Towards Robust and Accurate Implicit Gradient Methods for Second- and Third-Order Nodal-Gradient Cell-Centered Finite-Volume Discretizations on Tetrahedral Grids

Hiroaki Nishikawa*
National Institute of Aerospace, Hampton, VA 23666, USA

Abstract

In this paper, we introduce implicit gradient methods as alternatives to conventional least-squares gradient methods for second- and third-order nodal-gradient cell-centered finite-volume discretizations, where solutions are stored at cells but gradients are stored at nodes. Because of the unique configuration of solutions and gradients, implicit gradient systems developed for the node-centered edge-based discretization method can be directly applied once the numerical solutions are interpolated from cells to nodes with sufficient accuracy. The resulting defect-correction solver can be loosely coupled with a flow-equation solver, and at convergence, solutions and gradients that satisfy the corresponding residual equations are obtained. Each iteration is relatively cheap compared with least-squares methods involving hundreds of neighbors. Numerical results are presented for accuracy verification studies and some simple but realistic flow problems.

1 Introduction

In a recent paper [1], we introduced a unique and economical third-order cell-centered finite-volume (CCFV) discretization method for tetrahedral grids, by extending the face/cell-averaged nodal-gradient (F/C-ANG) CCFV method [2, 3, 4] with an efficient quadratic solution interpolation scheme utilizing nodal gradients. This third-order method is economical compared with conventional third-order CCFV unstructured-grid discretization methods [5, 6, 7, 8, 9, 10, 11, 12] for several reasons: (1) there is no need to compute and store second derivatives of solution variables, (2) it only requires a single numerical flux per face, (3) it achieves third-order accuracy with linear tetrahedral grids even for curved geometries, similarly to other economical third-order methods [13, 14], and (4) the third-order residual equations can be solved robustly by an implicit defect-correction solver, which is widely employed and available in practical second-order CFD solvers [15]. However, least-squares (LSQ) methods used for computing solution gradients at nodes require a large number of neighbor cells. Often, hundreds of neighbor cells are needed in quadratic LSQ methods for robustness; therefore, it is desirable to reduce the cost of gradient computations and simplify the algorithm without needing to access hundreds of neighbors. There are two possible approaches to address this problem. One is to reduce the number of neighbors through smart stencil augmentation techniques such as those in Refs [16, 17]. The other is to employ implicit gradient methods [18, 19, 20, 21, 22, 23], where we solve a globally-coupled linear system to obtain gradients. As discussed in Refs. [19, 20, 22], the latter can be an efficient alternative to LSQ methods if a single linear relaxation is performed per flow-solver iteration. Moreover, we typically need to access only the face neighbor cells in each relaxation. Therefore, it can potentially be more efficient than processing hundreds of neighbors as in the LSQ methods. Also, it is known that implicit defect-correction flow-equation solvers tend to be more stable when using implicit gradients instead of using explicit LSQ gradients [19, 20, 22]. However, implicit gradients methods are currently not available for the economical third-order CCFV method and their second-order versions, where solutions are stored at cells but gradients are stored at nodes. In this paper, we introduce implicit gradient methods suitable for these discretization methods.

For second-order discretization methods [2, 3, 4], we seek algorithms that can produce exact gradients for linear functions. For the economical third-order CCFV method [1, 15], we need quadratically-exact algorithms and wish to preserve the second-derivative-free feature. Typically, a quadratic implicit gradient method requires

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

the storage of second derivatives [18, 19, 20, 21, 22, 23] and the solution of a global linear system for both gradients and second derivatives. This approach greatly increases the cost of storing unknowns and solving the linear system with six additional unknowns (six second derivatives) per cell/node, compared with a linear implicit gradient method (3 unknowns per cell/node). For a conventional third-order CCFV method, the additional cost cannot be avoided because both gradients and second derivatives are needed for the third-order discretization. However, for the economical third-order discretization method [1, 15], which does not require second derivatives, it is desirable to avoid the additional cost of storing and computing second derivatives because we only need gradients in the discretization. To achieve this, we take advantage of the special configuration of the economical third-order discretization method: gradients at nodes and solutions at cells, and consider utilizing the implicit edge-based gradient (IEBG) methods developed for the node-centered third-order edge-based discretization method [24]. In the IEBG methods, gradients are computed in a linearly or quadratically exact manner for a given set of functions stored at nodes. The quadratic IEBG method is particularly efficient because it does not require second derivatives at all and produces second-order accurate gradients on arbitrary triangular/tetrahedral grids. It is perhaps the only second-derivative-free quadratic implicit gradient method currently available. The IEBG methods are directly applicable to the nodal-gradient CCFV methods once the solution is transferred from cells to nodes. In this paper, we focus on a simple approach where we average solutions interpolated linearly or quadratically from the cells to the nodes and apply the IEBG methods to compute the gradients at nodes. This results in a defect-correction solver, where a preconditioner matrix is relaxed directly by using the IEBG method [24]. It is more expensive than the direct relaxation as done in Ref. [24], but is not more expensive than LSQ methods per flow-solver iteration, and has the capability of delivering superior gradient accuracy without destabilizing an implicit flow-equation solver as we show in this paper.

2 Target Governing Equations

In this paper, we consider the Euler equations:

$$\int_V \frac{\partial \mathbf{u}}{\partial t}\, dV + \oint_{\partial V} \mathcal{F}\hat{\mathbf{n}} ds = \int_V \mathbf{s}\, dV, \tag{1}$$

where V and ∂V denote a control volume and its boundary, respectively, $\mathbf{s}$ is a vector of source terms, $\hat{\mathbf{n}}$ is a unit outward normal vector of the infinitesimal surface area ds,

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \end{bmatrix}, \quad \mathcal{F} = \begin{bmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\otimes\mathbf{v} + p\mathbf{I} \\ \rho\mathbf{v}H \end{bmatrix}, \tag{2}$$

where $\otimes$ denotes the dyadic product, $\mathbf{I}$ is the 3×3 identity matrix, ρ is the density, $\mathbf{v} = (u, v, w)^t$ is the column vector of the velocity (the superscript t denotes transpose) with Cartesian components u, v, and w, p is the static pressure, and $H = \gamma p/\{\rho(\gamma-1)\} + \mathbf{v}\cdot\mathbf{v}/2$ is the specific total enthalpy with $\gamma = 1.4$ (air), E is the specific total energy, $E = H - p/\rho$, and $\mathbf{s}$ is a source term vector. The system is nondimensionalized by free-stream values in a similar manner as described in Ref. [25] and closed by the nondimensionalized ideal gas law:

$$p = \rho\mathcal{T}/\gamma, \tag{3}$$

where $\mathcal{T}$ is the static temperature.

3 Economical Third-Order Cell-Centered Finite-Volume Discretization

In this section, we give a brief overview of the third-order CCFV method originally developed in Ref. [1]. In a tetrahedral grid, storing the numerical solutions at cells and gradients at nodes, we discretize the system (1) over a cell j as

$$V_j \frac{\partial \mathbf{u}_j}{\partial t} + \sum_{k\in\{k_j\}} \Phi_{jk}(\mathbf{w}_L, \mathbf{w}_R)|\mathbf{n}_T| - \mathbf{s}_j V_j = \delta\mathbf{f}_j + \delta\mathbf{s}'_j, \tag{4}$$

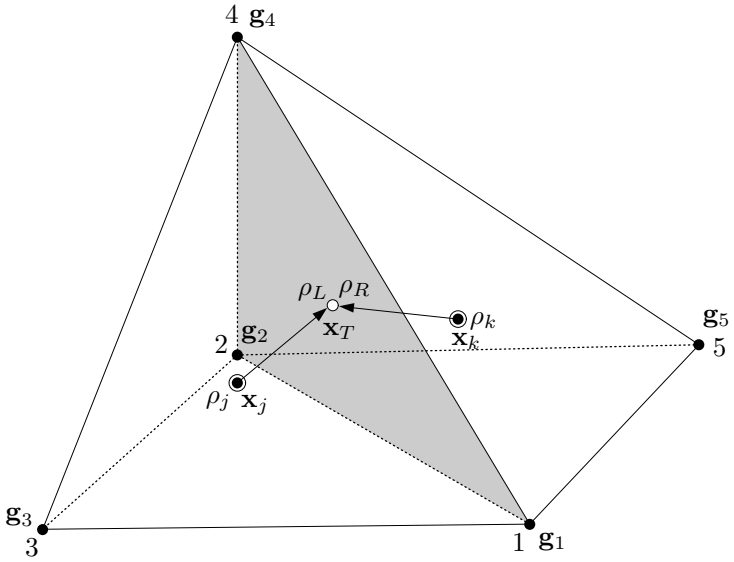


Figure 1: A face between two tetrahedral cells, j and k: ρ_j and ρ_k are the solutions stored at the centroids $\mathbf{x}_j$ and $\mathbf{x}_k$; $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{g}_3$, $\mathbf{g}_4$, and $\mathbf{g}_5$ are the gradients stored at nodes. As an example, the solutions reconstructed at the face centroid from the left and right cells are indicated as ρ_L and ρ_R, respectively.

where $\delta\mathbf{f}_j$ and $\delta\mathbf{s}'_j$ are the flux and source corrections, respectively, needed for third-order accuracy:

$$\delta\mathbf{f}_j = -\frac{1}{24}\sum_{k\in\{k_j\}}\sum_{i\in\{i_T\}}\left(\frac{\partial\mathbf{f}}{\partial\mathbf{w}}\right)_i\nabla\mathbf{w}_i(\mathbf{x}_i-\mathbf{x}_T)|\mathbf{n}_T|, \quad \delta\mathbf{s}'_j = \frac{1}{40}\sum_{i\in\{i_j\}}\nabla\left[\mathbf{s}_i-\left(\frac{\partial\mathbf{u}_i}{\partial t}\right)\right](\mathbf{x}_i-\mathbf{x}_j)V_j. \tag{5}$$

Here, $\mathbf{u}_j$ and $\mathbf{s}_j$ are point values of the conservative variable vector and the source term at the cell centroid $\mathbf{x}_j$, Φ_{jk} is a numerical flux evaluated at the geometric centroid $\mathbf{x}_T$ of the triangular face T between the cell j and its face neighbor cell $k\in\{k_j\}$, V_j is the volume of the cell j, $\{k_j\}$ is a set of neighbor cells, $\mathbf{f}=\mathcal{F}\mathbf{n}_T/|\mathbf{n}_T|$ is a flux projected along the face normal direction, $\mathbf{n}_T$ is an outward normal vector of the face T, $|\mathbf{n}_T|$ is the area of the face, $\mathbf{w}$ denotes the vector of primitive variables $\mathbf{w}=(\rho,\mathbf{v},p)$, $\{i_T\}$ is a set of nodes of the face T, and $\mathbf{x}_i$ is the position of the node $i\in\{i_T\}$, and $\{i_j\}$ in $\delta\mathbf{s}'_j$ is a set of nodes of the cell j. See Figure 1. The left and right states, $\mathbf{w}_L$ and $\mathbf{w}_R$, are computed by the following interpolation scheme, expressed for the density ρ,

$$\rho_L = \rho_j + \left[C\,\overline{\mathbf{g}}_T + (1-C)\overline{\mathbf{g}}_j\right]\cdot(\mathbf{x}_T-\mathbf{x}_j), \tag{6}$$

$$\rho_R = \rho_k + \left[C\,\overline{\mathbf{g}}_T + (1-C)\overline{\mathbf{g}}_k\right]\cdot(\mathbf{x}_T-\mathbf{x}_k), \tag{7}$$

where $\mathbf{g}=\nabla\rho$, C is a constant, and

$$\overline{\mathbf{g}}_T = \frac{1}{3}\sum_{i\in\{i_T\}}\mathbf{g}_i, \quad \overline{\mathbf{g}}_j = \frac{1}{4}\sum_{i\in\{i_j\}}\mathbf{g}_i, \quad \overline{\mathbf{g}}_k = \frac{1}{4}\sum_{i\in\{i_k\}}\mathbf{g}_i, \tag{8}$$

where $\{i_k\}$ is a set of nodes of the cell k. The left and right values of the other solution variables, u, v, w, p, are computed similarly. The solution interpolations as defined in the above are exact for quadratic solutions with $C=1/2$, which is used for the third-order method; otherwise, they are exact for linear solutions. The cases $C=0$ and $C=1$ correspond, respectively, to the cell- and face-averaged nodal gradient schemes (CANG and FANG) [2, 3].

Note that the left hand side of Eq. (4) is a typical second-order finite-volume discretization if a linear solution interpolation (e.g., $C=0$) is employed instead of the quadratic solution interpolation ($C=1/2$). The first term on the right hand side, $\delta\mathbf{f}_j$, is a flux correction term required for quadratic exactness of the flux integral, and the second term, $\delta\mathbf{s}'_j$, consists of correction terms for the source and time-derivative terms. These forms of high-order surface and volume integration formulas have been derived in Ref. [1], by following the k-exact finite-volume discretization approach [26, 12, 27, 28]. The formulation with correction terms is useful because we can upgrade an existing second-order code to third-order accuracy just by replacing a linear

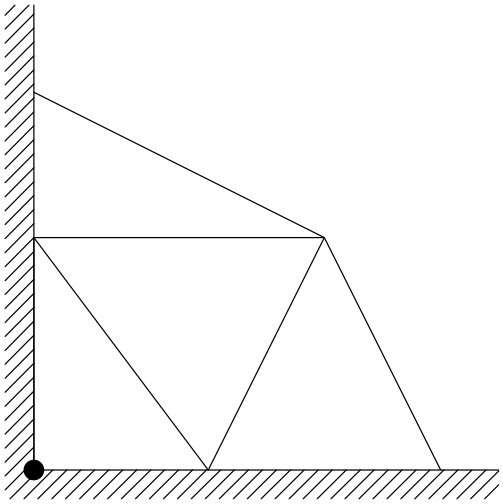


Figure 2: A node at a corner with only one neighbor cell.

solution interpolation/reconstruction ($C \neq 1/2$) by the quadratic solution interpolation ($C = 1/2$) and adding the correction terms. Note that these correction terms do not require second derivatives, making the third-order accurate discretization (4) completely second-derivative free [1].

The choice of the gradient method is left open at this point, but storing gradients at nodes brings the benefit for tetrahedral grids that the memory required to store the nodal gradients is greatly reduced, compared with that of a conventional CCFV method, where gradients are computed and stored at cells because the number of nodes is toughly 1/6 of the number of cells [1, 2, 3, 4]. This is one of the favorable features of the nodal-gradient CCFV methods. In the next section, we first review the explicit LSQ methods, which have been chosen for the nodal gradient in the previous studies, and then discuss implicit gradient methods.

4 Explicit Gradient Methods for Nodal Gradients

In the previous studies, we computed the nodal gradients by LSQ methods: linear and quadratic LSQ methods for second- and third-order discretizations, respectively [1, 2, 3, 4]. For the linear LSQ method, we fit a linear polynomial over a set of neighbors including the solution at a node as an additional unknown. For the quadratic method, we add second derivatives as additional unknowns. The LSQ problems are solved for a given grid and the resulting coefficients are stored only for gradients, leading to the form for both the linear and quadratic methods:

$$\mathbf{g}_i = \sum_{k \in \{k_i^{LSQ}\}} \mathbf{Q}_{ik}^{LSQ} \rho_i, \tag{9}$$

at a node i, where $\{k_i^{LSQ}\}$ is a set of neighbors and $\mathbf{Q}_{ik}^{LSQ}$ is a vector of LSQ coefficients corresponding to the gradient. For the linear LSQ method, the set $\{k_i^{LSQ}\}$ is defined typically by the cells sharing the node i. At a boundary node, however, typically there are insufficient neighboring cells and the LSQ problem can be ill-posed. For example, a node at a corner can have only one cell as shown in Figure 2. To ensure the LSQ problem is well posed, we add all cells sharing at least one node with the cells around a boundary node. In Figure 2, this approach results in the total of four cells for the corner node, which is sufficient for the linear LSQ problem to be well posed. See for Ref. [1] for details.

For the quadratic LSQ method, for the sake of robustness, we add cells sharing at least one node with the cells sharing the node i, which is called the node-neighbor augmentation [4, 29]. As suggested in Ref. [29], the node-neighbor augmentation is employed here also in the linear LSQ method for robustness. At a boundary node, the node-neighbor augmentation is applied to the stencil already augmented as explained in the above. The augmentation does provide robustness for distorted grids, but it dramatically increases the number of neighbors up to a few hundreds for a tetrahedral grid. This approach is not only computationally expensive but also relatively difficult to implement (although feasible) in a parallel code as it requires access to many neighbors across partition boundaries.

5 Implicit Gradient Method for Nodal Gradients

In this section, we first describe the implicit edge-based gradient method, which is the baseline method for developing the proposed implicit gradient method. Then, we present the proposed implicit gradient method for nodal gradients. Both methods are described for computing the gradients of the density ρ. The same algorithm is applied to other variables.

5.1 Implicit Edge-Based Gradient Method

To develop implicit gradient algorithms for the nodal-gradient CCFV discretization, we employ the implicit edge-based gradient method [24], which is very efficient because it can be exact for quadratic functions on arbitrary tetrahedral grids without second derivatives. The implicit edge-based gradient system can be written, for the density ρ, as

$$\mathbf{M}_{ii}\mathbf{g}_i + \sum_{\ell\in\{\ell_i\}} \mathbf{M}_{i\ell}\mathbf{g}_\ell = \mathbf{b}_i, \tag{10}$$

where $\mathbf{g}_i = \nabla\rho_i$ and $\mathbf{g}_\ell = \nabla\rho_\ell$ are the gradients stored at nodes i and ℓ, respectively,

$$\mathbf{M}_{ii} = \left(\frac{c+\kappa-1}{2}\right)\mathbf{I} + \frac{2-c}{2V_i}\sum_{\ell\in\{\ell_i\}}\left[\sum_{m\in\{\ell_i\}}\left(\mathbf{L}_{i\ell}^{LSQ}\cdot\Delta\mathbf{r}_{im}\right)V_{im}\right]\mathbf{I}, \tag{11}$$

$$\mathbf{M}_{i\ell} = -\frac{\kappa-1}{4V_i}\left(\mathbf{n}_{i\ell}\otimes\Delta\mathbf{r}_{i\ell}\right) + \frac{2-c}{2V_i}V_{i\ell}\mathbf{I} - \frac{2-c}{2V_i}\left[\sum_{m\in\{\ell_i\}}\left(\mathbf{L}_{i\ell}^{LSQ}\cdot\Delta\mathbf{r}_{im}\right)V_{im}\right]\mathbf{I}. \tag{12}$$

$$\Delta\mathbf{r}_{i\ell} = \mathbf{x}_\ell - \mathbf{x}_i, \quad V_{i\ell} = \frac{1}{6}\mathbf{n}_{i\ell}\cdot\Delta\mathbf{r}_{i\ell}, \quad V_i = \sum_{\ell\in\{\ell_i\}} V_{i\ell}, \quad \mathbf{b}_i = \frac{1}{V_i}\sum_{\ell\in\{\ell_i\}}\frac{\rho_i+\rho_\ell}{2}\mathbf{n}_{i\ell}, \tag{13}$$

c and κ are parameters, $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{n}_{i\ell}$ is the directed-area vector pointing from i to ℓ, which is the sum of the directed-areas corresponding to the dual-triangular faces associated with all tetrahedral elements sharing the edge $[\,i,\ell\,]$, and $\mathbf{L}_{i\ell}^{LSQ} = [L_{x_{i\ell}}, L_{y_{i\ell}}, L_{z_{i\ell}}]$ is a vector of linear LSQ coefficients (not a quadratic LSQ coefficients as in Eq. 9) such that the gradient at a node i is computed in a linearly-exact manner for a set of solution values stored at nodes:

$$\nabla\rho_i^{LSQ} = \sum_{\ell\in\{\ell_i\}}\mathbf{L}_{i\ell}^{LSQ}(\rho_\ell-\rho_i) = \left[\sum_{\ell\in\{\ell_i\}}L_{x_{i\ell}}(\rho_\ell-\rho_i), \sum_{\ell\in\{\ell_i\}}L_{y_{i\ell}}(\rho_\ell-\rho_i), \sum_{m\in\{\ell_i\}}L_{z_{i\ell}}(\rho_\ell-\rho_i)\right]. \tag{14}$$

Note that these linear LSQ coefficients are used only in the construction of the implicit gradient system matrix and therefore there is no need to store them once the implicit gradient system matrix is assembled.

As shown in Ref. [24], for quadratic exactness (second-order gradient accuracy on irregular grids), the second term in Eq.(11) and the third term in Eq.(12) are needed and we must set $\kappa = 0$ (c is arbitrary). In this paper, we only consider the quadratic IEBG method and it is referred to as IEBG. Also shown in Ref. [24] is an optimal choice of the parameters: $c = 2.6$ and $\kappa = 0$, which leads to fourth-order gradient accuracy on a regular tetrahedral grid. At a boundary node, the system needs to be closed by an accuracy-preserving quadrature formula. See Ref. [24] for details.

The implicit edge-based gradient system can be solved by a relaxation scheme in the form:

$$\mathbf{g}_i^{m+1} = \mathbf{g}_i^m - \mathbf{M}_{ii}^{-1}\left(\sum_{\ell\in\{\ell_i\}}\mathbf{M}_{i\ell}\mathbf{g}_\ell^m + \mathbf{b}_i\right), \tag{15}$$

where m is the iteration counter. This is the Jacobi iteration but it can be implemented also as the Gauss-Seidel iteration. In this work, we employ a multi-color Gauss-Seidel method by updating $\mathbf{g}_i$ immediately before moving onto the next node. Note that $\mathbf{M}_{ii}$ is a diagonal matrix with the same entry and therefore applying the inverse matrix $\mathbf{M}_{ii}^{-1}$ is equivalent to a simple scalar division.

Obviously, the method cannot be employed directly for our purpose because we have solutions at cells instead of nodes. However, if we transfer the solution values from cells to nodes with sufficient accuracy, we can directly employ the implicit edge-based gradient scheme and compute the nodal gradients. This is the approach considered here and it leads to a defect-correction gradient solver as we describe in the next section.

5.2 Implicit Gradient Systems for Nodal Gradients: Defect Correction

In this paper, we consider a defect-correction iteration as one of the simplest ways of utilizing the implicit edge-based gradient scheme:

$$\mathbf{g}_i^{n+1} = \mathbf{g}_i^n + \Delta\mathbf{g}_i, \quad n = 0, 1, 2, 3, \cdots, N, \tag{16}$$

where the correction is computed by the relaxation,

$$\Delta\mathbf{g}_i^{m+1} = \Delta\mathbf{g}_i^m - \mathbf{M}_{ii}^{-1}\left(\sum_{\ell\in\{\ell_i\}}\mathbf{M}_{i\ell}\Delta\mathbf{g}_\ell^m + \mathbf{r}_i\right), \quad \mathbf{r}_i = \mathbf{M}_{ii}\mathbf{g}_i^n + \sum_{\ell\in\{\ell_i\}}\mathbf{M}_{i\ell}\mathbf{g}_\ell^n - \mathbf{b}_i, \quad m = 0, 1, 2, 3, \cdots, M_g, \tag{17}$$

where N is the number of defect-correction gradient iterations and M_g is the number of the preconditioner relaxations. In this work, we set $M_g = 4$ and employ the Gauss-Seidel relaxation scheme. Note that $\mathbf{b}_i$ defined by

$$\mathbf{b}_i = \frac{1}{V_i}\sum_{\ell\in\{\ell_i\}}\frac{\rho_i + \rho_\ell}{2}\mathbf{n}_{i\ell}, \tag{18}$$

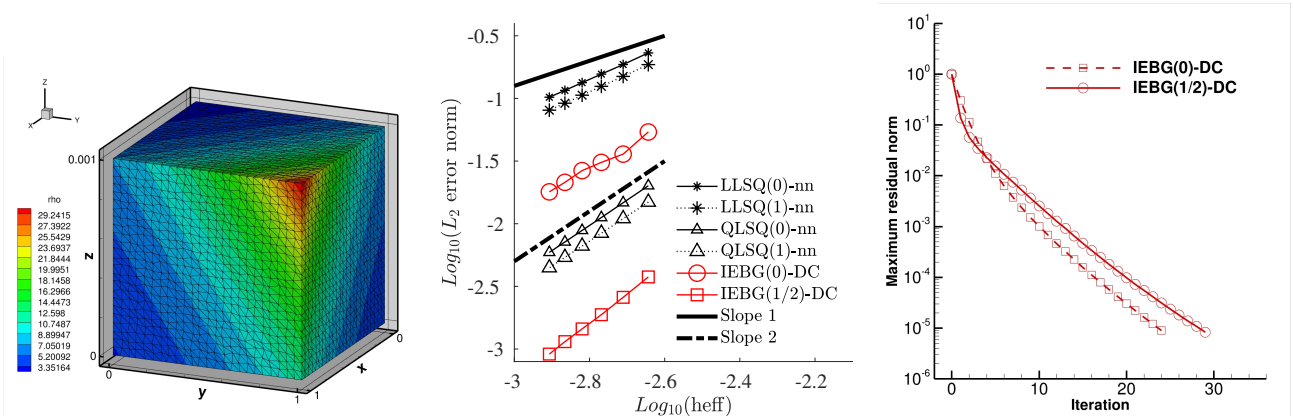
depends on the nodal gradients because the nodal values, ρ_i and ρ_ℓ, are computed by averaging the values interpolated from cells: e.g.,

$$\rho_i = \frac{1}{N_{c_i}}\sum_{j\in\{c_i\}}\left[\rho_j + \left\{C\,\mathbf{g}_i^n + (1-C)\overline{\mathbf{g}}_j^n\right\}\cdot(\mathbf{x}_i - \mathbf{x}_j)\right], \tag{19}$$

where $\{c_i\}$ is a set of cells sharing the node i, N_{c_i} is the number of cells in $\{c_i\}$, $\mathbf{g}_i^n$ is the gradient at the node i at the nth iteration, and $\overline{\mathbf{g}}_j^n$ is the averaged nodal gradient over the cell j at the nth iteration. The above formula is exact for quadratic functions if we set $C = 1/2$; otherwise, it is exact only for linear functions.Therefore, the order of gradient accuracy in the resulting implicit method depends on the interpolation accuracy although both are based on the quadratic IEBG method. In this study, we consider a quadratic algorithm with $C = 1/2$ for the third-order scheme, which will be referred to as IEBG(1/2)-DC while we consider a linear algorithm with $C = 0$ for a second-order scheme, which will be referred to as IEBG(0)-DC.

6 Results

In the results presented in this paper, all boundary conditions are imposed weakly through the right state as described in Ref. [1] for the third-order scheme, and the gradients are computed by LSQ or implicit methods at all nodes without incorporating physical boundary conditions. Both LSQ and implicit gradient methods are taken as purely mathematical algorithms that compute gradients at all nodes for a given set of solutions at cells.



(a) Coarsest grid. (b) Error convergence for $\partial_y\rho$. (c) Iterative convergence.

Figure 3: Results for gradient accuracy verification.

6.1 Gradient Accuracy Verification

To verify the first- and second-order accuracy of IEBG(0)-DC and IEBG(1/2)-DC, respectively, we consider computing gradients at nodes for the following function:

$$u(x, y, z) = \exp\left(c_x x + c_y y + c_z z\right), \tag{20}$$

where $c_x = 1.5$, $c_y = 1.6$, and $c_z = 1000.3$, in a unit cube domain, using a series of irregular tetrahedral grids with 82,944, 131,712, 196,608, 279,936, 384,999, and 511,104 cells. The coarsest grid and the function contours are shown in Figure 3(a). For comparison, we consider linear and quadratic LSQ methods with the node-neighbor augmentation, which has been found to improve robustness of the nodal-gradient cell-centered methods [29]. For the weighting, we consider both unweighted and inverse-distance-weighted methods. Weighted and unweighted linear LSQ methods are referred to as LLSQ(0)-nn and LLSQ(1)-nn, respectively, and weighted and unweighted quadratic LSQ methods are referred to as QLSQ(0)-nn and QLSQ(1)-nn, respectively. On all grids, the number of neighbor cells involved in these LSQ methods is roughly about 180 on average.

Figure 3(b) shows the error convergence results for $\partial_y \rho$ (results are similar for other derivatives). As can be expected, weighted LSQ gradients are slightly more accurate than unweighted LSQ gradients. On the other hand, the gradients computed by the implicit methods are significantly more accurate than LSQ gradients. The results demonstrate the superior accuracy of the implicit gradient methods, and are in good agreement with results reported in other papers [21, 24].

Figure 3(c) shows the iterative convergence of the defect-correction gradient solver for the implicit gradient methods on the finest grid. Note that we only compute gradients for a given function here, and do not solve the Euler equaitons. The solver converges rapidly within 30 iterations on such a highly-stretched and highly-skewed grid. Here, the residuals, which are defined as in Eq. (17), are reduced by five orders of magnitude, which was found sufficient for observing the design orders of gradient accuracy.

6.2 Accuracy Verification with Manufactured Solutions

Next, we consider employing the implicit gradient methods for solving the Euler equations as alternatives to LSQ gradient methods. To verify solution accuracy and gradient accuracy, we used the method of manufactured solutions [30], introducing a forcing term vector $\mathbf{s}$ to the Euler equations such that the following functions are the exact solutions:

$$\begin{aligned}
\rho^{exact} &= 1.0 + 0.1\exp(1.5x + 1.5y + 1000.5z), && (21)\\
u^{exact} &= 0.2 + 0.1\exp(1.5x + 1.5y + 1000.5z), && (22)\\
v^{exact} &= 0.1 + 0.1\exp(1.5x + 1.5y + 1000.5z), && (23)\\
w^{exact} &= 0.1 + 0.1\exp(1.5x + 1.5y + 1000.5z), && (24)\\
p^{exact} &= 1.0 + 0.1\exp(1.5x + 1.5y + 1000.5z). && (25)
\end{aligned}$$

We solve the residual equations (4) for the Euler equations by an implicit defect-correction solver with the residual Jacobian exact for the first-order accurate residuals. The linearized system for the Euler equations is relaxed by a multi-color Gauss-Seidel relaxation scheme 15 times. For the implicit gradient methods, we perform only one defect-correction gradient iteration per flow-solver iteration with 5 preconditioner relaxations. Specificaly, setting the initial values,

$$\mathbf{g}_i^0 = \mathbf{0}, \quad i = 1, 2, 3, \cdots, N_{nodes},$$

$$\mathbf{w}_j^0 = (\rho^{exact}(\mathbf{x}_j) + \epsilon_j, u^{exact}(\mathbf{x}_j) + \epsilon_j, v^{exact}(\mathbf{x}_j) + \epsilon_j, w^{exact}(\mathbf{x}_j) + \epsilon_j, p^{exact}(\mathbf{x}_j) + \epsilon_j), \quad j = 1, 2, 3, \cdots, N_{cells},$$

where $\rho^{exact}(\mathbf{x}_j)$ is the exact solution at the cell center $\mathbf{x}_j$ (similarly for the other variables), ϵ_j is a random perturbation, N_{nodes} and N_{cells} are, respectively, the number of nodes and the number of cells in a given grid, we iterate on gradients and solutions as follows. For $n = 0, 1, 2, 3, \cdots$, until all the residuals, the gradient system residuals (26) and flow-equation residuals (4), are reduced by six orders of magnitude, we first relax the preconditioner system as

$$\Delta\mathbf{g}_i^{m+1} = \Delta\mathbf{g}_i^m - \mathbf{M}_{ii}^{-1}\left(\sum_{\ell\in\{\ell_i\}} \mathbf{M}_{i\ell}\Delta\mathbf{g}_\ell^m + \mathbf{r}_i\right), \quad \mathbf{r}_i = \mathbf{M}_{ii}\mathbf{g}_i^n + \sum_{\ell\in\{\ell_i\}} \mathbf{M}_{i\ell}\mathbf{g}_\ell^n - \mathbf{b}_i, \quad m = 0, 1, 2, 3, \cdots, M_g, \tag{26}$$

where $M_g = 4$, and then update the gradients,

$$\mathbf{g}_i^{n+1} = \mathbf{g}_i^n + \Delta\mathbf{g}_i, \quad i = 1, 2, 3, \cdots, N_{nodes}. \tag{27}$$

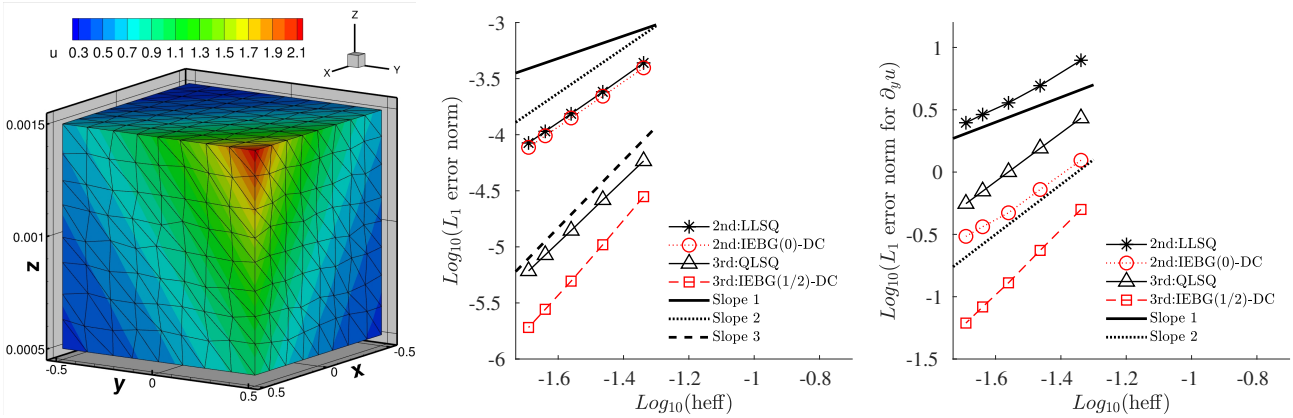
We repeat this defect-correction gradient iteration for all the primitive variables, ρ, u, v, w, and p. Then, using the updated gradients, we compute the flow-equation residual vector, $\mathbf{Res}(\mathbf{U}^n)$, where $\mathbf{U}$ is a global vector of numerical solutions, and relax the linearized system of the flow-equation residuals by a multi-color Gauss-Seidel relaxation scheme:

$$\left(\mathbf{D} + \frac{\partial\overline{\mathbf{Res}}}{\partial\mathbf{U}}\right)\Delta\mathbf{U} = -\mathbf{Res}(\mathbf{U}^n), \tag{28}$$

for a specified number of relaxations $M_f = 15$, where $\mathbf{D}$ is a diagonal matrix of the ratio of the cell volume to the local pseudo-time step defined in the corresponding cell, e.g., $\mathbf{D}_j = V_j/\Delta\tau_j$, and $\frac{\partial\overline{\mathbf{Res}}}{\partial\mathbf{U}}$ is the first-order residual Jacobian. Finally, we update the solution,

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta\mathbf{U}. \tag{29}$$

This completes one loosely-coupled iteration for the gradients and the solution. Therefore, the gradients are updated just once per flow-solver iteration. There is no need to fully solve the gradient system per flow-equation iteration because the flow numerical solutions are iterated. If the numerical solution is well converged, then the gradients would also be well converged as they are iterated from the initial values $\mathbf{g}_i^0$ and never reinitialized just like the numerical solutions. In this test problem, we set $\Delta\tau_j = 10^{15}$.



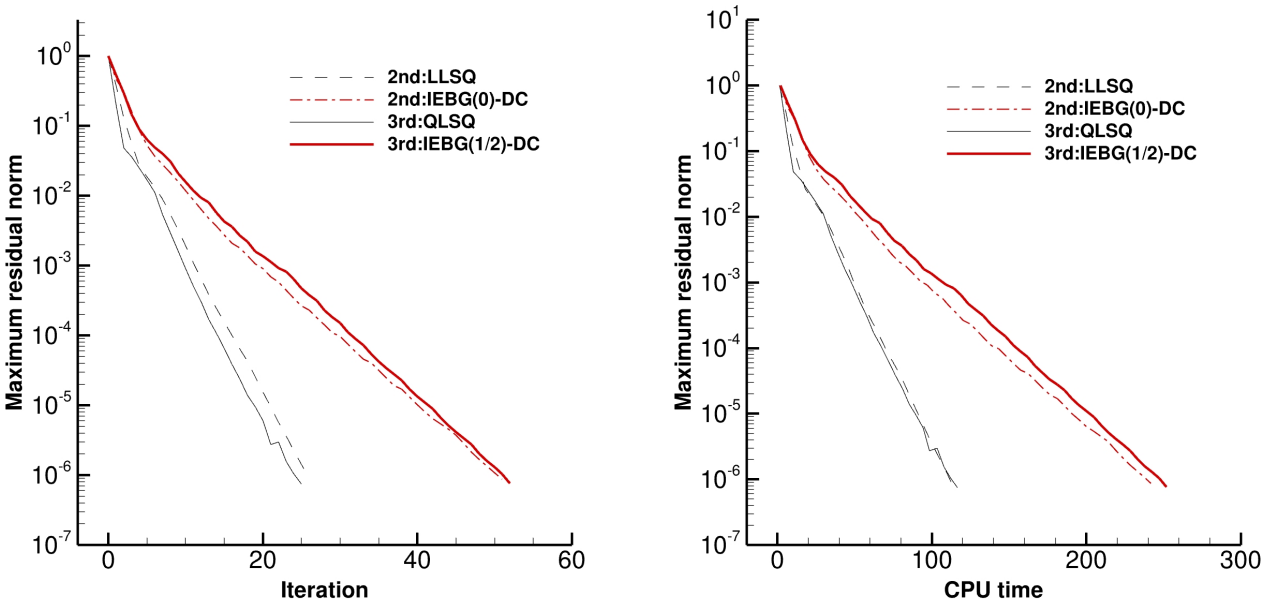
(a) Coarsest grid. (b) Error convergence for u. (c) Error convergence for $\partial_z u$.

Figure 4: Coarsest grid and results for the MMS test case.

Again, we compare the implicit gradient methods with LSQ methods. For this problem, we consider only unweighted LSQ gradients for a stability reason [31]. Figure 4(b) shows error convergence results for the velocity component u (results are similar for other variables). As expected, the second-order scheme produces second-order accurate solutions with both LLSQ and IEBG(0)-DC. Similarly, the third-order scheme produces third-order accurate solutions with both QLSQ and IEBG(1/2)-DC. While there is not much difference in the results with LLSQ and IEBG(0)-DC, IEBG(1/2)-DC gives significantly lower errors than QLSQ.

Figure 4(c) show error convergence results for the derivative $\partial_z u$ (results are similar to other derivatives). In general, the order of gradient accuracy is one-order lower than the order of solution accuracy; as expected, these results show first- and second-order gradient accuracy for the second- and third-order schemes, respectively. In contrast to the solution errors, significant improvements are observed in both second- and third-order schemes. Both IEBG(0)-DC and IEBG(1/2)-DC produce nearly an order of magnitude more accurate gradients than explicit LSQ gradients.

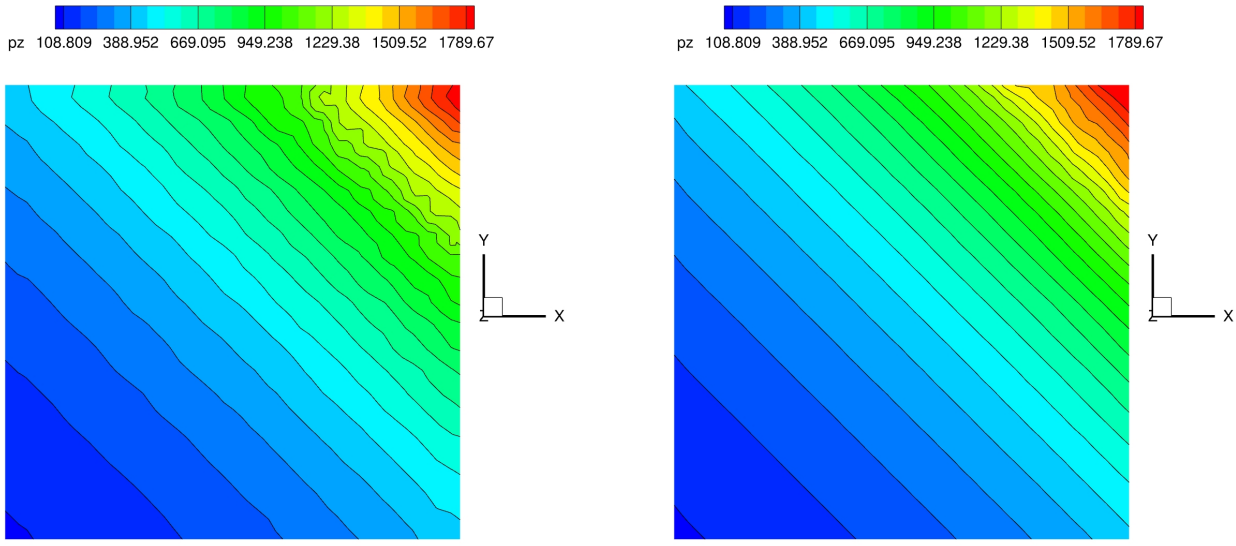
Figure 5(a) shows iterative convergence plot for the maximum residual norm (scaled by the initial norm) taken over the flow residuals (4) as well as the gradient residuals (26). Unexpectedly, the solver took more iterations to converge with the implicit gradient methods. It is not clear exactly why, but the code is not optimized at this point and also the large number of neighbors in LSQ gradients (≈ 180) might improve the solver stability. In any case, it should be noted that the solutions and gradients are significantly more accurate with the implicit gradient methods. Getting less accurate results faster is not necessarily considered as an advantage.



(a) Residual versus iteration for the finest grid.

(b) Residual versus CPU time for the finest grid.

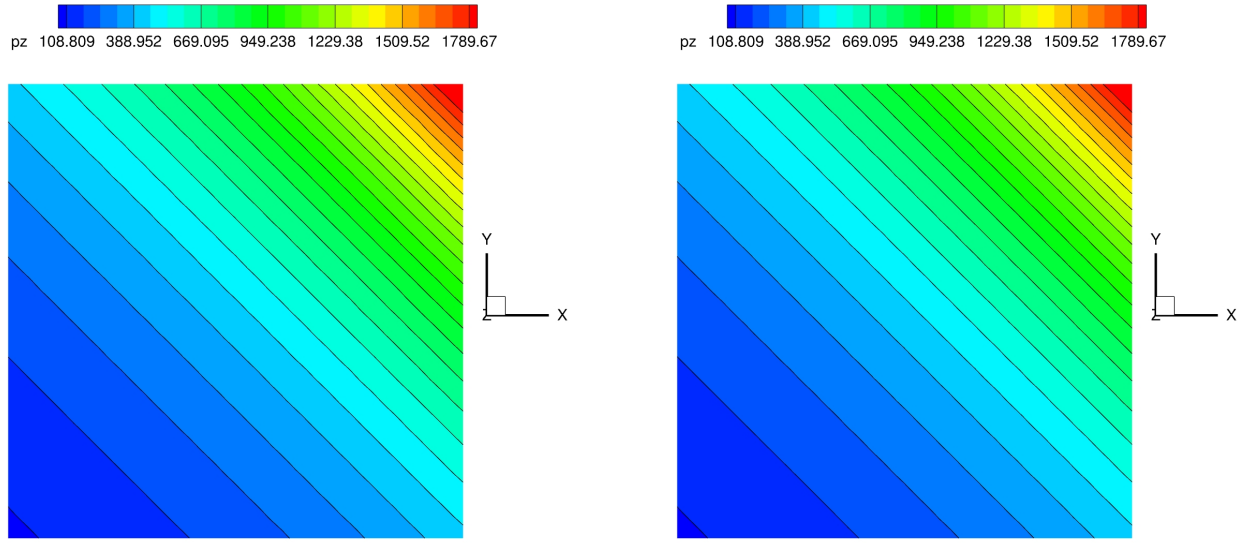
Figure 5: Iterative convergence results for the MMS test case.



(a) 2nd: LLSQ.

(b) 2nd: IEBG(0)-DC

Figure 6: Contours of $\partial_z \rho$ at $z = 0.00148$ for the MMS test case.



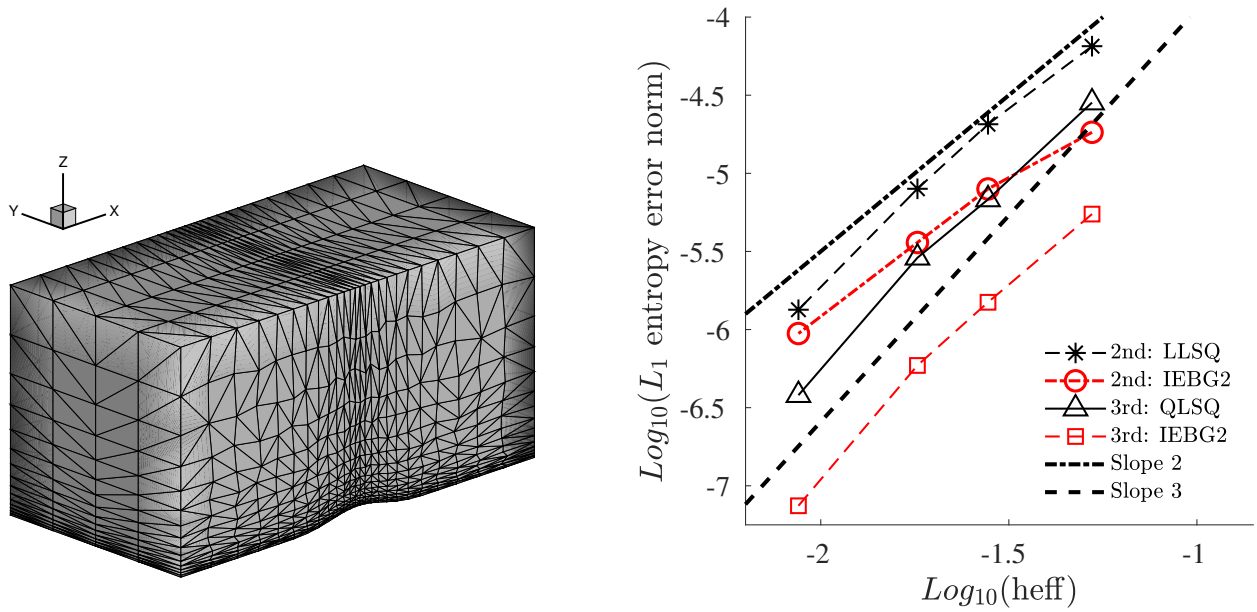
(a) 3rd: QLSQ.

(b) 3rd: IEBG(1/2)-DC.

Figure 7: Contours of $\partial_z \rho$ at $z = 0.00148$ for the MMS test case.

Finally, Figure 6 shows the contours of $\partial_z p$ at a section taken at $z = 0.00148$ (close to the top boundary). As can be seen in Figure 6(a), the gradient contours are not smooth for LLSQ, which is typical for LLSQ gradients on highly-skewed irregular grids. On the other hand, IEBG(0)-DC produces improved contours (see Figure 6(b)) although not perfectly smooth. Figures 7(a) and 7(b) show results for QLSQ and IEBG(1/2)-DC. Both contours are very smooth, compared with those obtained by linear methods. No significant differences are observed between QLSQ and IEBG(1.2)-DC while significantly lower gradient errors of IEBG(1/2)-DC as observed in the error convergence plot.

6.3 Subsonic Flow over a Smooth Bump



(a) Coarsest grid.

(b) Entropy error convergence.

Figure 8: Grid and error convergence for the subsonic smooth bump case.

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

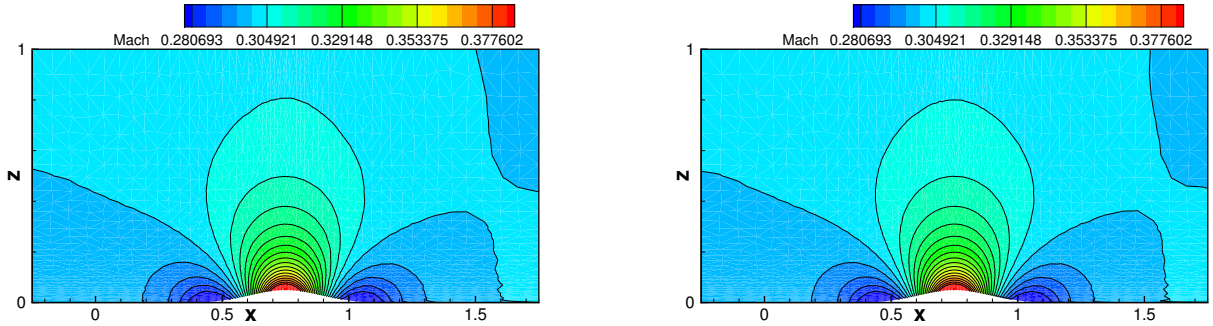
$$z_b = 0.05 \sin^4 \left[\frac{\pi(10x - 3)}{9} \right], \quad x \in [0.3, 1.2], \tag{30}$$

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 50 Gauss-Seidel linear relaxations and CFL=100 ($\Delta\tau_j = \text{CFL} \times V_j / (\sum_{k \in \{k_j\}} [\text{max wave speed}] \times |\mathbf{n}_{jk}|)$) for relatively coarse irregular tetrahedral grids with 9,000, 59,040, 221,616, and 1,932,300 cells. The coarsest grid is shown in Figure 8(a). All boundary conditions are imposed weakly through the right state in the numerical flux: the free stream condition at $x = -0.25$, the back pressure condition 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 $(-\, dz_b/dx|_i\,, 0, 1)$, which is used to define the right state in the numerical flux at a slip-wall node. See [2] for details on the implementation of a slip-wall condition that preserves third-order accuracy on a triangular surface grid. The solver is taken to be converged when the L_1 norms of the residual equations, including the implicit-gradient residuals, are reduced by six orders of magnitude. For LSQ gradients, for a stability reason, we employed unweighted LSQ methods. For the implicit gradient method, we employ IEBG(1/2)-DC for both second- and third-order schemes, and it is here referred to as IEBG2. There is no strong reason to employ the linear method, IEBG(0)-DC, for a second-order scheme. In fact, it is the advantage of the implicit gradient approach that the solver tends to be stable for accurate gradients whereas it tends to diverge if gradients are accurate in the case of explicit gradient methods.

Figure 8(b) shows entropy error convergence results, where the entropy error is defined by $|\gamma p/\rho^\gamma - 1|$, which should be zero throughout the domain. For the second-order scheme, the entropy error is smaller with IEBG2 than with LLSQ. Further results would be necessary to determine their behaviors on finer grids. For the third-order scheme, third-order error convergence is observed for both QLSQ and IEBG2 and the latter produces consistently lower levels of error for all the grids. The results are in good agreement with those obtained for the node-centered edge-based methods [24].

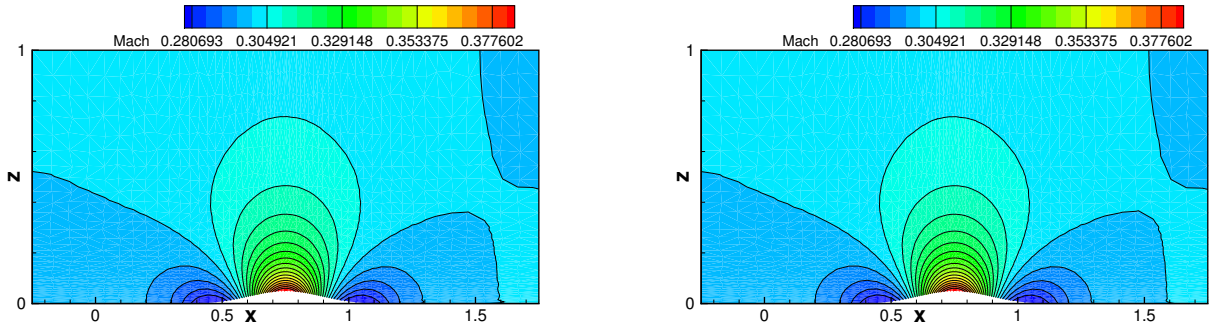
Figures 9 and 10 show Mach contours on the third grid with 221,616 cells. These contours may look similar but they exhibit more noise in the second-order results and particularly with LSQ gradients. These differences can be seen clearly in the entropy error contours as shown in Figures 11 and 12. Figures 11(a) and 11(b) compare the entropy error contours for the second-order scheme. Clearly, larger entropy errors are observed in the result obtained with LLSQ. Figures 12(a) and 12(b) compare the entropy errors for the third-order scheme. Note that the contour range in Figure 12 is different from that in the second-order results. As can be seen, larger entropy errors are observed in the result obtained with QLSQ. In both second- and third-order schemes, the implicit gradient methods produce cleaner solutions with lower levels of the entropy error.

Figure 13(a) shows residual convergence results, where the maximum residual norm is taken over the flow residuals (4) as well as the gradient residuals (26). In this problem, the solver converged with fewer numbers of iterations with the implicit gradient methods. As can be seen in Figure 13(b), where the residual convergence is plotted with CPU time, the solver converged faster in time with the implicit gradient methods. It demonstrates that one defect-correction gradient iteration is not too expensive compared with LSQ methods. These are encouraging results, indicating that LSQ gradient methods may be replaced by the implicit gradient methods to obtain improved accuracy faster.



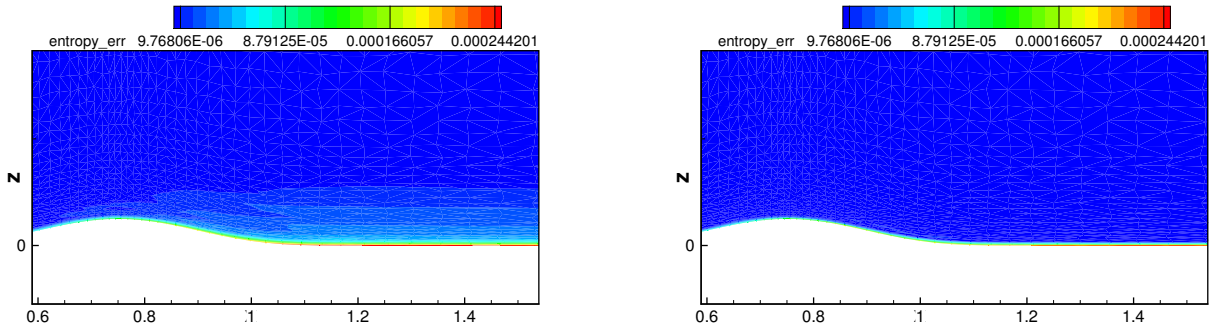
(a) 2nd:LLSQ. (b) 2nd: IEBG2.

Figure 9: Mach contours for the subsonic smooth bump case on the third grid with 221,616 cells.



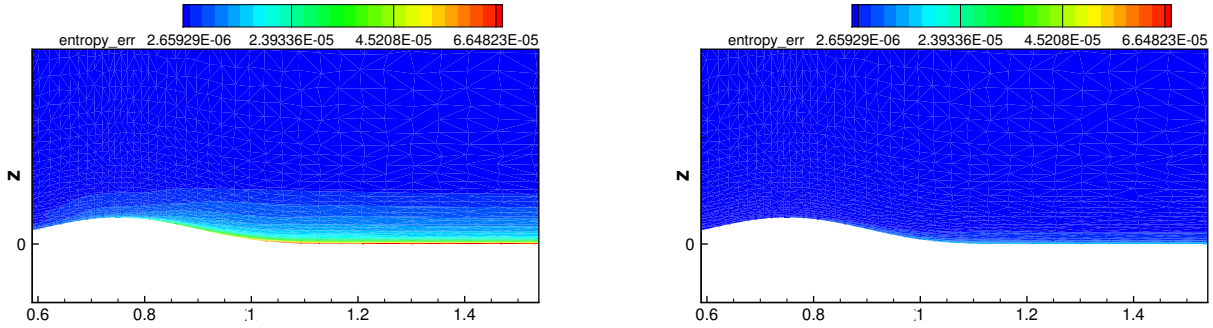
(a) 3rd: QLSQ. (b) 3rd: IEBG2.

Figure 10: Mach contours for the subsonic smooth bump case on the third grid with 221,616 cells.



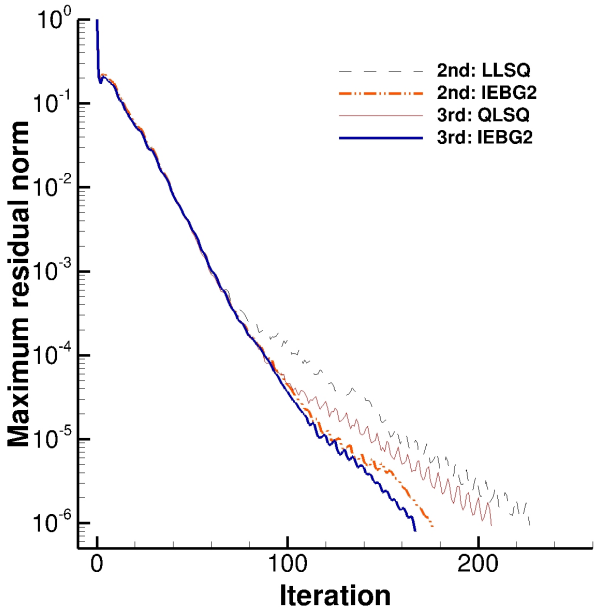
(a) 2nd: LLSQ. (b) 2nd: IEBG2.

Figure 11: Entropy contours for the subsonic smooth bump case on the third grid with 221,616 cells.

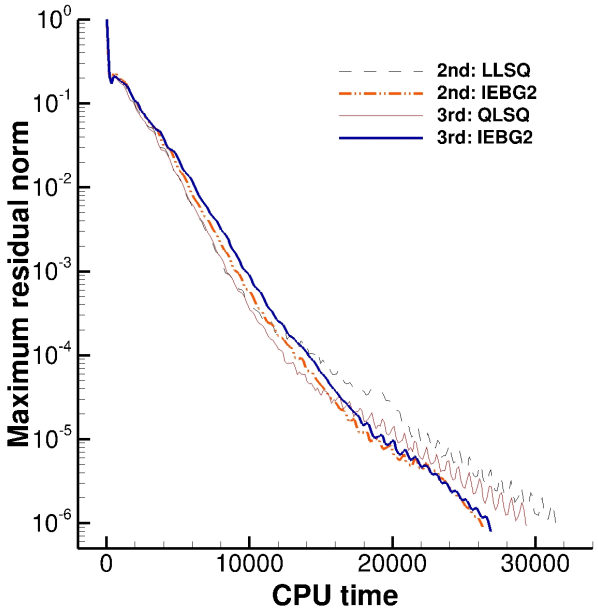


(a) 3rd: QLSQ. (b) 3rd: IEBG2.

Figure 12: Entropy contours for the subsonic smooth bump case on the third grid with 221,616 cells.



(a) Iterative convergence: residual versus iteration.



(b) Iterative convergence: residual versus CPU time.

Figure 13: Iterative convergence results.

7 Conclusions

In this paper, we have developed implicit gradient methods for the second- and third-order nodal-gradient cell-centered finite-volume methods, extending the implicit edge-based gradient method originally developed for the node-centered edge-based methods. Linear and quadratic implicit gradient methods been developed for computing gradients at nodes from solutions at cells by directly employing the implicit edge-based gradient method with nodal solutions computed by averaging values interpolated from cells to nodes. For the linear method, we compute the solution at a node by averaging linearly interpolated solutions from cells sharing the node. For the quadratic method, we utilize the efficient quadratic interpolation scheme developed in Ref.[1], which does not require second derivatives. As a result, the third-order nodal-gradient cell-centered finite-volume method remains efficient and completely second-derivative free. Numerical results indicate that accuracy in solutions and gradients can be greatly improved by employing the implicit gradient methods and such improved results can potentially be obtained faster than results with LSQ methods.

Future work includes a few potential algorithmic improvements. First, in the defect-correction approach considered in this paper, the gradient system residual is not compact, depending on the nodes beyond the edge-connected neighbors. It is possible to develop a gradient system with a compact stencil. It will allow a direct relaxation and it can potentially be cheaper than the defect-correction iteration. Second, it may be worth considering the development of a Jacobian-Free Newton-Krylov (JFNK) solver for solving the flow-residuals and gradient-residuals together in a tightly-coupled manner with the loosely-coupled solver as a preconditioner. An attractive feature of the JFNK solver is that the number of extra unknowns added to the flow variables is substantially smaller because the gradients are stored at nodes, not at cells, and the number of nodes is typically 1/6 of the number of cells in a tetrahedral grid. Finally, the implicit gradient methods need to be demonstrated for a more realistic flow problems including those involving shock waves.

Acknowledgments

The author gratefully acknowledges support by the Hypersonic Technology Project, through the Hypersonic Airbreathing Propulsion Branch of the NASA Langley Research Center, funded under Prime Contract No. 80LARC23DA003.

References

[1] 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.

[2] Nishikawa, H. and White, J. A., "An Efficient Cell-Centered Finite-Volume Method with Face-Averaged Nodal-Gradients for Triangular Grids," *J. Comput. Phys.*, Vol. 411, 2020, pp. 109423.

[3] White, J., Nishikawa, H., and Baurle, R., "A 3-D Nodal-Averaged Gradient Approach for Unstructured-grid Cell-centered Finite-volume Methods for Application to Turbulent Hypersonic Flow," *SciTech 2020 Forum*, AIAA Paper 2020-0652, Orlando, FL, 2020.

[4] White, J. A., Nishikawa, H., and O'Connell, M., "F-ANG+: A 3-D Augmented-Stencil Face-Averaged Nodal-Gradient Cell-Centered Finite-Volume Method for Hypersonic Flows," *SciTech 2022 Forum*, AIAA Paper 2022-1848, San Diego, CA, 2022.

[5] Barth, T. J. and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids using Quadratic Reconstruction," AIAA Paper 90-0013, 1990.

[6] Delanaye, M. and Essers, J. A., "Quadratic-Reconstruction Finite Volume Scheme for Compressible Flows on Unstructured Adaptive Grids," *AIAA J.*, Vol. 35, No. 4, 1997, pp. 631–639.

[7] Ollivier Gooch, C. and Van Altena, M., "A High-Order-Accurate Unstructured Mesh Finite-Volume Scheme for the Advection-Diffusion Equation," *J. Comput. Phys.*, Vol. 181, 2002, pp. 729–752.

[8] Caraeni, D. and Hill, D. C., "Unstructured-Grid Third-Order Finite Volume Discretization Using a Multi-step Quadratic Data-Reconstruction Method," *AIAA J.*, Vol. 48, No. 4, 2010, pp. 808–817.

[9] Haider, F., Brenner, P., Courbet, B., and Croisille, J.-P., "Efficient Implementation of High Order Reconstruction in Finite Volume Methods," *Proc. of Finite Volumes for Complex Applications VI Problems & Perspectives, FVCA 6, International Symposium*, Prague, June 2011, pp. 553–560.

[10] Charest, M. R. J., Groth, C. P. T., and and, P. Q. G., "High-Order CENO Finite-Volume Scheme for Low-Speed Viscous Flows on Three-Dimensional Unstructured Mesh," *Proc. of Seventh International Conference on Computational Fluid Dynamics*, Big Island, Hawaii, 2012.

[11] Jalali, A. and Ollivier-Gooch, C., "Higher-Order Unstructured Finite Volume RANS Solution of Turbulent Compressible Flows," *Comput. Fluids*, Vol. 143, 2017, pp. 32–47.

[12] Pont, G., Brenner, P., Cinnella, P., Maugars, B., and Robinet, J.-C., "Multiple-Correction Hybrid k-Exact Schemes for High-Order Compressible RANS-LES Simulations on Fully Unstructured Grids," *J. Comput. Phys.*, Vol. 350, 2017, pp. 45–83.

[13] Caraeni, D. and Fuchs, L., "Compact Third-Order Multidimensional Upwind Discretization for Steady and Unsteady Flow Simulations," *Comput. Fluids*, Vol. 34, 2005, pp. 419–441.

[14] 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.

[15] 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.

[16] Schwöppe, A. and Diskin, B., "Accuracy of the Cell-Centered Grid Metric in the DLR TAU-Code," *New Results in Numerical and Experimental Fluid Mechanics VIII. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Volume 121*, edited by A. Dillmann, G. Heller, H. P. Kreplin, W. N. W., and I. Peltzer, Springer, 2013, pp. 429–437.

[17] Nishikawa, H., "Efficient Gradient Stencils for Robust Implicit Finite-Volume Solver Convergence on Distorted Grids," *J. Comput. Phys.*, Vol. 386, 2019, pp. 486–501.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] Nishikawa, H., "Updates to Implicit Edge-Based Gradient Methods," *AIAA SciTech 2024 Forum*, AIAA Paper 2024-1748, Orlando, FL, 2024.

[25] Krist, S. L., Biedron, R. T., and Rumsey, C. L., "CFL3D User's Manual (Version 5.0)," *NASA/TM-1998-208444*, June 1998.

[26] Brenner, P., "Towards Rather Comprehensive Methods for Finite Volume Methods," October 2010, doi:10.13140/RG.2.2.29412.53127.

[27] Setzwein, F., Ess, P., and Gerlinger, P., "An Implicit High-Order k-Exact Finite-Volume Approach on Vertex-Centered Unstructured Grids for Incompressible Flows," *J. Comput. Phys.*, Vol. 446, 2021, pp. 110629.

[28] Nishikawa, H., "A Flux Correction for Finite-Volume Discretizations: Achieving Second-Order Accuracy on Arbitrary Polyhedral Grids," *J. Comput. Phys.*, Vol. 468, 2022, pp. 111481.

[29] Nishikawa, H. and White, J. A., "A Simplified FANG Cell-Centered Finite-Volume Method and Comparison with Other Methods for Trouble-Prone Grids," *Proc. of AIAA Aviation 2021 Forum*, AIAA Paper 2021-2720, 2021.

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

[31] 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.