Preprint accepted in Journal of Computational Physics. https://doi.org/10.1016/j.jcp.2018.06.019

From Hyperbolic Diffusion Scheme to Gradient Method: Implicit Green-Gauss Gradients for Unstructured Grids

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

June 18, 2018

Abstract

This paper introduces a new approach to constructing algorithms for gradient computations on unstructured grids. The proposed approach is to derive a gradient algorithm from a hyperbolic diffusion scheme, which solves the diffusion equation and equations for the solution gradient, by ignoring the residual component for the diffusion equation. The resulting scheme forms a globally coupled linear system of equations for the gradients with the Green-Gauss gradient formula on the right hand side, which can be solved efficiently by iterative techniques. Accuracy and iterative performance can be controlled by parameters inherited from the generating hyperbolic diffusion scheme. Second- or fourth-order gradient accuracy can be obtained on regular grids, and first-order accuracy, which is sufficient for second-order finite-volume schemes, can be obtained on irregular grids. Numerical results indicate that the method gives highly accurate gradients especially for highly-curved high-aspect-ratio grids typical in high-Reynolds-number boundary layer flows over complex geometries, and can be employed in finite-volume solvers as an efficient alternative to least-squares or Green-Gauss gradient methods. It is also shown that one of the parameters has a role of dissipation, and can be tuned to yield a smooth gradient distribution over a discontinuous function, thereby allowing an implicit finite-volume solver to converge for a discontinuous solution with a mild overshoot. The paper concludes with a discussion for future developments.

1 Introduction

The performance of finite-volume solvers replies heavily on gradient computation methods, which are required in the inviscid flux reconstruction, the evaluations of viscous fluxes, and turbulence model source terms, especially on unstructured grids typical of applications involving complex geometries. Despite great progress made over the last decades, represented by extensive studies on least-squares gradients in Ref.[1], current state-of-the-art unstructured-grid finite-volume solvers still encounter robustness issues when dealing with highly-distorted grids. One of the popular remedies, other than ignoring gradients, is the Green-Gauss method, which is known to lose consistency on irregular grids, but can be employed for robustness, i.e., for the sake of obtaining numerical solutions [2, 3, 4]. As a result, the solutions obtained would lose second-order accuracy (i.e., reduce to zeroth- or first-order accuracy) even for smooth flows. Such remedies may also negatively affect approximations to the viscous terms. For turbulence model source terms, accurate gradients are highly desired because inaccurate source term evaluations can lead to instability of implicit solvers. These considerations imply that a robust solver has to pay the price for computing different types of gradient approximations for different terms: inaccurate but robust gradients for the inviscid terms, and accurate gradients for the viscous terms and the source terms [2, 5]. Note that the gradients of all variables (e.g., the density, velocity components, and pressure) are needed in the inviscid part, and those of the velocity components and temperature are needed in the viscous part, thus implying that at least the velocity gradients need to be computed twice with two different gradient methods. The need for different gradient algorithms also arises from adapting different discretization methods for different terms [6, 7, 8, 9]. To achieve truly second-order accurate simulations without performing multiple gradient computations, a gradient method is sought that maintains consistency and accuracy on unstructured grids of arbitrary distortion and leads to stable iterative convergence of a flow solver.

Two classes of gradient algorithms have widely been used in unstructured-grid finite-volume codes: Green-Gauss (GG) [4, 10, 11] and least-squares (LSQ) methods [12, 13, 14, 15, 16]. The basic GG method is formulated by the midpoint quadrature approximation to the integral definition of the gradient operator with the face-midpoint solution value evaluated by an average of the solutions in the two cells sharing the face, or a distance-weighted interpolation [4]. As is well known for cell-centered schemes, the method loses consistency on irregular

grids, where the interpolation misses the face-midpoint as the line between the centroids of the two cells does not pass through the face-midpoint (see, e.g., Ref.[17]). Consistency can be recovered by evaluating the face value by the average of two values linearly interpolated from the cells using the gradients (which are unknowns). It is consistent and exact for linear functions on arbitrary grids, but it requires iterations to obtain the gradients. Such implicit gradient methods are described in Ref.[4], but not widely used and their performance on highly distorted grids is unknown. More importantly, these iterative schemes lack flexibility, i.e., no parameters to adjust, with which potential issues encountered on highly-distorted grids can be resolved. A node-averaging-based GG method as described for tetrahedral grids in Ref.[11] is consistent on arbitrary tetrahedral grids and can be applied to other types of grids [18] at a cost of further increasing the stencil. The LSQ method is a very popular method for unstructured grids, where the gradients are computed as a solution to a LSQ problem of fitting a polynomial over solution values in nearby cells. Many variants exist, depending on how the LSQ problem is weighted, typically by the distances from the cell of interest to its neighbors. The LSQ method is more flexible than the GG method in that it is designed to be exact for linear functions on any type of grid, and also can be extended to high-order by fitting high-order polynomials. A drawback is, however, that high-order LSQ methods require large stencils, thus being less attractive for parallel implementation and for the construction of effective implicit solvers. Moreover, even the linear LSQ method requires a careful construction of the stencil: a cell at a corner may have only one neighbor, which leads to an ill-posed LSQ problem, and additional cells need to be added. Even more critically, it has been shown in Ref.[19] that finite-volume schemes with LSQ gradients computed over face neighbors only are unstable, especially in three-dimensional tetrahedral grids. This is a rather disappointing result. To construct a stable finite-volume scheme, the LSQ stencil is required to involve cells beyond face neighbors (i.e., non-compact), but a robust guiding principle to select a minimal set of extra cells is still missing although some progress has been reported in Refs.[1, 20]; Or a nonlinear algorithm such as a gradient limiter may be employed to maintain the stability [21, 22]. A hybrid method that combines the GG and LSQ methods is described in Ref.[3]. Although impressive results are reported for relatively regular grids, this approach does not extend the stencil beyond face neighbors and therefore cannot address the stability issue on tetrahedral grids. Note that Ref.[3] demonstrates stable finite-volume solver convergence with the hybrid gradient method for quadrilateral and mixed grids, for which a finite-volume solver is known to be stable [19]. In this regard, Ref.[23] proposed a new promising approach called the variational reconstruction (VR) method. In this method, a globally coupled system of linear equations for gradients is derived by minimizing the solution jumps at faces. The system is iteratively solved along with a finite-volume flow solver. If one iteration is performed for the gradients per solver iteration, then the cost per iteration is almost the same as the explicit methods such as the GG and LSQ methods, or more efficient than LSQ methods with extended stencils beyond face neighbors. The stability issue is effectively circumvented by the gradient stencil spanning the entire grid [24, 25]. A further advantage lies in simple extensions to high-order through minimizing the jumps of high-order polynomials. Inspired by the VR method, we explore a similar implicit gradient methodology, but with a totally different approach to deriving an implicit gradient system.

In this paper, we propose to derive a gradient method from a discretization of a hyperbolic diffusion system. In the hyperbolic diffusion method [26], the diffusion equation is converted to a hyperbolic system by adding extra variables corresponding to the solution gradient, and then the system is discretized by upwind methods. The discretization represents discrete approximations to the diffusion equation and equations for the gradient components. A system of gradient equations are then obtained by assuming that the numerical solution is given and thus ignoring the diffusion residual. In particular, if we consider a cell-centered finite-volume hyperbolic diffusion scheme, the resulting gradient system becomes a globally-coupled linear system for the cell gradients with the Green-Gauss gradient method on the right hand side. Unlike the conventional Green-Gauss method, the resulting method is exact for linear functions on arbitrary grids. Furthermore, the relaxation time associated with the hyperbolic diffusion formulation and a one-parameter-family of source term quadrature formulas provide ways to control the gradient algorithm. For example, there is a special definition of the relaxation time that leads to fourth-order gradient accuracy on a regular grid. In this paper, we illustrate the idea of deriving a gradient method from a hyperbolic diffusion scheme, and investigate the performance of the derived implicit Green-Gauss method for various unstructured grids, including highly-distorted grids. Targeting applications to second-order finite-volume schemes, we focus on gradient algorithms that are exact for linear functions on arbitrary grids.

The paper is organized as follows. In Section 2, we describe a second-order cell-centered hyperbolic diffusion scheme. In Section 3, a gradient algorithm is derived from the hyperbolic diffusion scheme. In Section 4, the gradient method is analyzed for regular grids. In Section 5, three ways to deal with boundaries are discussed. In Section 6, numerical results are shown and discussed. Finally, in Section 7, we conclude the paper and discuss

future developments.

2 Hyperbolic Diffusion Scheme

In this section, we follow the hyperbolic method [26], and construct a second-order cell-centered finite-volume discretization for a hyperbolic diffusion system. The resulting residual will serve as the basis for deriving a gradient method as discussed in the subsequent section.

2.1 Hyperbolic Diffusion System

In the hyperbolic method [26], the steady diffusion equation:

$$0 = \nu(\partial_{xx}u + \partial_{yy}u), \tag{1}$$

where ν is a constant diffusion coefficient and u is a scalar solution variable, is discretized in the form of a first-order system:

$$\partial_\tau u = \nu(\partial_x p + \partial_y q), \quad \partial_\tau p = \frac{1}{T_r}(\partial_x u - p), \quad \partial_\tau q = \frac{1}{T_r}(\partial_y u - q), \tag{2}$$

where τ is a pseudo time, and T_r is called the relaxation time. The extra variables p and q correspond to the solution gradient components, and are called the gradient variables. For dimensional consistency [26], we set

$$T_r = \frac{L_r^2}{\nu}, \tag{3}$$

where L_r is a length scale to be discussed later. The system is equivalent, with the pseudo-time derivatives dropped, to the diffusion equation. Therefore, a consistent discretization for the diffusion equation (1) is obtained by discretizing the first-order system and then dropping the pseudo-time derivatives. The first-order system is written in the vector form:

$$\partial_\tau \mathbf{u} + \partial_x \mathbf{f} + \partial_y \mathbf{g} = \mathbf{s}, \tag{4}$$

where

$$\mathbf{u} = \begin{bmatrix} u \\ p \\ q \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} -\nu p \\ -u/T_r \\ 0 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} -\nu q \\ 0 \\ -u/T_r \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} 0 \\ -p/T_r \\ -q/T_r \end{bmatrix}. \tag{5}$$

This system is hyperbolic in τ, and therefore can be discretized by methods for hyperbolic systems, e.g., upwind methods [26].

Remark: The system (4) is valid for a constant diffusion coefficient. For nonlinear diffusion equations, a nonlinear hyperbolic formulation must be used, which was first introduced in Ref.[27] for the compressible Navier-Stokes equations, and later applied to nonlinear diffusion equations [28, 29]. These formulations use the diffusive fluxes as additional variables, and may be employed to derive algorithms for computing diffusive fluxes instead of gradients. Such a reconstruction method can be useful for some applications, e.g., problems with interfaces through which the diffusive flux is continuous [30], and should be explored in future. In this work, we consider only the linear diffusion equation as it is sufficient to derive a gradient algorithm that is general and can be applied to any problem or discretization scheme, where gradients need to be computed, including Euler and Navier-Stokes schemes.

2.2 Residual

We discretize the first-order hyperbolic system (4) by a cell-centered finite-volume discretization on unstructured grids as shown in Figure 1, where the residual is defined as an approximation to the system (4) integrated over a computational cell j by the midpoint rule:

$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \mathbf{\Phi}_{jk} A_{jk} - \int_{V_j} \mathbf{s}\, dV, \tag{6}$$

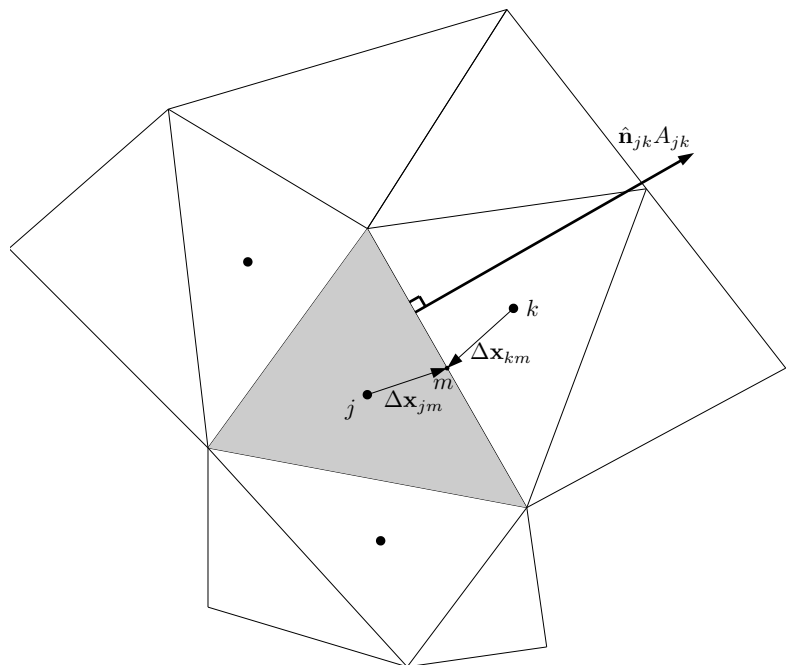


Figure 1: Stencil for cell-centered finite-volume discretization.

where $\{k_j\}$ is a set of face neighbors of the cell j, A_{jk} is the length of the face across j and k, V_j is the area of the cell j, and the source integration will be discussed later. The numerical flux $\mathbf{\Phi}_{jk}$ is given by the upwind flux (see Ref.[31]):

$$\begin{aligned}\mathbf{\Phi}_{jk} &= \frac{1}{2}\left[\mathbf{f}_n(\mathbf{u}_L)+\mathbf{f}_n(\mathbf{u}_R)\right]-\frac{1}{2}\left|\mathbf{A}_n\right|\left(\mathbf{u}_R-\mathbf{u}_L\right)\\ &= \frac{1}{2}\begin{bmatrix}-\nu(p_{nL}+p_{nR})\\ -\frac{1}{T_r}(u_L+u_R)\hat{n}_x\\ -\frac{1}{T_r}(u_L+u_R)\hat{n}_y\end{bmatrix}-\frac{\nu}{2L_r}\begin{bmatrix}u_R-u_L\\ (p_{nR}-p_{nL})\hat{n}_x\\ (p_{nR}-p_{nL})\hat{n}_y\end{bmatrix},\end{aligned} \tag{7}$$

where $\mathbf{A}_n=\partial\mathbf{f}_n/\partial\mathbf{u}$, $\mathbf{f}_n=\mathbf{f}\hat{n}_x+\mathbf{g}\hat{n}_y$, $\hat{\mathbf{n}}_{jk}=(\hat{n}_x,\hat{n}_y)$ is the unit vector normal to the face pointing from j to k, $p_{nL}=(p_L,q_L)\cdot\hat{\mathbf{n}}_{jk}$, and $p_{nR}=(p_R,q_R)\cdot\hat{\mathbf{n}}_{jk}$. The left and right states are computed as

$$\begin{aligned}&u_L=u_j+(p_j,q_j)\cdot\Delta\mathbf{x}_{jm}, && u_R=u_k+(p_k,q_k)\cdot\Delta\mathbf{x}_{km},\\ &p_L=p_j, && p_R=p_k,\\ &q_L=q_j, && q_R=q_k,\end{aligned} \tag{8}$$

where $\Delta\mathbf{x}_{jm}=\mathbf{x}_m-\mathbf{x}_j$, $\Delta\mathbf{x}_{km}=\mathbf{x}_m-\mathbf{x}_k$, $\mathbf{x}_m$ is the face-midpoint location, and $\mathbf{x}_j$ and $\mathbf{x}_k$ are the centroid coordinates of the cells j and k, respectively. The above face reconstruction corresponds to the first-order version of Scheme II in Ref.[31], which is an economical version of the hyperbolic diffusion scheme that does not require gradient reconstruction for the solution u. For the source term discretization, to better control the gradient algorithm to be derived later, we employ the following formula:

$$\int_{V_j}\mathbf{s}\,dV=\frac{1}{2}\sum_{k\in\{k_j\}}\left\{c_j\mathbf{s}_j+(1-c_j)\mathbf{s}_k\right\}(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk})A_{jk}, \tag{9}$$

where c_j is a parameter defined in the cell j. This formula is exact for constant (p,q) and thus for linear functions of u.

In the residual (6), the pseudo time derivatives have been dropped, and therefore it is a consistent approximation to the diffusion equation (1) in the form of the first-order system. To see this, we write out the components of the residual, $\mathbf{Res}_j$:

$$Res_j(1) = -\sum_{k\in\{k_j\}} \left[\nu(p_{nL}+p_{nR}) + \frac{\nu}{L_r}(u_R-u_L)\right] A_{jk}, \tag{10}$$

$$Res_j(2) = \left[-\frac{1}{2V_j}\sum_{k\in\{k_j\}} \left[(u_L+u_R)\hat{n}_x + L_r(p_{nR}-p_{nL})\hat{n}_x + \{c_j p_j + (1-c_j)p_k\}\left(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\right)\right] A_{jk}\right]\frac{V_j}{T_r}, \tag{11}$$

$$Res_j(3) = \left[-\frac{1}{2V_j}\sum_{k\in\{k_j\}} \left[(u_L+u_R)\hat{n}_y + L_r(p_{nR}-p_{nL})\hat{n}_y + \{c_j q_j + (1-c_j)q_k\}\left(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\right)\right] A_{jk}\right]\frac{V_j}{T_r}. \tag{12}$$

It is clear, since jump quantities will vanish in the grid refinement, that the discretization is a consistent approximation to the diffusion equation written as a first-order system:

$$Res_j(1)/V_j = -\nu(\partial_x p + \partial_y q) + T.E., \tag{13}$$

$$Res_j(2)/V_j = -\frac{1}{T_r}(\partial_x u - p) + T.E., \tag{14}$$

$$Res_j(3)/V_j = -\frac{1}{T_r}(\partial_y u - q) + T.E., \tag{15}$$

where $T.E.$ denotes a truncation error. Note finally that the residual vanishes for linear functions of u and thus constant (p, q) for any L_r and c_j.

As demonstrated in the previous papers, the above hyperbolic scheme has significant advantages over conventional diffusion schemes: superior gradient accuracy on irregular grids and $O(1/h)$ speed-up in iterative convergence, where h is a typical mesh spacing. The speed-up is achieved by the elimination of second-derivatives and a unique scaling of L_r: $L_r = O(1)$, e.g., $L_r = 1/(2\pi)$, which was derived to optimize error propagation [26, 31]. The scheme described above corresponds to DG(P0P1)+DG(P0) in Ref.[22], and Hybrid P1+P0 in Ref.[32], and is known to achieve second-order accuracy in u and first-order accuracy in p and q. It should be noted that these accuracy orders are exceptional and better than the node-centered counterpart, which gives first-order accuracy for all variables [31]. See Refs.[28, 29, 31, 33, 34, 35, 36] for further details on the hyperbolic method.

The hyperbolic method has also been found useful in the derivation of conventional diffusion schemes. Refs.[37, 38] demonstrate that a robust and accurate scalar diffusion scheme can be derived from a hyperbolic scheme by discarding the residual components corresponding to the gradients. For example, in the scheme considered above, a scalar diffusion scheme is obtained if we discard $Res_j(2)$ and $Res_j(3)$, and instead reconstruct the gradients from the solution u, e.g., by a LSQ method. The resulting scheme, which consists of $Res_j(1)$ and the LSQ gradients, is a conventional diffusion scheme. In this approach, the relaxation time is defined as $T_r = L_r^2/(\alpha^2\nu)$ where α is a free parameter [37, 38], so that the dissipation coefficient becomes $\alpha\nu/L_r$ instead of ν/L_r. This dissipation term is called a damping term as it behaves as a high-frequency damping term as shown in Refs.[37, 38]. The damping term has been found to be essential to robust and accurate computations on highly irregular grids, where L_r is defined by

$$L_r = \frac{1}{2}|\mathbf{e}_{jk}\cdot\hat{\mathbf{n}}_{jk}|, \quad \mathbf{e}_{jk} = \mathbf{x}_k - \mathbf{x}_j, \tag{16}$$

where $|\mathbf{e}_{jk}\cdot\hat{\mathbf{n}}_{jk}|$ is a skewness parameter and has the effect of increasing the damping for highly-skewed grids. It has also been shown in Refs.[37, 38] that there exists a special value of α that improves the order of accuracy from second order to fourth order on regular grids, not only for finite-volume methods but also for discontinuous Galerkin, and spectral-volume methods. See Ref.[39] for the extension to the compressible Navier-Stokes equations, Ref.[40] for the effects of damping on iterative convergence of implicit viscous solvers, and Refs.[41, 42, 43] for applications to three-dimensional practical turbulent-flow solvers.

In this paper, we explore the alternative. Namely, we discard $Res_j(1)$ by assuming that the solution u is given, and solve the remaining residual equations, $Res_j(2) = 0$ and $Res_j(3) = 0$, to compute gradients for a given solution. This is the new approach to deriving gradient algorithms as further discussed in the next section.

3 Derivation of Gradient Algorithm

To derive a gradient algorithm, we assume that the primal solution u is given at cells by some other means, e.g., a known function or a numerical solution for any problem. Then, we can ignore $Res_j(1)$, and use the rest of the residual equations to determine p and q:

$$-\frac{1}{2V_j}\sum_{k\in\{k_j\}}\left[(u_L+u_R)\hat{n}_x+L_r(p_{nR}-p_{nL})\hat{n}_x-\{c_jp_j+(1-c_j)p_k\}\,\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\right]A_{jk} = 0, \tag{17}$$

$$-\frac{1}{2V_j}\sum_{k\in\{k_j\}}\left[(u_L+u_R)\hat{n}_y+L_r(p_{nR}-p_{nL})\hat{n}_y-\{c_jq_j+(1-c_j)q_k\}\,\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\right]A_{jk} = 0. \tag{18}$$

These residuals form a globally-coupled linear system for the gradients, which can be expressed at the cell j as

$$\mathbf{M}_{jj}\mathbf{g}_j+\sum_{k\in\{k_j\}}\mathbf{M}_{jk}\mathbf{g}_k=\mathbf{b}_j, \tag{19}$$

where

$$\mathbf{M}_{jj}=\frac{1}{2V_j}\sum_{k\in\{k_j\}}\left\{-\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix}+L_r\begin{bmatrix}\hat{n}_x^2 & \hat{n}_x\hat{n}_y\\ \hat{n}_x\hat{n}_y & \hat{n}_y^2\end{bmatrix}+c_j\begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk}, \tag{20}$$

$$\mathbf{M}_{jk}=\frac{1}{2V_j}\left\{-\begin{bmatrix}\Delta x_{km}\hat{n}_x & \Delta y_{km}\hat{n}_x\\ \Delta x_{km}\hat{n}_y & \Delta y_{km}\hat{n}_y\end{bmatrix}-L_r\begin{bmatrix}\hat{n}_x^2 & \hat{n}_x\hat{n}_y\\ \hat{n}_x\hat{n}_y & \hat{n}_y^2\end{bmatrix}+(1-c_j)\begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk}, \tag{21}$$

$$\mathbf{g}_j=\begin{bmatrix}p_j\\ q_j\end{bmatrix},\quad \mathbf{g}_k=\begin{bmatrix}p_k\\ q_k\end{bmatrix},\quad \mathbf{b}_j=\frac{1}{2V_j}\sum_{k\in\{k_j\}}\begin{bmatrix}(u_j+u_k)\hat{n}_x\\ (u_j+u_k)\hat{n}_y\end{bmatrix}A_{jk}. \tag{22}$$

The right hand side $\mathbf{b}_j$ is the GG gradient over the cell j, and the gradients are coupled with neighbors on the left hand side, meaning that an implicit extension of the GG gradient method has just been derived. In this paper, the method thus derived is referred to as the implicit Green-Gauss (IGG) method. The IGG method is similar to the iterative Green-Gauss techniques in Ref.[4] in that both are implicit, but quite different in that the IGG method is derived from a hyperbolic diffusion scheme, includes a jump term derived from the dissipation term, and is a more flexible algorithm having adjustable parameters in the source quadrature and the jump coefficient, which will be discussed further below. The IGG method is also similar to the VR method, but different in that the IGG method has the GG gradient on the right hand side while the VR method has a LSQ gradient on the right hand side (see Appendix B). It is emphasized that the diffusion equation has no influence on the IGG method because the diffusion equation residual has been ignored and is not used at all. The IGG method is a purely mathematical algorithm for computing gradients for a given function or any kind of numerical solution.

Note that the gradient system (19) is exact for linear functions: p and q that satisfy the gradient system are exact for linear u. This means, just like linear LSQ methods, that the gradients are obtained with first-order accuracy on irregular grids for a given function or a second-order accurate numerical solution (i.e., a numerical solution obtained by an algorithm that is exact for linear functions). Second-order accuracy may be expected on regular grids by error cancellation (see Appendix A). Note also that $\mathbf{M}_{jj}$ can be simplified, by the identity:

$$\sum_{k\in\{k_j\}}\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix}A_{jk}=\frac{1}{2}\sum_{k\in\{k_j\}}\begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\mathbf{n}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\mathbf{n}_{jk}\end{bmatrix}=\begin{bmatrix}V_j & 0\\ 0 & V_j\end{bmatrix}, \tag{23}$$

where $\mathbf{n}_{jk}=\hat{\mathbf{n}}_{jk}A_{jk}$, as

$$\mathbf{M}_{jj}=\begin{bmatrix}c_j-1/2 & 0\\ 0 & c_j-1/2\end{bmatrix}+\frac{1}{2V_j}\sum_{k\in\{k_j\}}L_r\begin{bmatrix}\hat{n}_x^2 & \hat{n}_x\hat{n}_y\\ \hat{n}_x\hat{n}_y & \hat{n}_y^2\end{bmatrix}A_{jk}. \tag{24}$$

The gradient system can be solved efficiently by iterative methods. The block Jacobi iteration can be performed in the form:

$$\mathbf{g}_j^{n+1} = \left(- \sum_{k \in \{k_j\}} \mathbf{C}_{jk} \mathbf{g}_k^n + \mathbf{c}_{jk}(u_j + u_k) \right), \tag{25}$$

where

$$\mathbf{C}_{jk} = \mathbf{M}_{jj}^{-1} \mathbf{M}_{jk}, \quad \mathbf{c}_{jk} = \mathbf{M}_{jj}^{-1} \frac{1}{2V_j} \sum_{k \in \{k_j\}} \hat{\mathbf{n}}_{jk} A_{jk}, \tag{26}$$

or in the relaxed form:

$$\mathbf{g}_j^{n+1} = \mathbf{g}_j^n + \omega \left[\left(- \sum_{k \in \{k_j\}} \mathbf{C}_{jk} \mathbf{g}_k^n + \mathbf{c}_{jk}(u_j + u_k) \right) - \mathbf{g}_j^n \right], \tag{27}$$

where ω is a relaxation factor, and n is the iteration counter. In this work, we update $\mathbf{g}_j$ immediately in looping over cells, which corresponds to the Gauss-Seidel iteration. The relaxation factor ω is set to be 1.0 in this work, but a smaller value can help the method converge when $\omega = 1$ fails (e.g., for some extremely irregular grids); it may be adjusted based the residual behavior during the iteration for robustness. Typically, a couple of orders of magnitude reduction in the linear residual is sufficient to obtain the expected accuracy, except for a special case where the method yields fourth-order accuracy as we will discuss later.

The IGG method requires to store in each cell the coefficient matrix $\mathbf{C}_{jk}$ and the vector $\mathbf{c}_{jk}$ for all the face neighbors. This appears a large storage requirement, but it is exactly the same as LSQ methods with extended neighbors. Considering cells away from boundaries, we find that a LSQ method with face neighbors and their face neighbors requires 12 and 9 LSQ coefficient vectors to be stored in a cell on regular quadrilateral (Figure 5(a)) and triangular (Figure 8(a)) grids, respectively. On the other hand, the IGG method, as it depends only on the face neighbors, requires 4 and 3 matrices $\mathbf{C}_{jk}$ and vectors $\mathbf{c}_{jk}$, on regular quadrilateral and triangular grids, respectively, which are equivalent to 12 and 9 vectors in total. Therefore, the IGG method does not require any additional storage. In terms of computational cost, this means that the IGG method costs exactly the same number of vector-scalar multiplications as the LSQ method. The same is true in three dimensions. In realistic flow simulations, if three different LSQ methods are used for inviscid, viscous, and source terms, the storage requirement and the cost are tripled. Then, the IGG method, if a single method applies to all terms, will be three times more efficient.

In the gradient algorithm, we define the parameter L_r as

$$L_r = \alpha_g |\mathbf{e}_{jk} \cdot \hat{\mathbf{n}}_{jk}|, \tag{28}$$

where α_g is a constant. There exists a special value of α_g that will achieve fourth-order accuracy in the gradients on regular grids. Further discussion will be given in the next section.

For regular skewed grids, such as equilateral-triangular grids or right isosceles triangular grids, the Gauss-Seidel iteration converges relatively well although it slows down slightly compared with Cartesian grids. In these skewed grids, interior cells have one perfectly or nearly orthogonal face. A serious convergence difficulty may be encountered in the case that cells have all skewed faces, e.g., for high-aspect-ratio irregular triangular grids. To increase the diagonal dominance and improve iterative convergence for these cases, we define c_j as follows:

$$c_j = 35(s_j - 1)^6 - (s_j - 1) + 1, \tag{29}$$

where

$$s_j = \begin{cases} 0.75 s_j^{min} + 0.25 s_j^{max} & \text{if } |s_j^{max} - 1| < 10^{-6} \\ s_j^{min} & \text{otherwise,} \end{cases}, \tag{30}$$

$$s_j^{min} = \min_k |\hat{\mathbf{e}}_{jk} \cdot \hat{\mathbf{n}}_{jk}|, \quad s_j^{max} = \max_k |\hat{\mathbf{e}}_{jk} \cdot \hat{\mathbf{n}}_{jk}|, \tag{31}$$

where $\hat{\mathbf{e}}_{jk} = \mathbf{e}_{jk}/|\mathbf{e}_{jk}|$, and $|\hat{\mathbf{e}}_{jk} \cdot \hat{\mathbf{n}}_{jk}|$ is the skewness measure at the face between the cells j and k, which decreases towards zero as the skewness increases. Equation (30) is designed to yield a very small value of s_j

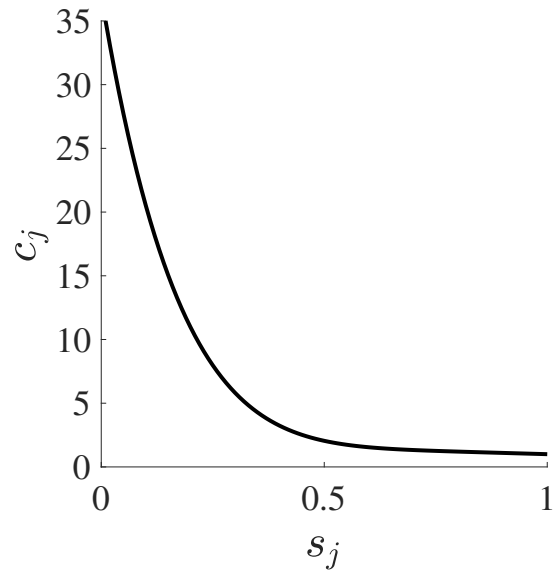


Figure 2: Equation (29): c_j as a function of the skewness measure s_j.

(thus a large c_j) for cells having all highly-skewed faces. As we will see later in numerical results, it is the skewness, not the cell-aspect-ratio by itself, that greatly affects the performance of the IGG method; but cells with all faces skewed arise typically on high aspect ratio grids. Figure 2 shows the variation of the function (29). In this way, we have $c_j = 1$ for Cartesian grids, a slightly large value for cells with both skewed and non-skewed faces, and a very large value for cells with all highly-skewed faces.

Finally, we emphasize again that the IGG method is a general gradient method applicable to any problem, where gradient computations are required whether for a given function or for a numerical solution. It can be directly applied to each variable in a finite-volume scheme for the Euler or Navier-Stokes equations, which will be discussed in a separate paper. It is also possible to derive a gradient algorithm from a discretization of a hyperbolic Navier-Stokes system [35, 44, 45]. In this case, since the extra variables used to form a hyperbolic viscous system are viscous stresses [44] (velocity gradients scaled by the viscosity [35, 45]) and heat fluxes, a system of equations will be derived not for gradients but for the viscous stresses and heat fluxes. Such an algorithm may turn out to be useful for problems where gradients are discontinuous but the viscous stresses are continuous (e.g., multi-phase flows) as briefly discussed in Ref.[30].

4 Regular Quadrilateral Grids

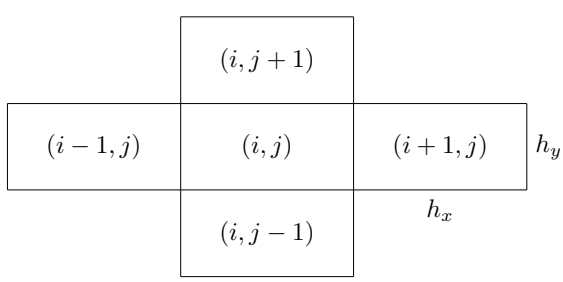


Figure 3: Regular quadrilateral grid.

On a quadrilateral grid uniformly spaced in each coordinate direction as in Figure 3, we have $c_j = 1$ and the

gradient system decouples as

$$\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)p_{i-1,j}+\left(\frac{1}{2}+\alpha_g\right)p_{i,j}+\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)p_{i+1,j}=\frac{u_{i+1,j}-u_{i-1,j}}{2h_x}, \tag{32}$$

$$\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)q_{i,j-1}+\left(\frac{1}{2}+\alpha_g\right)q_{i,j}+\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)q_{i,j+1}=\frac{u_{i,j+1}-u_{i,j-1}}{2h_y}. \tag{33}$$

Each system is strictly diagonally dominant for $\alpha_g > 0$:

$$\begin{aligned}\left(\frac{1}{2}+\alpha_g\right) &> \frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)+\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)=\frac{1}{2}-\alpha_g \quad \text{for} \quad 0<\alpha_g<1/2,\\ \left(\frac{1}{2}+\alpha_g\right) &> -\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)-\frac{1}{2}\left(\frac{1}{2}-\alpha_g\right)=\alpha_g-\frac{1}{2} \quad \text{for} \quad \alpha_g\geq 1/2,\end{aligned} \tag{34}$$

and therefore the Gauss-Seidel iteration is guaranteed to converge. Of course, it can be solved alternatively by an efficient tridiagonal solver. It is easy to see from Equations (32) and (33) that there is a special value of α_g:

$$\alpha_g=\frac{1}{2}, \tag{35}$$

which eliminates the implicit coupling and leads to the gradient system reduced to the central difference formulas:

$$p_{i,j}=\frac{u_{i+1,j}-u_{i-1,j}}{2h_x}, \tag{36}$$

$$q_{i,j}=\frac{u_{i,j+1}-u_{i,j-1}}{2h_y}. \tag{37}$$

A better choice is

$$\alpha_g=\frac{1}{6}, \tag{38}$$

which yields the classical fourth-order compact formula [46] for both p and q:

$$\frac{1}{6}\,p_{i-1,j}+\frac{2}{3}\,p_{i,j}+\frac{1}{6}\,p_{i+1,j}=\frac{u_{i+1,j}-u_{i-1,j}}{2h_x}, \tag{39}$$

$$\frac{1}{6}\,q_{i,j-1}+\frac{2}{3}\,q_{i,j}+\frac{1}{6}\,q_{i,j+1}=\frac{u_{i,j+1}-u_{i,j-1}}{2h_y}. \tag{40}$$

The superior accuracy, however, comes with an additional cost if iterative methods are used to solve the system. The tolerance needs to be at least eight orders of magnitude reduction in order to observe fourth-order accuracy. Another important remark is that the IGG method with $\alpha_g = 1/6$ achieves fourth-order accuracy through the cells adjacent boundaries (where the classical fourth-order compact formula does not apply) if the boundary flux is evaluated directly by the boundary value of the solution as discussed in the next section.

The equivalence to the classical compact scheme implies that the IGG method can be considered as a compact finite-difference scheme extended to unstructured grids. The method can be used to compute a flux divergence of a conservation law, which can then be integrated in time to generate a time-stepping scheme. Such a scheme may be explored in future. Here, we apply the modified-wavenumber analysis typical in compact schemes [47] to the IGG method, and discuss its approximation property. Following the analysis in Ref.[47], we consider a Fourier mode, $u = u_\beta \exp(i\beta x/h_x)$, where u_β is the amplitude, and β is the wavenumber ranging from 0 to π. The exact derivative $\partial_x u$ is $i\beta/h_x u$. Substituting $p = i\beta'/h_x u$, where β' is the modified wavenumber, into the IGG residual (32), and solving for β', we obtain

$$\beta'=\frac{-2\sin(\beta)}{(2\alpha_g-1)\cos(\beta)-(2\alpha_g+1)}, \tag{41}$$

which is plotted for several different values of α_g in Figure 4. Clearly, the gradient is better approximated with a broader range of wavenumbers with a smaller value of α_g, and the approximation reaches fourth-order

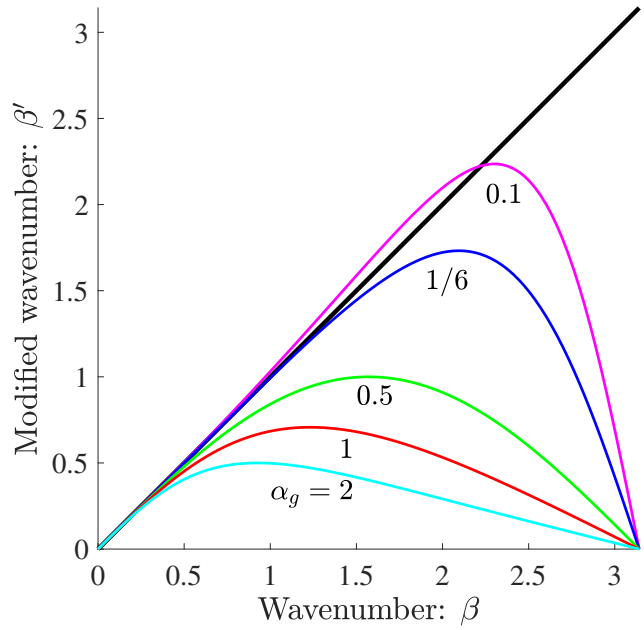


Figure 4: Modified wavenumber versus wavenumber. The black thick line indicates the exact relation: $\beta' = \beta$.

accuracy with $\alpha_g = 1/6$. Further decreasing α_g leads to more frequencies picked up although the order of accuracy reduces to second-order. These plots indicate that the gradient approximation will be smoother for a larger value of α_g as β' becomes significantly smaller than the exact value β for higher wavenumbers. It is interesting to note that the parameter α_g comes from the dissipation term in the generating hyperbolic diffusion scheme, and still plays a role of providing dissipation in the gradient algorithm. Later, we will show that the IGG method can yield a smooth gradient variation for a discontinuous function with a larger α_g, and it allows an implicit finite-volume solver to converge for a discontinuous solution with mild oscillations.

5 Boundary Treatment

As mentioned earlier, the IGG method is a purely mathematical algorithm for computing gradients at cells for a set of values $\{u_j\}$ given at cell centers. Therefore, there is no physics involved in the gradient computation, and no boundary conditions are required. The set of values may be given by a known function or by a numerical scheme solving a target partial differential equation; they enter the right hand side of the gradient system, i.e., the vector $\mathbf{b}_j$ in Equation (22), and serve as an input to the gradient system. Because the vector $\mathbf{b}_j$ involves the arithmetic average of the values across a face, the algorithm needs to be modified at a boundary face, where a face neighbor is not available. The focus here is, therefore, on how to evaluate the averaged value $(u_j + u_k)/2$ at a boundary face, where u_k is not available, and modify $\mathbf{M}_{jj}$ to preserve the design accuracy.

To derive suitable boundary modifications, it is convenient to re-derive the gradient system from the hyperbolic diffusion scheme with the Dirichlet or Neumann condition associated with diffusion problems imposed at a boundary face. Let us return, for a moment, to the diffusion problem and the hyperbolic scheme as described in Section 2. In the hyperbolic method, boundary conditions can be imposed strongly [26, 31] or weakly [22, 28]. For cell-centered methods considered here, the latter would be more suitable. Also, the Dirichlet condition is useful for our purpose, which can be imposed weakly through the numerical flux (7) with the right state $\mathbf{u}_R$ defined as

$$\mathbf{u}_R = (u_b, p_L, q_L), \tag{42}$$

where u_b is the value given by the boundary condition, and (p_L, q_L) are simply the copy of the left state as the gradient is not known at a boundary. This completes the residual equations (10)-(12) over a cell adjacent to a boundary. Then, to derive a gradient system, we ignore the residual equation for the diffusion equation, and

build a gradient system with the rest of the residual equations:

$$-\frac{1}{2V_j}\left(\sum_{k\in\{k_j^{int}\}}\phi_{jk}^{int}A_{jk} + \sum_{k\in\{k_j^b\}}\phi_{jk}^{b}A_{jk}\right) = 0, \tag{43}$$

$$-\frac{1}{2V_j}\left(\sum_{k\in\{k_j^{int}\}}\psi_{jk}^{int}A_{jk} + \sum_{k\in\{k_j^b\}}\psi_{jk}^{b}A_{jk}\right) = 0, \tag{44}$$

where ϕ_{jk}^{int} and ψ_{jk}^{int} are fluxes at interior faces $\{k_j^{int}\}$ of the cell j, and ϕ_{jk}^{b} and ψ_{jk}^{b} are fluxes at boundary faces $\{k_j^b\}$ of the cell j:

$$\phi_{jk}^{int} = (u_L+u_R)\hat{n}_x + L_r(p_{nR}-p_{nL})\hat{n}_x - \{c_jp_j+(1-c_j)p_k\}(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}), \tag{45}$$

$$\phi_{jk}^{b} = (u_L+u_b)\hat{n}_x - p_j(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}), \tag{46}$$

$$\psi_{jk}^{int} = (u_L+u_R)\hat{n}_y + L_r(p_{nR}-p_{nL})\hat{n}_y - \{c_jq_j+(1-c_j)q_k\}(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}), \tag{47}$$

$$\psi_{jk}^{b} = (u_L+u_b)\hat{n}_y - q_j(\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}), \tag{48}$$

where u_b denotes the boundary value at the k-th boundary face in $\{k_j^b\}$. Note that the gradient system (43) and (44) has nothing to do with the diffusion equation nor the Dirichlet boundary condition. It is just a system that determines gradients at cells for arbitrarily values of u given at cell centers and at the midpoints of the boundary faces. The system can be written in the form (19) with

$$\begin{aligned}\mathbf{M}_{jj} &= \frac{1}{2V_j}\sum_{k\in\{k_j^{int}\}}\left\{-\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix} + L_r\begin{bmatrix}\hat{n}_x^2 & \hat{n}_x\hat{n}_y\\ \hat{n}_x\hat{n}_y & \hat{n}_y^2\end{bmatrix} + c_j\begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk}\\ &+ \frac{1}{2V_j}\sum_{k\in\{k_j^{b}\}}\left\{-\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix} + \begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk},\end{aligned} \tag{49}$$

$$\mathbf{b}_j = \frac{1}{2V_j}\sum_{k\in\{k_j^{int}\}}\begin{bmatrix}(u_j+u_k)\hat{n}_x\\ (u_j+u_k)\hat{n}_y\end{bmatrix}A_{jk} + \frac{1}{2V_j}\sum_{k\in\{k_j^{b}\}}\begin{bmatrix}(u_j+u_b)\hat{n}_x\\ (u_j+u_b)\hat{n}_y\end{bmatrix}A_{jk}, \tag{50}$$

and $\mathbf{M}_{jk}=\mathbf{0}$ for $k\in\{k_j^b\}$. This is the method derived naturally from the hyperbolic scheme, and in this paper this boundary procedure is referred to as B0. This procedure requires the boundary value u_b to be available. If the gradients are sought for a given function $u(x,y)$, the boundary value can be directly computed by the function. However, for numerical solutions, the solution values are always available only at cell centers. If boundary values are available, e.g., from a no-slip condition in solving the Navier-Stokes equations, then u_b can be specified, but that is not always the case (e.g., no values are specified in a supersonic outflow condition).

In the case that no boundary values are available, we can set

$$u_b = u_L, \tag{51}$$

which gives

$$\begin{aligned}\mathbf{M}_{jj} &= \frac{1}{2V_j}\sum_{k\in\{k_j^{int}\}}\left\{-\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix} + L_r\begin{bmatrix}\hat{n}_x^2 & \hat{n}_x\hat{n}_y\\ \hat{n}_x\hat{n}_y & \hat{n}_y^2\end{bmatrix} + c_j\begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk}\\ &+ \frac{1}{2V_j}\sum_{k\in\{k_j^{b}\}}\left\{-2\begin{bmatrix}\Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x\\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y\end{bmatrix} + \begin{bmatrix}\Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk} & 0\\ 0 & \Delta\mathbf{x}_{jm}\cdot\hat{\mathbf{n}}_{jk}\end{bmatrix}\right\}A_{jk},\end{aligned} \tag{52}$$

$$\mathbf{b}_j = \frac{1}{2V_j} \sum_{k \in \{k_j^{int}\}} \begin{bmatrix} (u_j + u_k)\hat{n}_x \\ (u_j + u_k)\hat{n}_y \end{bmatrix} A_{jk} + \frac{1}{V_j} \sum_{k \in \{k_j^b\}} \begin{bmatrix} u_j \hat{n}_x \\ u_j \hat{n}_y \end{bmatrix} A_{jk}, \tag{53}$$

and $\mathbf{M}_{jk} = \mathbf{0}$ for $k \in \{k_j^b\}$. This procedure is referred to as B1. It is still exact for linear functions, and therefore second/first-order accuracy is maintained on regular/irregular grids. This procedure is useful especially when the IGG method is used to compute gradients required for the solution reconstruction in a finite-volume scheme. Note that physical boundary conditions in a problem that the finite-volume scheme is trying to solve need to be satisfied by the finite-volume scheme, not by the gradient method. Therefore, the gradient method is applicable as long as numerical solutions are available at cell centers. If solution values are available as a physical boundary condition (e.g., inflow condition), the gradient method can incorporate them, although it is not necessary, by the B0 procedure or the B2 procedure discussed below. However, such is not always possible: e.g., no solution variables are specified at a supersonic outflow. The procedure B1 is useful because it does not require any boundary solution information. Taking $u_b = u_L$ simply means that the gradients will be computed based on a set of values $\{u_j\}$ given at cell centers only, without any reference to the values at the boundary. Therefore, the procedure B1 can be applied to finite-volume schemes for any physical boundary conditions. It is possible that the resulting gradients may be under-estimated if the value u changes extremely rapidly from the cell center to the boundary face within a cell (e.g., a cell adjacent to a viscous wall in a highly under-resolved boundary layer) because the algorithm does not see the true boundary value. However, such an under-estimated gradient may be considered as more accurate or consistent. For example, finite-volume solutions for a high-Reynolds-number boundary-layer flow are known to converge to a slip flow on highly under-resolved mesh [48]; thus an under-estimated gradient is more consistent with the nearly-inviscid slip-flow solution.

If the boundary values are available, we can also directly evaluate the boundary fluxes ϕ_{jk}^b and ψ_{jk}^b, which corresponds to setting

$$u_L = u_R = u_b, \tag{54}$$

which leads to

$$\begin{aligned} \mathbf{M}_{jj} &= \frac{1}{2V_j} \sum_{k \in \{k_j^{int}\}} \left\{ - \begin{bmatrix} \Delta x_{jm}\hat{n}_x & \Delta y_{jm}\hat{n}_x \\ \Delta x_{jm}\hat{n}_y & \Delta y_{jm}\hat{n}_y \end{bmatrix} + L_r \begin{bmatrix} \hat{n}_x^2 & \hat{n}_x\hat{n}_y \\ \hat{n}_x\hat{n}_y & \hat{n}_y^2 \end{bmatrix} + c_j \begin{bmatrix} \Delta\mathbf{x}_{jm} \cdot \hat{\mathbf{n}}_{jk} & 0 \\ 0 & \Delta\mathbf{x}_{jm} \cdot \hat{\mathbf{n}}_{jk} \end{bmatrix} \right\} A_{jk} \\ &+ \frac{1}{2V_j} \sum_{k \in \{k_j^b\}} \begin{bmatrix} \Delta\mathbf{x}_{jm} \cdot \hat{\mathbf{n}}_{jk} & 0 \\ 0 & \Delta\mathbf{x}_{jm} \cdot \hat{\mathbf{n}}_{jk} \end{bmatrix} A_{jk}, \end{aligned} \tag{55}$$

$$\mathbf{b}_j = \frac{1}{2V_j} \sum_{k \in \{k_j^{int}\}} \begin{bmatrix} (u_j + u_k)\hat{n}_x \\ (u_j + u_k)\hat{n}_y \end{bmatrix} A_{jk} + \frac{1}{V_j} \sum_{k \in \{k_j^b\}} \begin{bmatrix} u_b \hat{n}_x \\ u_b \hat{n}_y \end{bmatrix} A_{jk}, \tag{56}$$

and $\mathbf{M}_{jk} = \mathbf{0}$ for $k \in \{k_j^b\}$. This procedure is referred to as B2. It will be shown in numerical experiments that fourth-order accuracy is achieved only with this boundary procedure on regular quadrilateral grids.

Finally, it is possible to derive a gradient system from the hyperbolic diffusion scheme incorporating the Neumann condition that specifies the gradient normal to the boundary [22, 28]. The resulting system may be useful if one wishes to impose the Neumann condition (e.g., adiabatic wall) not only in the finite-volume scheme but also in the gradient computation [49]. Here, such a constrained gradient computation is not considered here because physical boundary conditions for the finite-volume scheme do not need to be imposed on the gradient computation and also it is not clear if imposing them on the gradient brings significant benefits for extra work and storage to keep different gradient systems for different variables in a target system of partial differential equations with different types of boundary conditions applied to different variables.

6 Numerical Results

First, we investigate the IGG method for computing gradients for a known function with various types of unstructured grids, and compare it with other gradient methods. The investigation focuses on gradient accuracy and iterative convergence behavior for the IGG method. Then, we study its performance as an alternative to the GG or LSQ methods in an implicit finite-volume solver for the advection-diffusion and Burgers' equations.

6.1 Gradient of Functions

In this section, we investigate the performance of the IGG method for computing gradients for a known function on various types of unstructured grids. Our target gradient scheme is the IGG method with $\alpha_g = 1/6$ and the boundary closure B2 (i.e., boundary flux evaluated by the given function value), which is denoted by IGG. For IGG and VR iterations, we set the tolerance to be eight orders of magnitude residual reduction in the L_1 norm for regular quadrilateral grids and three orders for all other grids. Errors are measured in the vector L_1 norm separately for interior and boundary cells. The boundary cells are defined as those having at least one boundary face. Note that the gradients are considered as point values defined at the cell centroid. Results are compared with those obtained by the GG method, a LSQ method, and the VR method. The VR method used in this study is described in Appendix B. The LSQ method is a weighted non-compact LSQ method with face neighbors and their face neighbors as described in Appendix C.

6.1.1 Isotropic grids

We consider the following function:

$$u(x, y) = \sin(\pi x)\sin(\pi y), \tag{57}$$

defined in a unit square domain, and compute its gradients by the IGG method and others on uniform quadrilateral grids and irregular triangular grids.

Cartesian grids: Results for the quadrilateral grids are presented in Figure 5. Figure 5(a) shows the coarsest grid and the contours of the function (57). Iterative convergence results are shown in Figure 5(b), where the maximum of the two residuals for p and q normalized by the initial residuals are plotted versus the iteration number. The IGG method with two other boundary closures are also considered for this problem: IGG-B0 and IGG-B1. As can be seen from the figure, both the IGG and VR methods converge very rapidly and the IGG method converges slightly faster. Convergence results for the IGG-B0 and IGG-B1 methods are almost identical to those in the figure for IGG, and therefore are not shown.

Error convergence results are given in Figures 5(c) and 5(d) for the interior cells. As expected, the IGG method yields fourth-order accuracy. On the other hand, the IGG-B0 and IGG-B1 methods give third-order accuracy. All other methods yield second-order accuracy as expected on regular grids. Results for the boundary cells are shown in Figures 5(e) and 5(f). All methods, except the LSQ method and the IGG method, are second-order accurate. The LSQ method gives results close to first-order accuracy, which is typical since the LSQ stencil is not regular at boundary cells and therefore error cancellations cannot be expected. Remarkably, the IGG method gives fourth-order accuracy in the boundary cells, where the classical fourth-order scheme does not apply. In effect, the IGG method automatically forms a fourth-order accurate scheme at these cells when closed by the B2 boundary procedure.

Irregular triangular grids: We consider fully irregular triangular grids for the same function. The coarsest grid is shown in Figure 6(a). As in the previous case, the IGG and VR methods converge rapidly as shown in Figure 6(b). Error convergence results are given in Figure 6(c)-6(f). All methods except the GG method yield first-order accurate gradients as expected. Note that the best we can expect is first-order accuracy on irregular grids for methods that are exact linear functions. The GG method is inconsistent as it is not exact for linear functions on these grids, which can be clearly seen in the results.

6.1.2 High-aspect-ratio grids

In this section, we consider high-aspect-ratio grids in a domain $(x, y) \in [0, 1] \times [0, 0.0005]$ and the following function:

$$u(x, y) = \sin(\pi x)\sin(4000\pi y), \tag{58}$$

which models a typical boundary layer problem, where the solution variation is predominant in the direction of small grid spacing and an anisotropic grid is specifically tailored to represent the solution anisotropy. Four types of grids are considered: regular quadrilateral, regular triangular, isosceles triangular, and irregular triangular grids.

Regular quadrilateral grids: The coarsest grid and the function contours are shown in Figure 7(a). Results are very similar to those for the Cartesian grids discussed in the previous section: rapid iterative convergence as in Figure 7(b); fourth-order accuracy by the IGG method, third-order accuracy by the IGG-B0 and IGG-B1 methods, second-order accuracy by all others in the interior cells as shown in Figures 7(c) and 7(d); second-order accuracy for all in the boundary cells as in Figures 7(e) and 7(f), except the IGG method, which is fourth-order accurate even in the boundary cells.

Regular triangular grids: Figure 8 shows the results for regular triangular grid with the coarsest grid shown in Figure 8(a). The IGG and VR methods converge rapidly as shown in Figure 8(b); with the VR method being slightly faster. Error convergence results shown Figures 8(c)-8(f) indicate that all methods yield first-order accuracy both in the interior and boundary cells. Exceptions are that the LSQ method is second-order in $\partial_y u$, and the IGG method give nearly second-order accuracy in the interior cells.

Isosceles triangular grids: Results are shown in Figure 9 for isosceles triangular grids, which are generated from equilateral triangular grids by rescaling the y-coordinates, with the coarsest grid shown in Figure 9(a). Figure 9(b) shows that the IGG and VR methods converge quickly, and that the VR method converges faster. As can be expected from the fact that these grids are irregular involving two different types of cell stencils, all methods are first-order accurate. An exception is that the IGG method is close to second-order in the interior cells.

Irregular triangular grids: A series of fully irregular triangular grids are considered. The coarsest grid is shown in Figure 10(a). These grids are highly-skewed in all faces of a cell. This is the case where the IGG method with $c_j = 1$ encountered a significant slow down in iterative convergence. The convergence is greatly improved with the formula (29), but it is slower than the VR method as can be seen in Figure 10(b). Error convergence results in Figures 10(c)-10(f) show that the GG method is inconsistent, which is expected, and all others yield first-order accuracy.

6.1.3 Highly curved grids

In this section, we investigate the gradient methods for highly-curved thin grids, known as high-Γ grids in Ref.[1]. It is well known that accuracy of gradients significantly deteriorates on high-Γ grids [2]. To study the performance of the IGG method, we consider three types of such grids: quadrilateral, triangular, and irregular triangular grids with the following function:

$$u(x, y) = \sin(100\pi r + \pi/6) + 0.5\sin(\theta), \tag{59}$$

where r is the radial distance from the origin and θ is the angle taken counterclockwise with respect to the positive x-axis.

Quadrilateral grids: Curved quadrilateral grids are generated from Cartesian grids by mapping a square domain to a thin curved domain. In these grids, the nodes are aligned straight from a node on the inner boundary to the corresponding node at the outer boundary. The domain and the function contours are shown in Figure 11(a). Figure 11(b) shows a close view of the grid. For these grids, the IGG method converges very rapidly, but VR method is very slow to converge as shown in Figure 11(c). Error convergence results in Figures 11(d)-11(g) reveal that the VR method is very inaccurate and the LSQ method also gives large errors although nearly first-order accuracy is observed. The IGG method, on the other hand, yields much lower errors with first-order error convergence. The GG method also gives accurate results.

Triangular grids: Curved triangular grids are generated from regular triangular grids by the same mapping as before. Again, the nodes are aligned straight from a node on the inner boundary to the corresponding node at the outer boundary. The domain and the function contours are shown in Figure 12(a). The close view of the grid is shown in Figure 12(b). As shown in Figure 12(c), the IGG method took more iterations to converge compared with the previous cases. The VR method took orders of magnitude more iterations than the IGG method, and slows down further for fine grids. Error convergence results shown in Figures 12(d)-12(g) show that the VR method is inaccurate, the LSQ method is slightly better, the GG method is inconsistent in the interior cells, and the IGG method maintains first-order accuracy for all cells.

Irregular triangular grids: Curved irregular triangular grids are generated from irregular triangular grids by the same mapping as before. In these grids, the nodes are perturbed randomly, and therefore the nodes are not aligned straight from a node on the inner boundary to the corresponding node at the outer boundary. The domain and the function contours are shown in Figure 13(a), and the close view is shown in Figure 13(b). Figure 13(c) shows that the IGG method took more iterations to converge compared with the previous case, and the VR method converges as slowly as in the previous case. Error convergence results are similar to those in the previous case. The GG method exhibits inconsistency again, and now even in the boundary cells.

6.1.4 Discontinuous functions

Consider a discontinuous function:

$$u = \begin{cases} 1.0 & x \leq 0.5, \\ -1.0 & x > 0.5, \end{cases} \tag{60}$$

in a unit square domain (see Figure 14(a)). It can be considered as a limit: $\lim_{\epsilon \to 0^+} -\tanh(x/\epsilon)$, and the derivative $\partial_x u$ has the limit

$$\partial_x u = \lim_{\epsilon \to 0^+} -\frac{1 - \tanh^2(x/\epsilon)}{\epsilon}, \tag{61}$$

which leads to the Dirac delta function with a minus sign. The y-derivative is zero and not considered here. This example is relevant to applications containing shock waves, for which a finite-volume scheme is often reverted to first order with gradients ignored for stability and better solution quality. We applied gradient algorithms to the discontinuous function to compute the gradient on a Cartesian grid. Results are shown in Figure 14. The GG method gives a very sharp gradient approximation with a finite peak as shown in Figure 14(b). Figure 14(c) shows that the LSQ method produces a slightly smeared approximation and exhibits a variation in y near the boundary. The VR method gives a more smeared derivative as in Figure 14(d), resembling the derivative of the hyperbolic tangent with a finite ϵ. The IGG method was tested with different values of α_g: 1/6, 1, and 2. As discusses in Section 4, the parameter α_g controls L_r, which comes from the dissipation term in the generating hyperbolic diffusion scheme, and the Fourier analysis suggests that high-frequency components will be greatly damped for large α_g. It is expected therefore that a larger value of α_g provides an dissipative effect in the gradients. This behavior is clearly seen in the results shown in Figures 14(e), 14(f), and 14(g) for $\alpha_g = 1/6$, $\alpha_g = 1$, and $\alpha_g = 2$, respectively. In the next section, we will show that the smoothed gradients computed by the VR and IGG methods allows an implicit finite-volume solver to converge for a discontinuous solution and produces solutions with little oscillations.

6.2 Implicit Finite-Volume Solver

In this section, we investigate the IGG method as an alternative to the LSQ or GG gradients in a second-order finite-volume scheme for a conservation law:

$$\partial_x \mathbf{f} + \partial_y \mathbf{g} = s, \tag{62}$$

where s is a forcing function. The fluxes are

$$\mathbf{f} = au - \nu \partial_x u, \quad \mathbf{g} = bu - \nu \partial_y u, \qquad \text{for advection diffusion equation,} \tag{63}$$

$$\mathbf{f} = \frac{u^2}{2}, \quad \mathbf{g} = u, \qquad \text{for Burgers' equation,} \tag{64}$$

where $(a, b) = (1.23, 2.57)$ and ν is a constant. A second-order cell-centered finite-volume discretization is given by

$$\mathbf{Res}_j = \sum_{k \in \{k_j\}} \phi_{jk} A_{jk} - s_j V_j. \tag{65}$$

The numerical flux ϕ_{jk} is given by the upwind flux for the advective term and the alpha-damping flux [38] for the diffusive term:

$$\phi_{jk} = \frac{1}{2}\left[f_n(u_L) + f_n(u_R)\right] - \frac{1}{2}\left(|a_n| + \frac{\nu\alpha}{\ell}\right)(u_R - u_L), \tag{66}$$

where $f_n = \mathbf{f}\hat{n}_x + \mathbf{g}\hat{n}_y$, $a_n = \partial f_n/\partial u$, $\alpha = 4/3$, $\ell = \frac{1}{2}|\mathbf{e}_{jk} \cdot \hat{\mathbf{n}}_{jk}|$, and

$$u_L = u_j + \nabla u_j \cdot \Delta \mathbf{x}_{jm}, \qquad u_R = u_k + \nabla u_k \cdot \Delta \mathbf{x}_{km}. \tag{67}$$

The gradients ∇u_j and ∇u_k are typically computed by the LSQ or GG methods, and here we apply the VR and IGG methods as alternatives. The residual equations are solved by an implicit solver,

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \Delta \mathbf{U}, \tag{68}$$

$$\frac{\partial \overline{\mathbf{Res}}}{\partial \mathbf{U}} \Delta \mathbf{U} = -\mathbf{Res}(\mathbf{U}^k), \tag{69}$$

where k is the iteration counter, $\mathbf{U}^k$ is the global vector of the solution u at the k-th iteration, and the Jacobian $\partial\overline{\mathbf{Res}}/\partial\mathbf{U}$ is the exact differentiation of the residual $\overline{\mathbf{Res}}$ with zero gradients. The linear system above is relaxed by the Gauss-Seidel method until the linear residual is reduced by an order of magnitude in the L_1 norm, with the maximum of 800 relaxations. The implicit solver is taken to be converged when the residuals are reduced by six orders of magnitude in the L_1 norm. In the rest of the paper, this implicit solver is referred to as the conservation-law implicit solver not to be confused with the implicit gradient solvers. Boundary conditions are implemented weakly through a numerical flux. Note that the numerical flux is formulated such that it naturally implements a characteristic boundary condition for the advection-dominated cases and Burgers' equation: by upwinding, the exact boundary value will not be used if the characteristic is going out of the domain. Further details on the discretization, the boundary condition implementation, and the construction of the implicit solver can be found in Ref.[40]. In all the gradient methods, the exact boundary value is not used; therefore, gradients are computed based only on the numerical solutions available at cells. This implementation corresponds to the boundary procedure B1 in the IGG method. For the parameter α_g, we set $\alpha_g = 1$ unless otherwise stated; the special value $\alpha_g = 1/6$ is not used since fourth-order accuracy cannot be expected on unstructured grids. At each conservation-law implicit solver iteration, the IGG and VR iteration is performed once for all problems. In the VR and IGG methods, the iteration begins with zero gradients at $k = 0$, and continues without reinitialization until the conservation-law implicit solver converges.

6.2.1 Advection-diffusion

First, we consider solving the linear advection-diffusion equation with $\nu = 10^{-2}\sqrt{a^2 + b^2}$ in a square domain for the exact solution [50]:

$$u(x, y) = \cos(2\pi\eta) \exp\left(\frac{-2\pi^2\nu}{1 + \sqrt{1 + 4\pi^2\nu^2}}\,\xi\right), \tag{70}$$

where $\xi = ax+by$, $\eta = bx-ay$. For this problem, the forcing term s is zero. The grids are the irregular triangular grids used in Section 6.1.1. These grids are relevant to practical unstructured-grid applications, e.g., a far-field grid. The coarsest grid and the solution contours are shown in Figure 15(a). The conservation-law implicit solver converged for all gradient methods: GG, LSQ, VR, and IGG; see Figure 15(b) for convergence histories for the finest grid. The conservation-law implicit solver was found to diverge with the GG gradients, and was made to converge by increasing the damping coefficient in Equation (66) to $\alpha = 5$. It is, however, the slowest to converge as seen in the figure. Figure 15(c) shows the same residual convergence in term of the CPU time; it demonstrates that the use of the VR and IGG methods does not greatly increase the computing time. Figures 15(d), 15(e), and 15(f) show error convergence results in the interior cells for u, $\partial_x u$, and $\partial_y u$, respectively. Inconsistency of the GG method leads to less than first-order error convergence in u. Other methods achieved expected orders of accuracy: second-order in u, and first-order in the gradients. Similar results are obtained in the boundary cells as shown in Figures 15(g), 15(h), and 15(i); the GG method gives first-order accuracy in u here.

Next, we consider the curved grids in Section 6.1.3: quadrilateral grids relevant to, for example, prismatic boundary layer grids, and irregular triangular grids relevant to fully unstructured anisotropic adaptation in boundary layers. The function (59) is set to be the exact solution by the method for manufactured solutions with the forcing term s_j computed numerically. The diffusion coefficient is given by $\nu = 10^{-5}\sqrt{a^2 + b^2}$.

For the quadrilateral grids, the conservation-law solution contours are shown in Figure 16(a). The conservation-law solver convergence histories are shown in Figure 16(b). The results are shown only for the second coarsest grid since the effect of high-curvature is more significant on coarser grids. The convergence was very rapid with

the LSQ method, and slowest with the IGG method. It is observed that the VR method leads to significant slow down in the later stage. The corresponding CPU time comparison is shown in Figure 16(c). For this problem, the LSQ and VR methods lead to the fastest and slowest computing times, respectively. Despite the slower convergence than the LSQ method, the IGG method gives the lowest level of errors as can be seen in Figures 16(d), 16(e), and 16(f) for the interior cells. The VR and LSQ gradients are very inaccurate, which are almost inconsistent, and the solution u is nearly first-order accurate. The GG gradients are more accurate, but the conservation-law solution is again first-order accurate. In contrast, with the IGG method, the solution u and the gradient are all second-order accurate. Results in the boundary cells are shown in Figures 16(g), 16(h), and 16(i). Again, the IGG method gives the most accurate results, but the solution u and the gradients are both first-order accurate.

For the triangular grids, Figure 17(a) shows the conservation-law solution contours, and Figure 17(b) shows the convergence histories for the second coarsest grid. Here, for the GG gradients, the conservation-law implicit solver diverged with $\alpha = 4/3$ and conservation-law convergence could only be obtained by increasing the damping coefficient to $\alpha = 80$. It can be seen that the conservation-law implicit solver slows down significantly with the VR gradient. Figure 17(c) shows the residual history versus the CPU time. The fastest conservation-law implicit solver convergence is obtained with the LSQ method, and the second fastest with the IGG method. The VR and GG gradients resulted in significantly slower convergence. Error convergence results, as shown in Figures 17(d), 17(e), and 17(f) for interior cells, show that second-order accuracy is achieved for the solution u in all methods except the GG method, which again gives first-order accuracy with inconsistent gradients. The LSQ and VR gradients resulted in large errors, but the conservation-law solution is very accurate. The IGG gradients produced the most accurate gradients, but the solution is not the most accurate. Results for the boundary cells are given in Figures 17(g), 17(h), and 17(i). As can be seen, the solution u is asymptotically first-order accurate for all methods. The IGG gradients are very accurate compared with others, but the accuracy of all the gradient methods deteriorates for finer grids.

To demonstrate the IGG method for highly irregular grids, we consider grids in an annular domain defined by the inner and outer circles of radii 0.5 and 3.0, respectively, for the solution,

$$u(x, y) = \exp\left(5(x^2 + y^2)\right), \tag{71}$$

which is made the exact solution by the method of manufactured solutions. A series of four grids have been generated with random diagonal swapping to increase irregularity and create severe skewness. The coarsest grid is shown in Figure 18(a), where the exact solution contours are over-plotted. The conservation-law implicit solver has been found to diverge on all grids with the GG method (which is likely to be inconsistent everywhere), and therefore only the results for other methods are shown. Figures 18(b) and 18(c) show iterative convergence results for the finest grid. The conservation-law implicit solver converged with the IGG, VR, and LSQ methods. A slight slow-down is observed for the VR method. Error convergence results are shown in Figures 18(d)-18(i). Second-order accuracy is observed for the solution and somewhat surprisingly for the gradients as well. No noticeable differences are seen among the three gradient methods.

Finally, these results demonstrate that the conservation-law implicit solver is stable with the VR and IGG gradients as expected. The implicit solver was found unstable with the LSQ gradients when a larger weight was used $m = 0.5$ (see Appendix C); this is the reason that a small weight $m = 0.25$ was used in the numerical study.

6.2.2 Burgers' equation with a discontinuous solution

To investigate effects of gradient algorithms in the presence of discontinuities, we consider solving Burgers' equation with $s = 0$ for an exact discontinuous solution similar to the function (60) with the left and right states given by 2.0 and -2.0, respectively. The problem was solved on a 32×32 Cartesian grid. All gradient methods were tested without a limiter, and the conservation-law implicit solver diverged with the GG gradient method. Therefore, results are shown only for the IGG, VR, and LSQ methods. The parameter α_g in the IGG method is set to be 25.0 for this problem. As shown in Figure 19(a), the conservation-law implicit solver converged with all the three gradient methods. The conservation-law solution plots are given in Figure 19(b) for IGG, Figure 19(d) for VR, and Figure 19(f) for LSQ. The gradient $\partial_x u$ is plotted in Figures 19(c) for IGG, 19(e) for VR, and 19(g) for LSQ. The solution involves over/under-shoots near the discontinuity in all cases, but is more accurate with the IGG method, which generates a much smoother gradient variation across the discontinuity. The results indicate that the IGG method can be tuned to smooth the gradient, in a manner somewhat similar to a gradient limiter, thereby allowing the conservation-law implicit solver to converge and

produce a solution with little oscillations. A similar approach has been proposed for the LSQ method in Ref.[51], where the LSQ weights are adaptively computed based on a local solution to develop a shock-capturing finite-volume scheme. Note, however, that monotonicity is not guaranteed for the IGG method because it is still a linear algorithm and such cannot lead to monotone second-order solutions by Godunov's order barrier theorem [52]. To guarantee monotonicity, the parameter α_g needs to be chosen adaptively based on a local solution variation or a conventional limiter [53] needs to be applied to the gradients at the reconstruction stage. The latter approach has been demonstrated for the VR gradients in Ref.[24]. The development of monotonicity preserving schemes with the IGG method is left as future work.

Next, to demonstrate that the method is applicable to unstructured grids, we consider the same discontinuous solution problem for an unstructured triangular grid with 512 elements. Again, no limiters are used. This time, the conservation-law implicit solver diverged with the GG, LSQ, and VR gradient methods. Figure 20(a) shows that the conservation-law implicit solver converged with the IGG method ($\alpha_g = 25$). As expected, the numerical solution is not completely monotone as shown in Figure 20(b). The gradient has a small variation across the discontinuity, but shows relatively large peaks near the boundaries. See Figure 20(c). This test case shows that the the IGG method is more robust than other gradient methods. To ensure monotonicity, as mentioned earlier, an adaptive α_g or a conventional limiter is required and such is left as future work.

Finally, although the iterative convergence with the IGG method is not always the fastest, the convergence is rapid at least over the first one or two orders of residual reduction. This property is important and encouraging for the Jacobian-Free Newton-Krylov methods and multigrid methods, where the current conservation-law implicit solver can be employed as a variable preconditioner and a smoother, respectively. In all these methods, the conservation-law implicit implicit solver is used only to reduce the residual by an order of magnitude or at a fixed number of iteration. Therefore, slow convergence for lower tolerance is not a major concern. Further details will be discussed in a separate paper.

7 Concluding Remarks

A new approach was proposed to constructing a gradient algorithm for unstructured grids, which is to derive a linear system of equations for gradients from a hyperbolic diffusion scheme. The resulting gradient system is implicit with the gradients coupled with face-neighbors on the left hand side, and the classical Green-Gauss gradient formula on the right hand side, which can be solved efficiently by the Gauss-Seidel iteration. The algorithm is adjustable with parameters inherited from the generating hyperbolic diffusion scheme. These parameters can be tuned to improve the gradient accuracy or to improve iterative convergence on highly skewed and distorted irregular grids as well as for discontinuous solutions. It has been demonstrated that second- or fourth-order gradient accuracy can be obtained on regular grids, and first-order accuracy, which is sufficient for second-order finite-volume schemes, can be obtained on irregular grids. The new gradient algorithm has been demonstrated for gradient computations on various types of regular and irregular grids. Fourth-order accuracy has been demonstrated for regular quadrilateral grids, and superior accuracy has been demonstrated especially for highly-curved high-aspect-ratio grids. The algorithm has been demonstrated as an alternative to least-squares or Green-Gauss gradients in a second-order finite-volume scheme. For a discontinuous solution, it has been shown that the method can be tuned to produce a smoothed gradient, allowing an implicit finite-volume solver to converge without a gradient limiter even on an unstructured grid, where the solver diverged with other gradient methods.

Future work includes applications to realistic fluid-dynamics simulations, and in particular flows with shock waves. For such applications, the parameter α_g may need to be adjusted to smooth the gradient only in the vicinity of a shock wave. The relationship of α_g to a monotonicity property remains to be discovered. In a practical point of view, a simpler approach is to apply a conventional limiter directly to the gradients in the reconstruction stage as it has been done for the VR gradients in Ref.[24]. Extensions to three dimensions are possible, but it must be performed carefully for non-simplex faces in order to preserve exactness for linear functions. For example, a quadrilateral face must be split into two triangular faces, and the numerical flux must be computed at centroids of the two triangular faces [54]. Extensions to grids of polyhedral elements are also possible by constructing a finite-volume discretization of the hyperbolic diffusion system on a polyhedral grid, and then discarding the residual component for the diffusion equation. To further improve convergence of an implicit finite-volume solver, a tightly-coupled solver may be developed, where the solution and the gradients are solved simultaneously by combining finite-volume residuals and the gradient system residuals into a single set of equations. The proposed gradient algorithm can be considered as a compact scheme extended to unstructured grids. It can be applied to fluxes in a conservation law to approximate the flux divergence, which

is then integrated in time to generate a time-accurate scheme. Such a scheme may found to be useful, especially when it is successfully extended to high-order. High-oder extensions are possible through the derivation of a gradient and higher-order derivative system from a high-order hyperbolic diffusion scheme: construct a high-order residual and discard the component for the diffusion equation. The high-order hyperbolic discontinuous Galerkin method [22, 29, 55], which generates residual equations for the gradient variables and their derivatives, is a promising example of a higher order extension.

Acknowledgments

The author acknowledges support by NASA under Contract No. 80LARC17C0004, and would like to thank Jeff White at NASA Langley Research Center for valuable comments and discussions. The author would also like to thank reviewers for useful and constructive comments and suggestions.

References

[1] B. Diskin and J. L. Thomas. Accuracy of gradient reconstruction on grids with high aspect ratio. *NIA Report No. 2008-12*, 2008.

[2] D. J. Mavriplis. Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. In *Proc. of 16th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2003-3986, Orlando, Florida, 2003.

[3] E. Shima, K. Kitamura, and T. Haga. Green-Gauss/weighted-least-squares hybrid gradient reconstruction for arbitrary polyhedra unstructured grids. *AIAA J.*, 51(11):2740–2747, 2013.

[4] F. Moukalled, L. Mangani, and M. Darwish. *The finite volume method in computational fluid dynamics: An introduction with OpenFOAM and matlab.* Fluid Mechanics and Its Applications, Volume 13. Springer International Publishing, 2015.

[5] Georg May and Antony Jameson. Unstructured algorithms for inviscid and viscous flows embedded in a unified solver architecture. In *Proc. of 43rd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2005-318, Reno, Nevada, 2005.

[6] N. T. Frink. Tetrahedral unstructured Navier-Stokes method for turbulent flows. *AIAA J.*, 36(11):1975–1982, 1998.

[7] Boris Diskin, James L. Thomas, C. Rumsey, and A. Schwöppe. Grid convergence for turbulent flows (invited). In *Proc. of 53rd AIAA Aerospace Sciences Meeting*, AIAA Paper 2015-1746, Kissimmee, Florida, January 2015.

[8] M. E. Braaten and S. D. Connel. Three-dimensional unstructured adaptive multigrid scheme for the Navier-Stokes equations. *AIAA J.*, 34(2):281–290, February 1996.

[9] Y. Kergaravat, F. Jacon, N. Okong'o, and D. Knight. A fully-implicit 2D Navier-Stokes algorithm for unstructured grids. *Int. J. Comput. Fluid Dyn.*, 9:179–196, 1998.

[10] T. J. Barth. Numerical aspects of computing viscous high Reynolds number flows on unstructured meshes. AIAA Paper 91-0721, 1991.

[11] N. T. Frink. Aerodynamic analysis of complex configurations using unstructured grids. AIAA Paper 91-3292, 1991.

[12] T. J. Barth and D. C. Jesperson. The design and application of upwind schemes on unstructured meshes. AIAA Paper 89-0366, 1989.

[13] W. K. Anderson and D. L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Comput. Fluids*, 23:1–21, 1994.

[14] M. Delanaye and J. A. Essers. Quadratic-reconstruction finite volume scheme for compressible flows on unstructured adaptive grids. *AIAA J.*, 35(4):631–639, 1997.

[15] A. Haselbacher and J. Blazek. Accurate and efficient discretization of Navier-Stokes equations on mixed grids. *AIAA J.*, 38(11):2094–2102, 2000.

[16] Carl Ollivier-Gooch, Amir Nejat, and Krzysztof Michalak. Obtaining and verifying high-order unstructured finite volume solutions to the euler equations. *AIAA J.*, 47(9):2015–2120, 2009.

[17] Emre Sozer, Christoph Brehm, and Cetin C. Kiris. Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered cfd solvers. In *Proc. of 52nd AIAA Aerospace Sciences Meeting*, AIAA Paper 2014-1440, National Harbor, Maryland, 2014.

[18] Mohagna J. Pandya, Neal T. Frink, Eijiang Ding, and Edward B. Parlette. Toward verification of USM3D extensions for mixed element grids. In *31st AIAA Applied Aerodynamics Conference*, AIAA Paper 2013-2541, San Diego, CA, 2013.

[19] F. Haider, J.-P. Croisille, and B. Courbet. Stability analysis of the cell centered finite-volume MUSCL method on unstructured grids. *Numerische Mathematik*, 113(4):555–600, 2009.

[20] A. Schwöppe and B. Diskin. Accuracy of the cell-centered grid metric in the DLR TAU-code. In A. Dillmann, G. Heller, H. P. Kreplin, W. Nitsche W., and I. Peltzer, editors, *New Results in Numerical and Experimental Fluid Mechanics VIII. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Volume 121*, pages 429–437. Springer, 2013.

[21] Hong Luo, Yidong Xia, Seth Spiegel, Robert Nourgaliev, and Zonglin Jiang. A reconstructed discontinuous Galerkin method based on a hierarchical WENO reconstruction for compressible flows on tetrahedral grids. *J. Comput. Phys.*, 236(1):477 – 492, 2013.

[22] Jialin Lou, Xiaodong Liu, Hong Luo, and Hiroaki Nishikawa. Reconstructed discontinuous Galerkin methods for hyperbolic diffusion equations on unstructured grids. In *55th AIAA Aerospace Sciences Meeting*, AIAA Paper 2017-0310, Grapevine, Texas, 2017.

[23] Q. Wang, Y.-X. Ren, J. Pan, and W. Li. Compact high order finite volume method on unstructured grids III: Variational reconstruction. *J. Comput. Phys.*, 337:1–26, 2017.

[24] Lingquan Li, Xiaodong Liu, Jialin Lou, Hong Luo, Hiroaki Nishikawa, and Yuxin Ren. A finite volume method based on variational reconstruction for compressible flows on arbitrary grids. In *23rd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2017-3097, Denver, Colorado, 2017.

[25] Lingquan Li, Xiaodong Liu, Jialin Lou, Hong Luo, Hiroaki Nishikawa, and Yuxin Ren. A discontinuous Galerkin method based on variational reconstruction for compressible flows on arbitrary grids. In *56th AIAA Aerospace Sciences Meeting*, AIAA Paper 2018-0831, Kissimmee, Florida, 2018.

[26] H. Nishikawa. A first-order system approach for diffusion equation. I: Second order residual distribution schemes. *J. Comput. Phys.*, 227:315–352, 2007.

[27] H. Nishikawa. New-generation hyperbolic Navier-Stokes schemes: $O(1/h)$ speed-up and accurate viscous/heat fluxes. In *Proc. of 20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3043, Honolulu, Hawaii, 2011.

[28] H. Nishikawa and Y. Nakashima. Dimensional scaling and numerical similarity in hyperbolic method for diffusion. *J. Comput. Phys.*, 355:121–143, 2018.

[29] Jialin Lou, Lingquan Li, Hong Luo, and Hiroaki Nishikawa. First-order hyperbolic system based reconstructed discontinuous Galerkin methods for nonlinear diffusion equations on unstructured grids. In *56th AIAA Aerospace Sciences Meeting*, AIAA Paper 2018-2094, Kissimmee, Florida, 2018.

[30] H. Nishikawa. On hyperbolic method for diffusion with discontinuous coefficients. *J. Comput. Phys.*, 367:102–108, 2018.

[31] H. Nishikawa. First-, second-, and third-order finite-volume schemes for diffusion. *J. Comput. Phys.*, 256:791–805, 2014.

[32] E. Lee, H. T. Ahn, and H. Luo. Cell-centered high-order hyperbolic finite volume method for diffusion equation on unstructured grids. *J. Comput. Phys.*, 355:464–491, 2018.

[33] Yi Liu and Hiroaki Nishikawa. Third-order inviscid and second-order hyperbolic Navier-Stokes solvers for three-dimensional inviscid and viscous flows. In *46th AIAA Fluid Dynamics Conference*, AIAA Paper 2016-3969, Washington, D.C., 2016.

[34] Hubert Baty and Hiroaki Nishikawa. Hyperbolic method for magnetic reconnection process in steady state magnetohydrodynamics. *Mon. Not. R. Astron. Soc.*, 459:624–637, 2016.

[35] Y. Nakashima, N. Watanabe, and H. Nishikawa. Hyperbolic Navier-Stokes solver for three-dimensional flows. In *54th AIAA Aerospace Sciences Meeting*, AIAA Paper 2016-1101, San Diego, CA, 2016.

[36] H. Nishikawa and P. L. Roe. Third-order active-flux scheme for advection diffusion: Hyperbolic diffusion, boundary condition, and Newton solver. *Computers and Fluids*, 125:71–81, 2016.

[37] H. Nishikawa. Beyond interface gradient: A general principle for constructing diffusion schemes. In *Proc. of 40th AIAA Fluid Dynamics Conference and Exhibit*, AIAA Paper 2010-5093, Chicago, 2010.

[38] H. Nishikawa. Robust and accurate viscous discretization via upwind scheme - I: Basic principle. *Comput. Fluids*, 49(1):62–86, October 2011.

[39] H. Nishikawa. Two ways to extend diffusion schemes to Navier-Stokes schemes: Gradient formula or upwinding. In *20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3044, Honolulu, Hawaii, 2011.

[40] H. Nishikawa, Y. Nakashima, and N. Watanabe. Effects of high-frequency damping on iterative convergence of implicit viscous solver. *J. Comput. Phys.*, 348:66–81, 2017.

[41] Y. Nakashima, N. Watanabe, and H. Nishikawa. Development of an effective implicit solver for general-purpose unstructured CFD software. In *The 28th Computational Fluid Dynamics Symposium*, C08-1, Tokyo, Japan, 2014.

[42] Alireza Jalali and Carl Ollivier-Gooch. Higher-order unstructured finite volume RANS solution of turbulent compressible flows. *Comput. Fluids*, 143:32–47, 2017.

[43] Jeffery A. White, Robert Baurle, Bradley J. Passe, Seth C. Spiegel, and Hiroaki Nishikawa. Geometrically flexible and efficient flow analysis of high speed vehicles via domain decomposition, part 1, unstructured-grid solver for high speed flows. In *JANNAF 48th Combustion 36th Airbreathing Propulsion, 36th Exhaust Plume and Signatures, 30th Propulsion Systems Hazards, Joint Subcommittee Meeting, Programmatic and Industrial Base Meeting*, Newport News, VA, 2017.

[44] H. Nishikawa. First, second, and third order finite-volume schemes for Navier-Stokes equations. In *Proc. of 7th AIAA Theoretical Fluid Mechanics Conference, AIAA Aviation and Aeronautics Forum and Exposition 2014*, AIAA Paper 2014-2091, Atlanta, GA, 2014.

[45] H. Nishikawa. Alternative formulations for first-, second-, and third-order hyperbolic Navier-Stokes schemes. In *Proc. of 22nd AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2015-2451, Dallas, TX, 2015.

[46] Richard S. Hirsh. Higher order accurate difference solutions of fluid mechanics problems by a compact differencing technique. *J. Comput. Phys.*, 19:90–109, 1975.

[47] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.*, 103:16–42, 1993.

[48] Jan Nordstrom, Sofia Eriksson, and Peter Eliasson. Weak and strong wall boundary procedures and convergence to steady-state of the Navier-Stokes equations. *J. Comput. Phys.*, 231:4867–4884, 2012.

[49] A. Haselbacher. On constrained reconstruction operators. In *Proc. of 44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 2006-1274, Reno, Nevada, 2006.

[50] H. Nishikawa and P. L. Roe. On high-order fluctuation-splitting schemes for Navier-Stokes equations. In C. Groth and D. W. Zingg, editors, *Computational Fluid Dynamics 2004*, pages 799–804. Springer-Verlag, 2004.

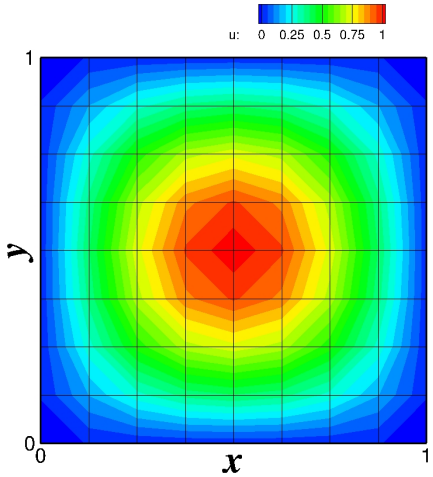
[51] J. C. Mandal and J. Subramanian. On the link between weighted least-squares and limiters used in higher-order reconstructions for finite volume computations of hyperbolic equations. *Appl. Numer. Math.*, 58:705–725, 2008.

[52] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sb. (N.S.)*, 47(89)(3):271–306, 1959.

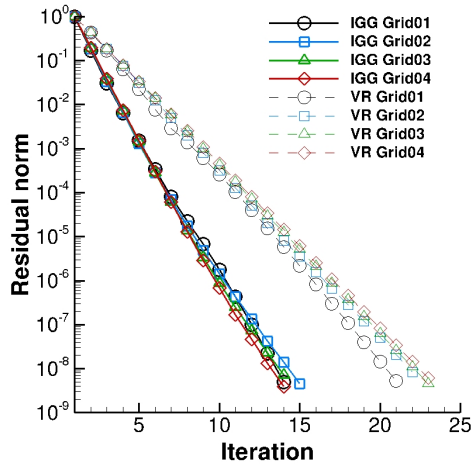
[53] V. Venkatakrishnan. Convergence to steady state solutions of the euler equations on unstructured grids with limiters. *J. Comput. Phys.*, 118:120–130, 1995.

[54] A. Katz and V. Sankaran. Discretization methodology for high aspect ratio prismatic grids. In *Proc. of 20th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2011-3378, Hawaii, 2011.

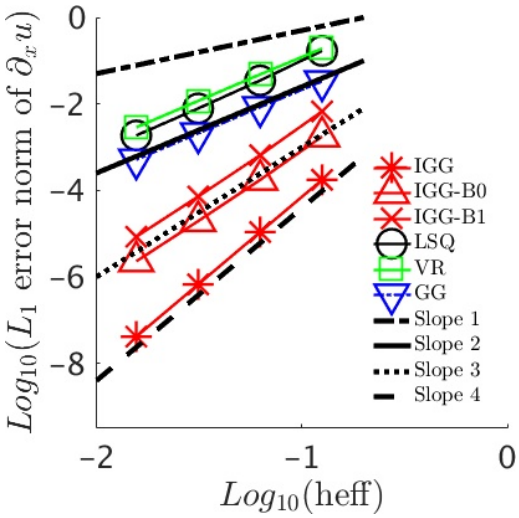
[55] Jialin Lou, Lingquan Li, Hong Luo, and Hiroaki Nishikawa. Reconstructed discontinuous Galerkin methods for linear advection-diffusion equations based on first-order hyperbolic system. *J. Comput. Phys.*, 369:103–124, 2018.



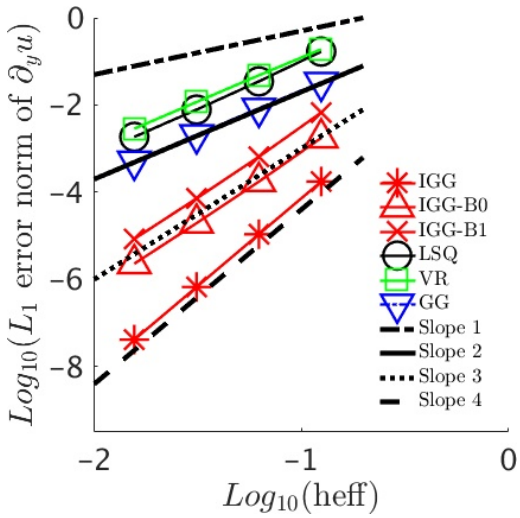
(a) Function contours and coarsest grid.



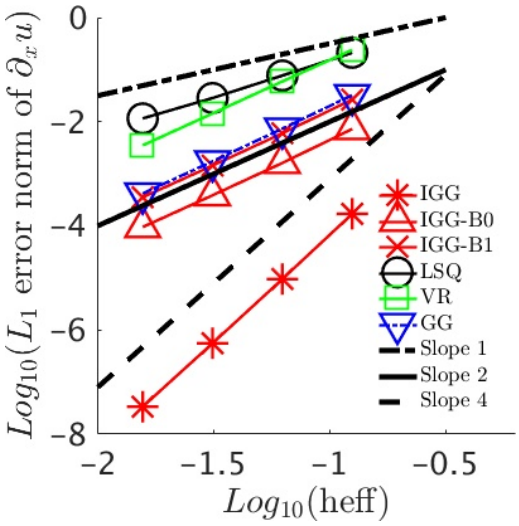
(b) Iterative convergence.



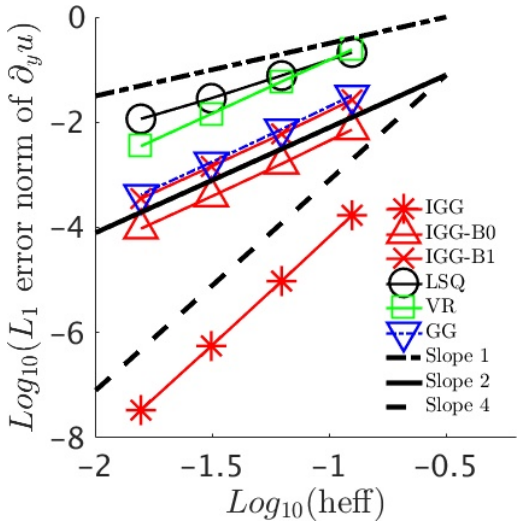
(c) Error convergence, $\partial_x u$, interior cells.



(d) Error convergence, $\partial_y u$, interior cells.

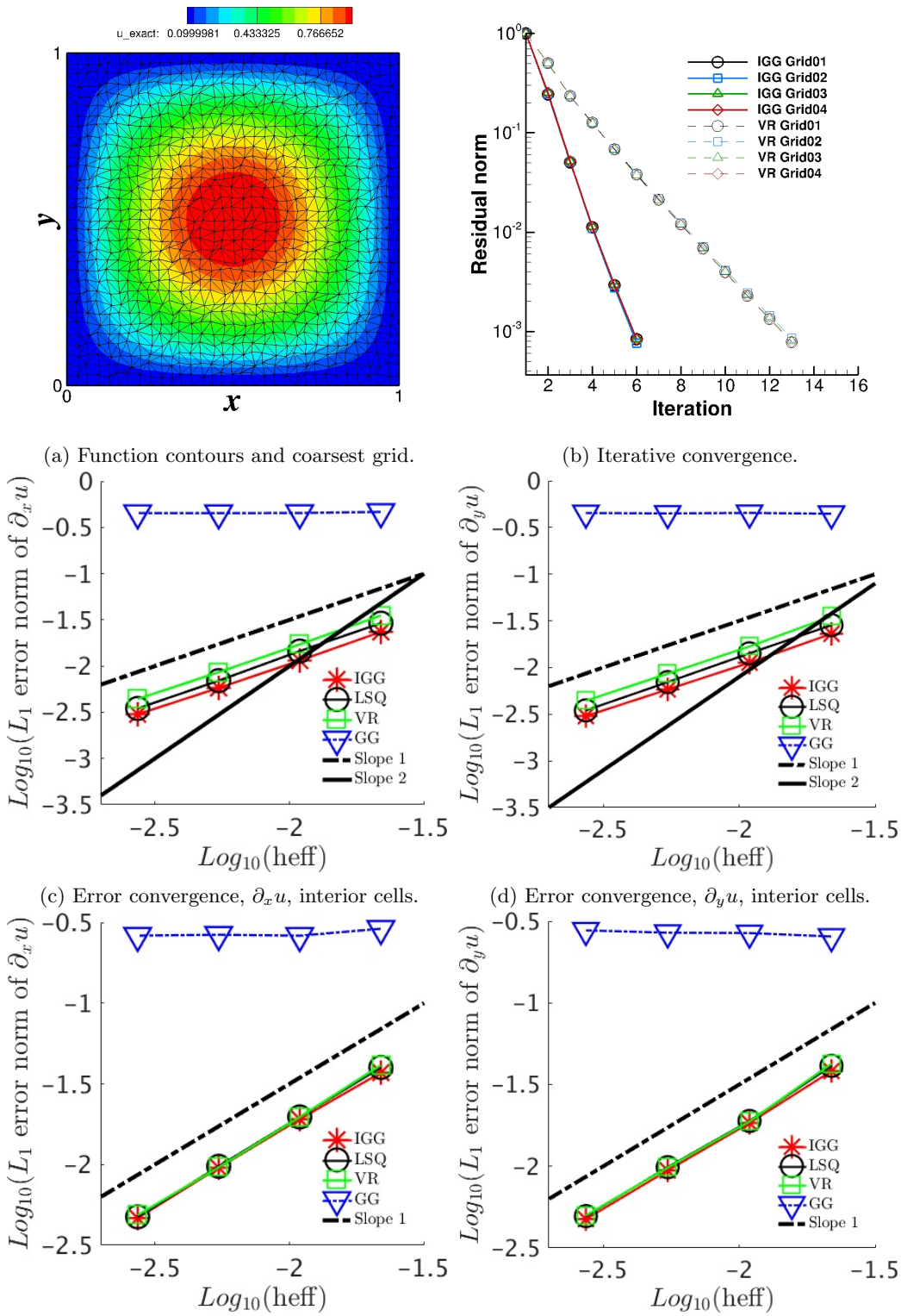


(e) Error convergence, $\partial_x u$, boundary cells.



(f) Error convergence, $\partial_y u$, boundary cells.

Figure 5: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(\pi y)$ on regular quadrilateral grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function contours and coarsest grid.

(b) Iterative convergence.

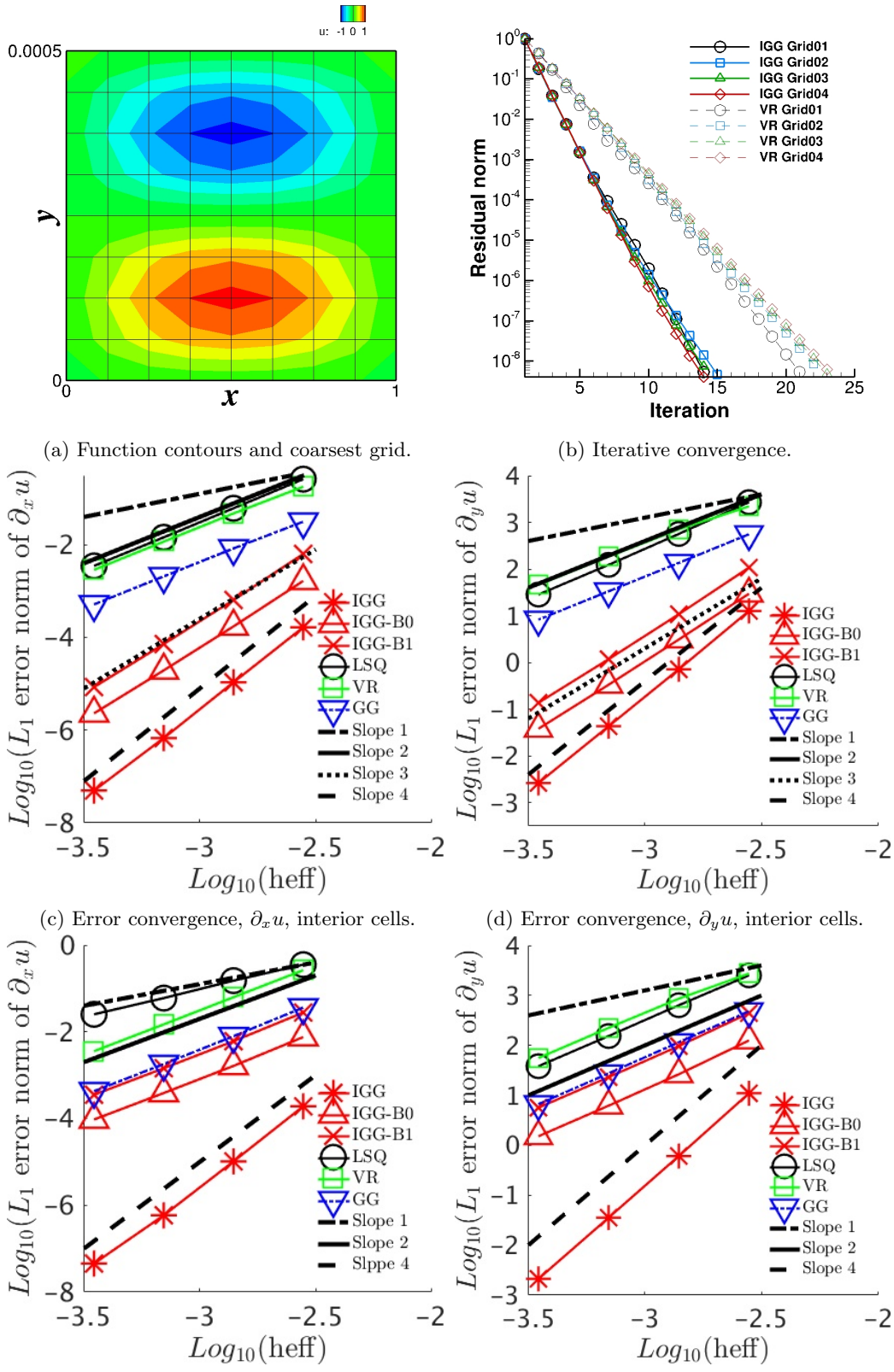
(c) Error convergence, $\partial_x u$, interior cells.

(d) Error convergence, $\partial_y u$, interior cells.

(e) Error convergence, $\partial_x u$, boundary cells.

(f) Error convergence, $\partial_y u$, boundary cells.

Figure 6: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(\pi y)$ on irregular isotropic triangular grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function contours and coarsest grid.

(b) Iterative convergence.

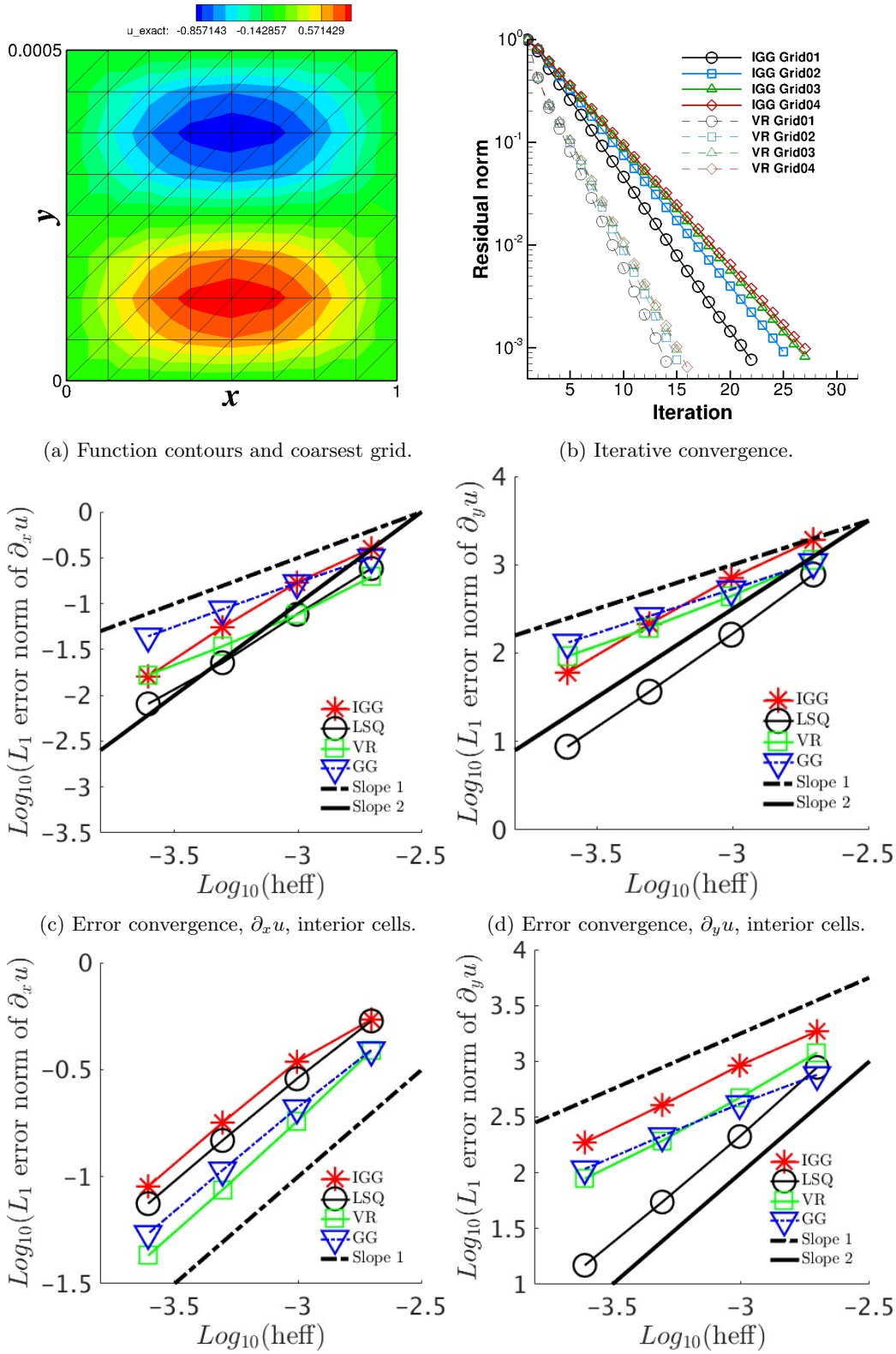
(c) Error convergence, $\partial_x u$, interior cells.

(d) Error convergence, $\partial_y u$, interior cells.

(e) Error convergence, $\partial_x u$, boundary cells.

(f) Error convergence, $\partial_y u$, boundary cells.

Figure 7: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(4000\pi y)$ on high-aspect-ratio regular quadrilateral grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function contours and coarsest grid.

(b) Iterative convergence.

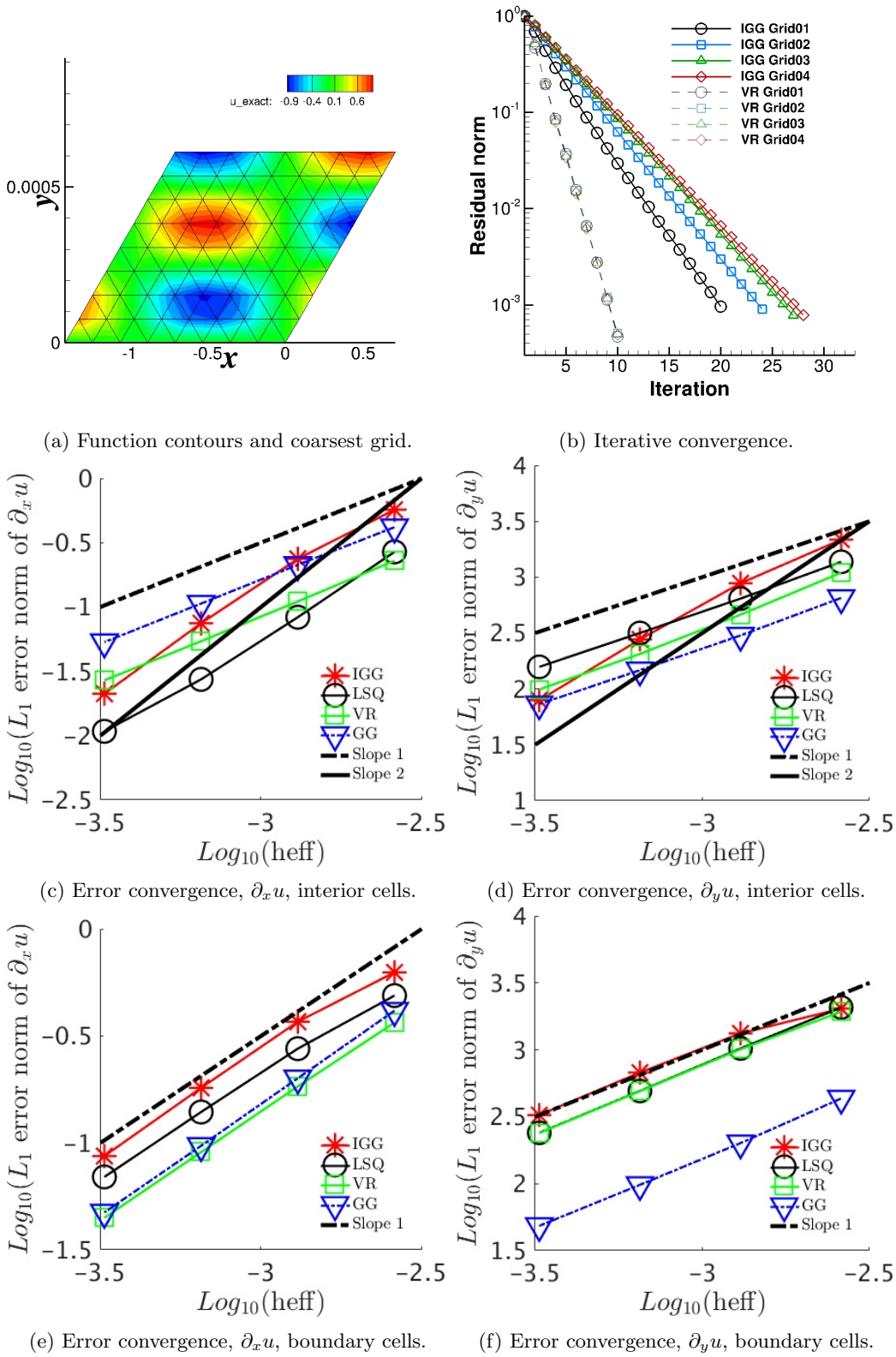
(c) Error convergence, $\partial_x u$, interior cells.

(d) Error convergence, $\partial_y u$, interior cells.

(e) Error convergence, $\partial_x u$, boundary cells.

(f) Error convergence, $\partial_y u$, boundary cells.

Figure 8: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(4000\pi y)$ on high-aspect-ratio regular triangular grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function contours and coarsest grid.

(b) Iterative convergence.

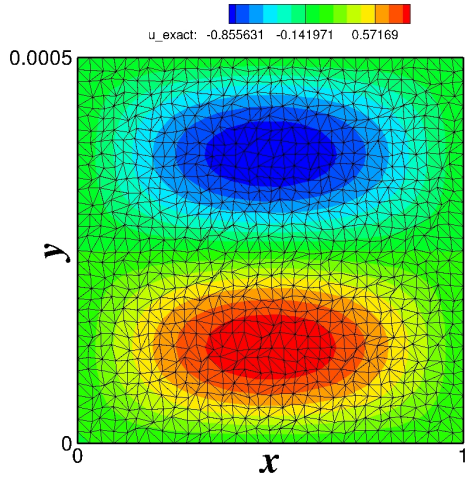
(c) Error convergence, $\partial_x u$, interior cells.

(d) Error convergence, $\partial_y u$, interior cells.

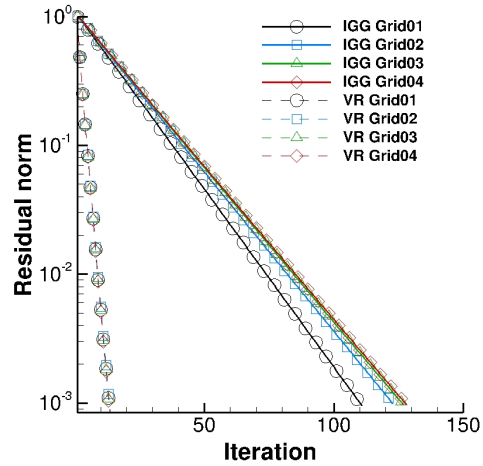
(e) Error convergence, $\partial_x u$, boundary cells.

(f) Error convergence, $\partial_y u$, boundary cells.

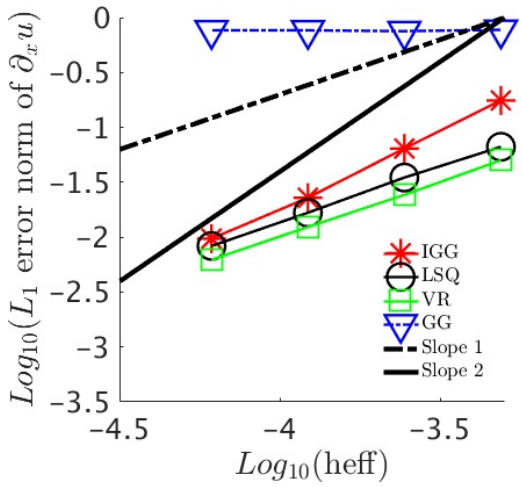
Figure 9: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(4000\pi y)$ on high-aspect-ratio isosceles triangular grids. IGG denotes the IGG method with the boundary closure B2.



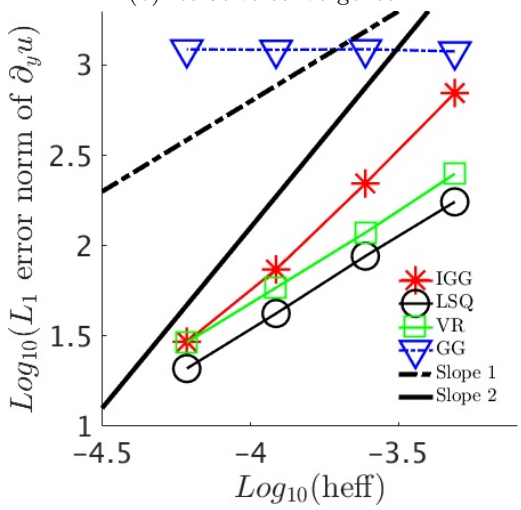
(a) Function contours and coarsest grid.



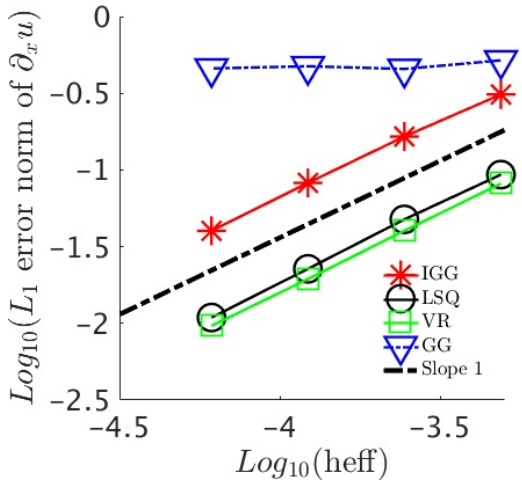
(b) Iterative convergence.



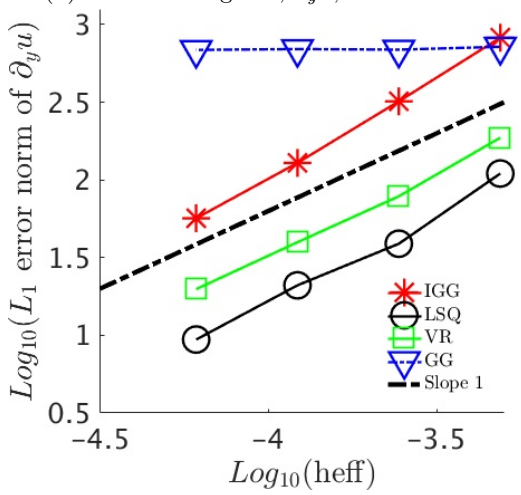
(c) Error convergence, $\partial_x u$, interior cells.



(d) Error convergence, $\partial_y u$, interior cells.

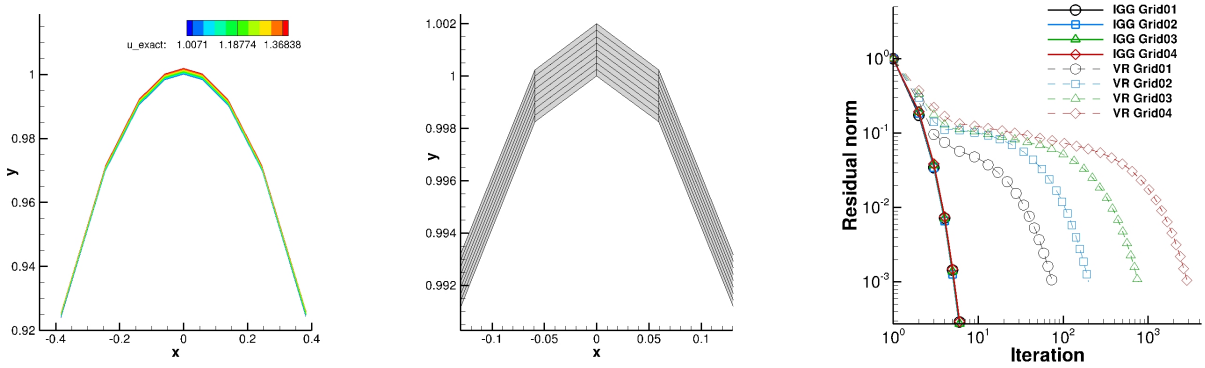


(e) Error convergence, $\partial_x u$, boundary cells.

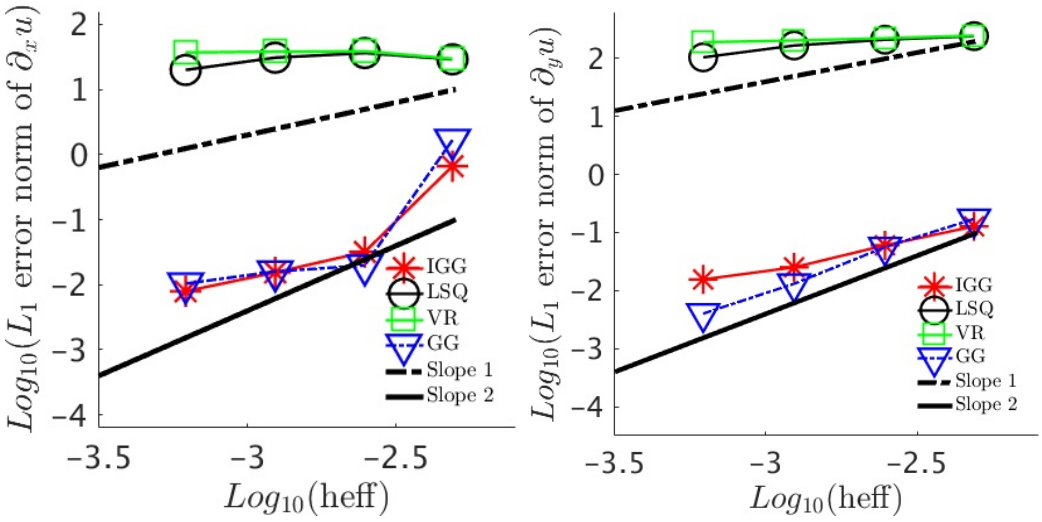


(f) Error convergence, $\partial_y u$, boundary cells.

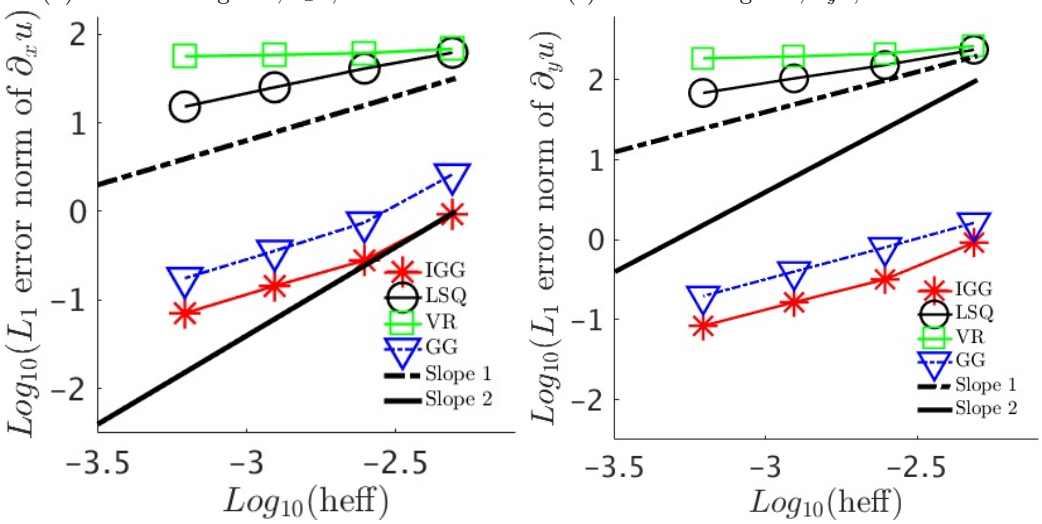
Figure 10: Gradients calculated by various methods for the function $u(x, y) = \sin(\pi x)\sin(4000\pi y)$ on high-aspect-ratio irregular triangular grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function and coarsest grid. (b) Close view of the grid. (c) Iterative convergence.

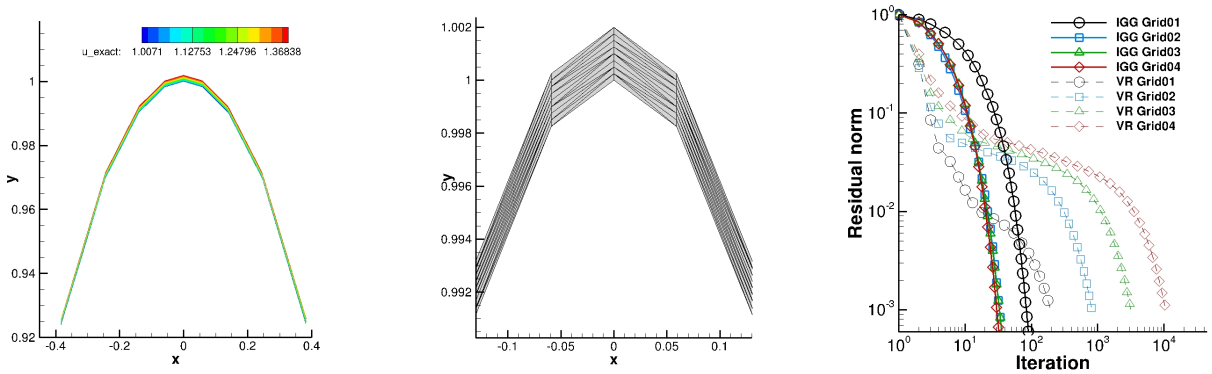


(d) Error convergence, $\partial_x u$, interior cells. (e) Error convergence, $\partial_y u$, interior cells.

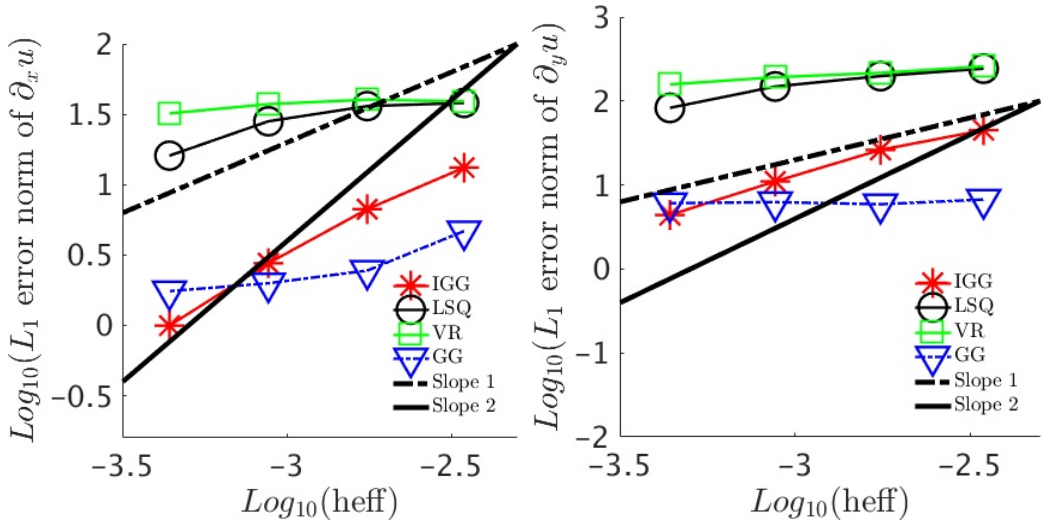


(f) Error convergence, $\partial_x u$, boundary cells. (g) Error convergence, $\partial_y u$, boundary cells.

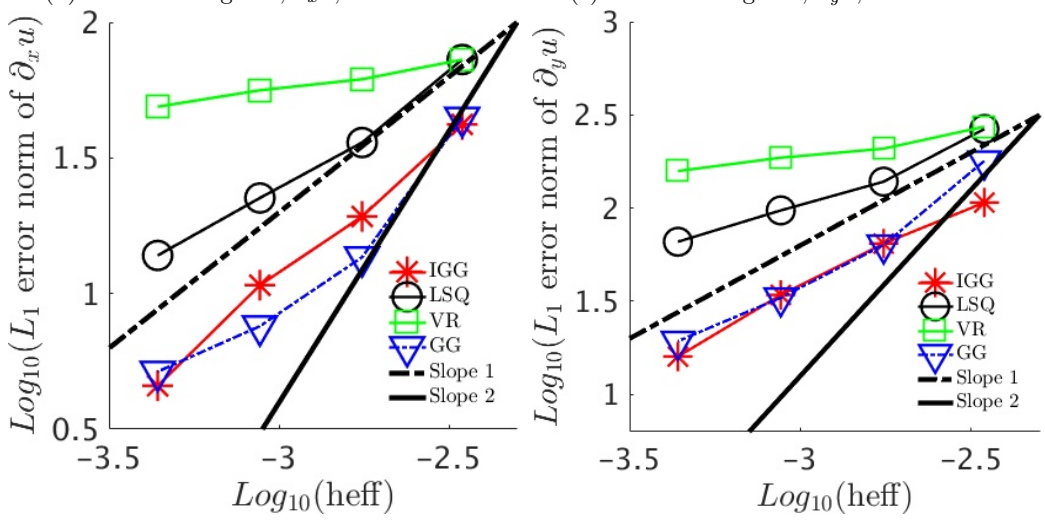
Figure 11: Gradients calculated by various methods for the function $u(x, y) = \sin(100\pi r + \pi/6) + 0.5\sin(\theta)$ on high-aspect-ratio curved regular quadrilateral grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function and coarsest grid. (b) Close view of the grid. (c) Iterative convergence.

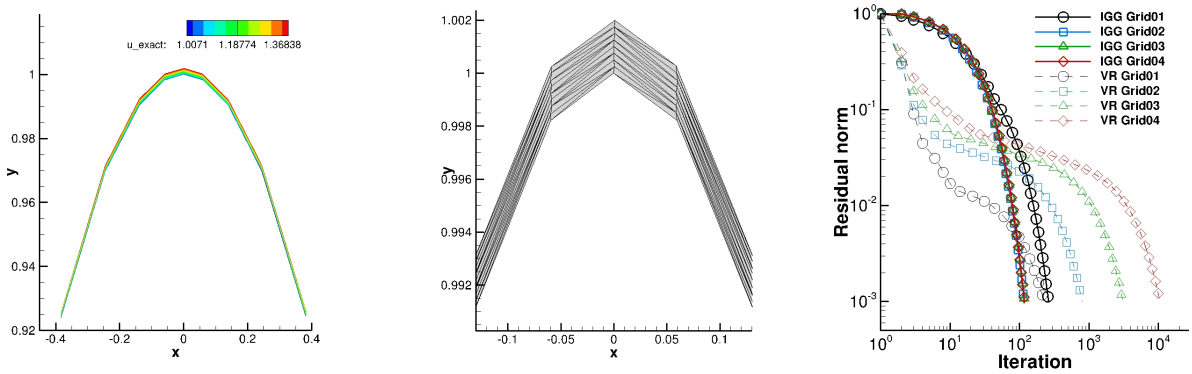


(d) Error convergence, $\partial_x u$, interior cells. (e) Error convergence, $\partial_y u$, interior cells.

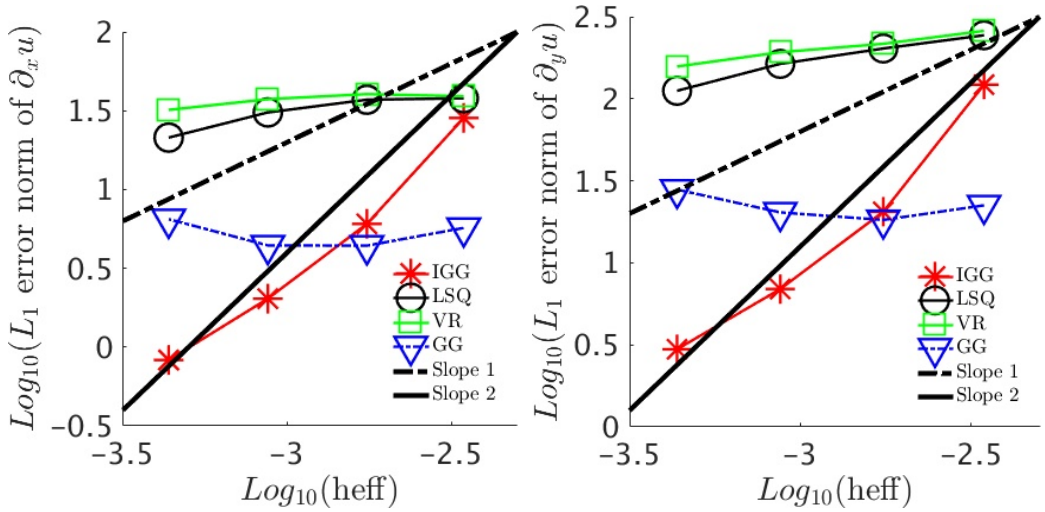


(f) Error convergence, $\partial_x u$, boundary cells. (g) Error convergence, $\partial_y u$, boundary cells.

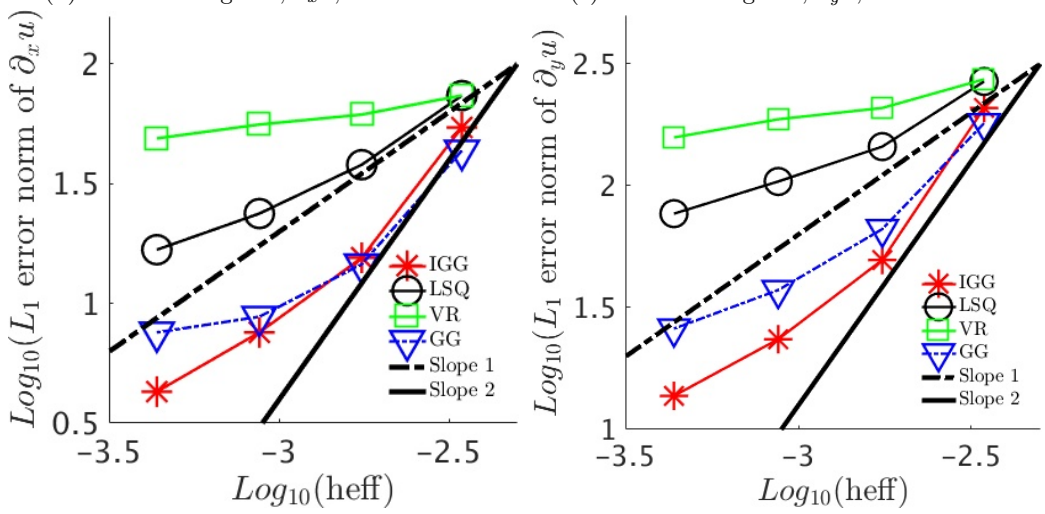
Figure 12: Gradients calculated by various methods for the function $u(x, y) = \sin(100\pi r + \pi/6) + 0.5\sin(\theta)$ on high-aspect-ratio curved regular triangular grids. IGG denotes the IGG method with the boundary closure B2.



(a) Function and coarsest grid. (b) Close view of the grid. (c) Iterative convergence.

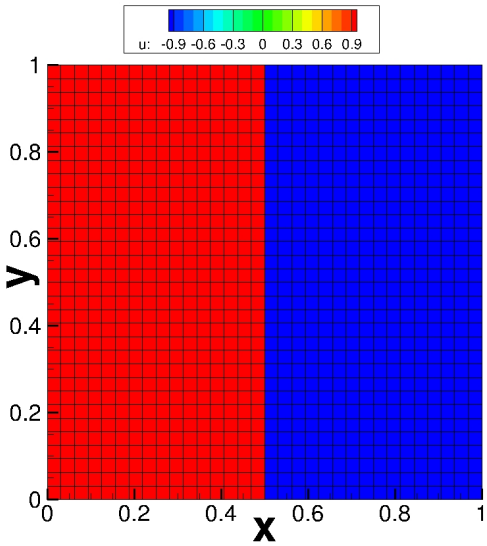


(d) Error convergence, $\partial_x u$, interior cells. (e) Error convergence, $\partial_y u$, interior cells.

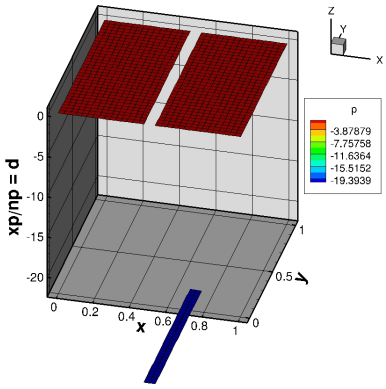


(f) Error convergence, $\partial_x u$, boundary cells. (g) Error convergence, $\partial_y u$, boundary cells.

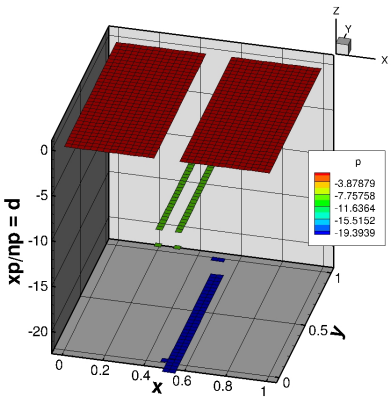
Figure 13: Gradients calculated by various methods for the function $u(x, y) = \sin(100\pi r + \pi/6) + 0.5\sin(\theta)$ on high-aspect-ratio curved irregular triangular grids. IGG denotes the IGG method with the boundary closure B2.



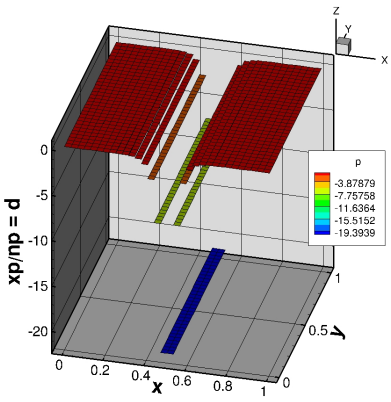
(a) Discontinuous function.



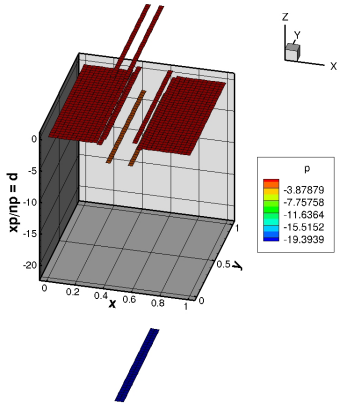
(b) GG



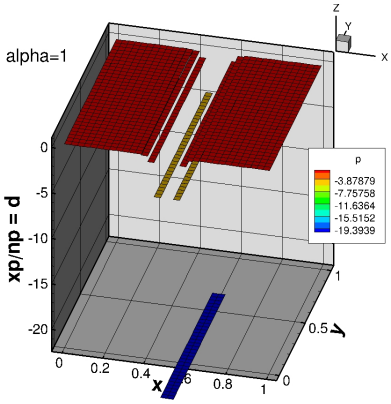
(c) LSQ



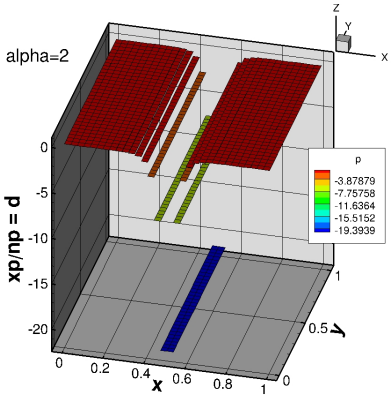
(d) VR



(e) IGG with $\alpha = 1/6$.

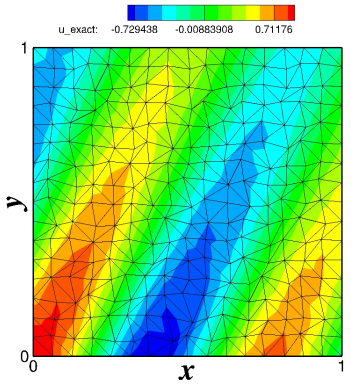


(f) IGG with $\alpha = 1$.

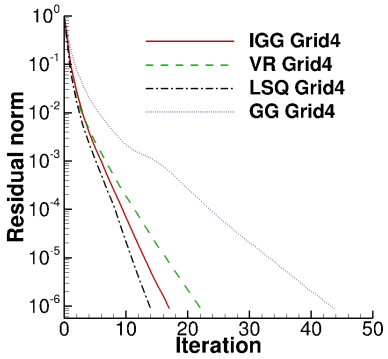


(g) IGG with $\alpha = 2$.

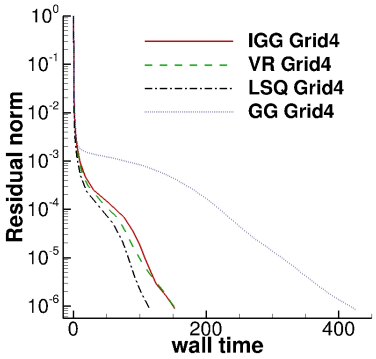
Figure 14: Gradients calculated by GG, LSQ, VR, and IGG methods for a discontinuous function.



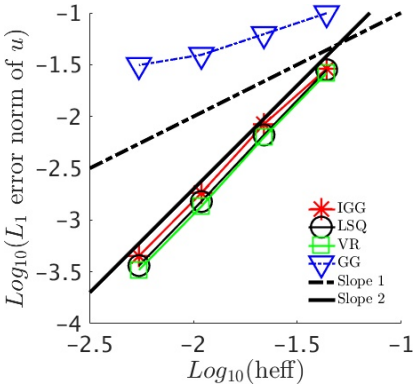
(a) Exact solution contours and coarsest grid.



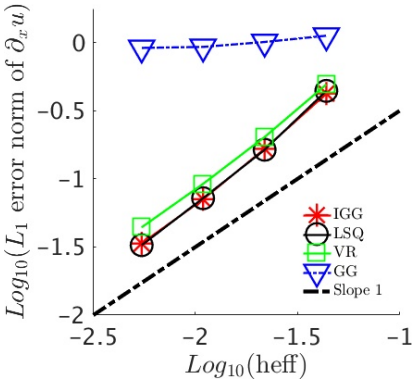
(b) Iterative convergence for Grid 4.



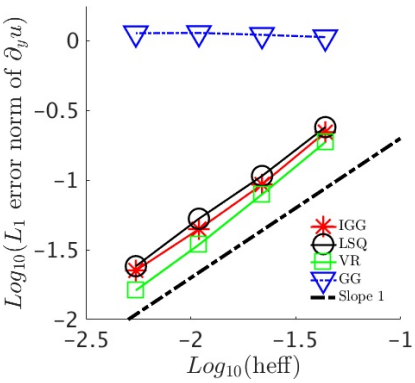
(c) Residual convergence in CPU time.



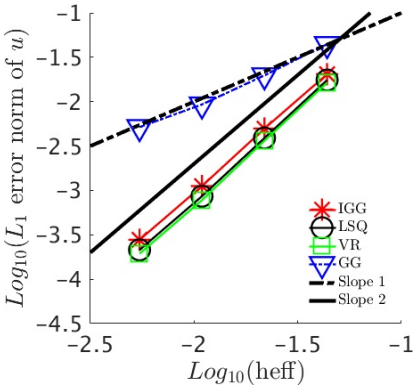
(d) Error in u, interior cells.



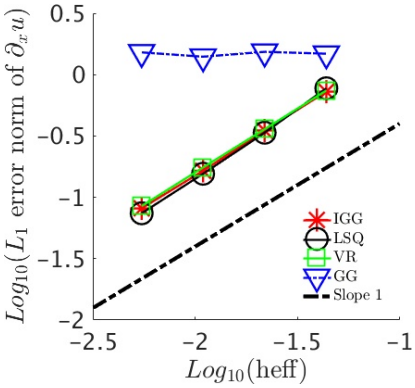
(e) Error in $\partial_x u$, interior cells.



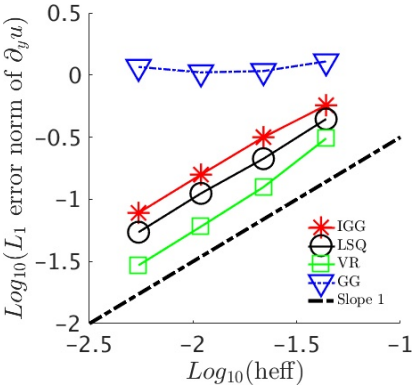
(f) Error in $\partial_y u$, interior cells.



(g) Error in u, boundary cells.

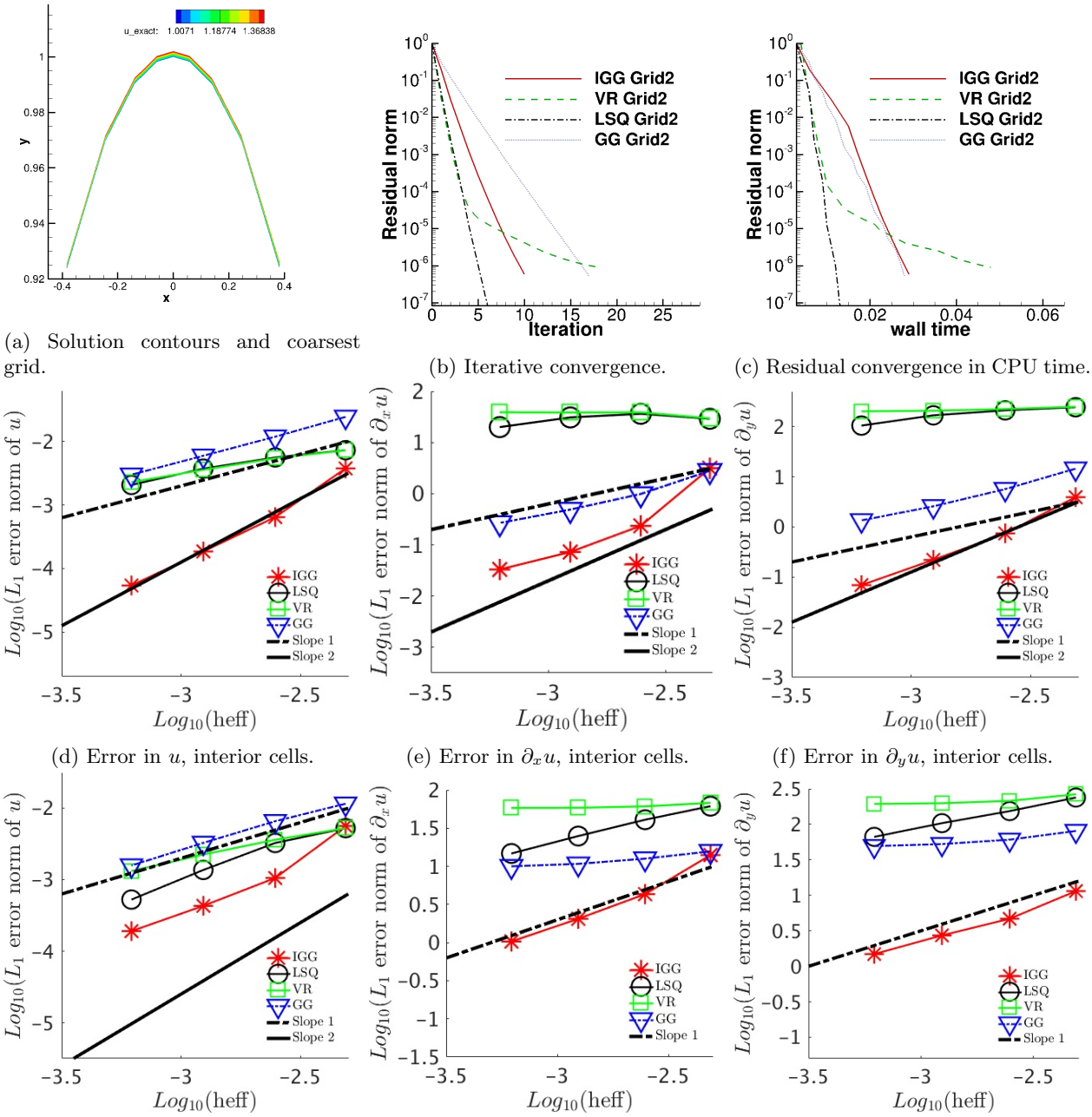


(h) Error in $\partial_x u$, boundary cells.



(i) Error in $\partial_y u$, boundary cells.

Figure 15: Finite-volume solutions and iterative convergence with various gradient methods on irregular isotropic triangular grids. IGG denotes the IGG method with the boundary closure B1 (no boundary values are used).



(a) Solution contours and coarsest grid.

(b) Iterative convergence.

(c) Residual convergence in CPU time.

(d) Error in u, interior cells.

(e) Error in $\partial_x u$, interior cells.

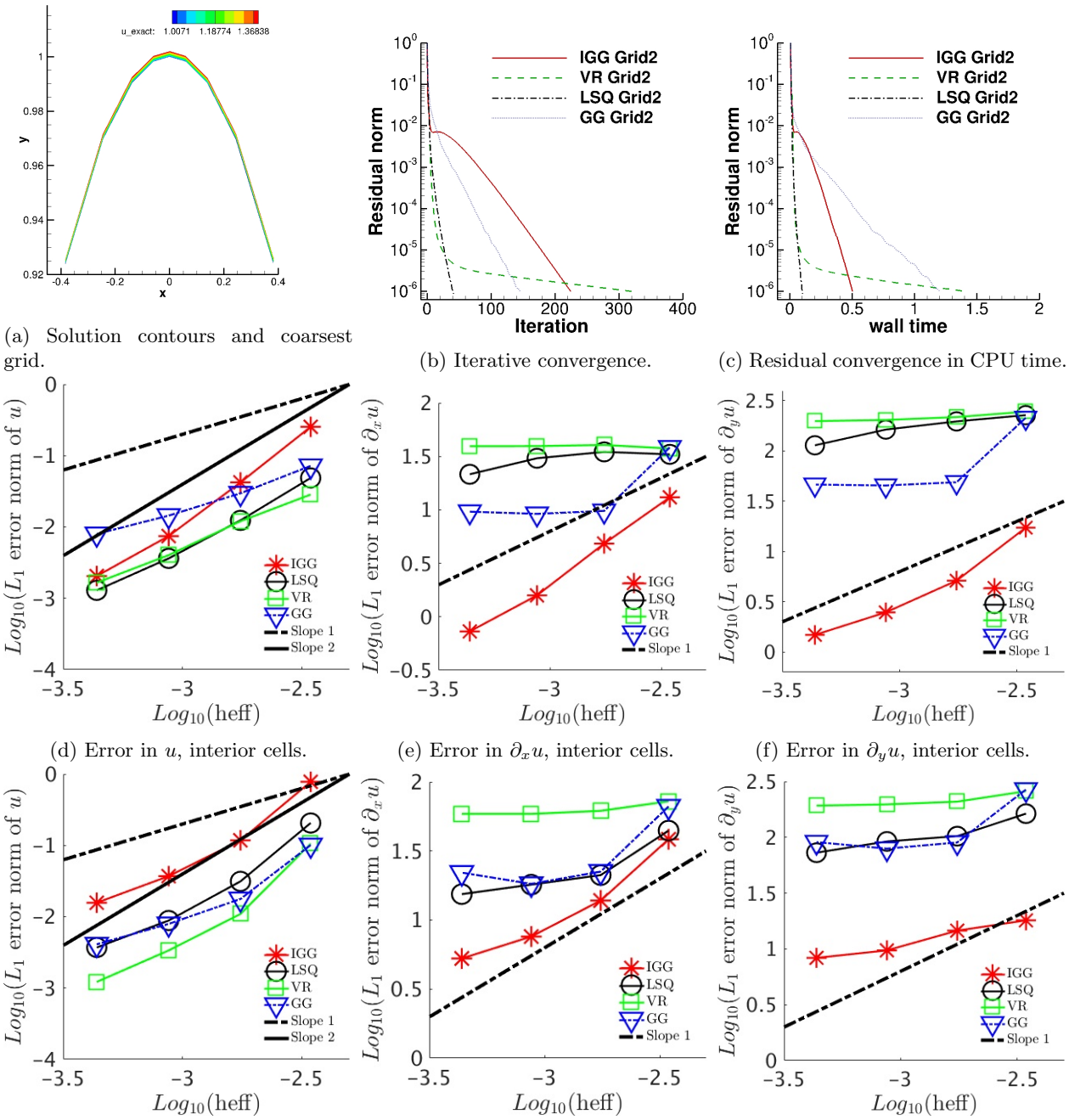
(f) Error in $\partial_y u$, interior cells.

(g) Error in u, boundary cells.

(h) Error in $\partial_x u$, boundary cells.

(i) Error in $\partial_y u$, boundary cells.

Figure 16: Finite-volume solutions and iterative convergence with various gradient methods on curved quadrilateral grids. IGG denotes the IGG method with the boundary closure B1 (no boundary values are used).



(a) Solution contours and coarsest grid.

(b) Iterative convergence.

(c) Residual convergence in CPU time.

(d) Error in u, interior cells.

(e) Error in $\partial_x u$, interior cells.

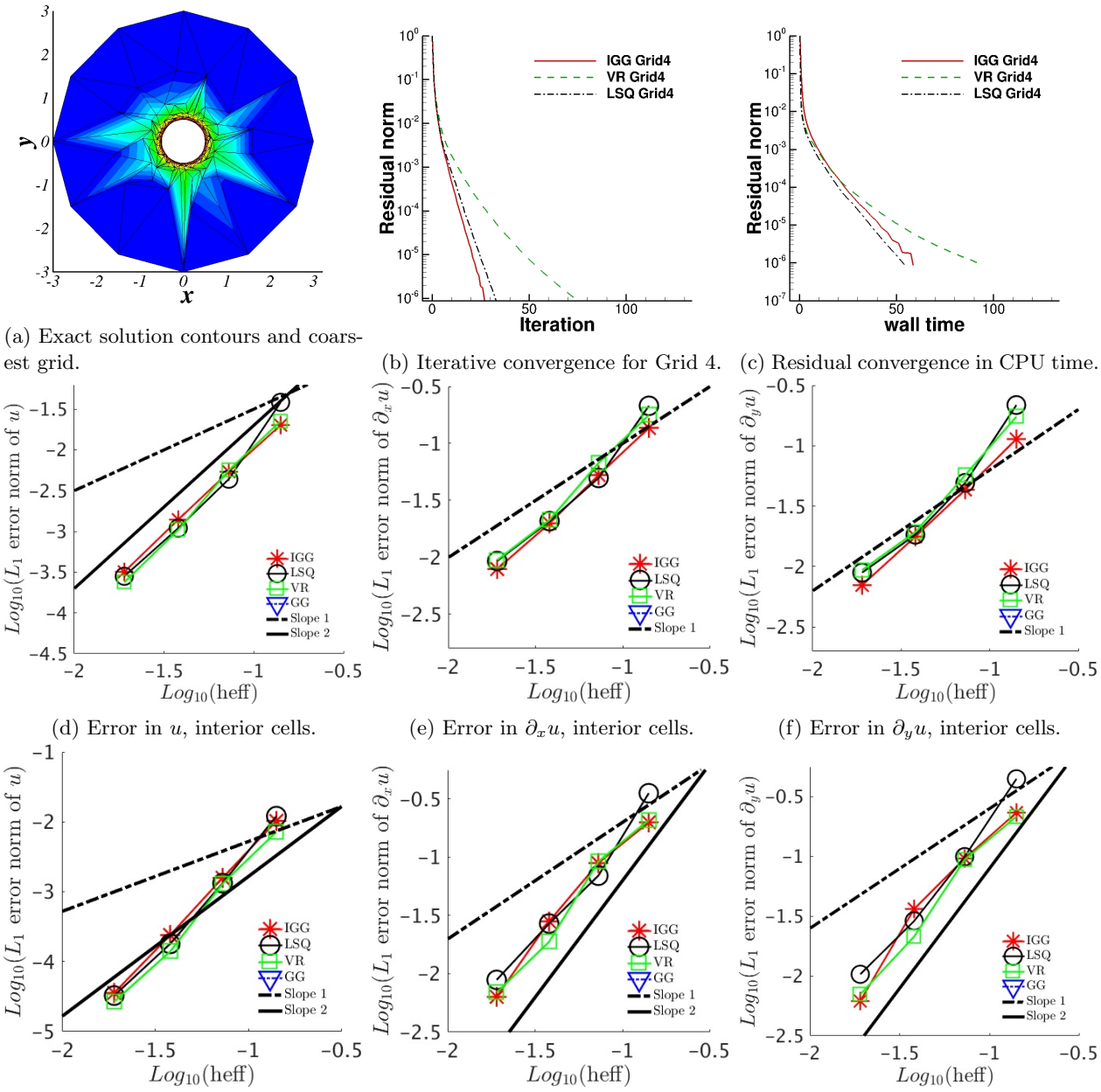
(f) Error in $\partial_y u$, interior cells.

(g) Error in u, boundary cells.

(h) Error in $\partial_x u$, boundary cells.

(i) Error in $\partial_y u$, boundary cells.

Figure 17: Finite-volume solutions and iterative convergence with various gradient methods on curved triangular grids. IGG denotes the IGG method with the boundary closure B1 (no boundary values are used).



(a) Exact solution contours and coarsest grid.

(b) Iterative convergence for Grid 4.

(c) Residual convergence in CPU time.

(d) Error in u, interior cells.

(e) Error in $\partial_x u$, interior cells.

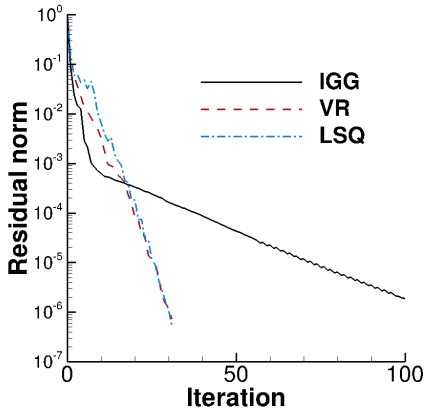
(f) Error in $\partial_y u$, interior cells.

(g) Error in u, boundary cells.

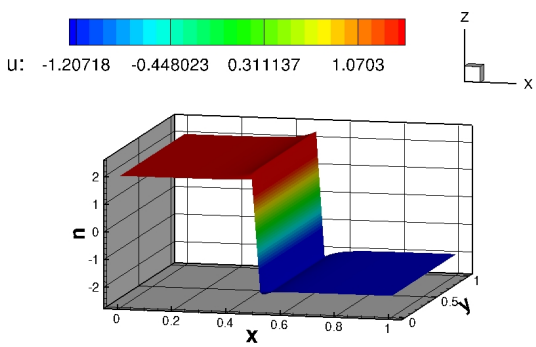
(h) Error in $\partial_x u$, boundary cells.

(i) Error in $\partial_y u$, boundary cells.

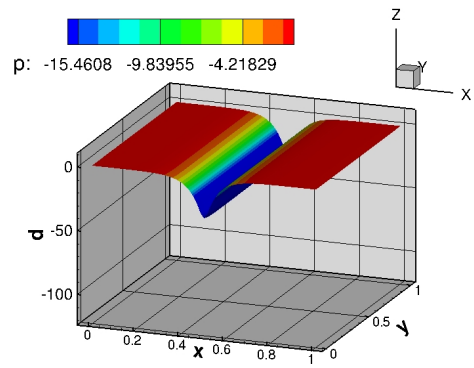
Figure 18: Finite-volume solutions and iterative convergence with various gradient methods on irregular triangular grids in an annular domain. IGG denotes the IGG method with the boundary closure B1 (no boundary values are used).



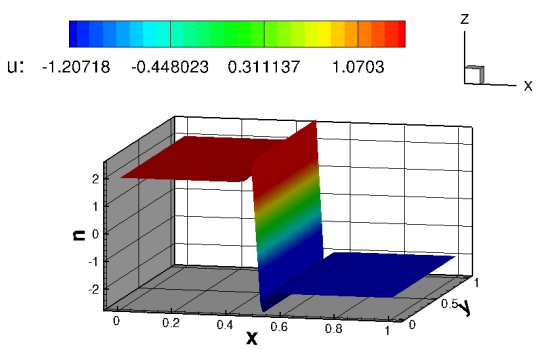
(a) Convergence history.



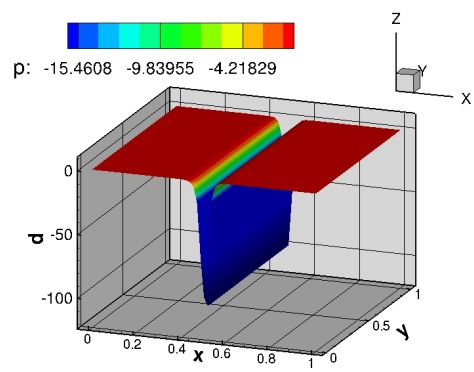
(b) IGG: Solution u.



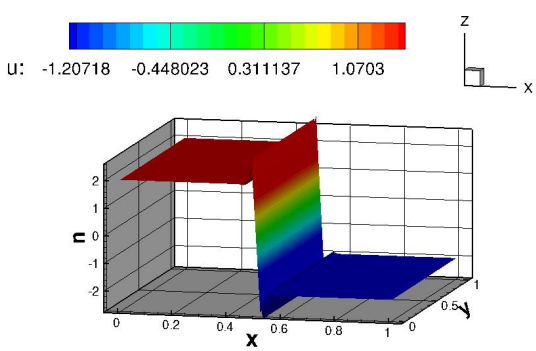
(c) IGG: Gradient $\partial_x u$.



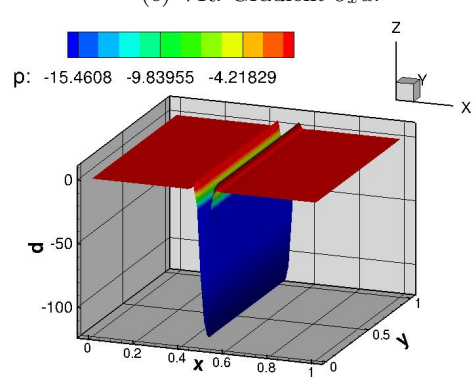
(d) VR: Solution u.



(e) VR: Gradient $\partial_x u$.

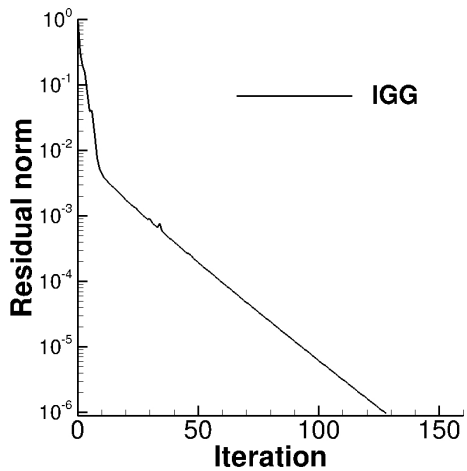


(f) LSQ: Solution u.

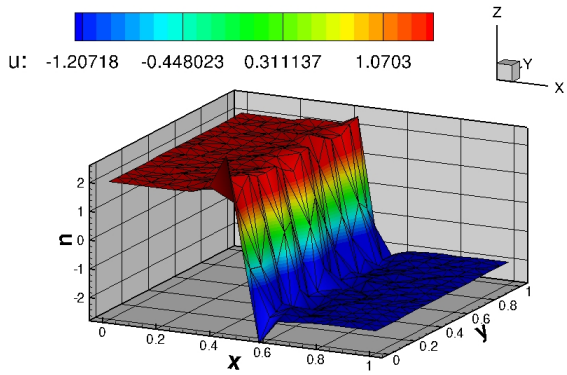


(g) LSQ: Gradient $\partial_x u$.

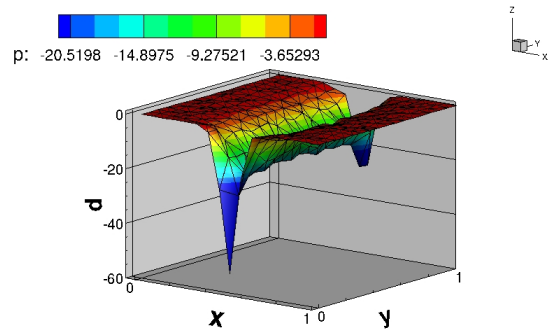
Figure 19: Finite-volume solutions and iterative convergence with various gradient methods on a regular grid for a discontinuous solution of Burgers' equation. The IGG method uses the boundary closure B1 (no boundary values are used).



(a) Convergence history.



(b) IGG: Solution u.



(c) IGG: Gradient $\partial_x u$.

Figure 20: Finite-volume solution and iterative convergence with the IGG method on an unstructured triangular grid for a discontinuous solution of Burgers' equation. The IGG method uses the boundary closure B1 (no boundary values are used).

Appendix A: Gradient Accuracy on Regular and Irregular Grids

Consider a set of function values $\{u_j\}$ given at points in a one-dimensional grid. Let $\mathcal{G}_j$ denote a gradient algorithm applied at a point $x = x_j$ that is exact for linear functions. Then, if $\{u_j\}$ is given by a linear function $u(x) = a + bx$, where a and b are constants, we have

$$\mathcal{G}_j(\{u_j\}) = b. \tag{A.1}$$

For a quadratic function $u(x) = a + bx + cx^2$, where c is another constant, we have

$$\mathcal{G}_j(\{u_j\}) = b + \mathcal{E}, \tag{A.2}$$

where $\mathcal{E}$ is an error term proportional to c. If the gradient algorithm is exact for quadratic functions, it will produce $\mathcal{E} = 2cx_j$ and the gradient is obtained exactly. However, linearly-exact gradient algorithms cannot produce $2cx_j$, and therefore the gradient is expected to have a first-order error:

$$\mathcal{G}_j(\{u_j\}) = b + O(h), \tag{A.3}$$

where h is a representative mesh spacing. Consider, for example, a central-difference-type formula on a non-uniform grid:

$$\frac{u_{j+1} - u_{j-1}}{h_L + h_R}, \tag{A.4}$$

where $x_{j+1} - x_j = h_R$ and $x_j - x_{j-1} = h_L$. This formula is clearly exact for linear functions. Without loss of generality, let us take $x_j = 0$, so that the exact gradient is b for the quadratic function $u(x) = a + bx + cx^2$ and also for higher-order functions. Applying the central-difference-type formula to the quadratic function, we obtain

$$\frac{u_{j+1} - u_{j-1}}{h_L + h_R} = b + c\frac{h_R^2 - h_L^2}{h_L + h_R}. \tag{A.5}$$

The second term is an $O(h)$ error. It shows that the central-difference-type formula can be exact for quadratic functions on uniform grids, $h_L = h_R = h$:

$$\frac{u_{j+1} - u_{j-1}}{2h} = b. \tag{A.6}$$

This is the error cancellation property typically observed on uniform grids. For more general functions, the leading error term comes from a cubic term and is of $O(h^2)$. However, such an error cancellation does not always occur. Consider the following gradient formula:

$$\frac{h_L^2(u_{j+1} - u_j) - h_R^2(u_{j-1} - u_j)}{h_L h_R(h_R + h_L)}, \tag{A.7}$$

which is exact for quadratic functions on irregular grids: if applied to $u(x) = a + bx + cx^2$, it gives

$$\frac{h_L^2(u_{j+1} - u_j) - h_R^2(u_{j-1} - u_j)}{h_L h_R(h_R + h_L)} = b. \tag{A.8}$$

For a cubic function, $u(x) = a + bx + cx^2 + dx^3$, it gives

$$\frac{h_L^2(u_{j+1} - u_j) - h_R^2(u_{j-1} - u_j)}{h_L h_R(h_R + h_L)} = b + d\, h_L h_R. \tag{A.9}$$

The term $d\, h_L h_R$ is an error term. Clearly, this error term does not cancel on uniform grids, and therefore the formula is second-order accurate on both irregular and regular grids. In general, a gradient algorithm is p-th order accurate if it is exact for polynomials of degree p, and exhibits higher-order accuracy on uniform grids when an error cancellation occurs. The argument extends straightforwardly to higher dimensions and unstructured grids.

Appendix B: Variational Reconstruction for Linear Functions

Consider a linear polynomial defined over a cell j:

$$P_j(x,y) = u_j + u_{xj}(x - x_j) + u_{y_j}(y - y_j), \tag{B.1}$$

where u_j is a given cell-averaged solution value, (x_j, y_j) is the centroid coordinates of the cell, and u_{xj}, u_{y_j} are the coefficients we wish to determine. The polynomials are defined independently at cells, and therefore they do not match over the face that divides two adjacent cells, creating a jump. In the VR method [23], we determine u_{xj} and u_{y_j} by minimizing the following global functional:

$$\mathcal{I} = \sum_{f\in\{F\}} \frac{1}{2L_f} \int_f \Delta\mathbf{v}^T \Delta\mathbf{v}, \tag{B.2}$$

where $\{F\}$ denotes a set of faces on a given grid (excluding boundary faces), f is a face shared by the cells j and k, L_f is a length scale to be defined at the face, $\Delta\mathbf{v}$ is a weighted jump vector,

$$\Delta\mathbf{v} = \mathbf{W}\Delta\mathbf{u}, \quad \Delta\mathbf{u} = (\Delta u, \Delta u_x, \Delta u_y)^T = (u_R - u_L, u_{xk} - u_{xj}, u_{y_k} - u_{y_j})^T, \tag{B.3}$$

with u_L and u_R linearly interpolated to the face from cells j and k, respectively, and the digaonal matrix $\mathbf{W}$ serves to weight the components of $\Delta\mathbf{u}$:

$$\mathbf{W} = \mathrm{diag}\,(1,\, w_x L_x,\, w_y L_y)\,, \tag{B.4}$$

where L_x and L_y are length scales, and w_x and w_y are weights. If the integral over the face is evaluated by the midpoint rule, we obtain

$$\mathcal{I} = \frac{1}{2}\sum_{f\in\{F\}} \frac{\Delta\mathbf{v}^T\Delta\mathbf{v}A_f}{L_f} = \frac{1}{2}\sum_{f\in\{F\}} \Delta\mathbf{u}^T\mathbf{Q}\Delta\mathbf{u}, \quad \mathbf{Q} = \frac{\mathbf{W}^T\mathbf{W}A_f}{L_f}, \tag{B.5}$$

where A_f is the face area and the jumps are evaluated at the face midpoint. The polynomial coefficients $\mathbf{g}_j = (u_{xj}, u_{y_j})$ are determined by the stationary condition:

$$\frac{\partial\mathcal{I}}{\partial\mathbf{g}_j} = \sum_{f\in\{F_j\}} \left(\mathbf{Q}\frac{\partial\Delta\mathbf{u}}{\partial\mathbf{g}_j}\right)^T \Delta\mathbf{u} = 0, \tag{B.6}$$

where $\{F_j\}$ is a set of faces of the cell j. To write the equation in a practical form, we express the vector $\Delta\mathbf{u}$ as

$$\Delta\mathbf{u} = \mathbf{M}_{jm}\mathbf{g}_j - \mathbf{M}_{km}\mathbf{g}_k + (u_j - u_k)\boldsymbol{\delta}_1, \tag{B.7}$$

where $\boldsymbol{\delta}_1 = (1,0,0)^T$, and

$$\mathbf{M}_{jm} = \begin{bmatrix} \Delta x_{jm} & \Delta y_{jm} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_m - x_j & y_m - y_j \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{M}_{km} = \begin{bmatrix} \Delta x_{km} & \Delta y_{km} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_m - x_k & y_m - y_k \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{B.8}$$

After some algebra, we finally obtain from Equation (B.6)

$$\mathbf{A}_{jj}\mathbf{g}_j - \sum_{k\in\{k_j\}} \mathbf{A}_{jk}\mathbf{g}_k = \mathbf{b}_j, \tag{B.9}$$

where

$$\mathbf{A}_{jj} = \sum_{k\in\{k_j\}} \mathbf{M}_{jm}^T\mathbf{Q}\mathbf{M}_{jm} = \sum_{k\in\{k_j\}} \frac{A_f}{L_f} \begin{bmatrix} \Delta x_{jm}^2 + w_x^2 L_x^2 & \Delta x_{jm}\Delta y_{jm} \\ \Delta x_{jm}\Delta y_{jm} & \Delta y_{jm}^2 + w_y^2 L_y^2 \end{bmatrix}, \tag{B.10}$$

$$\mathbf{A}_{jk} = \mathbf{M}_{jm}^T\mathbf{Q}\mathbf{M}_{km} = \frac{A_f}{L_f} \begin{bmatrix} \Delta x_{jm}\Delta x_{km} + w_x^2 L_x^2 & \Delta x_{jm}\Delta y_{km} \\ \Delta x_{km}\Delta y_{jm} & \Delta y_{jm}\Delta y_{km} + w_y^2 L_y^2 \end{bmatrix}, \tag{B.11}$$

$$\mathbf{b}_j = \sum_{k\in\{k_j\}} \frac{A_f}{L_f} \begin{bmatrix} (u_k - u_j)\Delta x_{jm} \\ (u_k - u_j)\Delta y_{jm} \end{bmatrix}. \tag{B.12}$$

This shows that the right hand side is a LSQ-type gradient; more precisely it is in a similar form to the right hand side of a LSQ method. In this paper, we set $(L_x, L_y) = (\Delta x_{jk}, \Delta y_{jk})$, $L_f = A_f$, and $w_x = w_y = 1$. These choices are different from those in Ref.[23], but have been found to perform much better especially for high-aspect-ratio grids considered in this paper. Theoretically, it can be easily shown by the Taylor expansion that $w_x = w_y = \sqrt{3}/6$ gives fourth-order accuracy for regular quadrilateral grids in the interior cells. The resulting fourth-order method is equivalent to the classical fourth-order compact formula [46] as was shown for the IGG method in Equation (40).

Appendix C: Linear Least-Squares Method

LSQ methods are very popular and well described in the literature (e.g., see Refs[12, 16]). Here, we describe a linear LSQ method used in this paper. Consider fitting a linear polynomial over neighbor solution values around a cell j:

$$u_k = u_j + u_{xj}(x - x_j) + u_{y_j}(y - y_j), \quad k \in \{k_j\}, \tag{C.1}$$

where $\{k_j\}$ denotes a set of neighbors of the cell j (not necessarily face neighbors only), and u_k is a solution value at a cell k. The number of neighbors in $\{k_j\}$ is denoted by n_k. To determine u_{xj} and u_{y_j}, there must be at least two neighbors. As there are typically more than two neighbors, the set of equations (C.1) forms an overdetermined problem, which can be formulated as a set of weighted equations:

$$\mathbf{WBg}_j = \mathbf{Wz}, \tag{C.2}$$

where $\mathbf{g}_j = (u_{xj}, u_{y_j})$, $\mathbf{W} = diag(w_{j1}, w_{j2}, \cdots, w_{jn_k})$ is a diagonal weighting matrix, $\mathbf{B}$ is an $n_k \times 2$ matrix with the k-th row is given by $[x_k - x_j, y_k - y_j]$, and $\mathbf{z}$ is an n_k-dimensional vector of $(u_k - u_j)$. It can be solved in the least-squares sense by forming a normal equation:

$$\mathbf{Ag}_j = \mathbf{b}, \tag{C.3}$$

where

$$\mathbf{A} = \mathbf{B}^T\mathbf{W}^2\mathbf{B} = \sum_{k\in\{k_j\}} \begin{bmatrix} w_{jk}^2 \Delta x_{jk}^2 & w_{jk}^2 \Delta x_{jk} \Delta y_{jk} \\ w_{jk}^2 \Delta x_{jk} \Delta y_{jk} & w_{jk}^2 \Delta y_{jk}^2 \end{bmatrix}, \tag{C.4}$$

$$\mathbf{b} = \mathbf{B}^T\mathbf{W}^2\mathbf{z} = \sum_{k\in\{k_j\}} \begin{bmatrix} w_{jk}^2 (u_k - u_j) \Delta x_{jk} \\ w_{jk}^2 (u_k - u_j) \Delta y_{jk} \end{bmatrix}, \tag{C.5}$$

and $\Delta x_{jk} = x_k - x_j$ and $\Delta y_{jk} = y_k - y_j$. The weight w_{jk} is defined as the m-th power of the inverse distance:

$$w_{jk} = \frac{1}{r_{jk}^m}, \quad r_{jk} = \sqrt{\Delta x_{jk}^2 + \Delta y_{jk}^2}. \tag{C.6}$$

In this paper, we set $m = 0.25$ and define $\{k_j\}$ as a set of face neighbors and their face neighbors.