe Hall, AIAA Member.
[3] Professor of Aerospace Engineering, 2271 Howe Hall, Associate Fellow of AIAA.

efficient solution methods [2, 6, 19]. However, solution techniques are still not matured enough to realize the wide use of these high-order methods especially for high Reynolds number flows in 3D aerodynamic applications. It is known that the convergence performance of a common implicit solver e.g. a point Jacobi or a Gauss-Seidel, degrades dramatically for highly anisotropic meshes used to efficiently resolve thin boundary layers at high-Reynolds number flows. In our previous study [9], an implicit SV method was developed using the nonlinear LU-SGS scheme. Although the developed method demonstrated an order of magnitude increase in the convergence rate compared with an explicit Runge-Kutta method, apparent slow down of the convergence was observed for a Reynolds Averaged Navier-Stokes (RANS) computation with highly stretched anisotropic mesh. To alleviate the stiffness associated with stretched grids, a line implicit solver which treats clusters of elements implicitly with strongest coupling will be employed. Variants of line implicit relaxation methods have been successfully used in finite volume solvers [18] as well as in the DG methods [6, 24, 4].

The present work focuses on the solution methods for the 3D CPR discretiztions. In this study, the line implicit solver is developed for the CPR method on 3D hybrid grids together with a *p*-multigrid algorithm to speed up convergence and the performance is demonstrated for several benchmark flow problems.

II. Discretization of the Correction Procedure via Reconstruction (CPR)

A. Governing equations

The 3D compressible Navier-Stokes equations can be written as a system of partial differential equations in conservation form:

$$\frac{\partial Q}{\partial t} + \nabla \cdot \left(F^c(Q) - F^v(Q, \nabla Q) \right) = 0, \tag{2.1}$$

where $Q = (\rho, \rho u, \rho v, \rho w, e)$ is the conservative state vector, $F^c(Q)$ is the convective flux, $F^v(Q)$ is the viscous flux. For a perfect gas, the pressure is related to the total energy e by

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right). \tag{2.2}$$

The specific heat ratio is set to 1.4 for air and the Prandtl number in the viscous flux is assumed to be a constant of 0.72 for laminar flows. The computations for solving the Euler equations are performed by omitting the viscous flux.

B. Framework of the CPR Formulation

The CPR formulation can be derived from a weighted residual method by transforming the integral formulation into a differential one. A hyperbolic conservation law, which can be written as

$$\frac{\partial Q}{\partial t} + \vec{\nabla} \bullet \vec{F}(Q) = 0, \tag{2.3}$$

with suitable initial and boundary conditions. Q is the vector of conserved variables, and $\vec{F}$ is the flux vector. Assume that the computational domain Ω is discretized into N non-overlapping triangular (in 2D) or tetrahedral (in 3D) elements $\{V_i\}_{i=1}^N$. The weighted residual form of Eq. (2.3) on element V_i can be derived by multiplying Eq. (2.3) by an arbitrary weighting or test function W and integrating over V_i,

$$\begin{aligned} &\int_{V_i} \left(\frac{\partial Q}{\partial t} + \vec{\nabla} \bullet \vec{F}(Q) \right) W dV \\ &= \int_{V_i} \frac{\partial Q}{\partial t} W dV + \int_{\partial V_i} W \vec{F}(Q) \bullet \vec{n} dS - \int_{V_i} \vec{\nabla} W \bullet \vec{F}(Q) dW = 0. \end{aligned} \tag{2.4}$$

Let Q_i be an approximate solution to Q on V_i. On each element, the solution belongs to the space of polynomials of degree p or less, i.e., $Q_i \in P^p(V_i)$ with no continuity requirement across element interfaces. Let the dimension of P^p be K: $K=(p+1)(p+2)/2$ for a triangle or $K=p(p+1)(p+2)/3!$ for a tetrahedron). Then, we equire that the numerical solution Q_i must satisfy Eq. (2.4), i.e.,

$$\int_{V_i} \frac{\partial Q_i}{\partial t} W dV + \int_{\partial V_i} W \vec{F}(Q_i) \bullet \vec{n} dS - \int_{V_i} \vec{\nabla} W \bullet \vec{F}(Q_i) dV = 0. \tag{2.5}$$

Because the approximated solution is generally discontinuous across element interfaces, the fluxes at the interfaces are not well defined. Following the idea used in the Godunov method, the normal flux term in Eq. (2.5) is replaced with a common Riemann flux [13, 15, 21, 23], i.e.,

$$F^n(Q_i) \equiv \vec{F}(Q_i) \bullet \vec{n} \approx F^n_{com}(Q_i, Q_{i+}, \vec{n}), \tag{2.6}$$

where Q_{i+} denotes the solution outside the current element V_i. Thus, Eq. (2.5) becomes

$$\int_{V_i} \frac{\partial Q_i}{\partial t} W dV + \int_{\partial V_i} W F^n_{com} dS - \int_{V_i} \vec{\nabla} W \bullet \vec{F}(Q_i) dV = 0. \tag{2.7}$$

Applying integration by parts to the last term of (2.7), we obtain

$$\int_{V_i} \frac{\partial Q_i}{\partial t} W dV + \int_{V_i} W \vec{\nabla} \bullet \vec{F}(Q_i) dV + \int_{\partial V_i} W \left[F^n_{com} - F^n(Q_i) \right] dS = 0. \tag{2.8}$$

Note that the quantity $\vec{\nabla} \bullet \vec{F}(Q_i)$involves no influence from the data in the neighboring cells. The influence of these data is represented by the above boundary integral, which is also called a "penalty term", penalizing the normal flux differences.

The next step is critical in the elimination of the test function. The boundary integral above is cast as a volume integral via the introduction of a "correction field" on $V_i, \delta_i \in P^p(V_i)$,

$$\int_{V_i} W \delta_i dV = \int_{\partial V_i} W [F^n] dS. \tag{2.9}$$

where $[F^n] \equiv F^n_{com} - F^n(Q_i)$ is the normal flux difference. The above equation is sometimes referred to as the "lifting operator", which has the normal flux differences on the boundary as input and a member of $P^p(V_i)$ as output. Substituting Eq. (2.9) into Eq. (2.8), we obtain

$$\int_{V_i} \left[\frac{\partial Q_i}{\partial t} + \vec{\nabla} \bullet \vec{F}(Q_i) + \delta_i \right] W dV = 0. \tag{2.10}$$

If the flux vector is a linear function of the state variable, obviously $\vec{\nabla} \bullet \vec{F}(Q_i) \in P^p$. Therefore the terms inside the square bracket are all elements of P^p. Because the test space is selected to ensure a unique solution, Eq. (2.10) is equivalent to

$$\frac{\partial Q_i}{\partial t} + \vec{\nabla} \bullet \vec{F}(Q_i) + \delta_i = 0. \tag{2.11}$$

For nonlinear conservation laws, $\vec{\nabla} \bullet \vec{F}(Q_i)$ is usually not an element of P^p. As a result, (2.10) cannot be reduced to Eq. (2.11) generally. In this case, the most obviously choice is to project $\vec{\nabla} \bullet \vec{F}(Q_i)$ into P^p. Denote $\Pi\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right)$ a projection of $\vec{\nabla} \bullet \vec{F}(Q_i)$ to P^p. Then Eq. (2.10) reduces to

$$\frac{\partial Q_i}{\partial t} + \Pi\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right) + \delta_i = 0. \tag{2.12}$$

With the introduction of the correction field δ_i, and a projection of $\vec{\nabla} \bullet \vec{F}(Q_i)$ for nonlinear conservation laws, we have reduced the weighted residual formulation to a differential formulation, which involves no integrals.

Next, let the DOFs be the solutions at a set of solution points (SPs) $\{\vec{r}_{i,j}\}$ (j varies from 1 to K), as shown in Figure 1.1. Then Eq. (2.12) holds true at the SPs, i.e.,

$$\frac{\partial Q_{i,j}}{\partial t} + \Pi_j\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right) + \delta_{i,j} = 0, \tag{2.13}$$

where $\Pi_j\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right)$ denotes the values of $\Pi\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right)$ at SP j. The efficiency of the CPR approach hinges on how the correction field δ_i and the projection $\Pi\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right)$ are computed.

To compute δ_i, we define p+1 points named flux points (FPs) along each interface, where the normal flux differences are computed, as shown in Figure 1.1. We approximate (for nonlinear conservation laws) the normal flux difference $[F^n]$ with a degree p interpolation polynomial along each interface. For linear triangles with straight edges, once the solution points and flux points are chosen, the correction at the SPs can be written as

$$\delta_{i,j} = \frac{1}{|V_i|} \sum_{f \in \partial V_i} \sum_l \alpha_{j,f,l} [F^n]_{f,l} S_f, \tag{2.14}$$

where f is an face index, l is the FP index, $\alpha_{j,f,l}$ are lifting constants independent of the solution, S_f is the face area, $|V_i|$ is the volume of V_i. Note that the correction for each solution point, namely $\delta_{i,j}$, is a linear combination of all the normal flux differences on all the faces of the cell.

To compute $\Pi_j\left(\vec{\nabla} \bullet \vec{F}(Q_i)\right)$, two possible approaches were considered in the previous work [30]. One is the Lagrange polynomial (LP) approach and the other is the chain rule (CR) approach. Numerical experiments indicate that there is a slight loss of accuracy with the LP approach, but it is fully conservative. On the other hand, the CR approach is much more accurate than the LP approach, at the expense of full conservation. In this work, the CR approach is used.

It can be easily shown that the location of SPs does not affect the numerical scheme for linear conservation laws [28]. For efficiency, therefore, the solution points and flux points are always chosen to include corners of the cell. In addition, the solution points are chosen to coincide with the flux points along cell faces to avoid any solution reconstruction. Furthermore, in computations with hybrid meshes, the flux points are always of the same distribution for different cell types for ease of interface treatment. Due to the special choice of DOFs, the reconstruction cost in CPR is completely avoided, and the mass matrix is always the identity matrix.

The above formulation is the basis of the discretization on triangles in 2D and tetrahedrons in 3D. For quadrilaterals in 2D and hexahedrons in 3D, one dimensional CPR formulation (FR formulation [11]) is directly used based on a tensor-product bases approximation. For triangular prisms, 1D formulation and 2D formulation on triangles are combined using a tensor-product approach, which is shown later.

The CPR formulation has been extended to the Navier-Stokes equations on 2D mixed meshes [7] and also on 3D mixed meshes [8]. The details of the discretization can be found in the references. In this work, the Roe flux [21] is used to evaluate the common inviscid flux at the interfaces. And the BR2 scheme is used to compute the common viscous flux.

III. Discretization on Mixed Grids with Curved Boundary

The current development for 3D hybrid meshes accommodates two kinds of element shapes, i.e., tetrahedron and triangular prism. The use of prismatic cells in addition to tetrahedral cells has the advantage in both accuracy and computational costs to resolve boundary layers near solid walls. All elements are transformed from the physical domain (x, y, z) into a corresponding standard element in the computational domain (ξ, η, ζ) as shown in Fig. 1. Here we consider the transformations for the elements with curved sides (faces and edges). The discretization for the curved elements is conducted in the same way as the straight sided elements by applying the CPR formulation in the standard elements.

Based on a set of nodes defining the shape of an element, a set of shape functions can be obtained [33]. Once the shape functions $M_i(\xi,\eta,\varsigma)$ are given, the transformation can be written as

$$(\vec{r}) = \sum_j M_j(\vec{\xi})(\vec{r}_j), \tag{3.1}$$

where $\vec{r}_i$ are the physical coordinates used to define an element, and $M_j(\vec{\xi})$ is the shape function. Denote J the Jacobian matrix of the transformation, and $\vec{S}_\xi = |J|\vec{\nabla}\xi, \quad \vec{S}_\eta = |J|\vec{\nabla}\eta, \quad \vec{S}_\varsigma = |J|\vec{\nabla}\varsigma$.

Let the flux vector in the physical domain be $\vec{F} = (F,G,H)$. The transformed equations take the following form

$$\frac{\partial \tilde{Q}}{\partial t} + \vec{\nabla}^\xi \bullet \vec{\tilde{F}} = 0. \tag{3.2}$$

where $\tilde{Q} = |J| \cdot Q$, $\vec{\tilde{F}} = (\tilde{F},\tilde{G},\tilde{H}) \equiv (\vec{F}\bullet\vec{S}_\xi, \vec{F}\bullet\vec{S}_\eta, \vec{F}\bullet\vec{S}_\xi)$.

A. Discretization on the standard tetrahedron

On a standard tetrahedron, the CPR formulation can be expressed as

$$\frac{\partial \tilde{Q}_{i,j}}{\partial t} + \Pi_j\left(\nabla^\xi \bullet \vec{\tilde{F}}(\tilde{Q}_i)\right) + \frac{1}{|V_i^\xi|}\sum_{f\in\partial V}\sum_l \alpha_{j,f,l}[\tilde{F}^n]^\xi_{f,l} S^\xi_f = 0, \tag{3.3}$$

For the standard tetrahedron, $|V_i^\xi| = 1/6$, the areas for the four faces are 1/2, 1/2, 1/2 and $\sqrt{3}/2$ respectively. For the face on the plane $\xi = 0$ (denoted as face 1), the outgoing unit normal in the computational domain is $\vec{n}_1^\xi = (-1,0,0)$

$$\begin{aligned}[\tilde{F}^n]^\xi_{1,l} S^\xi_1 &= -(\tilde{F}_{com} - \tilde{F}(\tilde{Q}_i))_{1,l}\,\tfrac{1}{2} \\ &= \tfrac{1}{2}(F^n_{com} - F^n(Q_i))_{1,l}\left|\vec{S}_\xi\right|_{1,l} = \tfrac{1}{2}[F^n]_{1,l}\left|\vec{S}_\xi\right|_{1,l} \equiv \tfrac{1}{2}[F^n]_{1,l} S_{1,l}.\end{aligned} \tag{3.4}$$

A similar expression can be obtained for the other faces, with a properly defined $S_{f,l}$. For the diagonal face,

$$S_{f,l} = \left|\vec{S}_\xi + \vec{S}_\eta + \vec{S}_\varsigma\right|_{f,l}. \tag{3.5}$$

The final formulation can be written as

$$\frac{\partial Q_{i,j}}{\partial t} + \Pi_j\left(\nabla \bullet \vec{F}(Q_i)\right) + \frac{3}{|J|_{i,j}}\sum_{f\in\partial V}\sum_l \alpha_{j,f,l}[F^n]_{f,l} S_{f,l} = 0. \tag{3.6}$$

In 3D, to construct a complete polynomial of degree p, at least $p(p+1)(p+2)/3!$ SPs need to be chosen. In order to achieve the most efficient implementation, SPs on edges are chosen to be the Legendre-Gauss Lobatto (LGL) points. For 4th- ($p3$) or higher order schemes, nodes inside the boundary triangle are chosen from [10]. For 5th- ($p4$) or higher order schemes, nodes inside the tetrahedron are chosen from [32]. The nodal set of the 4th-order ($p3$) CPR scheme is shown in Fig. 2a. Note that the flux difference at a flux point corrects all solution points as shown in (3.6).

B. Discretization on the standard prism

For a standard triangular prism, the solution polynomial can be expressed as a tensor product of 1D and 2D Lagrange polynomials, i.e.,

$$\tilde{Q}_i(\xi,\eta,\zeta) = \sum_m \sum_j \tilde{Q}_{i;j,m} L_j^{SP}(\xi,\eta) L_m^{SP}(\zeta), \tag{3.7}$$

where $\tilde{Q}_{i;j,m}$ are the state variables at the solution point (j,m), with j the index in the ξ-η plane and m the index in ζ direction, $L_j^{SP}(\xi,\eta)$ is a 2D Lagrange polynomial based on the solution point in the base triangle and $L_m^{SP}(\zeta)$ is a 1D Lagrange polynomial based on the solution points in the prism height direction. Figure 2b shows the locations of the solution points for p=3. The nodal sets on the edge and the triangle are chosen in the same manner as the tetrahedral element.

The CPR formulation for a standard prism takes advantage of this tensor product basis, and is two dimensional in the ξ-η plane and one dimensional in ζ direction

$$\begin{aligned} &\frac{\partial \tilde{Q}_{i;j,m}}{\partial t} + \Pi_{j,m}\left(\nabla^{\xi} \bullet \vec{\tilde{F}}(\tilde{Q}_i)\right) + \frac{1}{\left|V_{Tri}^{\xi}\right|} \sum_{f \in \partial V_{Tri}} \sum_{l} \alpha_{j,f,l} [\tilde{F}^n(\xi_{f,l},\eta_{f,l},\zeta_m)] S_f^{\xi} \\ &- \left[\tilde{H}_{com}(\xi_j,\eta_j,-1) - \tilde{H}(\xi_j,\eta_j,-1)\right] \frac{\alpha_{L,m}}{2} \\ &+ \left[\tilde{H}_{com}(\xi_j,\eta_j,1) - \tilde{H}(\xi_j,\eta_j,1)\right] \frac{\alpha_{R,m}}{2} = 0. \end{aligned} \tag{3.8}$$

The correction process is done in a decoupled manner. The third term is the correction of the flux components in ξ and η direction, which is computed on a plane with fixed $\zeta = \zeta_m$. This is nothing but the correction used in the 2D CPR method for a triangular element. In Eq. (3.8), $\left|V_{Tri}^{\xi}\right|$ is the area of the base triangle, which is 1/2, S_f the length of the edge f of the base triangle, and l the index for flux points on f. Note that, $[\tilde{F}^n(\xi_{f,l},\eta_{f,l},\zeta_m)]$ corrects only the solution points on the triangle with fixed m instead of all solution points in the element as shown in Figure 2b. The

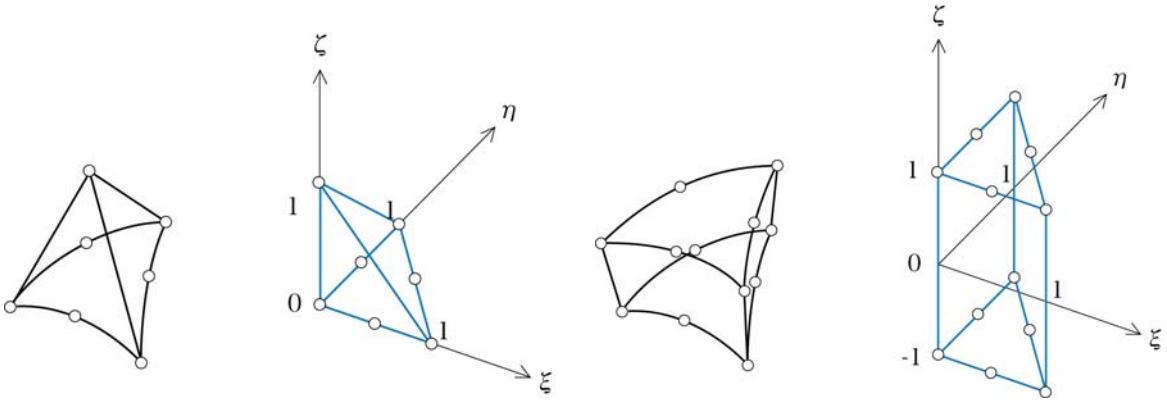
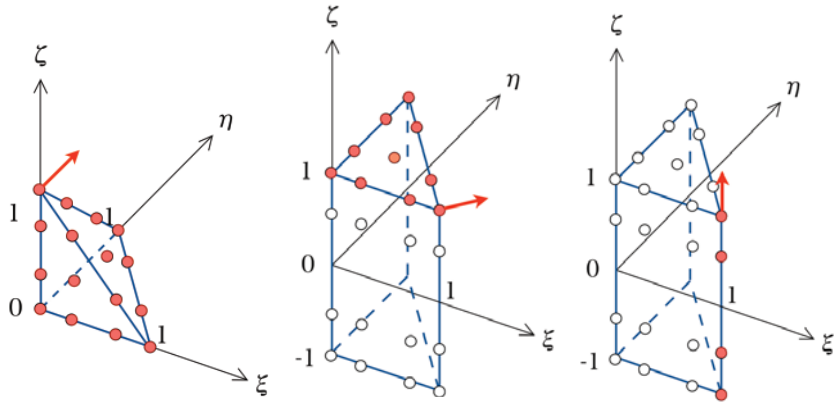


Figure 1. Transformations of a curve boundary tetrahedral and prismatic cell to the standard elements.

last two terms denote the correction in the ζ direction, which is evaluated by the 1D CPR method [11]. The flux difference at an end point corrects only the solution points on the segment with fixed j as shown in Figure 2c. For prism cells, the number of solution points corrected by a flux point is smaller than the one for tetrahedral cells due to the decoupled correction procedure. Hence, the method for prisms is more efficient per DOF than for tetrahedrons. This decoupled procedure also facilitates the implementation employing different degrees of polynomials in ξ-η and ζ directions to adapt to flow features. An attempt employing higher order polynomials in the wall normal direction to resolve the boundary layer with coarser prism cells is presented in [8].



(a) (b) (c)

Figure 2. Solution points in the standard tetrahedral and prism elements for p = 3 polynomial (only points on the visible faces are shown).

IV. Implicit Relaxation Solver

A. Block nonlinear LU-SGS solver

For the time integration, multi-stage Runge-Kutta schemes usually suffer from slow convergence to the steady state, especially for viscous problems in which grid points are clustered in the boundary layer. In order to circumvent the small time step restriction, we consider implicit solution methods. Applying the backward Euler differencing for the combined semi-discrete form of the governing equations gives

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta t} = R_i(Q^{n+1}), \quad (i = 1,\ldots N) \tag{4.1}$$

where R_i is the residual vector for all unknowns in the i-th element arising from the spatial discretization. To solve the nonlinear system for the n-th time step, usually the residual is linearized and an iterative linear solver such as an element Jacobi, a Gauss-Seidel or a Krylov subspace method (GMRES) is applied to the resulting sparse system. In this study, we employ a nonlinear LU-SGS method [25]. The final form of the method is written as follows for the i-th element,

$$\left(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i}\right)\left(Q_i^{k+1} - Q_i^*\right) = R_i(Q_i^*, Q_{nb}^*) - \frac{Q_i^* - Q_i^n}{\Delta t}, \quad (k = 1,\ldots,kmax) \tag{4.2}$$

where k denotes the index of the inner iterations between n-th and n+1-th time steps and * denotes the most recent update during the symmetric forward and backward sweeps over the global domain. In this study, Jacobian matrices are calculated using a numerical differencing and those are frozen to the value at the n-th time step. For each element, Eq. (4.2) is easily solved by the direct LU decomposition. If Eq. (4.2) is solved for the RHS to become zero for all elements, this is equivalent to solve the nonlinear system of Eq. (4.1) at the n-th step.

The size of the Jacobian matrix in the 3D Navier-Stokes equations is (5 × NDOFs in an element)^2. The memory requirement for the Jacobian matrices is not small, and it will not be affordable for k>3 polynomial approximations. In order to improve the computational efficiency, we freeze the Jacobian matrices for intervals of time steps 10-50.

The maximum number of the inner sweeps (*kmax*) is set to 3 in this study. To enable a computation with the impulsive start condition of freestream values, we use a ramping equation to gradually increase the CFL number.

$$CFL^{n+1} = \min\left(CFL^0 \times a^n, CFL_{\max}\right) \tag{4.3}$$

where α is a constant, which we set to 1.05 for usual cases. In practice, we typically use a value for CFL^0 of 1 and a value for $CFL_{\max}$ of about 100 to 10,000.

B. Line implicit solver

The nonlinear LU-SGS solver in the previous subsection is an elemental iterative method and is effective for isotropic meshes. However, it is known that its convergence performance degrades dramatically for highly anisotropic meshes commonly used in resolving thin boundary layers at high-Reynolds number flows. To alleviate the stiffness associated with stretched grids, we employ a line implicit solver which treat clusters of elements with strongest coupling implicitly.

We follow the algorithm described in [4] to construct lines of elements in a hybrid mesh. Since the anisotropic cells are used to resolve boundary layers, anisotropic regions will be found attached to the boundary faces on the solid surface. Lines connecting anisotropic cells are formed to originate from those boundary faces. Starting from one of those boundary faces, the face's normal vector that is orienting inside of the mesh is calculated and the cell attached to the boundary face is added to the line. To make a line solver effective, a line must propagate along the direction of strong coupling, which is assumed to be the wall normal direction in the boundary layer. By considering this, the next element to be added in the line is searched from the face sharing neighbors of the current (most recently added) cell. For the current cell, all the outgoing normal vectors of its faces are computed and the angles between the boundary face's normal vector and these faces' normal vectors on the cell are computed. If the minimum angle among those is less than a prescribed value θ, the neighboring cell across the corresponding face is added to the line and searched for the next cell. The above searching process is repeated until the computed minimum angle for the current cell is greater than θ degrees or the cell aspect ratio is less than 3. In this work, θ is chosen to be 30. This line creation is done for each boundary face on the solid surface.

After creating lines from the boundary faces, there can still be anisotropic regions in the mesh due to the physical phenomena such as wake regions or due to a meshing process employing the structured grid like topology. Although lines in such regions can be generated by using a more general approach [18], we don't employ the technique in this study.

Once the lines have been created the line-implicit solver is straight-forward to implement. Using the backward Euler time integration, the linearized equations for a line can be written as

$$\left(\frac{I}{\Delta t} - \frac{\partial R_i}{\partial Q_i}\right)\Delta Q_i - \frac{\partial R_i}{\partial Q_{n-}}\Delta Q_{n-} - \frac{\partial R_i}{\partial Q_{n+}}\Delta Q_{n+} = R_i(Q^n), \quad (i = 1,\ldots N_{line}) \tag{4.4}$$

where the subscript *i* denotes the local cell index in the line, *n*- and n+ denote the two neighbor cells and N_{line} is the total number of cells in the line. Thus, implicit lines form a block tri-diagonal matrix for each line. Each row of this block tri-diagonal matrix contains the block diagonal matrix of an element and one or two block off-diagonal matrices of neighboring elements in the line. The localized linear system for each line is solved directly using a block variant of the Thomas algorithm. This line implicit solver is used by coupled with the nonlinear LU-SGS solver in the previous section. That is, solution variables are updated for one cell or one line after another during the forward and backward sweeps over the cells and lines in the global mesh. In this approach, one or two more off-diagonal block Jacobian matrices need to be computed for each cell in the implicit lines. To save the computation time we store the Jacobian matrices and also freeze them for the intervals as with the LU-SGS scheme. Since this extra storage is required for only the region where lines are created, the cost will not be prohibitively expensive.

V. p-Multigrid Solver

While the standard iterative solvers such as the variants of element-Jacobi scheme or Gauss-Seidel scheme are quite effective in damping the non-smooth (high frequency) part of the error represented in a given mesh, they are not efficient in reducing the smooth (low frequency) part of the error. For this reason, the high frequencies of the error are smoothed out after a small number of iterations and the convergence rate deteriorates due to the persistent

low frequency error. This undesirable tendency becomes more prominent for fine meshes. Multigrid methods are well known as the effective remedy for this problem. In *h*-multigrid methods, a better convergence rate is obtained by adopting a sequence of progressively coarser grids, which facilitate an effective reduction of the solution error over the entire frequency field. The *p*-multigrid approach [22] is based on the same concept as the common *h*-multigrid but it makes use of "coarser" levels which are constructed by reducing the order of accuracy of the discretization rather than using physically coarser grids. Thus, all grid levels contain the same number of elements, which alleviates the need to perform complex interpolation between the multigrid levels and to implement agglomeration procedures.

1. Nonlinear formulation

Multigrid methods have been proven to be effective for accelerating convergence to steady state for both linear and nonlinear problems [26] and can be applied with many existing relaxation techniques (smoother). There are two ways of applying a multigrid method to the nonlinear set of equations. One is so called the Newton-multigrid, which uses the coarse grid correction (CGC) multigrid on the linearized problem obtained at each Newton iteration. Another is the full approximation storage (FAS) scheme, which applies multigrid directly to the nonlinear problem. In this study, we employ the FAS scheme [3] to solve the nonlinear governing equations.

In order to illustrate the two-level algorithm, let us consider a generic nonlinear problem $\mathbf{A}^p(\mathbf{u}^p) = \mathbf{f}^p$, where $\mathbf{u}^p$ is the discrete exact solution vector, $\mathbf{A}^p(\mathbf{u}^p)$ is the associated nonlinear algebraic operator and the superscript p indicates the level. Note the source term $\mathbf{f}^p$ at the fine level is 0 in the governing equations considered here. Let $\mathbf{v}^p$ be an approximation to the solution vector and define the discrete residual as $\mathbf{r}^p \equiv \mathbf{f}^p - \mathbf{A}^p(\mathbf{v}^p)$. Starting from the initial guess $\mathbf{v}_0^p$ on the fine level, the correction is performed according to the following steps:

- Iterate the fine level problem (update the approximate solution from $\mathbf{v}_0^p$ to $\mathbf{v}^p$) using any of the relaxation scheme and compute the residual:

$$\mathbf{A}^p(\mathbf{v}^p) = \mathbf{f}^p, \ \mathbf{r}^p = \mathbf{f}^p - \mathbf{A}^p(\mathbf{v}^p). \tag{5.1}$$

- Restrict the solution and the residual to the coarse level:

$$\mathbf{v}_0^{p-1} = \mathbf{I}_p^{p-1}\mathbf{v}^p, \ \mathbf{r}^{p-1} = \tilde{\mathbf{I}}_p^{p-1}\mathbf{r}^p, \tag{5.2}$$

  where $\mathbf{I}_p^{p-1}$ and $\tilde{\mathbf{I}}_p^{p-1}$ are the solution and the residual restriction operators from level p to level p-1, respectively.

- Compute the forcing term for the coarse level:

$$\mathbf{f}^{p-1} = \mathbf{A}^{p-1}(\mathbf{v}_0^{p-1}) + \mathbf{r}^{p-1}. \tag{5.3}$$

- Solve the coarse level problem (update the approximate solution from $\mathbf{v}_0^{p-1}$ to $\mathbf{v}^{p-1}$):

$$\mathbf{A}^{p-1}(\mathbf{v}^{p-1}) = \mathbf{f}^{p-1}. \tag{5.4}$$

- Calculate the coarse grid error:

$$\mathbf{e}^{p-1} = \mathbf{v}^{p-1} - \mathbf{v}_0^{p-1}. \tag{5.5}$$

- Prolongate the coarse grid error and correct the fine level approximation:

$$\mathbf{v}^p = \mathbf{v}^p + \mathbf{I}_{p-1}^p\mathbf{e}^{p-1}. \tag{5.6}$$

  where $\mathbf{I}_{p-1}^p$ is the error prolongation operator.

Note that the above multigrid formulation is applied to the steady Euler or Navier-Stokes equations in the form of $R(Q) = 0$, and hence there is no time derivative term. However, we still include the time step term in the LU-SGS

and the line implicit smoother to enhance the stability of the computation, though there is no physical meaning to keep the term during the iteration.

2. Transfer operators

Defining a prolongation operator is natural in the CPR method as in the case of spectral element methods. Let L_i^p denote the *i*-th Lagrange polynomial of order *p* in the discretization. Since the numerical approximation space are nested L_i^{p-1} can be expressed in terms of L_i^p

$$L_i^{p-1} = \sum_j \alpha_{ij}^{p-1} L_j^p . \tag{5.7}$$

Thus, the prolongation operator can be defined as

$$\mathbf{I}_{p-1}^p = (\alpha^{p-1})^T . \tag{5.8}$$

Defining a restriction operator is less straightforward. In this study, an L_2 projection was chosen for both the state and residual restriction operators, giving the following definition:

$$\mathbf{I}_p^{p-1} = (\mathbf{M}^{p-1})^{-1}\mathbf{N}^{p-1}, \text{ where } \mathbf{M}_{k,i}^{p-1} = \int_\Omega L_k^{p-1} L_i^{p-1} d\Omega, \ \mathbf{N}_{k,j}^{p-1} = \int_\Omega L_k^{p-1} L_j^p d\Omega. \tag{5.9}$$

3. Cycling strategy

In the two-level correction scheme, the coarse level problem is not much different from the original problem. To solve the coarse level problem efficiently, the two-level correction scheme can be applied recursively by moving to successively coarser levels. For *p*-multigrid methods, the recursive application of lower order discretization ends with the *p*=0 on the same grid as the fine level. A simple strategy is to use a saw-tooth V-cycle over all *p* levels. In this cycle, a fixed (small) number of smoothing iterations is performed on each approximation level before the restriction on the next coarse level (pre-smoothing) and the smoothing after fine level prolongations (post-smoothing) are deleted. A further increase in efficiency and robustness can be achieved by augment the solution process with a full multigrid (FMG) technique [3]. In FMG, a better initial solution is provided for the fine level by solving (usually approximately) the coarse level problem in advance. We chose the *p*=0 approximation as the coarsest level of FMG and use 2 V-cycles per level.

4. Hybrid smoothing strategy

Usually the same smoother is used for the all *p*-levels, though different smoothers can be used at different levels. It is known that using an explicit Runge-Kutta smoother for all *p*-level is not effective for the high-order DG discretization [17]. Using an implicit relaxation scheme at all levels will provide fast convergence rate, but the memory requirement for the implicit Jacobian matrices becomes rapidly huge as increasing the *p* approximation, especially for 3D problems. One approach considered in the literature is to use an implicit relaxation scheme at lower levels and use an explicit scheme at higher levels. Since the memory storage for the lower *p* approximations (*p*0 to *p*1) is small, substantial reduction of memory is expected. In this work, a hybrid smoothing strategy 'pMG(ERK-LUSGS)' using the three-stage explicit Runge-Kutta (ERK) scheme at the highest level and the implicit LU-SGS scheme in the section V is employed besides the ordinal approach 'pMG(LUSGS)' using the implicit LU-SGS smoother at all levels. In both strategies, 10 LU-SGS smoothing iterations are performed at the lowest *p*=0 level and 5 LU-SGS iterations are performed at the higher levels for one cycle. In the 'pMG(ERK-LUSGS)' strategy, 10 ERK iterations are performed at the highest level. These numbers of smoothing iteration were found empirically to minimize the overall computational time in the results section.

VI. Numerical Results

A. Inviscid flow over a sphere

A steady subsonic flow around a sphere is computed by solving the compressible Euler equations. The freestream Mach number is *M*=0.3. A computational mixed grid is generated around a quarter sphere considering

the two symmetric planes in the flowfield. The mixed grid is composed of five layers of prismatic cells around the quarter sphere and isotropic tetrahedral cells for the remaining region. To preserve the geometry of the sphere well, the curved wall boundaries are represented by quadratic polynomials. The computational grid and the computed density contours using the p=3 approximation are shown in Fig. 3. The computed flowfield is perfectly symmetric without visible numerical dissipation even on the relatively coarse grid.

The performance of the developed p-multigrid solver is presented in the following. Figure 4 shows the convergence history of the residual plotted versus CPU time for the $p1$ to $p3$ approximations. As a base line case, the nonlinear LU-SGS scheme is used as a single grid solver. Two different smoothing strategies introduced in the previous section are compared; 'pMG(LUSGS)' and 'pMG(ERK+LUSGS)'. It is shown that 'pMG(LUSGS)' strategy is the fastest and achieves a computational time reduction of about 40-60% with respect to the single grid case. As the approximation level p is increased, the performance of the 'pMG(LU-SGS)' strategy is improved. This will be due to the benefit of the increased total number of levels, even though the cost passing through one multigrid cycle increases too. As for 'pMG(ERK+LUSGS)' strategy, the speed-up against for the single grid solver is small in the $p2$ and $p3$ approximations and there is no improvement in the $p1$ case. However, in this approach the large memory storage for the highest level's implicit matrices can be saved.

The p-multigrid can also performed by skipping levels. Figure 5(a) shows the residual versus the number of multigrid cycles using all p-levels ($p2$-$p1$-$p0$) with 'pMG(LUSGS)', skipping $p1$ (using two levels of $p2$-$p0$) with 'pMG(LUSGS)' and skipping $p1$ with 'pMG(ERK+LUSGS)'. While the total number of cycles needed to converge to the machine zero increases by skipping the level, there is a benefit to reduce the computational time for the level. However, Fig. 5(b) shows there is no improvement and even worse results in terms of CPU time. Even though there is a loss of the efficiency, skipping levels may be worth to be considered for memory constraint applications.

B. Viscous flow over NACA 0012 airfoil

We examine the performance of the developed line solver in simulating a laminar flow around the NACA0012 airfoil. We assume the freestream Mach number of M=0.5, the Reynolds number of Re=5000 (based on the free stream velocity and the airfoil chord length) and zero angle of attack.

3D prism meshes are generated to compute the two dimensional flowfield. First a 2D O-type structure grid is generated around the airfoil and it is extruded in the span-wise direction by the length of 10% of the chord. Finally, the hexahedral cells are divided into prisms. The base grid has 72x24x2 cells with the maximum aspect ratio of ~10. In addition, two grids with higher aspect ratio cells are generated by changing the initial wall spacing and the number of cells along grid lines normal to the wall. These grids with the created lines are shown in Fig. 6. The maximum aspect ratio for the medium and the fine grid are ~50 and ~500, respectively. In the fine grid with large aspect ratio cells, we observed that high-order boundary treatment only on the faces attached to the curved solid surface near the leading edge caused bad shaped cells and instability in the computation. To remedy this problem, a high-order meshing of the prism layers is devised and implemented. The close looks of the resulting meshes are shown in Fig. 7.

The efficiency of the line implicit solver is confirmed compared with the point (elemental) nonlinear LU-SGS solver on the coarse mesh (Fig. 8 (a)-(d)). It is shown that the line implicit solver yields better convergence in both the

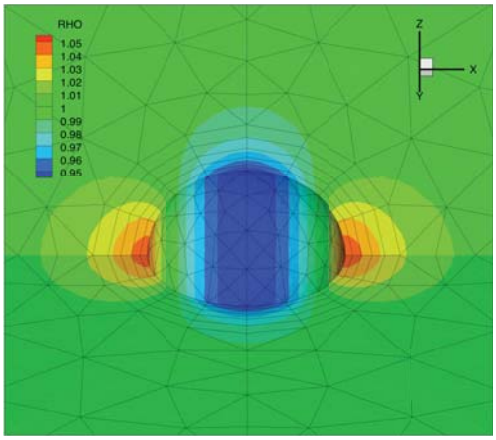


Figure 3. Computed density contours for the inviscid subsonic flow over a sphere using the p=3 approximation on the mixed mesh.

number of iterations and the total CPU time compared to the point implicit solver on this relatively small aspect ratio mesh.

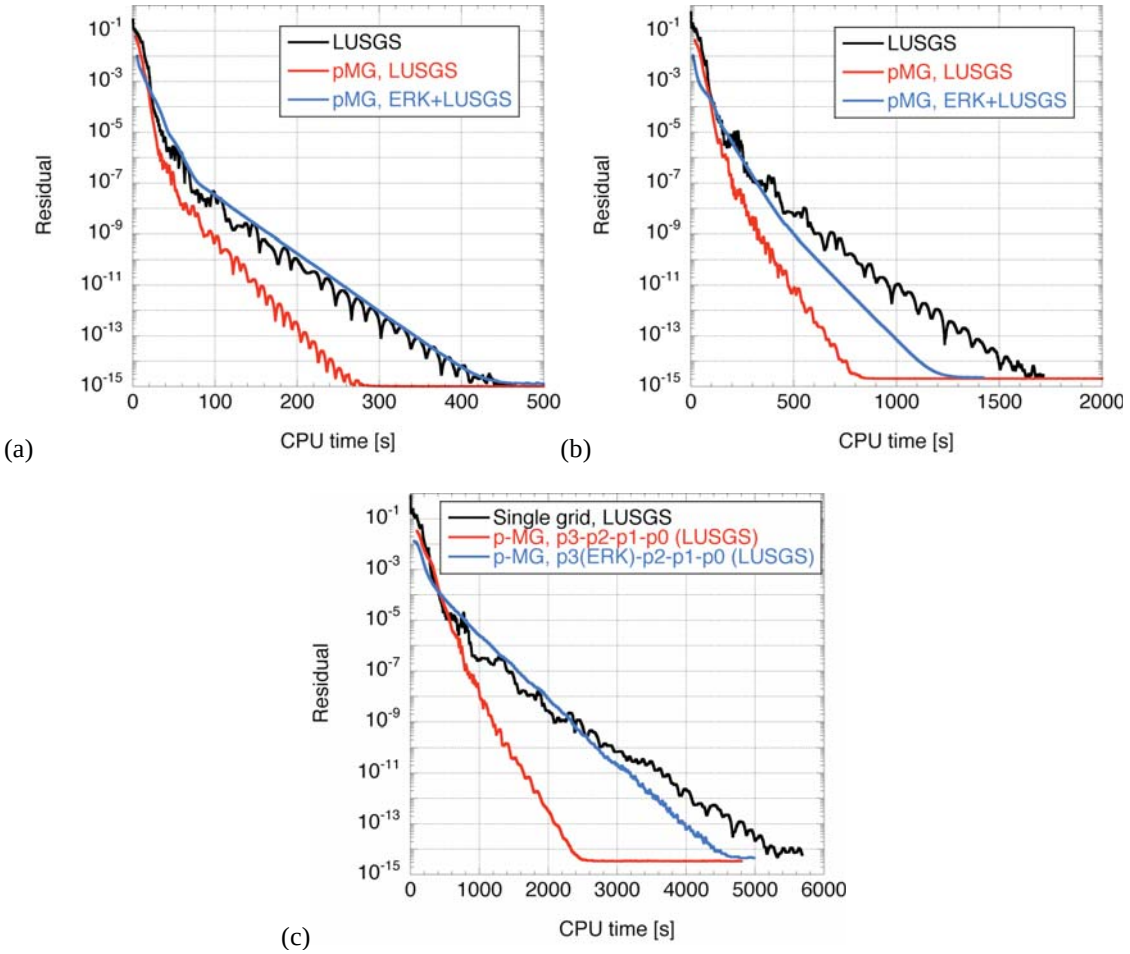


Figure 4. Efficiency comparison of the LUSGS single grid solver, *p*-multigrid solver with (LUSGS) smoother and p-multigrid solver with (ERK+LUSGS) smoother. The residual vs. the CPU time in solving the sphere problem for *p*=1-3 in (a)-(c).

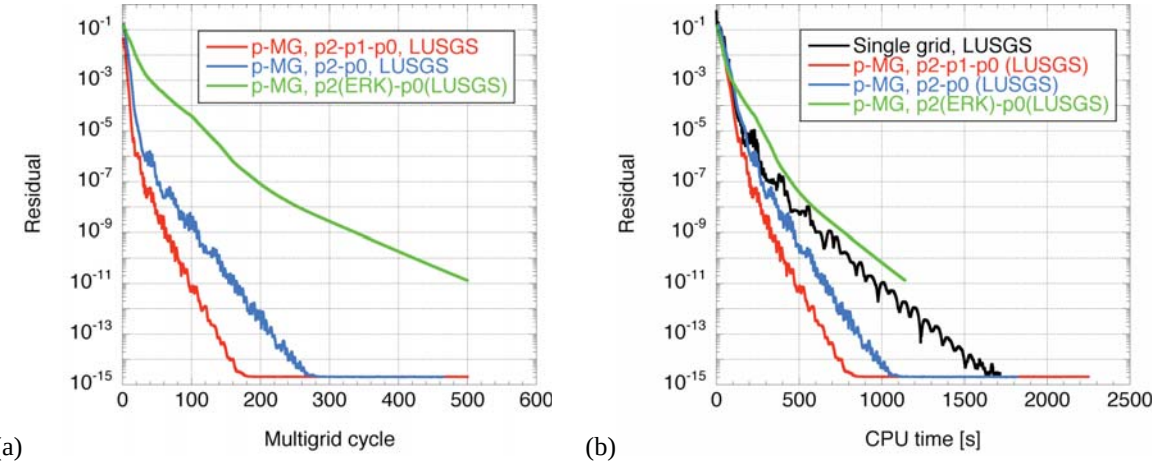


Figure 5. Comparison of the convergence of *p*-multigrid by skipping *p*1 level with (LUSGS) smoother and with (ERK+LUSGS) smoother. The residual vs. the multigrid cycles (a) and the CPU time (b) in solving the sphere case.

Now we examine the aspect ratio dependence of the line implicit solver using the three different meshes. In all the cases, the time step size was increased from the initial small value (1e-5 to 1e-3) to the prescribed maximum value of 0.5. Figure 9 (a)-(c) demonstrate the convergence history as a function of the number of time steps for p=1, 2, 3. For the highest aspect ratio case, the line implicit solver shows significant improvement compared with the elemental implicit solver. And nearly aspect ratio independent convergence is obtained for every p approximations.

A main drawback of the line implicit solver is the large memory requirement to store the Jacobian matrices not only for the block diagonal matrix of each element but also off diagonal matrices of neighbor elements in the line. To save the cost, a possible way is to use the line implicit solver as a smoother at coarser levels of the p-multigrid method.

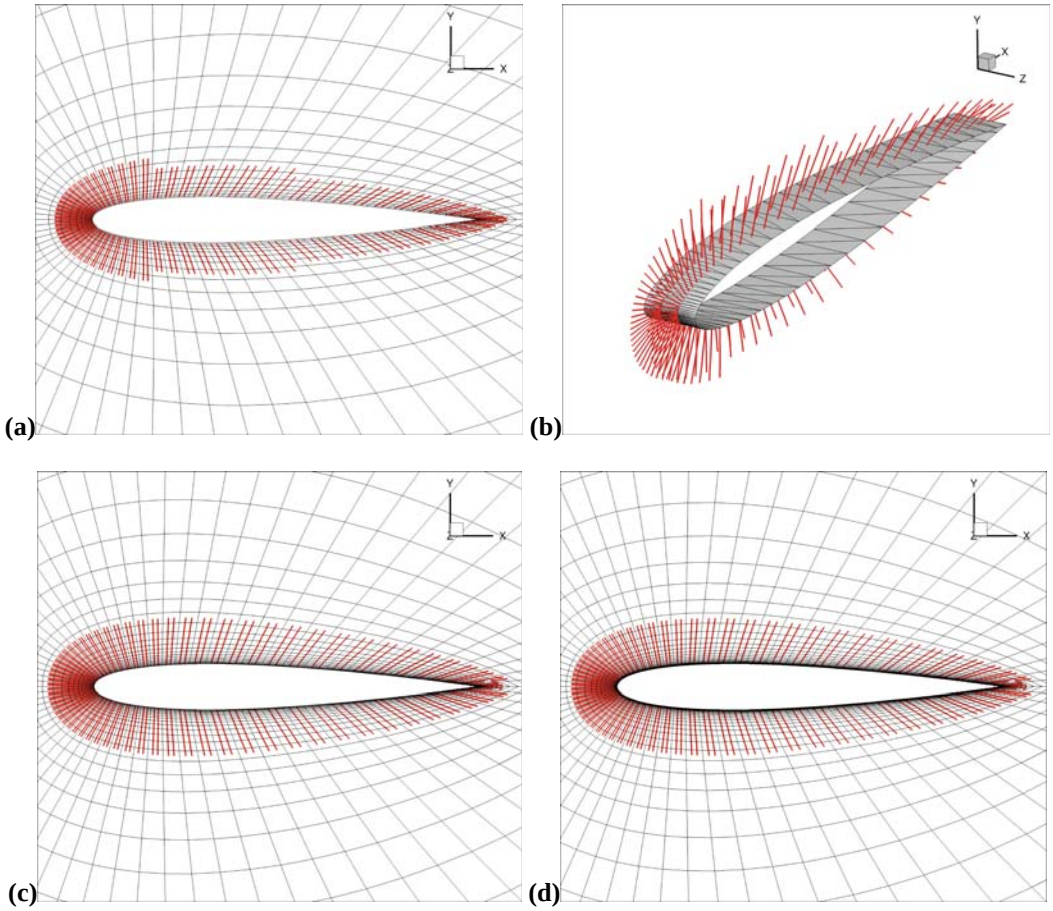


Figure 6. Computational grids with created lines for the NACA 0012 problem; (a) coarse grid (72x24x2 cells), (b) a perspective view of the coarse grid, (c) medium grid (72x34x2 cells), (d) fine grid (72x44x2 cells).

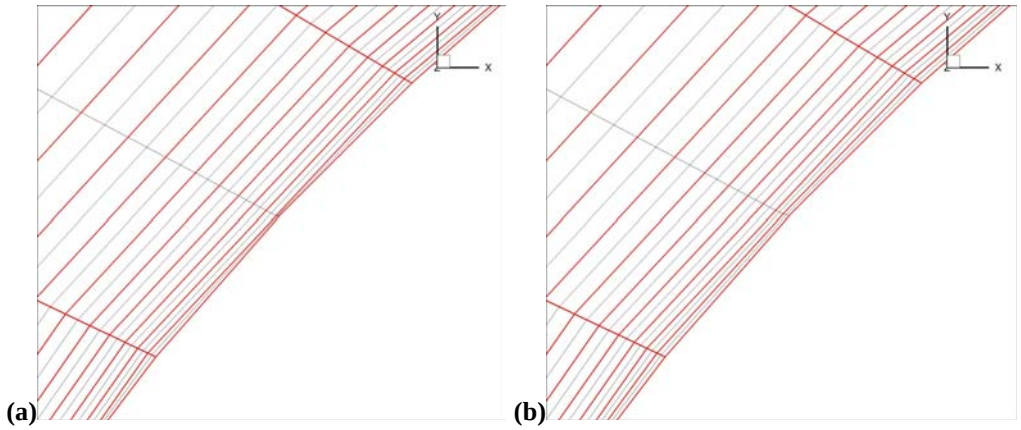


Figure 7. High-order boundary elements generated near the leading edge of the NACA 0012 airfoil; (a) only the boundary faces are modified to fit the curved solid surface. (b) Some of the prism cells are modified to maintain better shape near the curved surface.

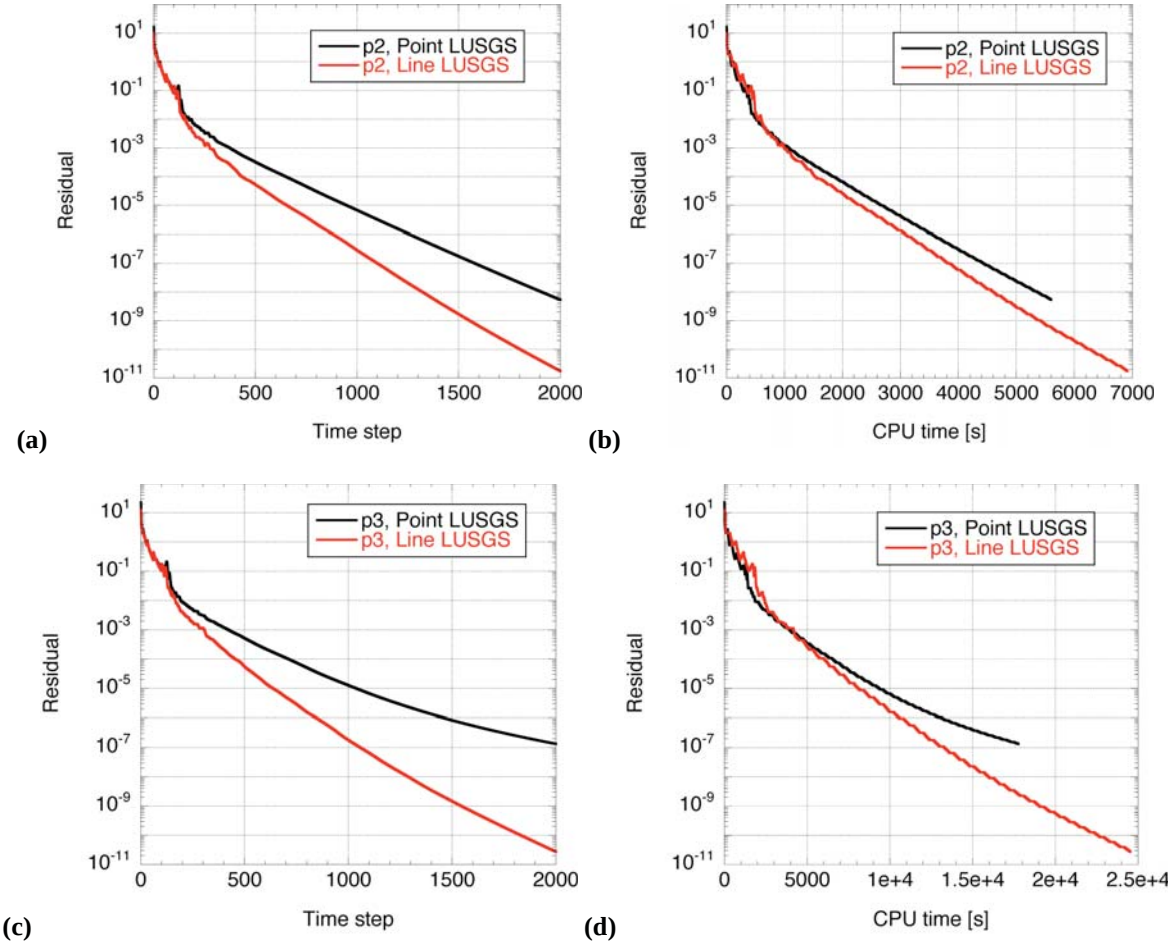


Figure 8. Efficiency comparison of the line implicit solver and the point (elemental) nonlinear LU-SGS solver for the NACA 0012 problem on the coarse mesh. The residual vs. the time step and the CPU time for the *p*=2 approximation (a, b) and the *p*=3 approximation (c, d).

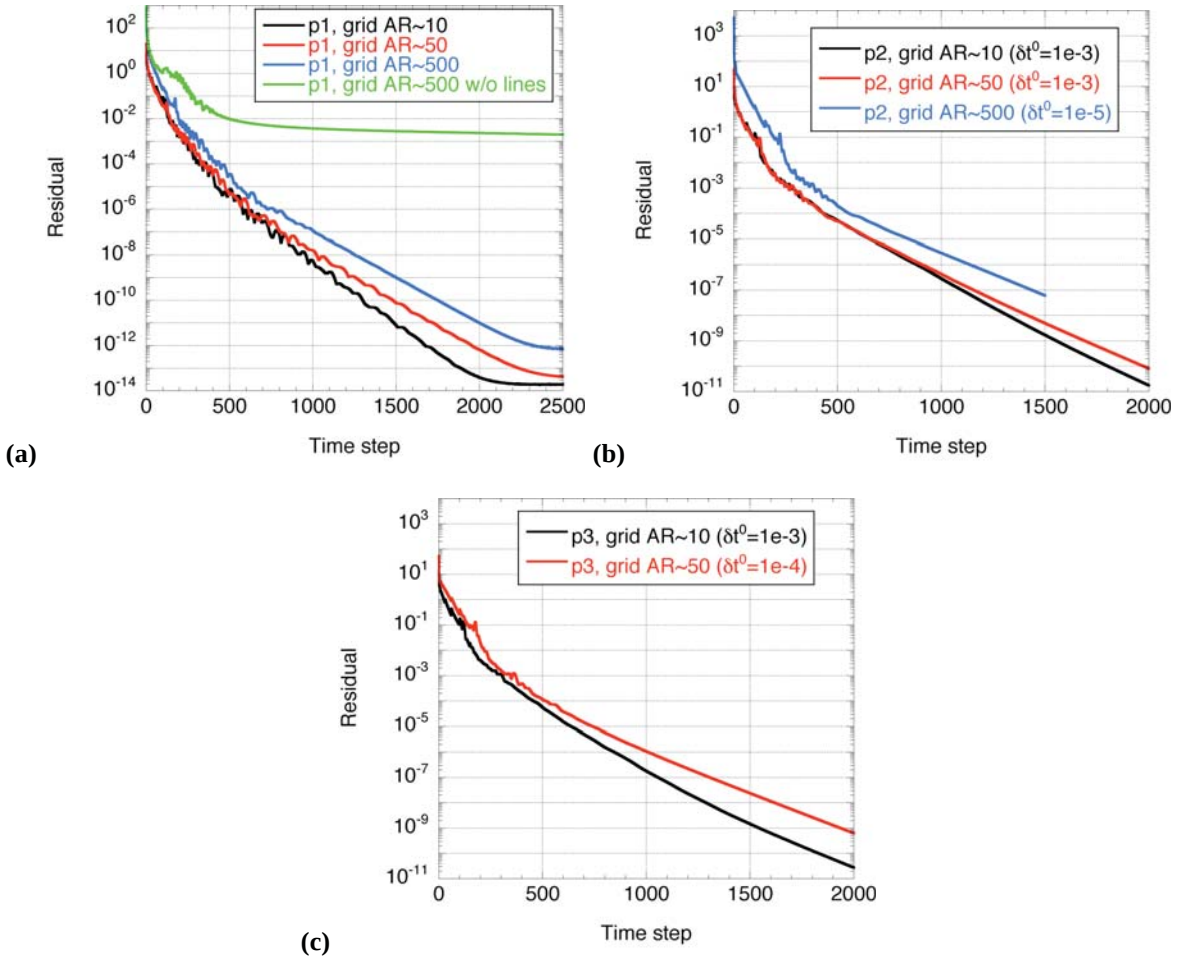


Figure 9. Aspect ratio dependence of the line implicit solver for the NACA 0012 problem. The residual vs. the time step for the approximation orders p=1, 2, and 3 in (a), (b), and (c), respectively.

C. Flow over a streamlined 3D body

The last test case is the computation of subsonic flow around a streamlined 3D body considered in the European ADIGMA project where it is called "BTC0" test case. The geometry is based on a 10% thick airfoil with boundaries constructed by a surface of revolution. The airfoil is defined analytically by an elliptical leading edge and straight lines. The computational grid generated using anisotropic prisms and isotropic tetrehedrons is shown in Fig. 10. The surface of the body is approximately treated by piecewise quadratic surface elements. The curved surfaces are reconstructed based on the ordinal grid with straight edges. The total number of the cells is 160,544 (149,152 tetrahedrons and 11,392 prisms). In this work, the mesh is decomposed into 4 sub-domains using a graph partitioning algorithm and parallel computations based on the MPI library are carried out on a Dell's PC cluster.

Preliminary computation of the inviscid flow of the freestream Mach number M=0.5 and zero angle of attack is performed here. The computed Mach contours plot using the $p1$ approximation is shown in Fig. 11. The convergence histories using the single grid elemental LU-SGS solver, the single grid line implicit solver and the p-multigrid with line implicit LU-SGS smoother are compared in Fig. 12. All the cases start from the initial CFL number of 1. In the elemental LU-SGS case, the CFL number cannot be increased to more than 10, while the other two cases using implicit lines are able to use much larger CFL number up to 10,000. In the p-multigrid case, the machine zero convergence was obtained in about 60 multigrid cycles.

VII. Conclusion

Efficient solution strategies for steady-state inviscid and viscous flows using high-order CPR discretizations have been developed. A line implicit solver have been devised and implemented to enable efficient solution techniques on anisotropic meshes. The numerical test using different aspect ratio meshes shows the line solver can provide significantly better performance for the high aspect ratio mesh and also its convergence rate has less dependence on the mesh aspect ratio. The *p*-multigrid solver with different smoothing strategies also has been developed. The performance test shows the *p*-multigrid with an implicit LU-SGS smoother achieves about twofold speed-up compared to the LU-SGS single grid solver. Hybrid smoothing strategy employing an explicit Runge-Kutta smooter at the highest level show the similar or slightly better performance compared to the single grid solver with reduction of the memory storage for the implicit matrices. Future work will concentrate on the development of low-storage solution algorithms, *hp*-multigrid algorithms and *hp*-adaptation techniques.

Acknowledgments

This study has been supported by the Air Force Office of Scientific Research (AFOSR). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFOSR.

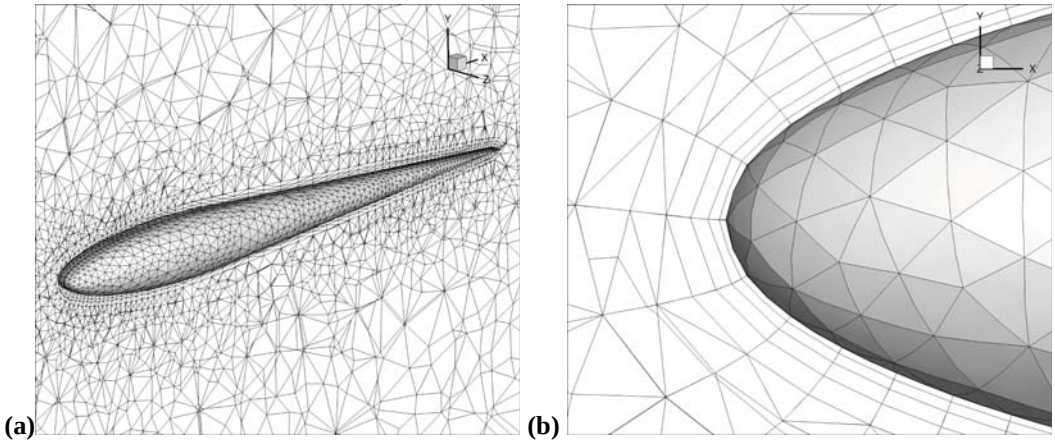


Figure 10. The computational grid for a streamlined 3D body. (a): a perspective view and (b): a close-up view near the leading edge. The surface geometry is approximated by quadratic boundary faces.

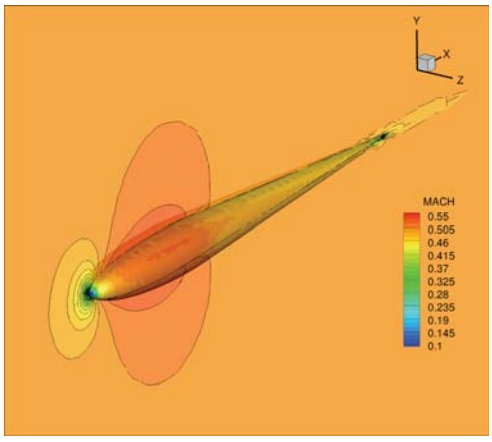


Figure 11. Computed Mach number contours around the streamlined body using the *p*1 approximation.

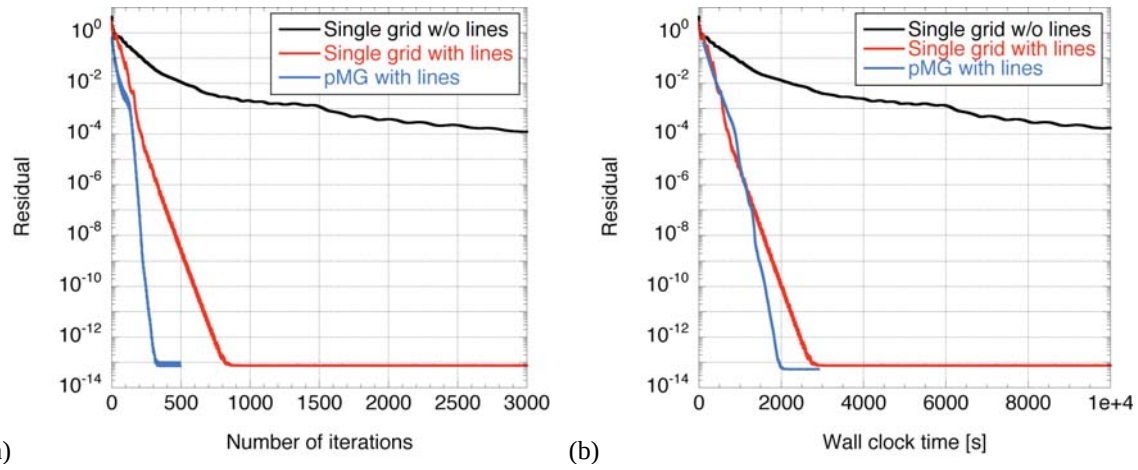


Figure 12. Convergence histories of the streamlined 3D body case using the *p*1 approximation for the single grid LU-SGS solver without lines, the single grid line implicit solver and the *p*-multigrid with line implicit LU-SGS smoother. (a) The residual vs. the number of iterations (for the *p*-multigrid case, the number of iterations at the *p*1 level is shown). (b) The residual vs. the wall clock time of the parallel computation.

References

[1] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, *J. Comput. Phys*. **131 (1)** (1997) 267–279.

[2] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations, in Lecture Note in Computational Science and Engineering 11, Springer Verlag, New York, (2000), 197-208.

[3] Briggs WL, Henson VE, MacCormick SF, A multigrid Tutorial (2nd edn). SIAM: Philadelphia, PA, 2000.

[4] N.K. Burgess, C.R. Nastase, D.J. Mavriplis, Efficient solution techniques for discontinuous Galerkin discretizations of the Naver-Stokes equations on hybrid anisotropic meshes, AIAA Paper 2010-1448.

[5] B. Cockburn, C.-W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework, *Math. Comput.* **52** (1989) 411–435.

[6] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys*. **207** (2005), 92-113.

[7] H. Gao, Z.J.Wang, A high-order lifting collocation penalty formulation for the Navier-Stokes equations on 2D mixed grids, AIAA Paper 2009-3784.

[8] T. Haga, H. Gao, Z.J.Wang, A high-order unifying discontinuous formulation for 3D mixed, AIAA Paper 2010-540.

[9] T. Haga, K. Sawada, Z.J. Wang, An implicit LU-SGS scheme for the spectral volume method on unstructured tetrahedral grids, *Commun. Comput. Phys*. **6(5)**, pp. 978-996 (2009).

[10] J.S. Hesthaven, From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex, *SIAM J. Numer. Anal*. **35 (2)** (1998) 655–676.

[11] H.T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods, AIAA Paper 2007-4079.

[12] H.T. Huynh, A Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin for Diffusion, AIAA Paper 2009-403.

[13] A. Jameson, Analysis and design of numerical schemes for gas dynamics. I. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence, *Int. J. Comput. Fluid Dyn*. **4** (1994) 171–218.

[14] D.A. Kopriva, J.H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, *J. Comput. Phys*. **125** (1996) 244.

[15] M.-S. Liou, A sequel to AUSM, Part II: AUSM+-up for all speeds, *J. Comput. Phys.* **214** (2006) 137–170.

[16] Y. Liu, M. Vinokur, Z.J. Wang, Discontinuous spectral difference method for conservation laws on unstructured grids, J. Comput. Phys. 216 (2006) 780– 801.

[17] H. Luo, J. D. Baum, R. Lohner, A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids, *J. Comput. Phys*. **211** (2006) pp767-783.

[18] D.J. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, *J. Comput. Phys.* **145** (1998) 141–165.

[19] C.R. Nastase, D.J. Mavriplis, High-order discontinuous Galerkin methods using an hp-multigrid approach, *J. Comput. Phys*. 2**13** (2006) 330–357.

[20] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation, Los Alamos Scientific Laboratory Report, LA-UR-73-479, 1973.

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

[22] E.M. Ronquist, A.T. Patera, Spectral element multigrid. I. formulation and numerical results. *SIAM J. Sci. Comput.* **2(4)** (1987) 389-406

[23] V.V. Rusanov, Calculation of interaction of non-steady shock waves with obstacles, *J. Comput. Math. Phys*. USSR 1 (1961) 261–279.

[24] K. Shahbazi, D.J. Mavriplis, N.K. Burgess, Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, *J. Comput. Phys*. **228** (1009) 7917–7940.

[25] Sun, Y., Wang, Z. J., and Liu, Y., "Efficient implicit non-linear LU-SGS approach for compressible flow computation using high-order spectral difference method," *Commun. Comput. Phys*. **5**, 2009, pp. 760–778.

[26] Trottenberg U, Oosterlle CW. Schuller A. Multigrid. Academic Press: London, U.K., 2001.

[27] K. Van den Abeele, C. Lacor, An accuracy and stability study of the 2D spectral volume method, *J. Comput. Phys.* **226** (1) (2007) 1007–1026.

[28] K. Van den Abeele, C. Lacor, Z.J. Wang, On the stability and accuracy of the spectral difference method, *J. Sci. Comput*. **37 (2)** (2008) 162–188.

[29] Z.J. Wang, High-order methods for the Euler and Navier–Stokes equations on unstructured grids, *J. Prog. Aerosp. Sci.* **43** (2007) 1–47.

[30] Z.J.Wang and H. Gao, A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids, *J. Comput. Phys*. **228** (2009) 8161-8186.

[31] Z.J. Wang, Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids II: extension to two-dimensional scalar equation, *J. Comput. Phys*. **179** (2002) 665–697.

[32] T. Warburton, An explicit construction of interpolation nodes on the simplex, *J. Eng. Math.* **56** (2006) 247–262.

[33] O.C. Zienkiewicz, R.L. Taylor, The Finite Element Method The Basics, vol. 1, Butterworth–Heinemann, Oxford, England, 2000.