Updates to Implicit Edge-Based Gradient Methods

Hiroaki Nishikawa*

National Institute of Aerospace, Hampton, VA 23666, USA

In this paper, we report updates to the implicit edge-based gradient methods originally introduced in [H. Nishikawa, AIAA Paper 2020-3048, 2020]. First, we clarify the relationship between gradient accuracy and truncation error, and show that the quadratic method involves a free parameter. Then, we provide a complete description and a simplified matrix form of the implicit gradient systems including a consistent boundary treatment, and derive a set of parameters for achieving fourth-order gradient accuracy on regular tetrahedral grids. A stability analysis is performed for a relaxation scheme used to solve the implicit gradient systems, and the result serves as a guide for choosing parameters. Numerical results are shown for accuracy verification and also for realistic inviscid flow problems in three dimensions, including flows with shock waves.

I. Introduction

The edge-based method is one of the most efficient discretization algorithms widely used in practical computational fluid dynamics (CFD) codes [1–12], where a residual is computed efficiently in a loop over edges followed by a loop over boundary elements. One of the most notable features of the edge-based method is that it achieves third-order accuracy on arbitrary simplex-element grids with a linear flux reconstruction and a quadratically-exact gradient without requiring computations of second derivatives at all [13–15]. Moreover, it does not require high-order curved grids even for curved geometries as demonstrated in Refs. [16,17]. Therefore, it is directly applicable to many practical grids currently in use for second-order finite-volume solvers. Despite advantages, the third-order edge-based scheme has not yet been employed routinely for practical applications because of robustness issues. First, as well known, an implicit defect-correction solver tends to get less stable if more accurate gradients [18], e.g., weighted least-squares (LSQ) methods, are used in the solution reconstructions for inviscid fluxes. Quadratic LSQ gradients are usually more accurate than linear LSQ gradients, and therefore a third-order edge-based solver tends to be less stable, especially on very thin and highly curved grids [13,15]. To alleviate the problem, we may employ an approximate mapping technique as mentioned in Ref. [15], or simply make the quadratic LSQ gradients less accurate by constructing larger LSQ stencils with neighbors of neighbors [16,19] (a larger gradient stencil is known to stabilize a solver [20]). The latter is a simple and effective approach, but it could deteriorate solution accuracy and also increases the cost of gradient computations. Moreover, it cannot be easily implemented in an efficient edge-loop, which is possible for linear LSQ methods involving only the edge-connected neighbors. While some compromise may be found by a weighted LSQ method with carefully chosen weights, e.g., the inverse distance to the power of 0.25, a more effective technique would be desired to overcome these conflicting requirements and take a full advantage of the economical third-order edge-based method.

As a superior alternative to LSQ gradients, implicit gradient methods have recently been developed and demonstrated in both low- and high-order reconstruction-based discretization methods [21–26]. In the implicit gradient methods, gradients are computed as a solution to a globally-coupled linear system and thus the gradient stencil virtually extends over a whole domain, possibly alleviating the stability problem yet improving gradient accuracy via coupling with neighbors. Indeed, as demonstrated in Refs. [24, 26], implicitly-computed gradients are significantly more accurate than explicit LSQ gradients, and even higher-order accuracy can be achieved on regular grids: fourth-order accuracy on a regular quadrilateral grid [24], and third/fourth-order accuracy on a regular triangular grid [26]. Although implicit gradient methods have been actively studied for cell-centered discretization methods, they have not yet been studied extensively for the node-centered edge-based method.

*Research Fellow (hiro@nianet.org), Associate Fellow AIAA

It is the objective of this paper to investigate the use of implicit gradient methods for the edge-based method. In particular, we are interested in the implicit edge-based gradient (IEBG) methods originally introduced in Ref. [26] and its applications to the third-order edge-based scheme. This method has a special property of achieving quadratic exactness, i.e., second-order gradient accuracy, on arbitrary simplex-element grids without increasing the number of unknowns, in contrast to other methods that require second derivatives to be added as additional unknowns for quadratic exactness. Therefore, in the edge-based method, the implicit gradient system consists of 3×3 blocks for both linear and quadratic methods. This is especially attractive to the third-order edge-based method, which is also a second-derivative-free algorithm.

Ref. [26] gives the basic idea behind the IEBG methods, but the formulation is incomplete, which has caused some confusion. First, the relationship between the gradient accuracy and the truncation error was not well understood and it was thought that there was no free parameter in the quadratic IEBG method. However, there is actually a free parameter and it can be used to increase robustness without losing second-order accuracy in the gradients. This is clarified in this paper. Another point is the boundary treatment. In Ref. [26], a standard second-order boundary-closure formula [27] or a general accuracy-preserving formula [17] with a ghost node were considered without any discussions on the best practice and any relation to the matrix of the implicit gradient systems. In this paper, we will show that the general accuracy-preserving formula can be used for both the linear and quadratic IEBG methods, leading to a great simplification in the diagonal blocks of the IEBG systems. Also, in this paper, we will focus on three dimensions while Ref. [26] focuses on two dimensions, and derive a set of optimal parameters to achieve fourth-order gradient accuracy on a regular tetrahedral grid. Finally, we will provide a practical matrix form of the IEBG systems, which can be more directly implemented in a code than those presented in Ref. [26].

The paper is organized as follows. In Section 2, we provide a brief derivation of the basic IEBG systems and derive a boundary-closure formula by using an edge-collapsing technique. In Section 3, we provide accuracy proofs for the IEBG methods. In particular, we will show that the gradient accuracy matches the truncation error of the corresponding implicit gradient system for irregular grids. Also, we derive a set of optimal parameters for achieving higher-order gradient accuracy on a regular tetrahedral grid. In Section 4, we derive a matrix form of the IEBG methods and in particular, show that the diagonal blocks reduce to diagonal matrices. In Section 5, we describe a relaxation scheme for solving the implicit gradient systems and present a stability analysis result. In Section 6, we present numerical results. In Section 7, finally, we will conclude the paper with remarks.

II. Linear and Quadratic Implicit Edge-Based Gradient Methods

In this section, following Ref. [24], we derive linear and quadratic IEBG systems from second- and third-order edge-based discretizations of a hyperbolic diffusion system, respectively. As we will show, it only takes a minor modification to upgrade the linear IEBG system to the quadratic IEBG system. A suitable boundary flux quadrature formula is also described; the same formula is used for both systems, which leads to a great simplification in the final implicit matrix as we will show later.

A. Derivation

As originally introduced in Ref. [24] and applied to the edge-based discretization in Ref. [26], the basic idea is to derive a system for gradients from a discretization of a hyperbolic diffusion system [28]. Consider the Laplace equation:

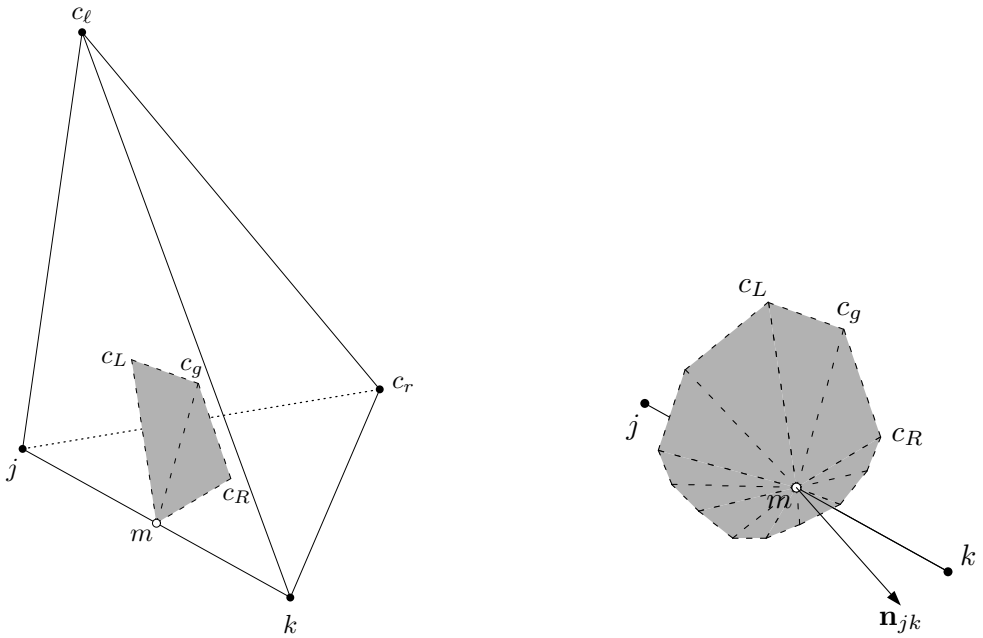
$$\operatorname{div}(\operatorname{grad} u) = 0, \tag{1}$$

where u is a solution variable. Adding pseudo-time terms, we write it as a first-order system:

$$\partial_\tau u = \operatorname{div} \mathbf{g}, \quad \partial_\tau \mathbf{g} = \frac{1}{T_r}\left[(\operatorname{grad} u)^t - \mathbf{g}\right], \tag{2}$$

where τ is a pseudotime variable, $T_r = L_r^2$ is a relaxation time, L_r is a length scale, $(\operatorname{grad} u)^t$ is a column vector of the gradient of u, and $\mathbf{g}$ is a column vector of gradient variables. The system can be written in the vector form:

$$\mathbf{P}^{-1}\partial_\tau \mathbf{U} + \operatorname{div} \mathcal{F} = \mathbf{S}, \tag{3}$$



(a) Dual face contribution from a tetrahedron. (b) Sum of the dual face contributions.

Figure 1: Dual face contribution (two shaded triangular faces) from a tetrahedron and the total dual face at the edge between j and k. The point m is the midpoint of the edge. The point c_g is the geometric centroid of the tetrahedron, and c_L and c_R are the geometric centroids of the triangular faces $[j, k, c_\ell]$ and $[j, c_r, k]$, respectively. The directed area vector $\mathbf{n}_{jk}$ is the sum of the directed area vectors of the dual faces (shaded triangles) pointing from j to k.

where

$$\mathbf{P}^{-1} = \mathrm{diag}(1, T_r, T_r, T_r), \quad \mathbf{U} = \begin{bmatrix} u \\ \mathbf{g} \end{bmatrix}, \quad \mathcal{F} = \begin{bmatrix} -\mathbf{g}^t \\ -u\mathbf{I} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 0 \\ -\mathbf{g} \end{bmatrix}, \tag{4}$$

the superscript t denotes transpose, $\mathbf{I}$ is the 3×3 identity matrix. The system is hyperbolic in the psuedotime τ and equivalent to the original Laplace equation (1) if $\partial_\tau u = \partial_\tau \mathbf{g} = 0$. It allows us to derive a consistent discretization of the Laplace equation by discretizing the first-order hyperbolic system by upwind schemes and then dropping the pseudotime derivatives. Then, the vector $\mathbf{g}$ corresponds to the gradient of u: $\mathbf{g} = \nabla u$, and it is called the gradient variable.

For an arbitrary tetrahedral grid, we store the values of u and $\mathbf{g}$ at nodes and construct the edge-based discretization of the system (3) at a node j as

$$\mathbf{Res}_j^{EB} = \frac{1}{V_j} \sum_{k \in \{k_j\}} \mathbf{\Phi}_{jk} A_{jk} - \frac{1}{2V_j} \sum_{k \in \{k_j\}} \{c\,\mathbf{S}_j + (2-c)\mathbf{S}_k\}\, V_{jk} = 0, \tag{5}$$

where $\{k_j\}$ denotes a set of edge-connected neighbors of j, $V_j = \sum_{k \in \{k_j\}} V_{jk}$ is the median dual volume around the node j, $V_{jk} = \frac{1}{6}(\Delta\mathbf{r}_{jk} \cdot \mathbf{n}_{jk})$, $\Delta\mathbf{r}_{jk} = \mathbf{x}_k - \mathbf{x}_j$, $\mathbf{n}_{jk}$ is the directed-area vector pointing from j to k, which is the sum of the directed-areas corresponding to the dual-triangular faces associated with all tetrahedral elements sharing the edge $[j, k]$ (see Figures 1(a) and 1(b)), $A_{jk} = |\mathbf{n}_{jk}|$, the source term has been discretized by a one-parameter-family quadrature formula with a parameter c [24], and $\mathbf{\Phi}_{jk}$ is a numerical flux defined by the upwind flux:

$$\mathbf{\Phi}_{jk} = \frac{1}{2}\left[\mathbf{F}_n(\mathbf{U}_R) + \mathbf{F}_n(\mathbf{U}_L)\right] - \frac{1}{2}\mathbf{P}^{-1}\left|\mathbf{P}\mathbf{A}_n\right|(\mathbf{U}_R - \mathbf{U}_L), \tag{6}$$

where the projected flux $\mathbf{F}_n = \mathcal{F} \cdot \mathbf{n}_{jk}$ and the Jacobian $\mathbf{A}_n = \partial\mathbf{F}_n/\partial\mathbf{U}$ are computed with the unit directed-area vector: $\hat{\mathbf{n}}_{jk} = \mathbf{n}_{jk}/|\mathbf{n}_{jk}|$. The left and right states, $\mathbf{U}_L = (u_L, \mathbf{g}_L^t)^t$ and $\mathbf{U}_R = (u_R, \mathbf{g}_R^t)^t$, are computed in a

linearly-exact manner by the U-MUSCL scheme [29] for u, implemented efficiently with $\mathbf{g}$ used as the gradient of u [30]:

$$u_L = \kappa\frac{u_j+u_k}{2} + (1-\kappa)\left(u_j + \frac{1}{2}\,\mathbf{g}_j\cdot\Delta\mathbf{r}_{jk}\right), \quad \mathbf{g}_L = \mathbf{g}_j, \tag{7}$$

$$u_R = \kappa\frac{u_j+u_k}{2} + (1-\kappa)\left(u_k - \frac{1}{2}\,\mathbf{g}_k\cdot\Delta\mathbf{r}_{jk}\right), \quad \mathbf{g}_R = \mathbf{g}_k, \tag{8}$$

where κ is a real-valued parameter. This construction is called Scheme-II; see Ref. [30] for details.

To derive the IEBG system, we assume that u is given at nodes and discard the first component of the residual, i.e., $Res^{EB}_{j,1}$ in $\mathbf{Res}^{EB}_j = \left(Res^{EB}_{j,1}, Res^{EB}_{j,2}, Res^{EB}_{j,3}, Res^{EB}_{j,4}\right)$, leaving three equations for the three component of $\mathbf{g}$, which we denote by $\mathbf{Res}^{(1)}_j = \left(Res^{EB}_{j,2}, Res^{EB}_{j,3}, Res^{EB}_{j,4}\right)$:

$$\mathbf{Res}^{(1)}_j = \frac{1}{2V_j}\sum_{k\in\{k_j\}}\left[-\left\{(u_L+u_R) + L_r(\mathbf{g}_R-\mathbf{g}_L)\cdot\hat{\mathbf{n}}_{jk}\right\}\mathbf{n}_{jk} + \left\{c\,\mathbf{g}_j + (2-c)\mathbf{g}_k\right\}V_{jk}\right] = 0. \tag{9}$$

This residual system is satisfied exactly by linear functions of u (i.e., constant gradients), and thus called the linear IEBG system. It is a linear system and solved for $\mathbf{g}_j$ at all nodes. First-order gradient accuracy is expected on arbitrary tetrahedral grids. As in Refs. [24, 26], the length scale L_r is defined as

$$L_r = \alpha_g|\Delta\mathbf{r}_{jk}\cdot\hat{\mathbf{n}}_{jk}|. \tag{10}$$

This parameter controls the spectral accuracy of the IEBG method on a regular Cartesian grid (see Refs. [24,26]). On a regular tetrahedral grid, however, it generates a second-order truncation error that cannot be eliminated as discussed in Ref. [26]. In this paper, we focus on tetrahedral grids and set $\alpha_g = 0$:

$$\mathbf{Res}^{(1)}_j = \frac{1}{V_j}\sum_{k\in\{k_j\}}\left[-\frac{u_L+u_R}{2}\mathbf{n}_{jk} + \frac{c\,\mathbf{g}_j+(2-c)\mathbf{g}_k}{2}V_{jk}\right] = 0. \tag{11}$$

To achieve second-order gradient accuracy, we need to modify the system for quadratic exactness. It is the advantage of the EB discretization that quadratic exactness is satisfied without second derivatives [17,31]. Only a single modifications is needed in the source discretization:

$$\mathbf{Res}^{(2)}_j = \frac{1}{V_j}\sum_{k\in\{k_j\}}\left[-\frac{u_L+u_R}{2}\mathbf{n}_{jk} + \frac{c\,\mathbf{g}_j+(2-c)\mathbf{g'}_k}{2}V_{jk}\right] = 0, \tag{12}$$

where $\mathbf{g}_k$ has been replaced by $\mathbf{g'}_k$ defined by

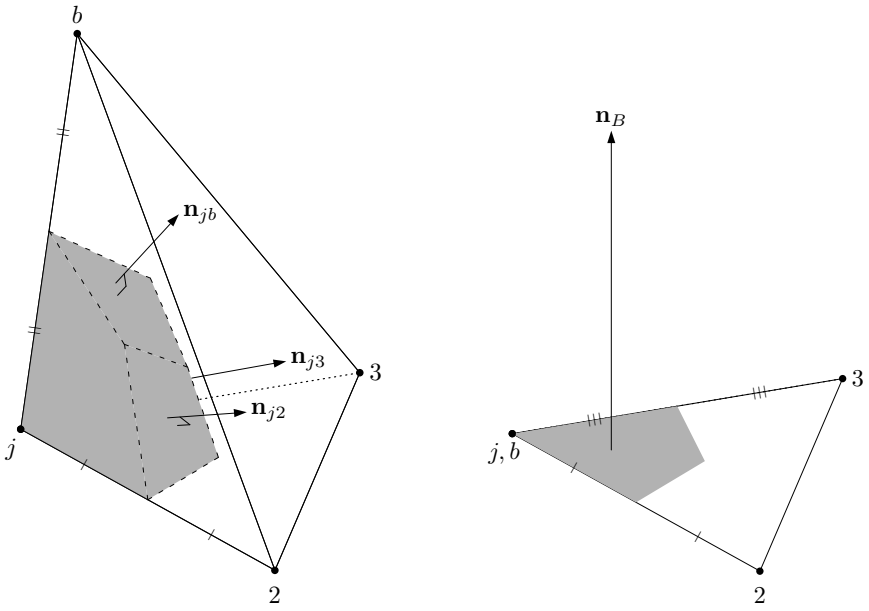
$$\mathbf{g}'_k = \mathbf{g}_k - \nabla\mathbf{g}^{LSQ}_j\cdot\Delta\mathbf{r}_{jk}. \tag{13}$$

Here, the gradient $\mathbf{g}^{LSQ}_j$ is computed by an unweighted linear least-squares fit over the neighbors $\{k_j\}$. The residual system (12) is exactly satisfied by quadratic functions of u (i.e., linear functions of $\mathbf{g}$), and therefore called the quadratic IEBG system. For arbitrary tetrahedral grids, second-order gradient accuracy is expected.

In Ref. [26], the parameter c in the source quadrature was set $c = (3D+4)/(D+2)$, where $D = 2$ in two dimensions and $D = 3$ in three dimensions. However, the modified quadrature in the above is actually exact for quadratic functions for any c, as we will discuss later.

B. Boundary Closure

Our aim is to compute gradients at all nodes including boundary nodes for a given set of function (or solution) values. Therefore, the IEBG systems need to be closed at boundaries. As discussed in Ref. [31], the source discretization does not need any boundary closure. Therefore, we will focus on the edge-based flux part, which is common to both linear and quadratic IEBG systems. Following the work in Ref. [17], we employ the edge collapsing technique to derive a boundary flux quadrature formula. We begin by constructing a full stencil by introducing a ghost node b outside a domain and connecting it to all the boundary neighbor nodes of a



(a) Tetrahedron outside the domain. (b) Collapsed tetrahedron.

Figure 2: Illustrations of the edge collapsing technique to derive a boundary formula. The triangle $[j, 2, 3]$ is a boundary element and the node b is located outside the domain. The tetrahedron is collapsed to a triangle (collapsing the edge $[j, b]$ by merging b to j), and an interior discretization scheme is converted to a boundary scheme. Note that the node b coincides with j and it is not located at the bottom of the arrow of $\mathbf{n}_B$.

boundary node j. Then, we construct the discretization over the full stencil, collapse the edge merging b to j, and identify the resulting discretization as a boundary formula. This process can be performed independently for each collapsing tetrahedral cell around j (located outside the domain of interest). An example is shown in Figure 2, where the edge between j and b is the one that is collapsed and the fluxes across the shaded dual faces will become the boundary fluxes for the discretization at the node j. Consider these flux contributions:

$$\mathbf{Res}_j^{(p)} = \mathbf{Res}_j^{(p)} + \frac{1}{V_j}\left[\phi_{jb}\mathbf{n}_{jb} + \phi_{j2}\mathbf{n}_{j2} + \phi_{j3}\mathbf{n}_{j3}\right], \tag{14}$$

where $p = 1$ or 2, ϕ_{jb} denotes the flux between j and b:

$$\begin{aligned}
\phi_{jb} &= \frac{u_L + u_R}{2} \\
&= \frac{1}{2}\left[\kappa\frac{u_j + u_b}{2} + (1-\kappa)\left(u_j + \frac{1}{2}\mathbf{g}_j \cdot \Delta\mathbf{r}_{jb}\right) + \kappa\frac{u_j + u_b}{2} + (1-\kappa)\left(u_b - \frac{1}{2}\mathbf{g}_b \cdot \Delta\mathbf{r}_{jb}\right)\right] \\
&= \frac{u_j + u_b}{2} + \frac{1-\kappa}{4}\left(\mathbf{g}_j - \mathbf{g}_b\right) \cdot \Delta\mathbf{r}_{jb},
\end{aligned} \tag{15}$$

and similarly for ϕ_{j2} and ϕ_{j3}:

$$\phi_{j2} = \frac{u_j + u_2}{2} + \frac{1-\kappa}{4}\left(\mathbf{g}_j - \mathbf{g}_2\right) \cdot \Delta\mathbf{r}_{j2}, \tag{16}$$

$$\phi_{j3} = \frac{u_j + u_3}{2} + \frac{1-\kappa}{4}\left(\mathbf{g}_j - \mathbf{g}_3\right) \cdot \Delta\mathbf{r}_{j3}. \tag{17}$$

Then, by collapsing the edge between j and b, we find (see Ref. [17])

$$\phi_{jb}\mathbf{n}_{jb} + \phi_{j2}\mathbf{n}_{j2} + \phi_{j3}\mathbf{n}_{j3} \to \phi_{jb}\left(\frac{1}{2}\mathbf{n}_B\right) + \phi_{j2}\left(\frac{1}{4}\mathbf{n}_B\right) + \phi_{j3}\left(\frac{1}{4}\mathbf{n}_B\right), \tag{18}$$

where $\phi_{jb} \to (u_j + u_b)/2$ because $\Delta\mathbf{r}_{jb} \to \mathbf{0}$. By setting $u_b = u_j$, we can write these boundary contributions as

$$\begin{aligned}\mathbf{Res}_j^{(p)} &= \mathbf{Res}_j^{(p)} + \frac{1}{V_j}\left[\phi_{jb}\left(\frac{1}{2}\mathbf{n}_B\right) + \phi_{j2}\left(\frac{1}{4}\mathbf{n}_B\right) + \phi_{j3}\left(\frac{1}{4}\mathbf{n}_B\right)\right] \\ &= \mathbf{Res}_j^{(p)} + \frac{1}{V_j}\left[\frac{6u_j + u_2 + u_3}{8}\mathbf{n}_B + \frac{1-\kappa}{16}\left\{\ (\mathbf{g}_j - \mathbf{g}_2)\cdot\Delta\mathbf{r}_{j2} + (\mathbf{g}_j - \mathbf{g}_3)\cdot\Delta\mathbf{r}_{j3}\ \right\}\mathbf{n}_B\right]. \quad (19)\end{aligned}$$

It is interesting that the first term is the well-known linearity-preserving boundary flux quadrature formula (see Appendix B of Ref. [27]). The second term is not necessary for linear exactness because then the gradient will be globally constant and the differences $\mathbf{g}_j - \mathbf{g}_2$ and $\mathbf{g}_j - \mathbf{g}_3$ will vanish. However, we will keep all the terms for both the linear and quadratic IEBG systems because the second term dramatically simplifies the diagonal block as we will see later.

Remark: We set $u_b = u_j$ because we are trying to compute the gradients for a given set of u values at all nodes. If one wishes to compute gradients for a numerical solution to a partial differential equation and if the solution value is known at a boundary node by a physical boundary condition, one can specify u_b. However, it is not necessary. In this paper, we will always set $u_b = u_j$.

III. Accuracy of Implicit Edge-Based Gradient Method

In Ref. [26], no discussions are given on the relationship between the gradient accuracy and the truncation error of the IEBG systems, and it was assumed that the accuracy-preserving source quadrature formula [31] had to be used for the quadratic IEBG system. As a result, the parameter c was fixed as $c = 2.6$ for tetrahedral grids and there was no free parameter available for the quadratic IEBG system. However, the parameter c can actually take values other than 2.6 and second-order gradient accuracy is still preserved[a]. To prove this, we will provide a detailed discussion below regarding the truncation error and its relation to the gradient accuracy.

A. Gradient Accuracy Order and Truncation Error

In unstructured grid discretizations, the truncation error order and the discretization error (solution error) order do not match and the difference depends on the type of target partial differential equations [32]. For example, the discretization error is one order higher than the truncation error for first-order hyperbolic partial differential equations. In the IEBG systems, if we consider u as a numerical solution to a first-order partial differential equation $\nabla u = \mathbf{g}$ and $\mathbf{g}$ as given, then the global vector of the IEBG residuals, $\mathbf{Res}_h^{(p)} = 0$, can be written as

$$\mathbf{J}_h\mathbf{U}_h = \tilde{\mathbf{b}}, \quad (20)$$

where $\mathbf{U}_h = (u_1, u_2, u_3, \cdots, u_N)$, N is the number of nodes, $\mathbf{J}_h$ is a residual Jacobian, and $\tilde{\mathbf{b}}$ is a global vector of known quantities associated with $\mathbf{g}$. Let $\mathbf{U}_e$ be the exact solution to $\nabla u = \mathbf{g}$. Then, we have

$$\mathbf{J}_h\mathbf{U}_e = \tilde{\mathbf{b}} + \tilde{\mathcal{T}}, \quad (21)$$

where $\tilde{\mathcal{T}}$ denotes the truncation error. Subtracting it from Eq. (20), we obtain

$$\mathbf{J}_h(\mathbf{U}_h - \mathbf{U}_e) = -\tilde{\mathcal{T}}, \quad (22)$$

or, by formally inverting the Jacobian,

$$\mathbf{U}_h - \mathbf{U}_e = -\mathbf{J}_h^{-1}\tilde{\mathcal{T}}. \quad (23)$$

Note that $\mathbf{J}_h^{-1}$ is of $O(h)$ for the discretization of first derivatives (i.e., ∇u), and therefore, the discretization error $\mathbf{U}_h - \mathbf{U}_e$ would be one order higher than the order of the truncation error $\tilde{\mathcal{T}}$ on unstructured grids [33].

[a]This was first observed by Emmett Padway (NASA) in numerical results obtained with the 2020 version of the QIEBG method [26] implemented in the NASA FUN3D code.

In our case, however, u is given and $\mathbf{g}$ is unknown. Then, the IEBG system $\mathbf{Res}_h = 0$ is still a linear system, which can be written as

$$\mathbf{A}_h \mathbf{G}_h = \mathbf{b}, \tag{24}$$

where $\mathbf{A}_h$ and $\mathbf{G}_h$ denote the residual Jacobian matrix and the global vector of gradients while $\mathbf{b}$ denotes a global vector of given quantities (i.e., the Green-Gauss gradients). Let $\mathbf{G}$ be the exact gradient vector. Then, we have

$$\mathbf{A}_h \mathbf{G} = \mathbf{b} + \mathcal{T}, \tag{25}$$

where $\mathcal{T}$ denotes the truncation error. Subtracting it from Eq. (24), we obtain

$$\mathbf{A}_h (\mathbf{G}_h - \mathbf{G}) = -\mathcal{T}, \tag{26}$$

or

$$\mathbf{G}_h - \mathbf{G} = -\mathbf{A}_h^{-1} \mathcal{T}. \tag{27}$$

Unlike the previous case, we have $\mathbf{A}_h = O(1)$ for the IEBG system. Therefore, the order of the gradient error, $\mathbf{G}_h - \mathbf{G}$, matches the truncation error order even on irregular grids. This means that the linear and quadratic IEBG systems must have $O(h)$ and $O(h^2)$ truncation errors, respectively, which will be verified in the next section.

B. Accuracy of Linear IEBG

By definition, we obtain a truncation error by substituting a smooth exact solution, which satisfies $\nabla u = \mathbf{g}$, into the IEBG system. Assuming smooth functions for u and $\mathbf{g}$, and defining the operator:

$$\partial_{jk} = \Delta \mathbf{r}_{jk} \cdot (\partial_x, \partial_y, \partial_z), \tag{28}$$

we write the Taylor expansions of u and $\mathbf{g}$ around a node j as

$$u_k = u_j + \partial_{jk} u_j + \frac{1}{2}\partial_{jk}^2 u_j + \frac{1}{6}\partial_{jk}^3 u_j + O(h^4), \tag{29}$$

$$\mathbf{g}_k = \mathbf{g}_j + \partial_{jk} \mathbf{g}_j + \frac{1}{2}\partial_{jk}^2 \mathbf{g}_j + \frac{1}{6}\partial_{jk}^3 \mathbf{g}_j + O(h^4), \tag{30}$$

or since $\mathbf{g} = \nabla u$,

$$\mathbf{g}_k = \nabla u_j + \partial_{jk} \nabla u_j + \frac{1}{2}\partial_{jk}^2 \nabla u_j + \frac{1}{6}\partial_{jk}^3 \nabla u_j + O(h^4). \tag{31}$$

Then, we find

$$\begin{aligned} \frac{1}{2}\,\mathbf{g}_k \cdot \Delta \mathbf{r}_{jk} &= \frac{1}{2}\nabla u_j \cdot \Delta \mathbf{r}_{jk} + \frac{1}{2}\partial_{jk} \nabla u_j \cdot \Delta \mathbf{r}_{jk} + \frac{1}{4}\partial_{jk}^2 \nabla u_j \cdot \Delta \mathbf{r}_{jk} + \frac{1}{12}\partial_{jk}^3 \nabla u_j \cdot \Delta \mathbf{r}_{jk} + O(h^5) \\ &= \frac{1}{2}\partial_{jk} u_j + \frac{1}{2}\partial_{jk}^2 u_j + \frac{1}{4}\partial_{jk}^3 u_j + \frac{1}{12}\partial_{jk}^4 u_j + O(h^5), \end{aligned} \tag{32}$$

$$\frac{u_j + u_k}{2} = u_j + \frac{1}{2}\partial_{jk} u + \frac{1}{4}\partial_{jk}^2 u + \frac{1}{12}\partial_{jk}^3 u + O(h^4), \tag{33}$$

$$u_k - \frac{1}{2}\,\mathbf{g}_k \cdot \Delta \mathbf{r}_{jk} = u_j + \frac{1}{2}\partial_{jk} u - \frac{1}{12}\partial_{jk}^3 u + O(h^4), \tag{34}$$

and therefore, substituting these expressions into the reconstruction schemes, Eqs. (7) and (8), we obtain

$$u_L = u_j + \frac{1}{2}\partial_{jk} u + \frac{\kappa}{4}\partial_{jk}^2 u + \frac{\kappa}{12}\partial_{jk}^3 u + O(h^4), \tag{35}$$

$$u_R = u_j + \frac{1}{2}\partial_{jk} u + \frac{\kappa}{4}\partial_{jk}^2 u + \frac{2\kappa - 1}{12}\partial_{jk}^3 u + O(h^4), \tag{36}$$

thus leading to

$$\frac{1}{2V_j}\sum_{k\in\{k_j\}}(u_L+u_R)\,\mathbf{n}_{jk} = \frac{1}{V_j}\sum_{k\in\{k_j\}}\left(\frac{1}{2}\partial_{jk}u\right)\mathbf{n}_{jk}+\frac{1}{V_j}\sum_{k\in\{k_j\}}\left(\frac{\kappa}{4}\partial_{jk}^2u\right)\mathbf{n}_{jk}+O(h^2). \tag{37}$$

It is a special property of the edge-based discretization that the first term reduces to ∇u_j on a tetrahedral grid (or a triangular grid in two dimensions) [17,31,34]:

$$\frac{1}{V_j}\sum_{k\in\{k_j\}}\frac{u_L+u_R}{2}\mathbf{n}_{jk} = \nabla u_j+\frac{\kappa}{4V_j}\sum_{k\in\{k_j\}}\left(\partial_{jk}^2u\right)\mathbf{n}_{jk}+O(h^2). \tag{38}$$

Note that the $O(h)$ term proportional to κ can be eliminated with $\kappa=0$ but it is not necessary for the linear IEBG method (it is necessary for the quadratic IEBG method). Next, we expand the source discretization as

$$\frac{1}{V_j}\sum_{k\in\{k_j\}}\frac{c\,\mathbf{g}_j+(2-c)\mathbf{g}_k}{2}V_{jk} = \mathbf{g}_j+\frac{2-c}{2V_j}\left(\sum_{k\in\{k_j\}}\partial_{jk}\mathbf{g}_jV_{jk}+O(h^2)\right) = \nabla u_j+O(h). \tag{39}$$

Therefore, the linear IEBG system is expanded on a general tetrahedral grid as

$$\begin{aligned}
\mathbf{Res}_j^{(1)} &= -\nabla u_j+\mathbf{g}_j-\frac{\kappa}{4V_j}\sum_{k\in\{k_j\}}\left(\partial_{jk}^2u\right)\mathbf{n}_{jk}+\frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\partial_{jk}\mathbf{g}_jV_{jk}+O(h^2)\\
&= -\nabla u_j+\nabla u_j-\frac{\kappa}{4V_j}\sum_{k\in\{k_j\}}\left(\partial_{jk}^2u\right)\mathbf{n}_{jk}+\frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\partial_{jk}\mathbf{g}_jV_{jk}+O(h^2)\\
&= -\frac{\kappa}{4V_j}\sum_{k\in\{k_j\}}\left(\partial_{jk}^2u\right)\mathbf{n}_{jk}+\frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\partial_{jk}\mathbf{g}_jV_{jk}+O(h^2)\\
&= O(h),
\end{aligned} \tag{40}$$

which proves that the gradient accuracy is expected to be first order. Here, both κ and c are free parameters. It achieves second-order gradient accuracy for $\kappa=0$ and $c=2$, but the resulting implicit system is difficult to relax as we will show later. On the other hand, the quadratic IEBG method achieves second-order accuracy for an arbitrary value of c as we will show in the next section.

C. Accuracy of Quadratic IEBG

For the quadratic IEBG system, as mentioned in the previous section, we must set $\kappa=0$ in order to eliminate the first-order error in Eq. (38). The first-order error in the source discretization (39) can also be eliminated with $c=2$, but it leads to instability in a relaxation scheme and we wish to be able to freely adjust c, so that we can achieve higher-order gradient accuracy in a regular tetrahedral grid and also we can overcome potential difficulties by adjusting c. To achieve second-order accuracy while keeping c as a free parameter, we can apply a modification using $\mathbf{g}_k'$ as in Eq. (13). To see how it eliminates the first-order truncation error, let us consider the following general formula for approximating $\mathbf{g}$ at a node j (see Ref. [31]):

$$\mathcal{I}_j = \frac{1}{2V_j}\sum_{k\in\{k_j\}}\left[a_L\mathbf{g}_j+b_L\nabla^{LSQ}\mathbf{g}_j\cdot\Delta\mathbf{r}_{jk}+a_R\mathbf{g}_k\right]V_{jk}, \tag{41}$$

where a_L, b_L, and a_R are constants to be determined. Note that $\nabla^{LSQ}\mathbf{g}_j$ is computed by an unweighted linear LSQ method and thus its error is $O(h)$. Expanding it around j, we obtain

$$\begin{aligned}
\mathcal{I}_j &= \frac{1}{2V_j}\sum_{k\in\{k_j\}}\left[a_L\mathbf{g}_j+b_L\left\{\nabla\mathbf{g}_j+O(h)\right\}\cdot\Delta\mathbf{r}_{jk}+a_R\left(\mathbf{g}_j+\partial_{jk}\mathbf{g}_j+\frac{1}{2}\partial_{jk}^2\mathbf{g}_j+O(h^3)\right)\right]V_{jk}\\
&= \frac{a_L+a_R}{2}\mathbf{g}_j+\frac{a_R+b_L}{2V_j}\sum_{k\in\{k_j\}}\left(\partial_{jk}\mathbf{g}_j\right)V_{jk}+O(h^2).
\end{aligned} \tag{42}$$

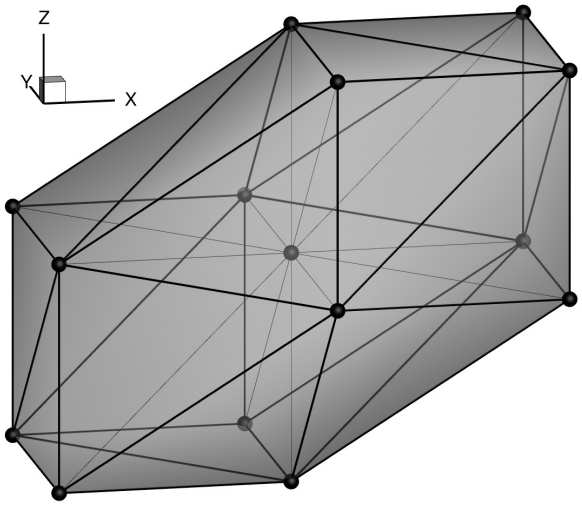


Figure 3: A regular tetrahedral grid.

Therefore, we must have $a_L + a_R = 2$ for consistency and also $a_R + b_L = 0$ for second-order accuracy, leading to

$$a_L = c, \quad a_R = 2 - c, \quad b_L = c - 2, \tag{43}$$

where c is an arbitrary constant, and thus

$$\begin{aligned} \mathcal{I}_j &= \frac{1}{2V_j} \sum_{k \in \{k_j\}} \left[c\,\mathbf{g}_j + (c-2)\nabla^{LSQ}\mathbf{g}_j \cdot \Delta\mathbf{r}_{jk} + (2-c)\mathbf{g}_k \right] V_{jk} \\ &= \frac{1}{2V_j} \sum_{k \in \{k_j\}} \left[c\,\mathbf{g}_j + (2-c)\mathbf{g}'_k \right] V_{jk}, \end{aligned} \tag{44}$$

where $\mathbf{g}'_k = \mathbf{g}_k - \nabla^{LSQ}\mathbf{g}_j \cdot \Delta\mathbf{r}_{jk}$. Finally, then, we find the truncation error of the quadratic IEBG system as

$$\mathbf{Res}_j^{(2)} = O(h^2). \tag{45}$$

Hence, the gradient accuracy is expected to be second order for an arbitrary c as long as $\kappa = 0$.

In Ref. [26], it was stated that c must be given by $c = (3D + 4)/(D + 2)$, where $D = 1$, 2, 3 in one, two, three dimensions, respectively. This choice is known to produce a factored second-order truncation error on a regular simplex-element grid, which is required to produce a third-order truncation error in the edge-based discretization [31]. This additional condition is not necessary in the IEBG method because we only need a second-order truncation error, as a method for gradients, on both regular and irregular grids. However, it implies that we can expect higher-order gradient accuracy on regular tetrahedral grids, which we discuss in the next section.

D. Higher-Order Accuracy on Regular Tetrahedral Grids

On a regular tetrahedral grid, which is symmetric with respect to the central node and invariant under translation [31], the extra term involving $\nabla\mathbf{g}_j^{LSQ}$ in the quadratic IEBG system will cancel out. As a result, the linear and quadratic IEBG systems reduce to the same system. For example, for the regular tetrahedral grid shown in Figure 3 (obtained by splitting each of regular hexahedra into six tetrahedra), the IEBG systems are expanded as

$$\mathbf{Res}_j^{(1)} = \mathbf{Res}_j^{(2)} = -\left(\frac{13 - 5c}{36} - \frac{\kappa}{4} \right) (\partial_{xx} + \partial_{yy} + \partial_{zz} - \partial_{xy} - \partial_{yz} + \partial_{zx}) (\nabla u) h^2 + O(h^4). \tag{46}$$

Therefore, both methods will be second-order accurate on a regular grid, and achieve fourth-order accuracy if we choose

$$c = \frac{13 - 9\kappa}{5}. \tag{47}$$

This matches the general expression derived in Ref. [31]:

$$c = \frac{3D(1-\kappa) + 4}{D+2}, \tag{48}$$

where $D = 1$ for a one-dimensional uniform grid, $D = 2$ for a regular triangular grid, and $D = 3$ for a regular tetrahedral grid. Observe that $D = 3$ gives Eq. (47). It also verifies that $D = 2$ gives the expression derived in Ref. [26] for the IEBG system for a regular triangular grid.

Eq. (47) implies that there is a one-parameter family of optimal choices, but we must set $\kappa = 0$ for the quadratic IEBG method, which gives

$$\kappa = 0, \quad c = \frac{13}{5} = 2.6. \tag{49}$$

For the linear IEBG method, we can choose other values of κ, but the above values are used also for the linear IEBG method in this work. In applying the IEBG methods to inviscid-flow problems, we find that c needs to be increased for robustness as we will discuss later.

IV. Matrix Forms

In Ref. [26], a Krylov solver was employed for solving the IEBG systems. While it is useful, we here demonstrate that a simple relaxation scheme such as the block Gauss-Seidel relation scheme can serve as an efficient and robust solver. For this purpose, we provide a matrix form of the IEBG system that can be directly implemented in a code, which is much more efficient than the automatic differentiation approach in Ref. [26]. In particular, we report here that the diagonal block can be dramatically simplified even at boundary nodes if the accuracy-preserving flux quadrature formula [17] is used in the linear and quadratic IEBG methods.

A. Matrix Form of Linear IEBG

The linear IEBG system $\mathbf{Res}_j^{(1)} = 0$ can be written as

$$\mathbf{M}_{jj}\mathbf{g}_j + \sum_{k\in\{k_j\}} \mathbf{M}_{jk}\mathbf{g}_k = \mathbf{b}_j, \tag{50}$$

where

$$\mathbf{M}_{jj} = \frac{\kappa-1}{4V_j}\sum_{k\in\{k_j\}} (\mathbf{n}_{jk}\otimes\Delta\mathbf{r}_{jk}) + \frac{c}{2V_j}\sum_{k\in\{k_j\}} V_{jk}\mathbf{I}, \tag{51}$$

$$= \frac{\kappa-1}{4V_j}\sum_{k\in\{k_j\}} (\mathbf{n}_{jk}\otimes\Delta\mathbf{r}_{jk}) + \frac{c}{2}\mathbf{I}, \tag{52}$$

$$\mathbf{M}_{jk} = -\frac{\kappa-1}{4V_j}(\mathbf{n}_{jk}\otimes\Delta\mathbf{r}_{jk}) + \frac{2-c}{2V_j}V_{jk}\mathbf{I}, \tag{53}$$

and

$$\mathbf{b}_j = \frac{1}{V_j}\sum_{k\in\{k_j\}} \frac{u_j+u_k}{2}\mathbf{n}_{jk}. \tag{54}$$

The diagonal block can be simplified, by the following identities,

$$\frac{1}{2}\sum_{k\in\{k_j\}} (\mathbf{n}_{jk}\otimes\Delta\mathbf{r}_{jk}) = V_j\mathbf{I}, \tag{55}$$

and $\sum_{k\in\{k_j\}} V_{jk}\mathbf{I} = V_j\mathbf{I}$ as

$$\mathbf{M}_{jj} = \frac{\kappa-1}{2V_j}\,V_j\mathbf{I} + \frac{c}{2}\mathbf{I} = \left(\frac{\kappa-1}{2}+\frac{c}{2}\right)\mathbf{I} = \left(\frac{c+\kappa-1}{2}\right)\mathbf{I}, \tag{56}$$

showing that the diagonal block is a diagonal matrix, which can be very easily inverted:

$$\mathbf{M}_{jj}^{-1} = \left(\frac{2}{c+\kappa-1}\right)\mathbf{I}, \tag{57}$$

where we assume $c+\kappa-1 \neq 0$. Later, we will show that values of c and κ satisfying $c+\kappa-1=0$ lead to instability in a relaxation scheme for solving the implicit gradient system. Note that the identity (55) holds not only at interior nodes but also at a boundary node if the accuracy-preserving boundary flux quadrature is employed because it is derived by collapsing an interior stencil. That is, collapsing an edge in an interior stencil does not have any impact on Eq.(56), which is obvious because Eq.(56) involves no geometric terms. This is the reason that we recommend the accuracy-preserving boundary flux quadrature even for the linear IEBG method.

B. Matrix Form of Quadratic IEBG

For the quadratic IEBG system, we have the same right hand side, but need to add extra contributions to $\mathbf{M}_{jj}$ and $\mathbf{M}_{jk}$ arising from the term involving $\nabla^{LSQ}\mathbf{g}_j$:

$$\frac{1}{V_j}\sum_{k\in\{k_j\}}\left[\frac{-(2-c)\nabla\mathbf{g}_j^{LSQ}\cdot\Delta\mathbf{r}_{jk}}{2}V_{jk}\right]. \tag{58}$$

To write this term in the matrix form, we define

$$\nabla\mathbf{g}_j^{LSQ} = \sum_{m\in\{k_j\}}(\mathbf{g}_m-\mathbf{g}_j)\otimes\mathbf{L}_{jm}^{LSQ}, \tag{59}$$

where $\mathbf{L}_{jm}^{LSQ}$ is a vector of the linear unweighted LSQ gradient coefficients such that the gradient of a scalar function a is computed as

$$\nabla a_j^{LSQ} = \sum_{m\in\{k_j\}}\mathbf{L}_{jm}^{LSQ}(a_m-a_j). \tag{60}$$

Then, we find

$$\begin{aligned}\frac{1}{V_j}\sum_{k\in\{k_j\}}\left[\frac{-(2-c)\nabla\mathbf{g}_j^{LSQ}\cdot\Delta\mathbf{r}_{jk}}{2}V_{jk}\right] &= -\frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\left[\sum_{m\in\{k_j\}}(\mathbf{g}_m-\mathbf{g}_j)\otimes\mathbf{L}_{jm}^{LSQ}\right]\cdot\Delta\mathbf{r}_{jk}V_{jk}\\ &= \frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\sum_{m\in\{k_j\}}\left(\mathbf{L}_{jm}^{LSQ}\cdot\Delta\mathbf{r}_{jk}\right)V_{jk}\,(\mathbf{g}_j-\mathbf{g}_m), \end{aligned}\tag{61}$$

which we write, by exchanging the indices k and m, as

$$\begin{aligned}\frac{1}{V_j}\sum_{k\in\{k_j\}}\left[\frac{-(2-c)\nabla\mathbf{g}_j^{LSQ}\cdot\Delta\mathbf{r}_{jk}}{2}V_{jk}\right] &= \frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\sum_{m\in\{k_j\}}\left(\mathbf{L}_{jk}^{LSQ}\cdot\Delta\mathbf{r}_{jm}\right)V_{jm}\,(\mathbf{g}_j-\mathbf{g}_k)\\ &= \frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\left[\sum_{m\in\{k_j\}}\left(\mathbf{L}_{jk}^{LSQ}\cdot\Delta\mathbf{r}_{jm}\right)V_{jm}\right]\mathbf{I}\,(\mathbf{g}_j-\mathbf{g}_k). \end{aligned}\tag{62}$$

Therefore, by adding the extra contributions to $\mathbf{M}_{jj}$ and $\mathbf{M}_{jk}$ of the linear IEBG system, we find the diagonal and off-diagonal blocks for the quadratic IEBG system:

$$\mathbf{M}_{jj} = \left(\frac{c+\kappa-1}{2}\right)\mathbf{I} + \frac{2-c}{2V_j}\sum_{k\in\{k_j\}}\left[\sum_{m\in\{k_j\}}\left(\mathbf{L}_{jk}^{LSQ}\cdot\Delta\mathbf{r}_{jm}\right)V_{jm}\right]\mathbf{I}, \tag{63}$$

and

$$\mathbf{M}_{jk} = -\frac{\kappa - 1}{4V_j} \left(\mathbf{n}_{jk} \otimes \Delta \mathbf{r}_{jk} \right) + \frac{2 - c}{2V_j} \left[V_{jk} - \sum_{m \in \{k_j\}} \left(\mathbf{L}_{jk}^{LSQ} \cdot \Delta \mathbf{r}_{jm} \right) V_{jm} \right] \mathbf{I}. \tag{64}$$

The extra term in the diagonal block does not seem to simplify further, but it is a diagonal matrix and therefore $\mathbf{M}_{jj}$ can be easily inverted.

C. Boundary Contributions

After computing the matrices $\mathbf{M}_{jj}$ and $\mathbf{M}_{jk}$, we loop over boundary triangular faces and add contributions from the accuracy-preserving boundary-flux quadrature (19) to a node j of the boundary face B (see Figure 2(b)):

$$\mathbf{M}_{j2} = \mathbf{M}_{j2} + \frac{1 - \kappa}{16} \mathbf{n}_B \otimes \Delta \mathbf{r}_{j2}, \quad \mathbf{M}_{j3} = \mathbf{M}_{j3} + \frac{1 - \kappa}{16} \mathbf{n}_B \otimes \Delta \mathbf{r}_{j3}, \tag{65}$$

and we also add the contribution to the right-hand side:

$$\mathbf{b}_j = \mathbf{b}_j + \frac{6u_j + u_2 + u_3}{8} \mathbf{n}_B. \tag{66}$$

These corrections should be made for the other two nodes of the boundary triangular face B (i.e., $j \leftarrow 2$ and $j \leftarrow 3$), and for both the linear and quadratic IEBG methods. The contribution to the diagonal bock $\mathbf{M}_{jj}$ would be

$$\mathbf{M}_{jj} = \mathbf{M}_{jj} - \frac{1 - \kappa}{16} \mathbf{n}_B \otimes \left(\Delta \mathbf{r}_{j2} + \Delta \mathbf{r}_{j3} \right), \tag{67}$$

but this is not necessary because the diagonal bock $\mathbf{M}_{jj}$ in the simplified form (56) already includes this contribution.

V. Relaxation Scheme and Stability

A. Relaxation Scheme

Both the linear and quadratic IEBG systems are given in the same form (50), which leads to the following block Gauss-Seidel relaxation scheme:

$$\mathbf{g}_j^{n+1} = - \sum_{k \in \{k_j^{n+1}\}} \mathbf{C}_{jk} \mathbf{g}_k^{n+1} - \sum_{k \in \{k_j^n\}} \mathbf{C}_{jk} \mathbf{g}_k^n + \overline{\mathbf{b}}_j, \quad \mathbf{C}_{jk} = \mathbf{M}_{jj}^{-1} \mathbf{M}_{jk}, \quad \overline{\mathbf{b}}_j = \mathbf{M}_{jj}^{-1} \mathbf{b}_j, \tag{68}$$

sequentially performed for $j = 1, 2, 3, \cdots, N$, where N is the total number of nodes, n is the iteration counter, $\{k_j^{n+1}\}$ is a set of neighbors where the gradients are already updated, and $\{k_j^n\}$ is the rest of the neighbors. Note that it requires only three multiplications to apply $\mathbf{M}_{jj}^{-1}$ to $\mathbf{b}_j$ and nine multiplications to $\mathbf{M}_{jk}$ because $\mathbf{M}_{jj}^{-1}$ is a 3×3 diagonal matrix as in Eq. (57). Note also that $\overline{\mathbf{b}}_j$ is computed just once at the beginning and fixed during the iteration. In our code, the relaxation scheme is implemented in the relaxed form:

$$\mathbf{g}_j^{n+1} = \mathbf{g}_j^n + \omega \, \mathbf{r}_j, \tag{69}$$

where ω is a relaxation factor and $\mathbf{R}_j$ is the linear-system residual:

$$\mathbf{r}_j = - \left[\sum_{k \in \{k_j^{n+1}\}} \mathbf{C}_{jk} \mathbf{g}_k^{n+1} + \sum_{k \in \{k_j^n\}} \mathbf{C}_{jk} \mathbf{g}_k^n \right] + \overline{\mathbf{b}}_j - \mathbf{g}_j^n. \tag{70}$$

The relaxation factor ω is set to 1 in this work, but a smaller value may be help the relaxation converge better in some future cases. In all the cases considered in this paper, we apply just one relaxation per flow-solver iteration.

B. Stability Analysis

The block Gauss-Seidel relaxation scheme is not always stable and the stability depends on the parameters κ and c. Intuitively, we can imagine that increasing κ towards 1 would stabilize the relaxation scheme because it makes the implicit coupling weaker and also increasing c would also stabilize the relaxation scheme because it increases the diagonal entries. For a regular tetrahedral grid, we can use a Fourier analysis to determine the stability region and verify these intuitive observations. The Fourier analysis is useful also for explaining why values of c and κ satisfying $c + \kappa - 1 = 0$ as well as the combination $\kappa = 0$ and $c = 2$, which renders the linear IEBG method quadratically exact, are not practical.

Consider a regular tetrahedral grid in Figure 3 and define the nodes as follows:

$$\mathbf{x}_1 = (0, 0, -h), \quad \mathbf{x}_2 = (0, -h, 0), \quad \mathbf{x}_3 = (-h, 0, -h), \quad \mathbf{x}_4 = (-h, 0, 0), \quad \mathbf{x}_5 = (-h, h, -h), \tag{71}$$

$$\mathbf{x}_6 = (0, h, -h), \quad \mathbf{x}_7 = (0, -h, h), \quad \mathbf{x}_8 = (-h, h, 0), \quad \mathbf{x}_9 = (0, 0, h), \quad \mathbf{x}_{10} = (0, h, 0), \tag{72}$$

$$\mathbf{x}_{11} = (h, -h, 0), \quad \mathbf{x}_{12} = (h, 0, 0), \quad \mathbf{x}_{13} = (h, -h, h), \quad \mathbf{x}_{14} = (h, 0, h), \quad \mathbf{x}_j = (0, 0, 0), \tag{73}$$

where h is a uniform mesh spacing. Define a Fourier mode $\mathbf{g}_\beta^n$ as

$$\mathbf{g}_\beta^n = \mathbf{A}_g^n \exp\left(\beta_x \frac{x}{h} + \beta_y \frac{y}{h} + \beta_z \frac{z}{h}\right), \tag{74}$$

where β_x, β_y, and β_z are discrete frequencies ranging from 0 to π, and $\mathbf{A}_g^n$ is a vector of amplitudes. Insert it into the scheme (68) with $\overline{\mathbf{b}}_j = 0$, which does not affect the stability, and find

$$\mathbf{A}_g^{n+1} = -\sum_{k=1}^{6} \mathbf{D}_k \mathbf{A}_g^{n+1} - \sum_{k=7}^{14} \mathbf{D}_k \mathbf{A}_g^n, \tag{75}$$

where $\mathbf{D}_j$ and $\mathbf{D}_k$ are matrices dependent on c, κ, β_x, β_y, and β_z. Note that we consider a sequential block Gauss-Seidel relaxation scheme in a lexicographic ordering and therefore $\{k_j^{n+1}\} = \{1, 2, 3, 4, 5, 6\}$ and $\{k_j^n\} = \{7, 8, 9, 10, 11, 12, 13, 14\}$: the gradients have already been updated at nodes 1 to 6. Then, we find

$$\mathbf{A}_g^{n+1} = \mathbf{M}\mathbf{A}_g^n, \tag{76}$$

where

$$\mathbf{M} = -\left(\mathbf{I} + \sum_{k=1}^{6} \mathbf{D}_k\right)^{-1} \sum_{k=7}^{14} \mathbf{D}_k. \tag{77}$$

Stability requires the spectral radius of $\mathbf{M}$ to be less than 1. We computed the spectral radius numerically with

$$\beta_x = \frac{i_\beta}{N_\beta}\pi, \quad i_\beta = 1, 2, 3, \cdots, N_\beta = 35, \tag{78}$$

$$\beta_y = \frac{j_\beta}{N_\beta}\pi, \quad j_\beta = 1, 2, 3, \cdots, N_\beta = 35, \tag{79}$$

$$\beta_z = \frac{k_\beta}{N_\beta}\pi, \quad k_\beta = 1, 2, 3, \cdots, N_\beta = 35, \tag{80}$$

and

$$c = 0.1 + \frac{5.9(i_c - 1)}{N_c - 1}, \quad i_c = 1, 2, 3, \cdots, N_c = 100, \tag{81}$$

$$\kappa = \frac{(i_\kappa - 1)}{N_\kappa - 1}, \quad i_\kappa = 1, 2, 3, \cdots, N_\kappa = 600. \tag{82}$$

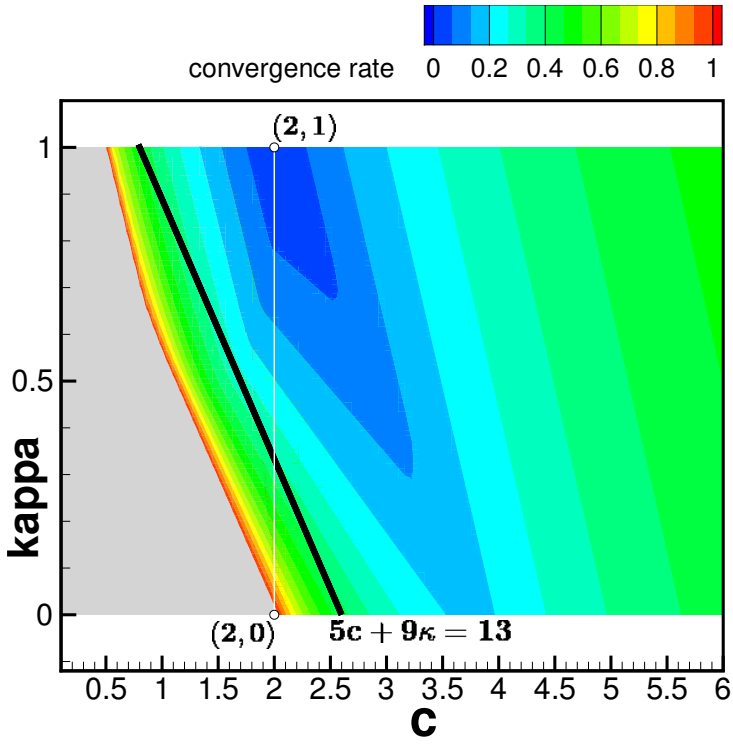


Figure 4: Stability analysis result: spectral radius contours.

Figure 4 shows the contours of the spectral radius (convergence rate). The thick black line indicates the line of the fourth-order accuracy: $c = (13 - 9\kappa)/5$, which lies inside the stability region. As can be seen, the relaxation scheme is stable for a wide range of parameters. Although only in a narrow region on the right side of the line $c = (13-9\kappa)/5$, increasing κ towards 1 or increasing c improves the convergence rate. In theory, the convergence rate should be zero at $(c, \kappa) = (2, 1)$ because then the method becomes explicit. This is well captured by the analysis as can be seen in Figure 4. In practical applications, we have found that a large value of c is needed for robustness, especially for the quadratic IEBG method, and therefore, we will employ $c = 5$. The relaxation scheme remains stable for this value as can be seen in Figure 4, at least for regular tetrahedral grids; but we have found from numerical experiments that it is stable and robust for irregular tetrahedral grids as well.

It is easy to see in Figure 4 that the line defined by $c + \kappa - 1 = 0$, which passes through two points $(c, \kappa) = (0, 1)$ and $(c, \kappa) = (1, 0)$, lies entirely in the grey area and thus the relaxation scheme is unstable. Therefore, we will not use values satisfying $c + \kappa - 1 = 0$ and the diagonal block is invertible, as in Eq. (57). Finally, the point $(c, \kappa) = (2, 0)$ corresponds to the special case that the linear IEBG achieves second-order gradient accuracy. Unfortunately, as can be seen in Figure 4, the relaxation scheme is not stable at this point. In numerical experiments, we found the relaxation scheme diverged quickly, indeed. In this paper, we consider two choices: $(c, \kappa) = (2.6, 0)$ and $(c, \kappa) = (5, 0)$ as a robust alternative for both the linear and quadratic IEBG methods.

VI. Results

In this paper, we focus on the Euler equations and consider second- and third-order edge-based schemes. For the second-order scheme, we employ the classical boundary-flux quadrature with weights $(6/8, 1/8, 1/8)$ [27], and for the third-order scheme, we use the accuracy-preserving formula [17]. Note that the classical formula is used only for the second-order discretization of the Euler equations and the accuracy-preserving formula is used in both linear and quadratic IEBG systems. See Ref. [31] for details of the third-order edge-based scheme. For comparison, we present results obtained with unweighted linear and quadratic LSQ methods. The linear LSQ method uses only the edge-connected neighbors but the quadratic LSQ method includes neighbors of the edge-

connected neighbors also. We implemented the quadratic LSQ method using the two step method described in Ref. [19], so that we only need to access the edge-connected neighbors in each step. As a result, it requires to loop over edge-connected neighbors and then at each neighbor, loop over its edge-connected neighbors. In a regular tetrahedral grid such as in Figure 3, where there are 14 edge-connected neighbors, it will go through the total of $14 \times 14 = 196$ neighbors. This algorithm is useful in that it can be easily implemented in a parallel code and is partition-independent, but it is difficult to implement it in an edge loop, which is possible for the linear LSQ method as well as the IEBG methods.

In all the results shown in this paper, we employ an implicit defect-correction solver with the exact Jacobian of a first-order scheme for the Euler equations. At every iteration, we perform a single Gauss-Seidel relaxation to update the gradients of the primitive variables. For the linear system in the implicit defect-correction solver, we perform 10 Gauss-Seidel relaxations per iteration. The numerical flux is computed by the Roe flux [35] with an eigenvalue smoothing applied to all waves for increasing numerical dissipation:

$$|\lambda_k|^* = \begin{cases} |\lambda_k| & \text{if } |\lambda_k| > \delta\lambda_k, \\ \frac{1}{2}\left(\frac{|\lambda_k|^2}{\delta\lambda_k} + \delta\lambda_k\right) & \text{otherwise,} \end{cases} \tag{83}$$

where λ_k is the k-th characteristic speed of the Euler equations in a face-normal direction, C_k is a constant, $\delta\lambda_k \equiv C_k\hat{a}$, and $\hat{a}$ is the Roe-averaged speed of sound. In this work, we set $C_k = 0.25$ for all waves and for all problems, except a hypersonic-flow problem, for which we set $C_k = 0.5$ for all waves. As a result, the Roe flux used in this work is more dissipative than the original Roe flux with an entropy fix (to prevent carbuncle phenomena).

A. Accuracy Verification

1. *Gradient Accuracy*

We begin by verifying the order of gradient accuracy for the linear and quadratic IEBG methods on regular and irregular tetrahedral grids. In all cases, we perform the Gauss-Seidel relaxation until the L_1 norm of the residual is reduced by eight orders of magnitude. We tested both LSQ and IEBG methods. The unweighted linear and quadratic LSQ gradient methods are referred to as LLSQ and QLSQ, which are both unweighted LSQ methods and may also be referred to as LLSQ(0) and QLSQ(0). The quadratic LSQ method is implemented based on a two-step method described in Ref. [19], thus involving neighbor nodes and their neighbor nodes. The linear and quadratic IEBG methods are referred to as LIEBG(c) and QIEBG(c), respectively, where c indicates the value of c. For the test, we compute the gradient of the following function:

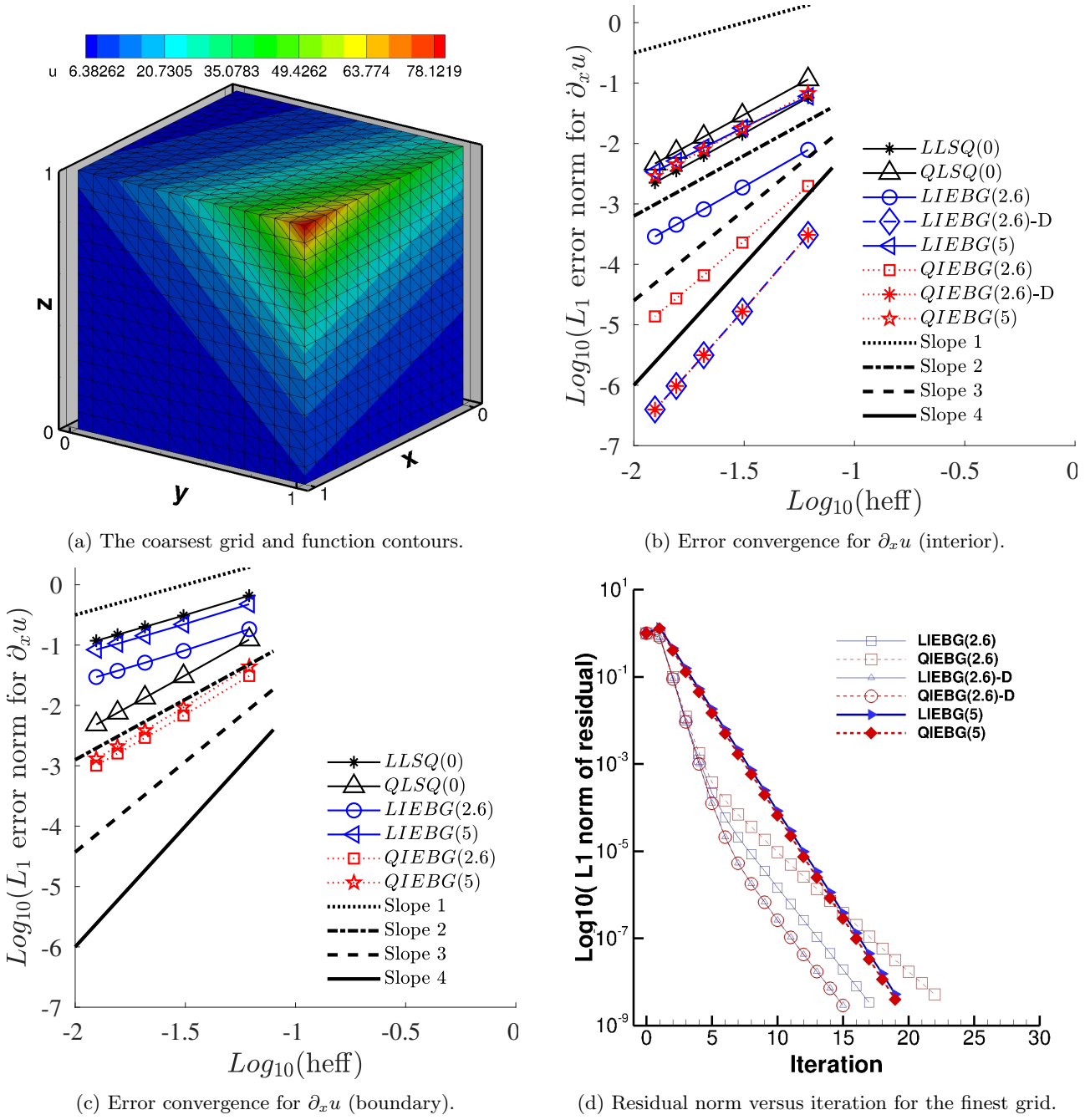
$$u = 1.5 + \sin^2(\xi) + 0.1\exp(\xi), \quad \xi = x + 2.7y + 3z, \tag{84}$$

on a series of regular/irregular tetrahedral grids with 4,096, 32,768, 110,592, and 512,000 nodes and determine the order of gradient error convergence. See Figure 5(a) for the coarsest regular grid and Figure 6(a) for the irregular grid.

Error convergence results for regular tetrahedral grids are shown in Figures 5(b) and 5(c), where only the errors in the x-derivative are presented as results for the other derivatives are similar. Here, LIEBG(2.6)-D and QIEBG(2.6)-D are the linear and quadratic IEBG methods, respectively, with the gradients specified at boundary nodes. As can be seen in Figure 5(b), which are the results for the interior nodes, these methods achieve fourth-order accuracy and also the two methods are identical as expected. For QIEBG(2.6), where the gradients at boundary nodes are not specified but computed by the implicit methods, the gradients are third-order accurate. LIEBG(2.6) gives second-order accurate gradients at interior nodes. These gradients are significantly more accurate than LLSQ and QLSQ gradients, which are both second-order accurate at interior nodes on regular tetrahedral grids. It is observed also that both LIEBG(5) and QIEBG(5) produce second-order accurate gradients at interior nodes.

Figure 5(c) shows the L_1 error convergence results for gradients at boundary nodes. The Gauss-Seidel relaxation scheme is taken to be converged when the residual norm is reduced by eight orders of magnitude. As can be seen, the LLSQ and LIEBG methods give first-order accurate gradients while QLSQ and QIEBG give second-order accurate gradients. These results are expected because stencils at boundaries are not regular and should be taken as an unstructured grid. The results show that increasing c from 2.6 to 5.0 increased the error levels for the LIEBG methods although not so significantly. From the results for interior and boundary

nodes, we see that the gradient accuracy can be one order higher at interior nodes than at boundary nodes on regular tetrahedral grids for the LIEBG method with any c and also for the QIEBG method if the optimal choice $c = 2.6$ is used. This is an interesting and somewhat surprising property for the implicit methods.



(a) The coarsest grid and function contours.

(b) Error convergence for $\partial_x u$ (interior).

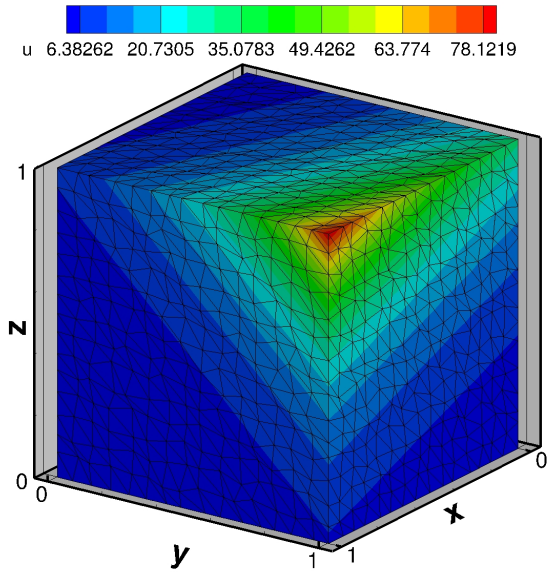
(c) Error convergence for $\partial_x u$ (boundary).

(d) Residual norm versus iteration for the finest grid.

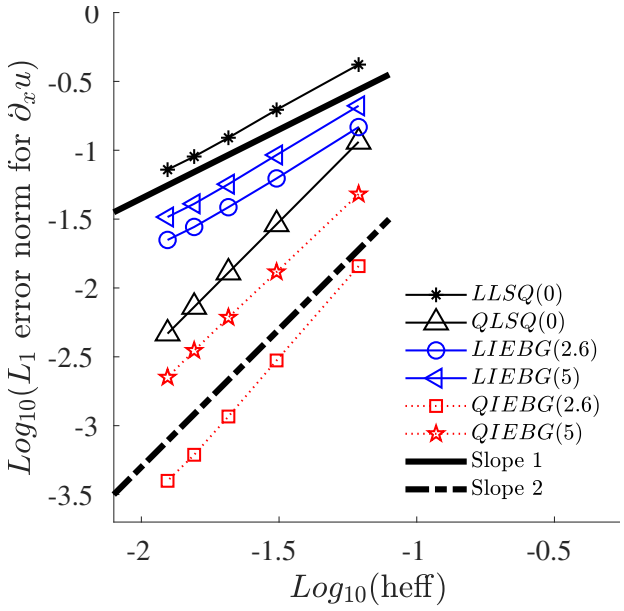
Figure 5: Gradient accuracy verification for regular grids.

Figure 5(d) shows the implicit system residual convergence for the finest grid, where the residual norm is defined as the sum of the L_1 residual norms of three gradient components. First of all, the Gauss-Seidel relaxation scheme converged very quickly for all cases within 25 relaxations. For LIEBG(2.6)-D and QIEBG(2.6)-D, we see identical residual convergence histories, which verifies again that the two methods are identical on regular tetrahedral grids. For $c = 5$, both the LIEBG and QIEBG methods converge rapidly although the convergence rate is larger in the early stage, which is consistent with the stability analysis result in Figure 4. Asymptotically,

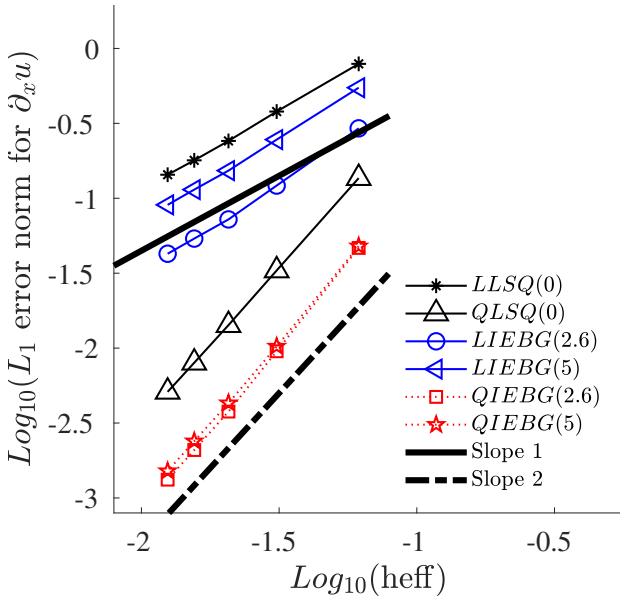
QIEBG(5) shows a faster rate of convergence than QIEBG(2.6) while LIEBG(2.6) and LIEBG(5) show a similar rate of convergence. Nevertheless, this is just one particular case and they do not necessarily behave in the same way for all problems.



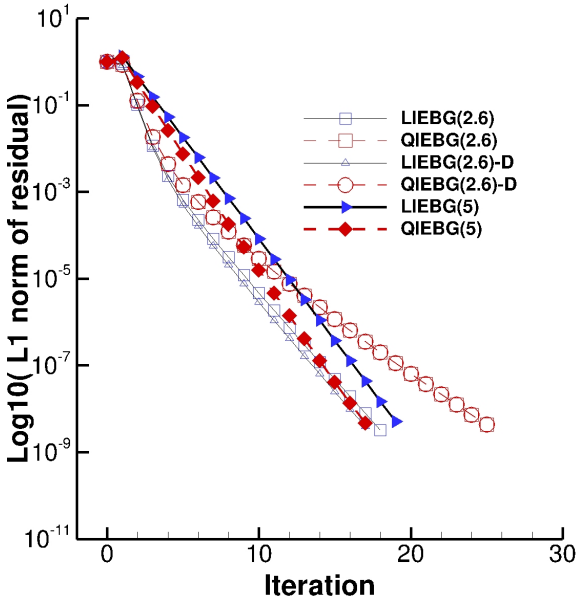
(a) The coarsest grid and function contours.



(b) Error convergence for $\partial_x u$ (interior).



(c) Error convergence for $\partial_x u$ (boundary).



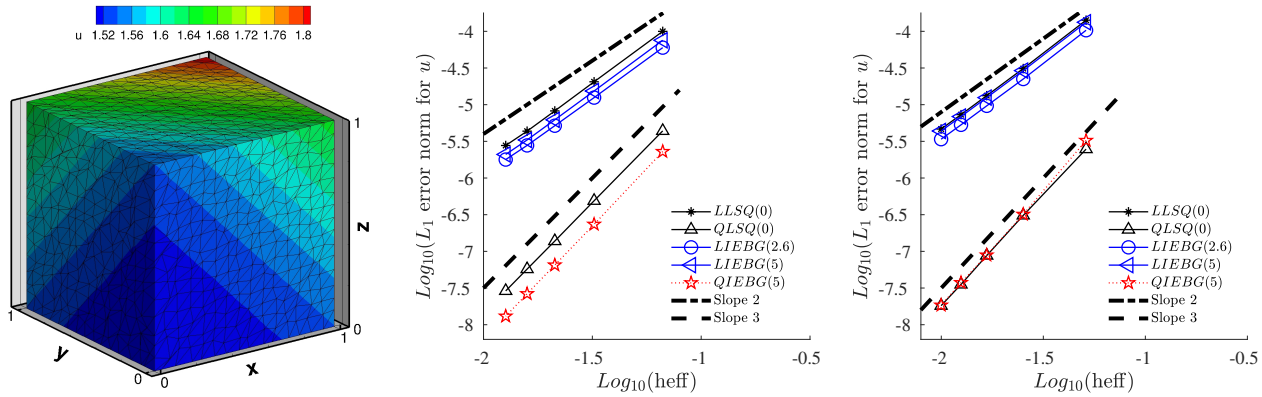
(d) Residual norm versus iteration for the finest grid.

Figure 6: Accuracy verification results for gradient methods (irregular grids).

Next, we consider irregular tetrahedral grids. See Figure 6(a) for the coarsest grid. Error convergence results are shown in Figure 6(b) for interiors nodes and Figure 6(c) for boundary nodes. All methods show expected orders of accuracy: the LLSQ and LIEBG methods are first-order accurate, and the QLSQ and QIEBG methods are second-order accurate. It is observed that the IEBG methods are significantly more accurate than LSQ methods especially with $c = 2.6$ and even with $c = 5$. Figure 6(d) shows the residual convergence for the finest grid. Again, the Gauss-Seidel relaxation scheme converged very rapidly within 30 iterations although it

slows slightly down for QIEBG(2.6).

2. Accuracy Verification for the Euler Equations



(a) Coarsest grid and solution contours.

(b) Error convergence: u, interior.

(c) Error convergence: u, boundary.

Figure 7: Accuracy verification results for second- and third-order edge-based schemes (irregular grids).

In this section, we present accuracy verification results for the second- and third-order edge-based schemes for the Euler equations using the LSQ and IEBG methods. To verify the order of accuracy, we employ the method of manufactured solutions [36], introducing a forcing term vector $\mathbf{s}$ to the Euler equations such that the following functions are the exact solutions:

$$\rho = 1.0 + 0.1\sin^2(\xi), \quad u = 0.3 + 0.1\sin^2(\xi), \quad v = 0.2 + 0.1\sin^2(\xi), \tag{85}$$

$$w = 0.1 + 0.1\sin^2(\xi), \quad p = 1.0 + 0.1\sin^2(\xi). \tag{86}$$

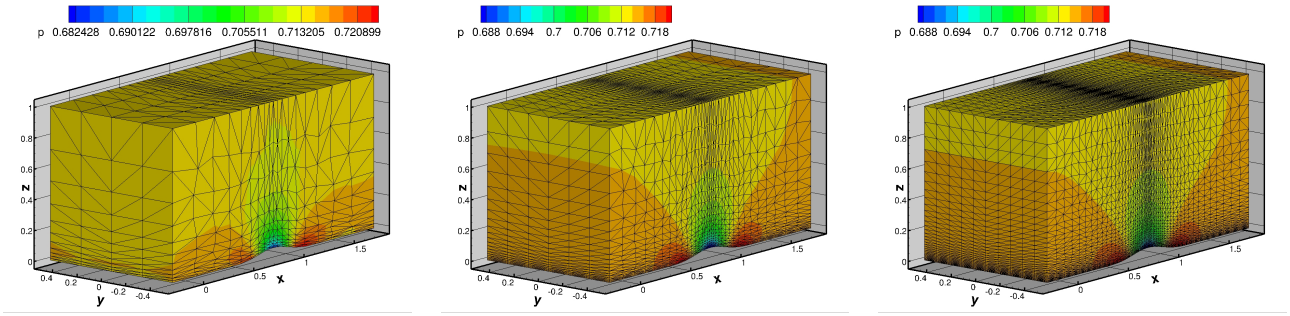
where $\xi = 0.2x + 0.2y + 0.2z$. We solve the residual equations by the implicit defect-correction solver over the same series of irregular tetrahedral grids used in the previous section. The coarsest grid and solution contours are shown in Figure 7(a). A steady state is taken to be reached when the L_1 norm of the residual is reduced by eight orders of magnitude for all equations, including the IEBG equations. At all boundaries, we impose boundary conditions weakly through the numerical flux with the right state given by the exact solution. Here, the implicit defect-correction solver was found to diverge with QIEBG of $c = 2.6$ and therefore we consider only QIEBG with $c = 5$. The solver was found unstable also for weighted LSQ gradients and therefore we consider only unweighted LSQ gradients. Figures 7(b) and 7(c) shows error convergence results for the x-velocity, u. As can be seen, the design orders of accuracy have been confirmed in all the cases. The errors are slightly smaller with the LIEBG and QIEBG methods than with the LSQ methods at interior nodes, but there is no significant difference at boundary nodes.

B. Subsonic Flow over a Smooth Bump

As a more realistic problem, we consider an $M_\infty = 0.3$ subsonic flow over a smooth bump through a duct. The smooth bump defined by

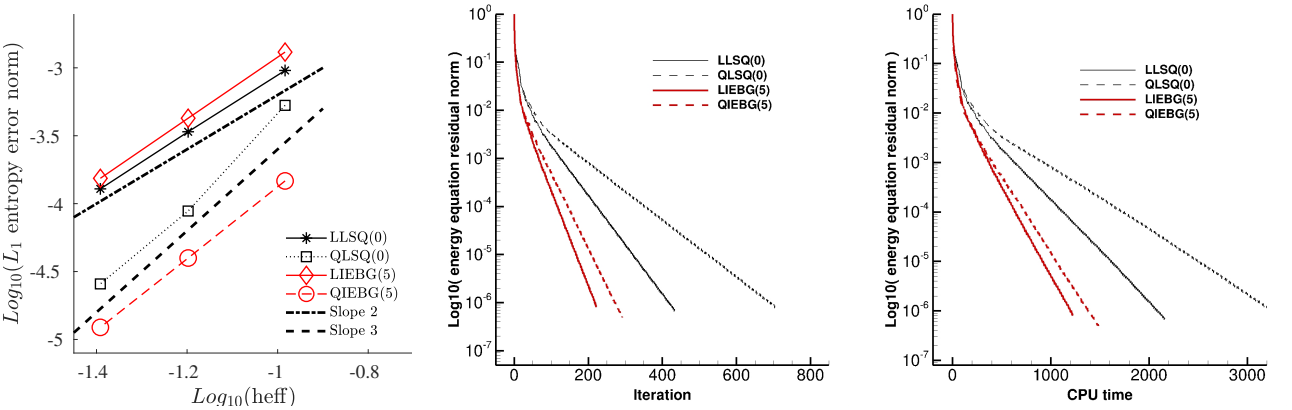
$$z_b = 0.05\sin^4\left[\frac{\pi(10x-3)}{9}\right], \quad x \in [0.3, 1.2], \tag{87}$$

is placed at the bottom of a rectangular domain $(x, y, z) = [-0.25, 1.75] \times [-0.5, 0.5] \times [0, 1]$. We solve this problem by the implicit defect-correction solver with 10 Gauss-Seidel linear relaxations and CFL=10^{15} for relatively coarse irregular tetrahedral grids with 1,140, 4,998, and 19,239 nodes, which are shown in Figures 8(a), 8(b), and 8(c), with the pressure contours obtained with the QIEBG method. All boundary conditions are imposed weakly through the right state: the free stream condition at $x = -0.25$, the back pressure condition



(a) Grid 1. (b) Grid 2. (c) Grid 3.

Figure 8: Subsonic flow over a smooth bump ($M_\infty = 0.3$): tetrahedral grids.



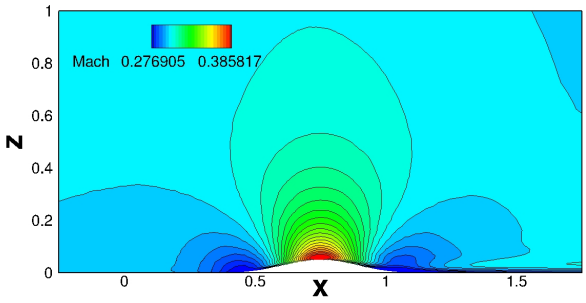
(a) Error convergence. (b) Residual versus iteration (Grid 3). (c) Residual versus CPU time (Grid 3).

Figure 9: Subsonic flow over a smooth bump ($M_\infty = 0.3$): error and residual convergence results.

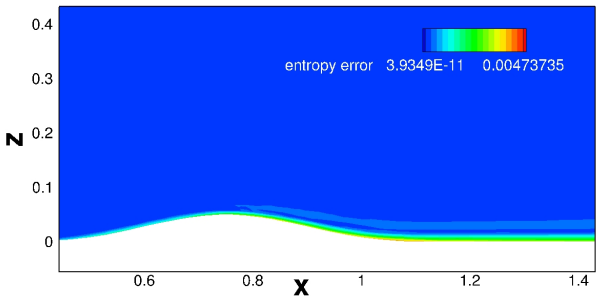
at $x = 1.75$, and the slip-wall condition applied at other boundaries. At a boundary node (x_i, y_i, z_i) over the bump, the exact surface normal vector is given by $(-dz_b/dx, 0, 1)$, which is used to define the right state at a slip-wall node. The solver is taken to be converged when the L_1 norms of the residual equations, including the IEBG equations, are reduced by six orders of magnitude. For this problem, the solver was found to diverge for QIEBG with $c = 2.6$, and we set $c = 5$ for both LIEBG and QIEBG for the sake of robustness. For LSQ gradients, we encountered instability with the weighted LSQ methods and therefore employed unweighted LSQ methods for both linear and quadratic methods.

Figure 9(a) shows entropy error convergence results, where the entropy error is defined by $|\gamma p/\rho^\gamma - 1|$, which should be zero throughout the domain. As can be seen, the LLSQ and IEBG methods yield second-order accuracy while the QLSQ ad QIEBG methods yield third-order accuracy. Although LIEBG gives slightly larger errors than LLSQ, QIEBG gives lower errors than QLSQ. Figure 9(b) shows iterative convergence histories for the energy equation residual. As can be seen, for this problem, the solver converged at significantly fewer iterations with LIEBG and QIEBG than with LLSQ and QLSQ. Also, the solver converged faster by nearly the same factor in CPU time as can be seen in Figure 9(c), implying that the per-iteration cost of the IEBG methods are not very different from that of LSQ methods.

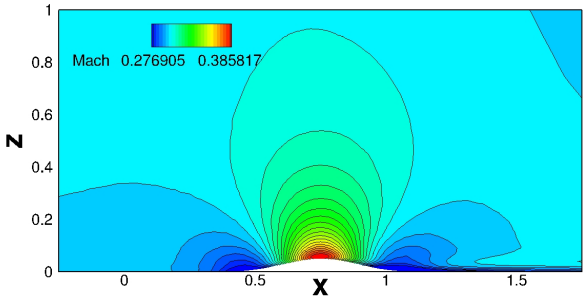
Figure 10 compares Mach and entropy-error contours for the second-order schemes with LLSQ and LIEBG. Here, no significant differences are observed. Figure 11 shows the comparison for the third-order scheme with QLSQ and QIEBG. It can be seen that QIEBG led to significantly smaller entropy errors and produced a more symmetric Mach contours with respect to the line $x = 0.75$.



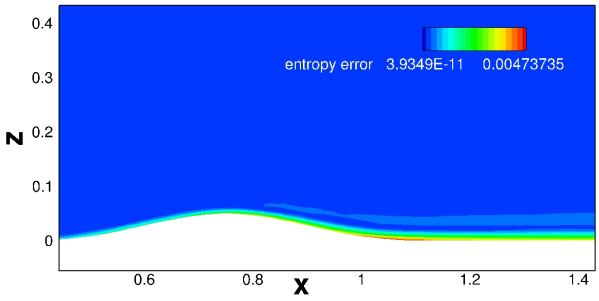
(a) LLSQ: Mach contours.



(b) LLSQ: Entropy error contours.

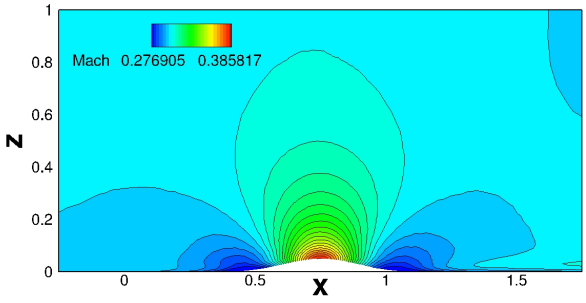


(c) LIEBG: Mach contours.

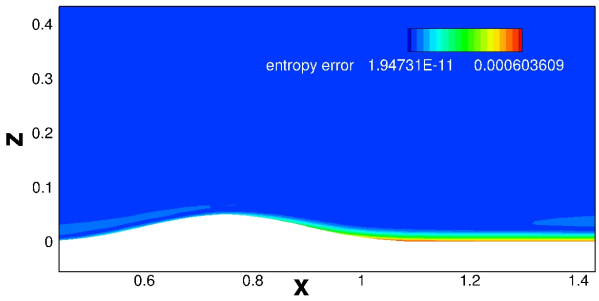


(d) LIEBG: Entropy error contours.

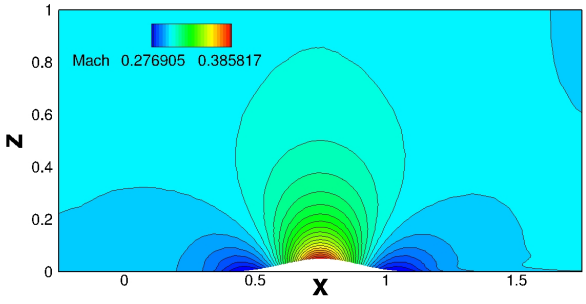
Figure 10: Subsonic flow over a smooth bump ($M_\infty = 0.3$): second-order scheme results (Grid 3).



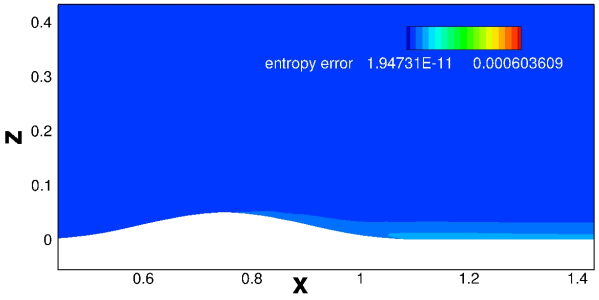
(a) QLSQ: Mach contours.



(b) QLSQ: Entropy error contours.



(c) QIEBG: Mach contours.



(d) QIEBG: Entropy error contours.

Figure 11: Subsonic flow over a smooth bump ($M_\infty = 0.3$): third-order scheme results (Grid 3).

C. Supersonic Flow over a Triangular Bump

Next, we consider an $M_\infty = 2.5$ supersonic flow over a triangular bump defined by the leading edge angle 15° and the height $0.25\tan(15^\circ)$ through a duct $(x, y, z) \in [-0.25, 1.75] \times [-0.5, 0.5] \times [0, 1]$. A supersonic flow enters at $x = -0.25$ and leaves at $x = 1.75$, and all other boundaries are treated as slip walls. The grid is a tetrahedral grid with 12,019 nodes adapted for a first-order accurate solution by the refine package (`https://github.com/nasa/refine`) developed at NASA Langley Research Center. See Figure 12. Note that the first-order scheme was used only for generating an adaptive grid; the second- and third-order schemes are tested on the same adaptive grid. For this problem and a hypersonic-flow problem in the next section, we employed the R_5 limiter [37] with the pressure limiter [38], following Ref. [39]. The CFL number in the approximate Jacobian of the implicit defect-correction solver is set to 5.

Figure 13 shows iterative residual convergence results. As can be seen, the solver converged by five orders of magnitude for LLSQ, LIEBG, and QIEBG while it eventually hangs with QLSQ. Again, the convergence results are similar in CPU time and we see that the per-iteration cost for the IEBG methods is not significantly more expensive than the LSQ methods. Figure 14 shows the comparison of pressure contours for the four methods: second-order accurate results on the left column and third-order accurate results on the right column. All results look similar; the reflected shock is captured more sharply by the third-order scheme with QIEBG.

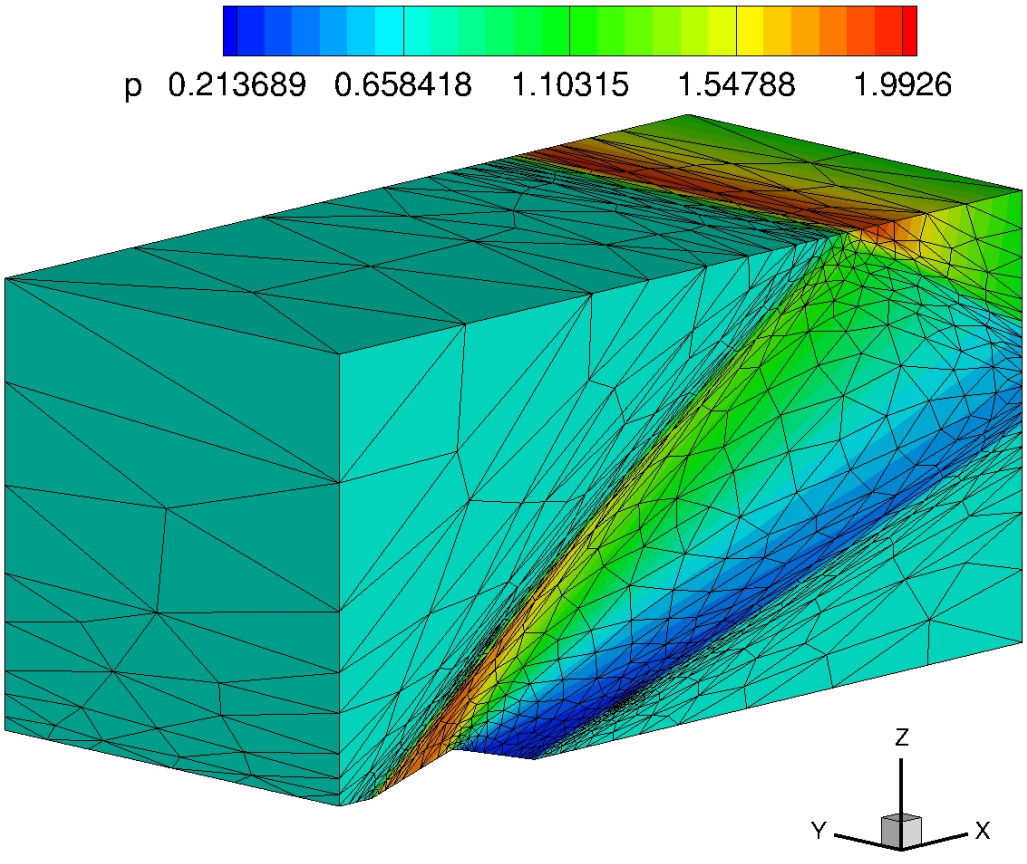
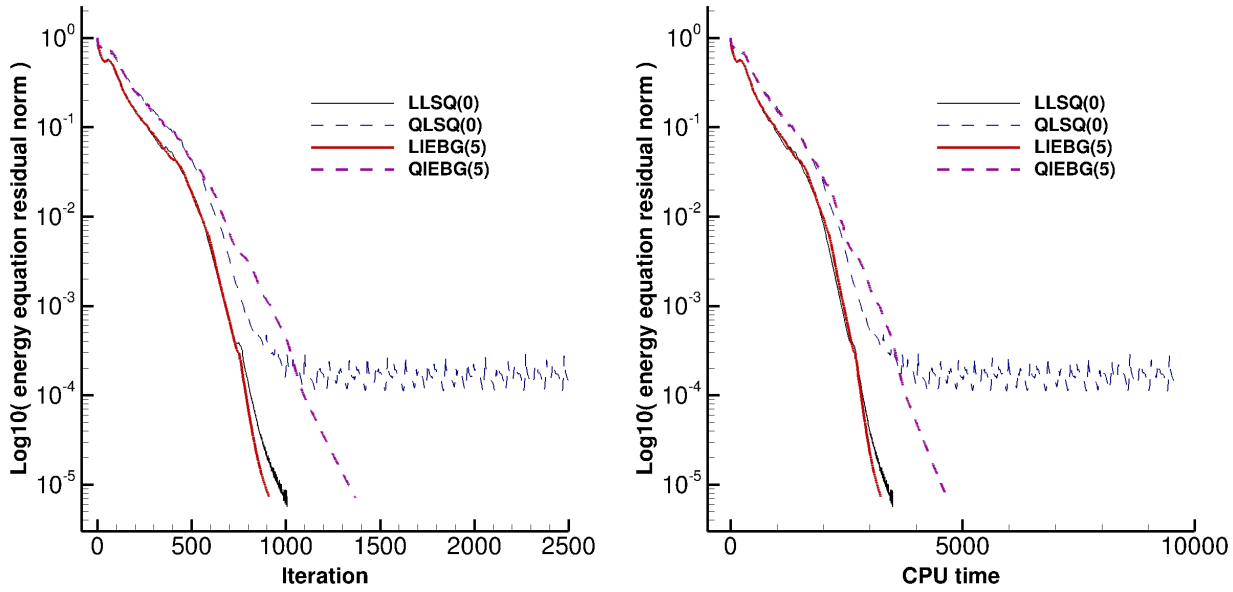


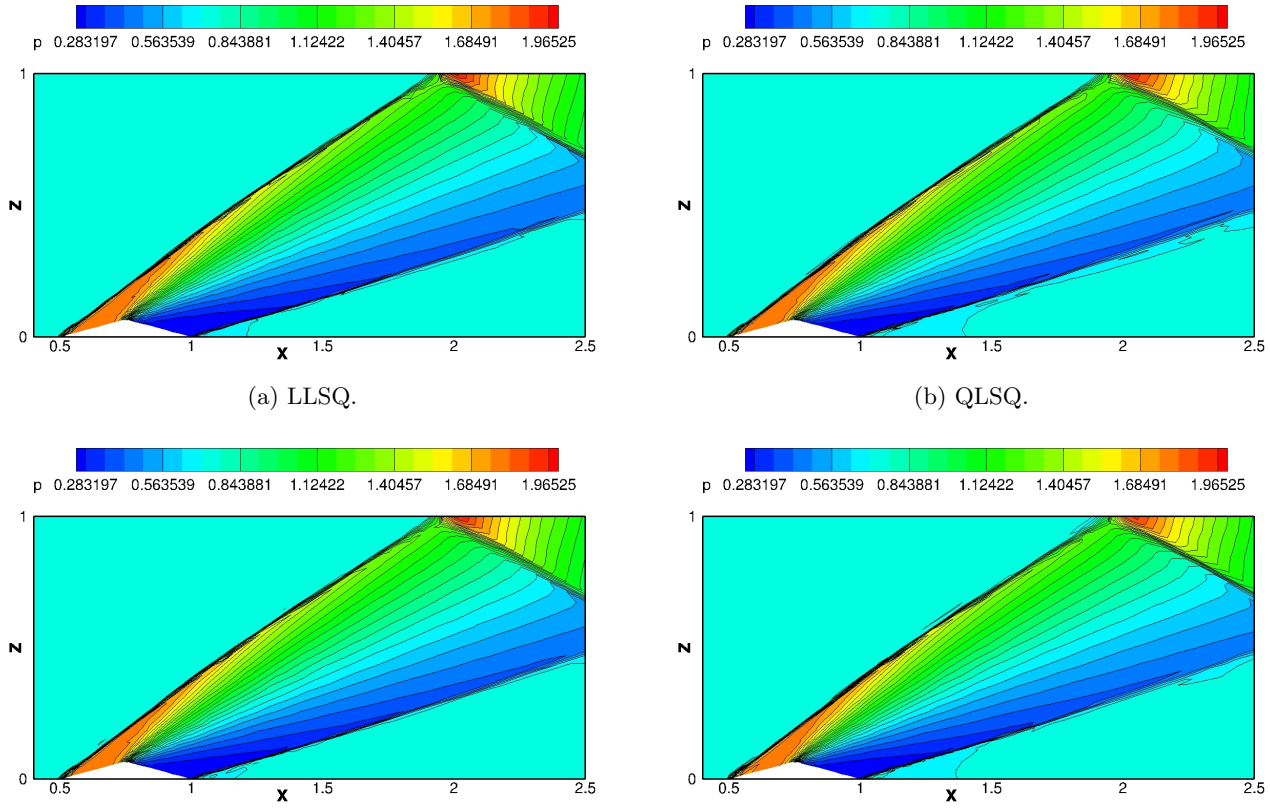
Figure 12: Supersonic flow over a triangular bump ($M_\infty = 2.5$): a tetrahedral grid.



(a) Residual versus iteration.

(b) Residual versus CPU time.

Figure 13: Supersonic flow over a triangular bump ($M_\infty = 2.5$): residual convergence results.



(a) LLSQ.

(b) QLSQ.

(c) LIEBG.

(d) QIEBG.

Figure 14: Supersonic flow over a triangular bump ($M_\infty = 2.5$): pressure contours.

D. Hypersonic Flow over a Cylinder

Finally, we consider an $M_\infty = 5.0$ hypersonic flow over a half circular cylinder. Figure 17(a) shows a tetrahedral grid with 11,275 nodes and a nonuniform spacing in the y-direction. For this problem, we employed the limiter as mentioned in the previous section and also increased the dissipation by setting $C_k = 0.5$ for all waves. The CFL number in the approximate Jacobian of the implicit defect-correction solver is set to 5.

Figures 16(a) and 16(b) show iterative residual convergence results. The solver converged and reduced the residual by five orders of magnitude, at least, with LLSQ, LIEBG, and QIEBG, but diverged with QLSQ. Again, there is no significant differences between the two figures and therefore the per-iteration cost is not significantly different between LSQ and IEBG. Figure 17 shows the comparison of pressure contours for the converged solutions. Here, no significant differences are observed in the solutions. As demonstrated in Ref. [39], the benefit of third-order accuracy may be observed in smooth solution contours on an adaptive anisotropic tetrahedral grid. Nevertheless, the results demonstrate robustness of the IEBG methods for a high-speed strong-shock problem. The IEBG methods should be further tested for viscous-flow problems, where target quantities such as heat fluxes require accurate gradients over a solid surface.

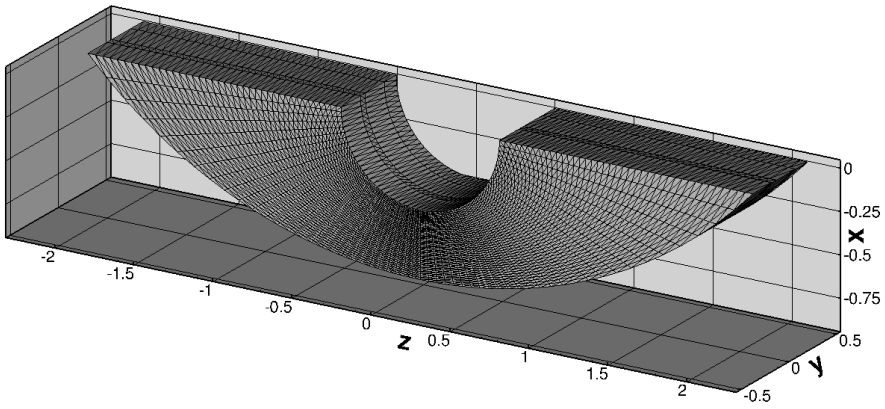
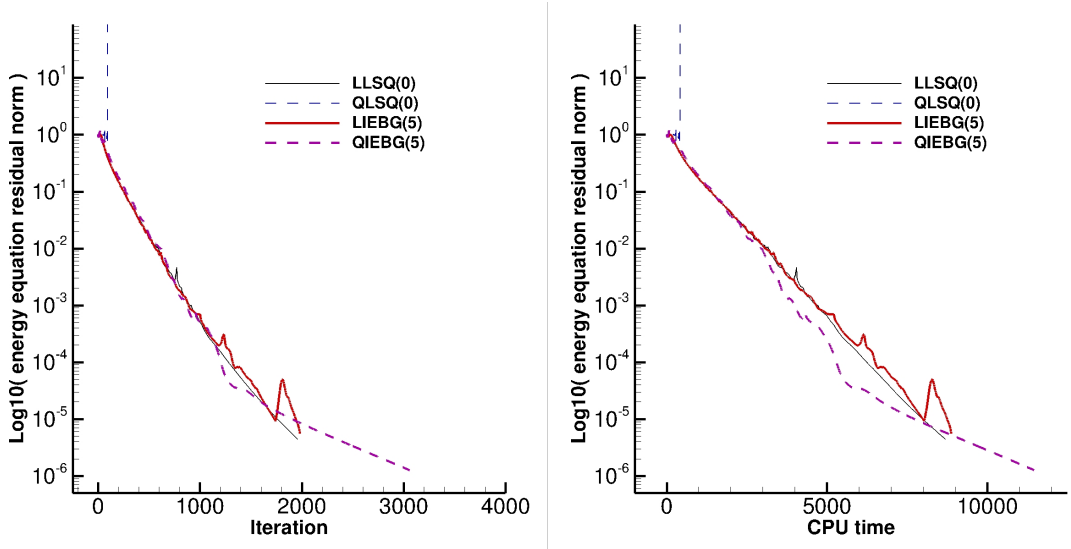


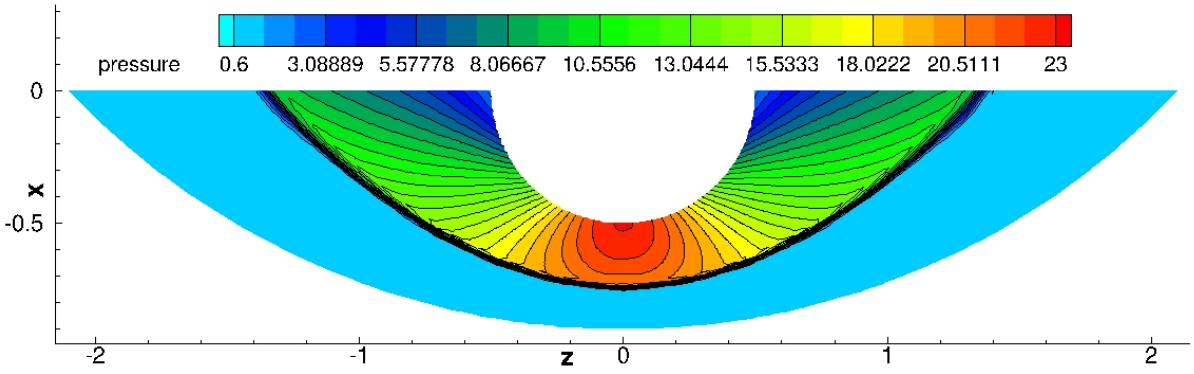
Figure 15: Hypersonic flow over a cylinder ($M_\infty = 5.0$): a tetrahedral grid.



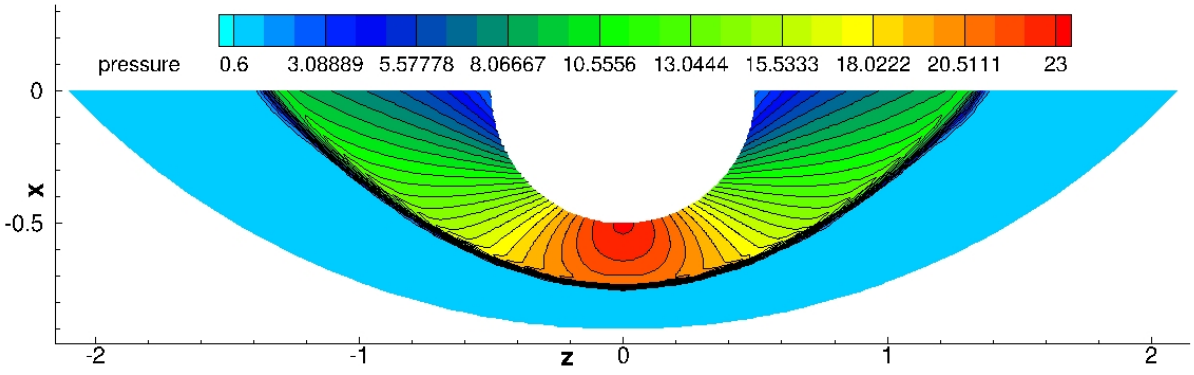
(a) Residual versus iteration.

(b) Residual versus CPU time.

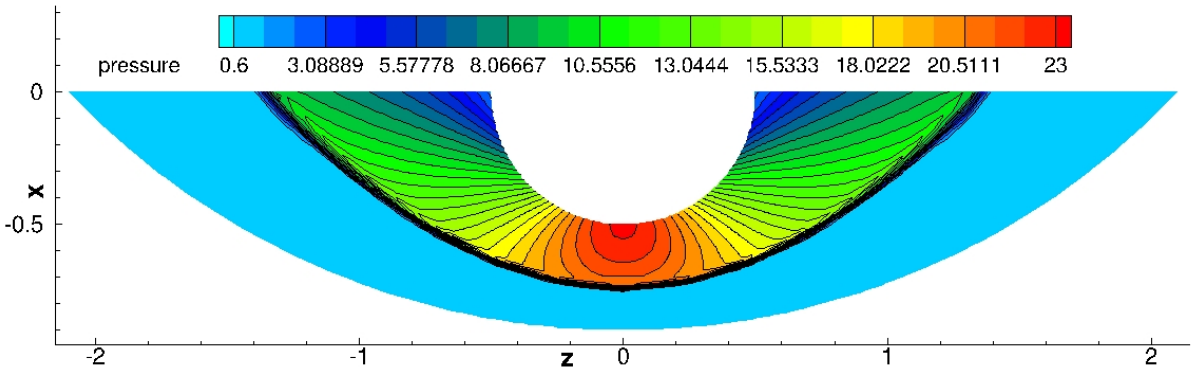
Figure 16: Hypersonic flow over a cylinder ($M_\infty = 5.0$): residual convergence results.



(a) LLSQ.



(b) LIEBG.



(c) QIEBG.

Figure 17: Hypersonic flow over a cylinder ($M_\infty = 5.0$): pressure contours.

VII. Concluding Remarks

In this paper, we reported progress in the development of the IEBG gradient methods for second- and third-order edge-based schemes applied on tetrahedral grids. The following updates are reported: (1) we have shown that the parameter c in the source quadrature formula is a free parameter and can be chosen to be a large value, e.g., $c = 5$, to increase the robustness without losing quadratic exactness, (2) we have shown that the diagonal blocks are actually diagonal matrices, which can be inverted very efficiently even at a boundary node if the accuracy-preserving boundary flux quadrature is employed, (3) an optimal set of parameters κ and c has been derived for achieving fourth-order accuracy on regular tetrahedral grids, and (4) a stability analysis is performed for the Gauss-Seidel relaxation scheme used to solve the IEBG gradient systems on a regular grid and a region of stability is revealed, showing that stability is ensured for a wide range of parameters. Numerical results have been presented from accuracy verification studies to more realistic inviscid-flow problems for subsonic, supersonic, and hypersonic flows. Fourth-order gradient accuracy has been verified at interior nodes of regular tetrahedral grids with exact gradients specified at boundary nodes. Third-order gradient accuracy has been observed at interior nodes in the case that the gradients at boundary nodes are computed as additional unknowns in the implicit gradient methods. For unstructured grids, the linear and quadratic IEBG methods have been shown to produce first- and second-order accurate gradients, as expected. For robustness, we have found that the parameter c needs to be increased and $c = 5$ has been found to work robustly for all the inviscid-flow problems in this paper.

Future work includes applications to viscous problems, where accuracy of gradients is of primal interest towards accurate predictions of the viscous stresses and heat fluxes. It should also be investigated whether the same gradients can be used for the inviscid and viscous schemes without destabilizing the implicit solver, whereas current practical solvers require different gradient methods for solver stability: unweighted LSQ gradients for inviscid schemes and weighted LSQ gradients for viscous schemes [40]. Furthermore, it may be worth developing an edge-loop implementation of the IEBG methods with a different relaxation scheme towards more efficient computations, for example, on GPU. Another area to explore is extensions of these IEBG methods to cell-centered finite-volume methods of nodal gradients [39, 41]. The focus here will be on switching from nodal solutions to cell-centered point-valued solutions and investigating its impact on the implicit system. The IEGB gradient methods have the potential for greatly reducing the cost of gradient computations in the nodal-gradient-based cell-centered finite-volume methods.

Acknowledgments

The author gratefully acknowledges support from the U.S. Army Research Office under the contract/grant number W911NF-19-1-0429 and support by the Hypersonic Technology Project, through the Hypersonic Air-breathing Propulsion Branch of the NASA Langley Research Center, under Prime Contract No. 80LARC23DA003. The author thanks Jeffery White (LaRC), Kyle Thompson (LaRC), and Kyle Anderson (LaRC) for their helpful comments.

References

[1]Barth, T. J., "Numerical Aspects of Computing Viscous High Reynolds Number Flows on Unstructured Meshes," AIAA Paper 91-0721, 1991.

[2]Biedron, R. T., Carlson, J.-R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., Rumsey, C. L., Thomas, J. L., and Wood, W. A., "FUN3D Manual: 12.9," *NASA-TM–2016–219012*, February 2016.

[3]Nakashima, Y., Watanabe, N., and Nishikawa, H., "Development of an Effective Implicit Solver for General-Purpose Unstructured CFD Software," *The 28th Computational Fluid Dynamics Symposium*, C08-1, Tokyo, Japan, 2014.

[4]Kroll, N., Abu-Zurayk, M., Dimitrov, D., Franz, T., Führer, T., Gerhold, T., Görtz, S., Heinrich, R., Ilic, C., Jepsen, J., Jägersküpper, J., Kruse, M., Krumbein, A., Langer, S., Liu, D., Liepelt, R., Reimer, L., Ritter, M., Schwöppe, A., Scherer, J., Spiering, F., Thormann, R., Togiti, V., Vollmer, D., and Wendisch, J.-H., "DLR Project Digital-X: Towards Virtual Aircraft Design and Flight Testing Based on High-Fidelity Methods," *CEAS Aeronautical Journal*, Vol. 7, No. 1, March 2016, pp. 3–27.

[5]Mavriplis, D. J. and Long, M., "NSU3D Results for the Fourth AIAA Drag Prediction Workshop," AIAA Paper 2010-4363, 2010.

[6]Luo, H., Baum, J. D., and Löhner, R., "High-Reynolds Number Viscous Flow Computations using an Unstructured-Grid Method," *Proc. of 42nd AIAA Aerospace Sciences Meeting*, AIAA Paper 2004-1103, Reno, NV, 2004.

[7]Kozubskaya, T., Abalakin, I., Dervieux, A., and Ouvrard, H., "Accuracy Improvement for Finite-Volume Vertex-Centered

Schemes Solving Aeroacoustics Problems on Unstructured Meshes," *Proc. of 16th AIAA/CEAS Aeroacoustics Conference*, AIAA Paper 2010-3933, 2010.

[8]Haselbacher, A. and Blazek, J., "Accurate and Efficient Discretization of Navier-Stokes Equations on Mixed Grids," *AIAA J.*, Vol. 38, No. 11, 2000, pp. 2094–2102.

[9]Smith, T., Ober, C., and Lorber, A., "SIERRA/Premo-A New General Purpose Compressible Flow Simulation Code," *Proc. of 32nd AIAA Fluid Dynamics Conference and Exhibit*, AIAA Paper 2002-3292, St. Louis, MO, 2002.

[10]Eliasson, P., "EDGE, a Navier-Stokes Solver, for Unstructured Grids," Technical Report FOI-R-0298-SE, Swedish Defence Research Agency, December 2001.

[11]Fezoui, L. and Stouffle, B., "A Class of Implicit Upwind Schemes for Euler Simulations with Unstructured Meshes," *J. Comput. Phys.*, Vol. 84, 1989, pp. 174–206.

[12]Gao, S., Habashi, W. G., Fossati, M., Isola, D., and Baruzzi, G. S., "Finite-Element Formulation of a Jacobian-free Solver for Supersonic Viscous Flows on Hybrid Grids," *55th AIAA Aerospace Sciences Meeting*, AIAA Paper 2017-0085, Grapevine, Texas, 2017.

[13]Katz, A. and Sankaran, V., "Mesh Quality Effects on the Accuracy of CFD Solutions on Unstructured Meshes," *J. Comput. Phys.*, Vol. 230, 2011, pp. 7670–7686.

[14]Katz, A. and Sankaran, V., "An Efficient Correction Method to Obtain a Formally Third-Order Accurate Flow Solver for Node-Centered Unstructured Grids," *J. Sci. Comput.*, Vol. 51, 2012, pp. 375–393.

[15]Diskin, B. and Thomas, J. L., "Effects of Mesh Regularity on Accuracy of Finite-Volume Schemes," *Proc. of 50th AIAA Aerospace Sciences Meeting*, AIAA Paper 2012-0609, Nashville, Tennessee, 2012.

[16]Liu, Y. and Nishikawa, H., "Third-Order Inviscid and Second-Order Hyperbolic Navier-Stokes Solvers for Three-Dimensional Inviscid and Viscous Flows," *46th AIAA Fluid Dynamics Conference*, AIAA Paper 2016-3969, Washington, D.C., 2016.

[17]Nishikawa, H., "Accuracy-Preserving Boundary Flux Quadrature for Finite-Volume Discretization on Unstructured Grids," *J. Comput. Phys.*, Vol. 281, 2015, pp. 518–555.

[18]Diskin, B. and Thomas, J. L., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Inviscid Fluxes," *AIAA J.*, Vol. 49, No. 4, 2011, pp. 836–854.

[19]Nishikawa, H., "First, Second, and Third Order Finite-Volume Schemes for Advection-Diffusion," *J. Comput. Phys.*, Vol. 273, 2014, pp. 287–309.

[20]Haider, F., Croisille, J.-P., and Courbet, B., "Stability Analysis of the Cell Centered Finite-Volume MUSCL Method on Unstructured Grids," *Numer. Math.*, Vol. 113, No. 4, 2009, pp. 555–600.

[21]Wang, Q., Ren, Y.-X., Pan, J., and Li, W., "Compact High Order Finite Volume Method on Unstructured Grids III: Variational Reconstruction," *J. Comput. Phys.*, Vol. 337, 2017, pp. 1–26.

[22]Li, L., Liu, X., Lou, J., Luo, H., Nishikawa, H., and Ren, Y., "A Finite Volume Method Based on Variational Reconstruction for Compressible Flows on Arbitrary Grids," *23rd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2017-3097, Denver, Colorado, 2017.

[23]Li, L., Liu, X., Lou, J., Luo, H., Nishikawa, H., and Ren, Y., "A Discontinuous Galerkin Method Based on Variational Reconstruction for Compressible Flows on Arbitrary Grids," *56th AIAA Aerospace Sciences Meeting*, AIAA Paper 2018-0831, Kissimmee, Florida, 2018.

[24]Nishikawa, H., "From Hyperbolic Diffusion Scheme to Gradient Method: Implicit Green-Gauss Gradients for Unstructured Grids," *J. Comput. Phys.*, Vol. 372, 2018, pp. 126–160.

[25]Nishikawa, H., "An Implicit Gradient Method for Cell-Centered Finite-Volume Solver on Unstructured Grids," *AIAA Scitech 2019 Forum*, AIAA Paper 2019-1155, San Diego, CA, 2019.

[26]Nishikawa, H., "Implicit Edge-Based Gradients for Simplex Grids," *AIAA Aviation 2020 Forum*, AIAA Paper 2020-3048, 2020.

[27]Nishikawa, H., "Beyond Interface Gradient: A General Principle for Constructing Diffusion Schemes," *Proc. of 40th AIAA Fluid Dynamics Conference and Exhibit*, AIAA Paper 2010-5093, Chicago, IL, 2010.

[28]Nishikawa, H., "A First-Order System Approach for Diffusion Equation. I: Second Order Residual Distribution Schemes," *J. Comput. Phys.*, Vol. 227, 2007, pp. 315–352.

[29]Burg, C. O. E., "Higher Order Variable Extrapolation for Unstructured Finite Volume RANS Flow Solvers," *17th AIAA Computational Fluid Dynaamics Conference*, AIAA Paper 2005-4999, Orlando, FL, 2005.

[30]Nishikawa, H., "First-, Second-, and Third-Order Finite-Volume Schemes for Diffusion," *J. Comput. Phys.*, Vol. 256, 2014, pp. 791–805.

[31]Nishikawa, H. and Liu, Y., "Accuracy-Preserving Source Term Quadrature for Third-Order Edge-Based Discretization," *J. Comput. Phys.*, Vol. 344, 2017, pp. 595–622.

[32]Diskin, B. and Thomas, J. L., "Notes on Accuracy of Finite-Volume Discretization Schemes on Irregular Grids," *Appl. Numer. Math.*, Vol. 60, 2010, pp. 224–226.

[33]Diskin, B. and Thomas, J. L., "Accuracy Analysis for Mixed-Element Finite-Volume Discretization Schemes," *NIA Report No. 2007-08*, 2007.

[34]Diskin, B., "Notes on a compact third order edge-based node-centered scheme," Unpublished, March 2011.

[35]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *J. Comput. Phys.*, Vol. 43, 1981, pp. 357–372.

[36]Oberkampf, W. L. and Roy, C. J., *Verification and Validation in Scientific Computing*, Cambridge University Press, 2010.

[37]Nishikawa, H., "New Unstructured-Grid Limiter Functions," *SciTech 2022 Forum*, AIAA Paper 2022-1374, San Diego, CA, 2022.

[38]Gnoffo, P. A., "Updates to Multi-Dimensional Flux Reconstruction for Hypersonic Simulations on Tetrahedral Grids," AIAA Paper 2010-1271, January 2010.

[39]Nishikawa, H., White, J. A., and O'connell, M. C., "Toward a Third-Order Accurate, Second Derivative-Free, Shock-Capturing Finite-Volume Method for Hypersonic Flows on Tetrahedral Grids," *Proc. of AIAA Aviation 2023 Forum*, AIAA Paper 2023-4418, San Diego, CA and Online, 2023.

[40]White, J. A., Nishikawa, H., and Baurle, R. A., "Weighted Least-squares Cell-Average Gradient Construction Methods for the VULCAN-CFD Second-Order Accurate Unstructured-Grid Cell-Centered Finite-Volume Solver," *AIAA Scitech 2019 Forum*, AIAA Paper 2019-0127, San Diego, CA, 2019.

[41]Nishikawa, H. and White, J. A., "An Efficient Quadratic Interpolation Scheme for a Third-Order Cell-Centered Finite-Volume Method on Tetrahedral Grids," *J. Comput. Phys.*, Vol. 490, 2023, pp. 112324.