Gradient Calculation Methods on Arbitrary Polyhedral Unstructured Meshes for Cell-Centered CFD Solvers

Emre Sozer*[1], Christoph Brehm*[1] and Cetin C. Kiris†[2]

[1] *Science and Technology Corporation, Moffett Field, CA 94035*
[2] *NASA Ames Research Center, Moffett Field, CA 94035*

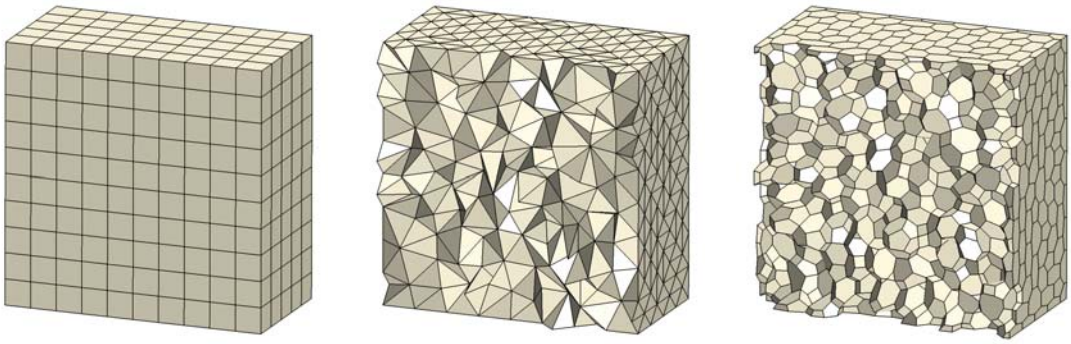
A survey of gradient reconstruction methods for cell-centered data on unstructured meshes is conducted within the scope of accuracy assessment. Formal order of accuracy, as well as error magnitudes for each of the studied methods, are evaluated on a complex mesh of various cell types through consecutive local scaling of an analytical test function. The tests highlighted several gradient operator choices that can consistently achieve 1^{st} order accuracy regardless of cell type and shape. The tests further offered error comparisons for given cell types, leading to the observation that the "ideal" gradient operator choice is not universal. Practical implications of the results are explored via CFD solutions of a 2D inviscid standing vortex, portraying the discretization error properties. A relatively naive, yet largely unexplored, approach of local curvilinear stencil transformation exhibited surprisingly favorable properties.

I. Introduction

Computational Fluid Dynamics (CFD) tools are now routinely used for direct support/guidance of fast-paced engineering design processes. Moreover, a CFD model is expected to represent an increasingly high level of geometric fidelity of a given design while in many cases, the analyst is afforded little time to setup the computational mesh. This encourages the trend of preferring approaches that require little, if any, investment in grid generation. The unstructured methods in CFD are becoming ubiquitous in the industry due to the attraction of semi-automated meshing coupled with the capability to resolve boundary layers efficiently through use of high aspect ratio prism cells.

An unstructured mesh, commonly depicted as consisting of tetrahedral elements, can be considered a superset encompassing any valid cell geometry including hexahedral, tetrahedral and arbitrary polyhedrals (see Figure 1). Utilization of unstructured meshes offers a great deal of ease and flexibility in terms of mesh generation compared to the traditional, block-structured approach. The lack of inherent assumptions about the element types and their connectivity helps achieve a somewhat automated meshing. While the price paid in return is often understood as reduced computational efficiency for a given mesh size, accuracy of the simulations may also suffer. This is largely due to the fact that an unstructured volume mesh quality is typically only controlled through limited, indirect inputs to the mesh generator. Furthermore, lacking a Cartesian structure or a mapping to one, unstructured meshes do not benefit from properties such as a regular stencil, exact cancellation of truncation errors and independence of spatial dimensions. This requires extra care to be

*Research Scientist, Applied Modeling and Simulation Branch, NAS Division, MS N258-2, AIAA Member
†Branch Chief, Applied Modeling and Simulation Branch, NAS Division, MS N258-2, AIAA Senior Member



(a) (b) (c)

Figure 1: Example meshes with (a) Hexahedral, (b) Tetrahedral and (c) Arbitrary polyhedral cell types.

taken in order to preserve accuracy for general cell types. In short, giving up the great level of user involvement and control over the mesh generation process is desirable in terms of the cost of mesh generation, but the burden is transferred to the unstructured solver to maintain accuracy and stability despite potentially sub-quality meshes.

During implementation of an unstructured solver, a choice about where to place the discrete data must be made. Alternative choices include vertices, face-centers, cell-centers or a combination. While advantages and drawbacks of each approach have been heavily debated in the CFD community,[1,2] the current authors feel that no clear "best" choice emerges. Since this study is targeted towards the Launch Vehicle and Ascent Aerodynamics (LAVA) CFD solver,[3–5] the focus will be limited to a cell-centered, finite-volume scheme. Gradient operator choice for an unstructured solver has a strong impact on accuracy, efficiency and robustness. While all of these are crucial factors, we limit our scope to the analysis of accuracy alone.

Aftosmis et al[6] investigated the behavior of linear reconstruction techniques on unstructured meshes. Their chief concern was the behavior of limiters and the effect of element types (triangular vs. quadrilateral) for CFD solutions, particularly for high aspect ratio or irregular elements. They did however, investigate the least squares (LSQR) and Green-Gauss methods for gradient calculation. The methods behaved similarly for regular meshes whereas the LSQR was found to be more tolerant to mesh distortions. Mavriplis[7] examined the LSQR procedure for gradient reconstruction, observing that the method produced accurate gradients for isotropic meshes but the accuracy deteriorated for highly stretched meshes in the presence of curvature. In the latter case, they found the Green-Gauss reconstruction method to be more accurate. Shima et al[8] devised an LSQR method where they incorporate weights based on face areas, attempting to inherit benefits of the Green-Gauss method for stretched meshes. While they note accuracy improvements, they still resort to a hybrid approach where Green-Gauss method is used for thin and distorted mesh regions.[9] A comprehensive survey of unstructured mesh gradient methods, in the context of computer graphics, is conducted by Correa et al.[10] They focus on cost and performance in volume rendering with respect to mesh resolution, element shapes, neighborhood size and scalar field complexity. They find the inverse weighted regression method to provide the highest accuracy for irregular meshes and the Green-Gauss method to perform poorly for badly shaped elements.

In the following sections, calculation of geometric properties (areas, volumes and centroids) for an arbitrary polyhedral computational cell is described. We then summarize a few common approaches (Green-Gauss with various face averaging schemes and Least Squares) to gradient reconstruction, as well as a simplistic, and largely ignored, approach of applying a local curvilinear transformation. The alternate methodologies are tested for formal order of accuracy by consistent local refinement of

a complex unstructured mesh of various cell types and shapes representing those that are commonly encountered in a practical unstructured CFD meshes. Linear-exactness of the gradient operator (i.e. ability to exactly reproduce the gradient of a linear function) is verified for several of the investigated methodologies. This often overlooked property is a necessary condition to achieve a 2^{nd} order accurate scheme.[11,12] Besides the formal order of accuracy, error magnitudes through the refinement levels for a set of different cell types are compared. The methodologies are also tested for a 2D standing inviscid vortex problem where the dissipation rate of the vortex, which is purely due to the numerical discretization error, is compared.

II. Geometric Properties

Given an unstructured mesh of arbitrary polyhedral cell types, the first problem faced with is the calculation of the geometric properties of cells and faces. While calculating these, care must be taken to respect some underlying assumptions of a conservative scheme such that $\Omega = \cup C_i$, $C_i \cap C_j = \emptyset$ for $i \neq j$ (i.e. the computational cells, C_i, should completely cover the domain, Ω, and that they must be non-overlapping). Each control volume should be water-tight and hence the vector sum of their face areas must be zero,

$$\sum_{i=1}^{N_{faces}} \vec{A}_i = 0 \tag{1}$$

Furthermore, the solver implementation must ensure that the neighboring cells use the same area, normal and centroid for a shared face. The procedures for calculating consistent geometric properties of faces and cells are explained below.

II.A. Polygonal Face Area and Centroid

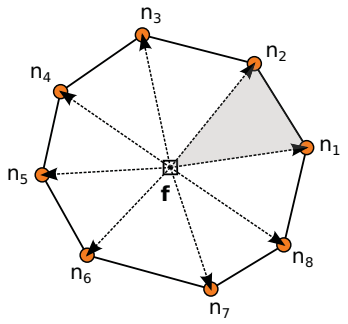


Figure 2: Triangulation of a polygon.

A polyhedral cell consists of a number of polygonal faces forming a closed volume. The area vector and the centroid location of each face needs to be computed. This can be achieved via triangulation of the polygon around a given point f as shown in Figure 2. A convenient starting location for the point f is the midpoint (simple average of the nodes of the polygon).

$$\vec{r}_f = \frac{1}{N_{fnodes}} \sum_{i=1}^{N_{fnodes}} \vec{r}_{n_i} \tag{2}$$

where N_{fnodes} denotes number of face nodes and $\vec{r}$ is the position vector. The area of each of the

triangular patches are added to get the area of the polygon face.

$$\vec{A}_{tri,i} = \frac{(\vec{r}_{n_i} - \vec{r}_f) \times (\vec{r}_{n_{i+1}} - \vec{r}_f)}{2} \quad \text{for} \;\; i = 1, N_{fnodes} \tag{3}$$

$$\vec{A}_f = \sum_{i=1}^{N_{fnodes}} \vec{A}_{tri,i} \tag{4}$$

and $\vec{r}_{N_{fnodes}+1} = \vec{r}_1$.

Centroid of the face is computed in a similar fashion as

$$\vec{r}_f = \frac{1}{|\vec{A}_f|} \sum_{i=1}^{N_{fnodes}} |\vec{A}_{tri,i}| \left(\vec{r}_{n_i} + \vec{r}_{n_{i+1}} + \vec{r}_f\right)/3 \tag{5}$$

Note that the face centroid $\vec{r}_f$ was initially taken as simply the midpoint of the nodes but it is updated at the end of the process. In the case of a planar polygon, this updated location reflects the true centroid of the polygon. However, while not desirable, polygon nodes may be highly non-coplanar in practice. This introduces ambiguity to the centroid location as no unique definition exists based solely on the knowledge of the node coordinates. In this case, simply iterating over Equations 3-5 until convergence provides a reasonable answer.

The triangulated polygonal face, even if non-coplanar, is still attached to each of the vertices defining it as opposed to an approach where one might fit a planar surface to the vertices. This ensures that, once all the faces of a cell is processed, a water-tight control volume is achieved. We note once again that regardless of the aforementioned ambiguity for non-coplanar polygons, consistency can be retained if the cells sharing a face use the same face centroid and area for their reconstruction and flux integration.

II.B. Polyhedral Volume and Centroid

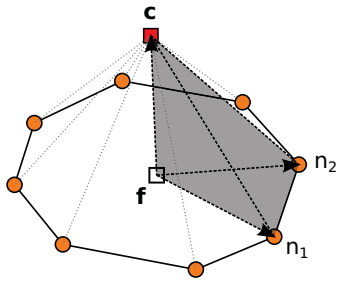


Figure 3: Tetrahedralization of a polyhedral (showing a single face).

The volume and the centroid location of a polyhedral cell can be computed via tetrahedralization, basically by extending the logic presented in the previous section to 3D. Figure 3 shows a single face of a polyhedral cell and the corresponding tetrahedralization around a midpoint m.

$$\vec{r}_m = \frac{1}{N_{cnodes}} \sum_{i=1}^{N_{cnodes}} \vec{r}_{n_i} \tag{6}$$

where N_{cnodes} is the number of cell nodes. Volume of each tetrahedral patch formed over the polygonal face f is then

$$V_{tet,f,i} = \frac{1}{3} \vec{A}_{tri,f,i} \cdot (\vec{r}_m - \vec{r}_f) \tag{7}$$

where $\vec{A}_{tri,f,i}$ and $\vec{r}_f$ for a given face f is obtained from Equation 3 and Equation 5 respectively. This usage of face triangulation around the previously calculated centroid ensures that a consistent volume is obtained.

The integrated volume and the centroid of the polyhedral cell is then calculated via summation of the contributions from each face as

$$V = \sum_{f=1}^{N_{faces}} \sum_{i=1}^{N_{fnodes}} V_{tet,f,i} \tag{8}$$

$$\vec{r}_c = \frac{1}{4V} \sum_{f=1}^{N_{faces}} \sum_{i=1}^{N_{fnodes}} \left(\vec{r}_{n_{f,i}} + \vec{r}_{n_{f,i+1}} + \vec{r}_f + \vec{r}_m\right) V_{tet,f,i} \tag{9}$$

where N_{faces} is the number of faces, N_{fnodes} is the number of face nodes and $\vec{r}_{f,N_{fnodes}+1} = \vec{r}_{f,1}$.

The arbitrary polyhedral meshes are typically generated via dualization of an initial tetrahedral mesh. Thus, the polyhedral cells generated this way are inherently tetrahedralizable. However, a more general procedure based on the Green-Gauss theorem that does not rely on this assumption could be adopted such that

$$V = \frac{1}{3} \sum_{f=1}^{N_{faces}} \vec{r}_f \cdot \vec{A}_f \tag{10}$$

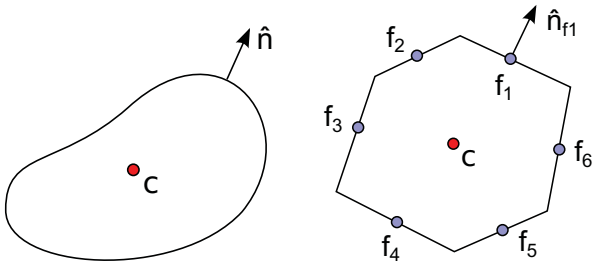
$$r_{c,i} = \frac{1}{2V} \sum_{f=1}^{N_{faces}} r_{f,i}^2 A_{f,i} \text{ for } i = 1, 2, 3 \tag{11}$$

III. Gradient Calculation

The difficulty in calculating gradients in an unstructured mesh stems from the lack of a consistent, inherent connectivity. The stencil for gradient calculation, as well as the corresponding coefficients vary cell-by-cell and are costly to compute. Hence, those are typically pre-computed and stored.

Two of the most common methods for gradient calculation are the Green-Gauss and the Least Squares approaches. Both have several common variations, some of which are explained in the following sections.

III.A. Green-Gauss Gradient Method



(a) (b)

Figure 4: Notations for (a) a control volume (b) a computational cell.

The Green-Gauss method represents an intuitive, sound basis for gradient calculation. According to the Green-Gauss theorem, average gradient of a scalar ϕ in a closed volume V (Figure 4a) can be

obtained by

$$\iiint_V \nabla\phi dV = \oiint_A \phi \hat{\mathrm{n}} dA \tag{12}$$

$$\overline{\nabla\phi} = \frac{1}{V} \oiint_A \phi \hat{\mathrm{n}} dA \tag{13}$$

where $\hat{\mathrm{n}}$ is the surface unit normal vector and A is the surface area. For a 2^{nd} order scheme with midpoint quadrature, the Green-Gauss method takes on the following discrete form for a polyhedral (Figure 4b):

$$\nabla\phi = \frac{1}{V} \sum_{f=1}^{N_{faces}} \bar{\phi}_f \hat{\mathrm{n}}_\mathrm{f} A_f \tag{14}$$

where N_{faces} is the number of faces and $\bar{\phi}_f$ is the average of the scalar over the face f.

Up to this point, average gradient of a linear function at the polyhedral cell centroid is represented exactly by Equation 14. The potential errors are introduced through the particular choice of a face averaging method to obtain $\bar{\phi}_f$. Several common alternatives in this regard are discussed below.

III.A.1. Simple Face Averaging

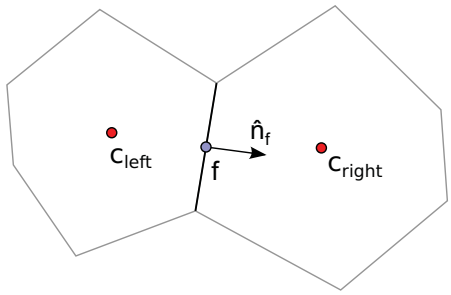


Figure 5: Simple face averaging.

Simple average of the cell center values at the left and right sides of the face is taken as the face center value (Figure 5).

$$\bar{\phi}_f = \frac{\phi_{c_{left}} + \phi_{c_{right}}}{2} \tag{15}$$

As the most basic approach, the simple averaging method is still commonly used due to its attractive properties of straightforward and cheap implementation. In fact, its usage is usually implied when cell-centered Green-Gauss gradient method is referred to without mention of the associated face averaging method. This, in some cases, leads to unfair characterizations of the Green-Gauss method itself.[7,8]

III.A.2. Inverse Distance Weighted (IDW) Face Interpolation

Another popular approach to face averaging, IDW method utilizes the entire neighbor stencil around face f (see Figure 6).

$$\bar{\phi}_f = \frac{\sum_{i=1}^{N} \phi_i / |\vec{d}_i|^2}{\sum_{i=1}^{N} 1/|\vec{d}_i|^2} \tag{16}$$

where the sum is carried out over the entire stencil and $\vec{d}_i = \vec{r}_i - \vec{r}_f$ represents the distance between the stencil point and the current face center.

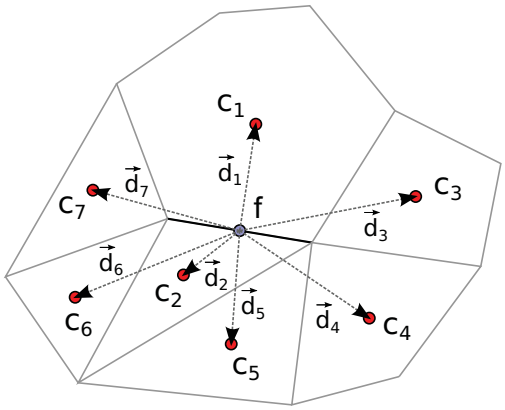


Figure 6: Inverse Distance Weighted (IDW) and Least Squares (LSQR) face interpolation stencils.

IDW has shortcomings when the neighboring cell centers are not evenly distributed around the face but clustered in certain directions; a scenario likely to occur for regions with poor grid quality or at the interfaces of different cell types in mixed cell type meshes. In addition to this non-isotropy, potentially large discrepancies in stencil distances may skew the weights significantly, a phenomena made worse by the usage of squared distances. Nonetheless, the squared weighting seems to be the most common approach taken with IDW and hence we chose to adopt it for our evaluation. Variations of this method involving different weightings (e.g. volume, inverse distance, ...) are possible but this does not change the fundamental flaws explained above.

III.A.3. Weighted Least Squares (LSQR) Face Interpolation

The weighted/unweighted least squares face interpolation coefficients are derived by considering the Taylor series expansion of ϕ_i at the i^{th} stencil point around the center of the face (see Figure 6):

$$\phi_i = \phi_f + \Delta x_i \phi_{f,x} + \Delta y_i \phi_{f,y} + h.o.t. \tag{17}$$

where Δx_i is the x-direction component of the distance vector $\vec{d_i}$ and $\phi_{f,x}$ is the x-derivative of the interpolated scalar ϕ at face f. By considering the Taylor series expansion, a functional can be obtained as

$$f\left(w_i, \phi_i, \phi_{f,x}, \phi_{f,y}, ...\right) = \sum_{i=1}^{N} w_i \left[\Delta\phi_i - \Delta x_i \phi_{f,x} - \Delta y_i \phi_{f,y} - h.o.t.\right]^2, \tag{18}$$

where w_i are the weights for each stencil point i and $\Delta\phi_i = \phi_i - \phi_f$. In order to derive the stencil coefficients, the functional $f\left(w_i, \phi_i, \phi_{f,x}, \phi_{f,y}, ...\right)$ is minimized.

$$\begin{aligned}
\frac{\partial f}{\partial \phi_f} &= \sum_{i=1}^{N} 2w_i \left[\Delta\phi_i - \Delta x_i \phi_x - \Delta y_i \phi_y - h.o.t.\right] = 0 \\
\frac{\partial f}{\partial \phi_{f,x}} &= \sum_{i=1}^{N} 2w_i \Delta x_i \left[\Delta\phi_i - \Delta x_i \phi_{f,x} - \Delta y_i \phi_{f,y} - h.o.t.\right] = 0 \\
\frac{\partial f}{\partial \phi_{f,y}} &= \sum_{i=1}^{N} 2w_i \Delta y_i \left[\Delta\phi_i - \Delta x_i \phi_{f,x} - \Delta y_i \phi_{f,y} - h.o.t.\right] = 0
\end{aligned} \tag{19}$$

Utilizing Equations 19, a linear system of equations in terms of the stencil values ϕ_i and ϕ_f, and the derivatives $\phi_{f,x}$ and $\phi_{f,y}$ can be written as

$$\underline{A}\vec{\Phi}_f = \underline{B}\vec{\phi}_s \quad \rightarrow \quad \vec{\Phi}_f = \underline{A}^{-1}\underline{B}\vec{\phi}_s, \text{ with } \vec{\Phi}_f = [\phi_f, \phi_{f,x}, \phi_{f,y}]^T, \ \vec{\phi}_s = [\phi_1, \phi_2, \phi_3, ...]^T, \tag{20}$$

$$\underline{A} = \begin{bmatrix} \sum w_i & \sum w_i\Delta x_i & \sum w_i\Delta y_i \\ \sum w_i\Delta x_i & \sum w_i\Delta x_i^2 & \sum w_i\Delta x_i\Delta y_i \\ \sum w_i\Delta y_i & \sum w_i\Delta x_i\Delta y_i & \sum w_i\Delta y_i^2 \end{bmatrix}, \text{ and } \quad \underline{B} = \begin{bmatrix} w_1 & w_2 & ... \\ w_1\Delta x_i & w_2\Delta x_i & ... \\ w_1\Delta y_i & w_2\Delta y_i & ... \end{bmatrix}. \tag{21}$$

The stencil coefficients $\mathcal{C}_i$ for each stencil point are contained in the first row of $\underline{A}^{-1}\underline{B}$. The coefficients can be expressed in the short form notation as:

$$\mathcal{C}_i = \underline{A}^{-1}_{1,k}\underline{B}_{k,i} \tag{22}$$

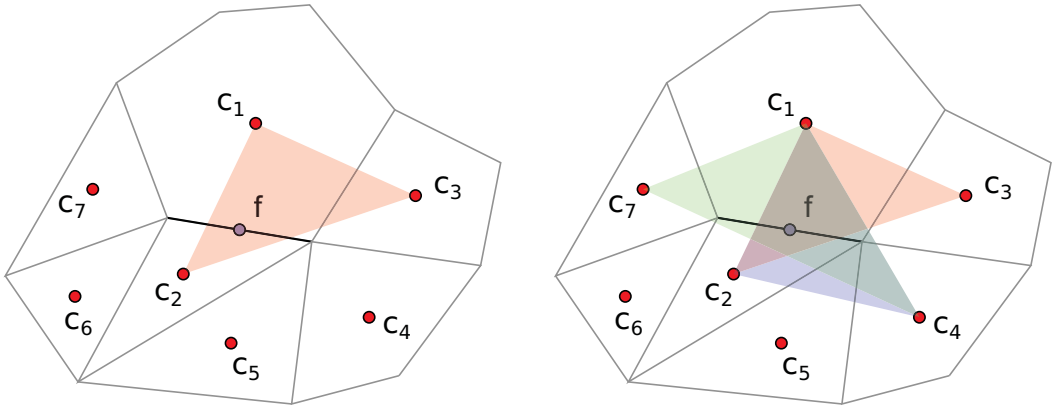
where $\underline{A}^{-1}_{1,k}$ is the value in the first row and k^{th} column of $\underline{A}$. The interpolated value at the face is then expressed as

$$\bar{\phi}_f = \sum_{i=1}^{N} \mathcal{C}_i\phi_i \tag{23}$$

Current implementation of this method in LAVA uses inverse distances as the stencil weights for the LSQR method:

$$w_i = 1/|\vec{d}_i| \tag{24}$$

III.A.4. Weighted Tri-Linear Face Interpolation (WTLI)



(a) Single triangle (b) Triangle combinations

Figure 7: Weighted Tri-Linear (WTLI) face interpolation stencil selection.

Given data at 3 non-colinear points forming a triangle (4 non-coplanar points in 3D forming a tetrahedron) around the face f (see Figure 7), a linear regression can be formed as

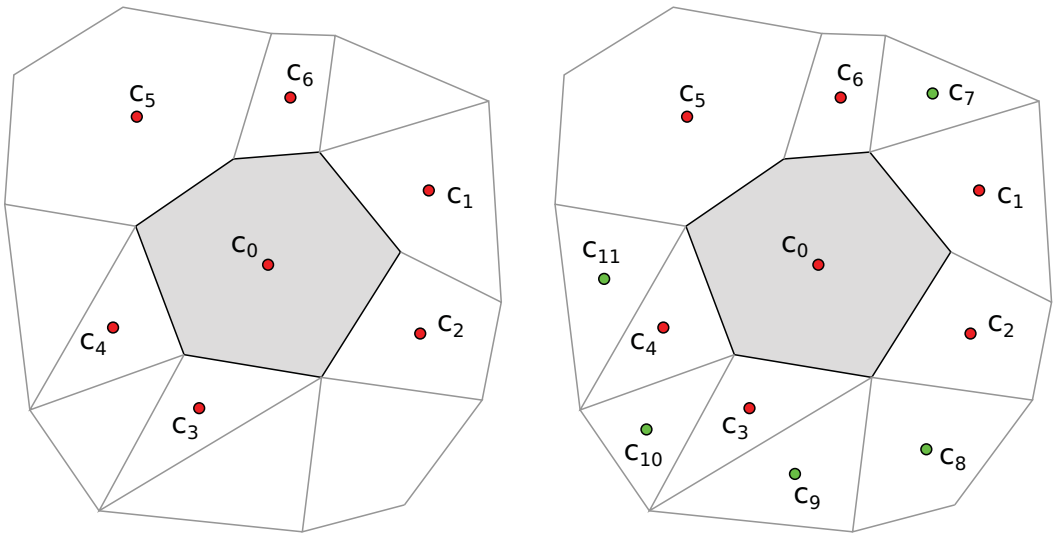
$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} \mathcal{C}_1 \\ \mathcal{C}_2 \\ \mathcal{C}_3 \end{Bmatrix} = \begin{Bmatrix} x_f \\ y_f \\ 1 \end{Bmatrix} \tag{25}$$

The stencil coefficients $\mathcal{C}_i$ can then be used to perform the face interpolation as

$$\bar{\phi}_f = \mathcal{C}_1\phi_1 + \mathcal{C}_1\phi_2 + \mathcal{C}_1\phi_3 \tag{26}$$

The coefficients $\mathcal{C}_i$ will be positive if the face center f lies within the triangle, ensuring a monotone interpolation. Figure 7b shows some of the other possible options for the interpolation triangle. LAVA implements this method by utilizing all the possible triangles (tetrahedra in 3D) enclosing point f. Each triangle's stencil coefficients are then weighted with the inverse of triangle center's distance from the point f and combined, hence the name Weighted Tri-Linear (WTLI) interpolation.

III.B. Least Squares (LSQR) Gradient Method



(a) Compact (b) Extended

Figure 8: Least Squares (LSQR) gradient stencils.

The Least Squares gradient method utilizes the same basic technique employed in obtaining the LSQR face averaging (used for the Green-Gauss method) as described in Section III.A.3. However, in applying LSQR for direct calculation of the gradient, the stencil is now centered around a cell center as opposed to the face center. Either a face neighborhood (referred as compact stencil, as shown in Figure 8a) or the node neighborhood (referred as the extended stencil, as shown in Figure 8b) of the cell c_0 can be used. Thus, the ϕ_f in Equations 17-21 is now replaced with ϕ_0 and the lower limit of summations now start from $i = 0$ (i.e. including the target cell center). The distance vector is now

$$\vec{d_i} = \vec{r_i} - \vec{r_0} \tag{27}$$

The resulting gradient stencil coefficients

$$\mathcal{C}_{i,x} = \underline{A}^{-1}_{2,k}\underline{B}_{k,i} \tag{28}$$

$$\mathcal{C}_{i,y} = \underline{A}^{-1}_{3,k}\underline{B}_{k,i} \tag{29}$$

are used the obtain:

$$\bar{\phi}_{1,x} = \sum_{i=0}^{N} \mathcal{C}_{i,x}\phi_i \tag{30}$$

$$\bar{\phi}_{1,y} = \sum_{i=0}^{N} \mathcal{C}_{i,y}\phi_i \tag{31}$$

The same inverse distance weighting as shown in Equation 24 is used here as well.

III.C. Curvilinear Gradient Method

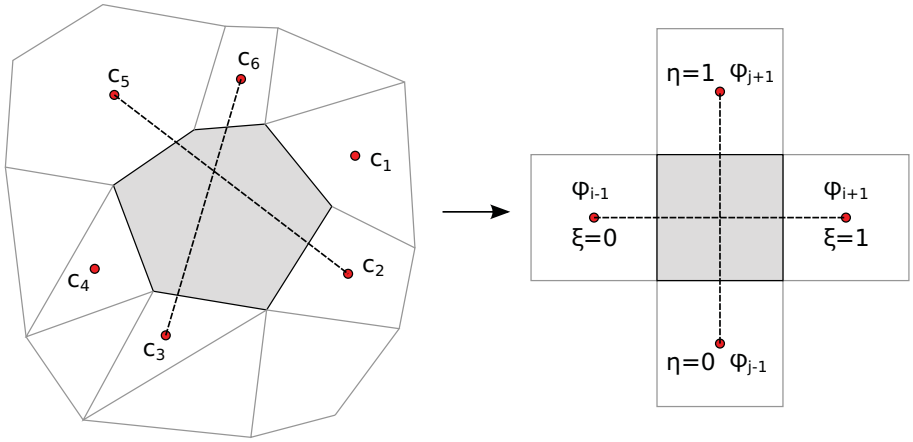


Figure 9: Curvilinear gradient stencil and mapping.

The curvilinear gradient method imitates a structured-grid approach by mapping the local cell gradient stencil to a uniform Cartesian mesh. For this mapping, a 4 point stencil (6 points in 3D) needs to be selected from the available cell neighbors. Figure 9 shows one such selection. The selection procedure evaluates every possible combination of stencil point pairs to pick curvilinear directions ξ and η. Once the stencil is selected, central differencing yields

$$\frac{\partial \phi}{\partial \xi} = \phi_{i+1} - \phi_{i-1} \tag{32}$$

$$\frac{\partial \phi}{\partial \eta} = \phi_{j+1} - \phi_{j-1} \tag{33}$$

The gradient in Cartesian coordinates can then be arrived as

$$\begin{Bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{Bmatrix} = \mathcal{J}^{-1} \begin{Bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{Bmatrix} \quad \text{with} \quad \mathcal{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} (x_{i+1} - x_{i-1}) & (y_{i+1} - y_{i-1}) \\ (x_{j+1} - x_{j-1}) & (y_{j+1} - y_{j-1}) \end{bmatrix} \tag{34}$$

In the current implementation, the stencil pairs that produce the largest determinant of the Jacobian $|\mathcal{J}|$ are selected out of all the possible options. Considering that only a compact neighborhood consisting of face sharing cells are considered for the stencil selection. The largest Jacobian determinant criteria usually identifies the most orthogonal ξ and η directions.

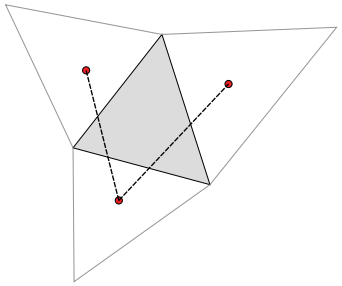


Figure 10: Shared-point curvilinear gradient stencil for a triangle.

Note that the stencil pairs are allowed to share points (see Figure 10). This enables application of the curvilinear method to the cell types which have less than 6 faces (e.g. tetrahedron). Among the

gradient calculation methods described herein, the curvilinear method results in the most compact stencil (at most 6 points in 3D).

IV. Order of Accuracy Assessment

IV.A. Methodology

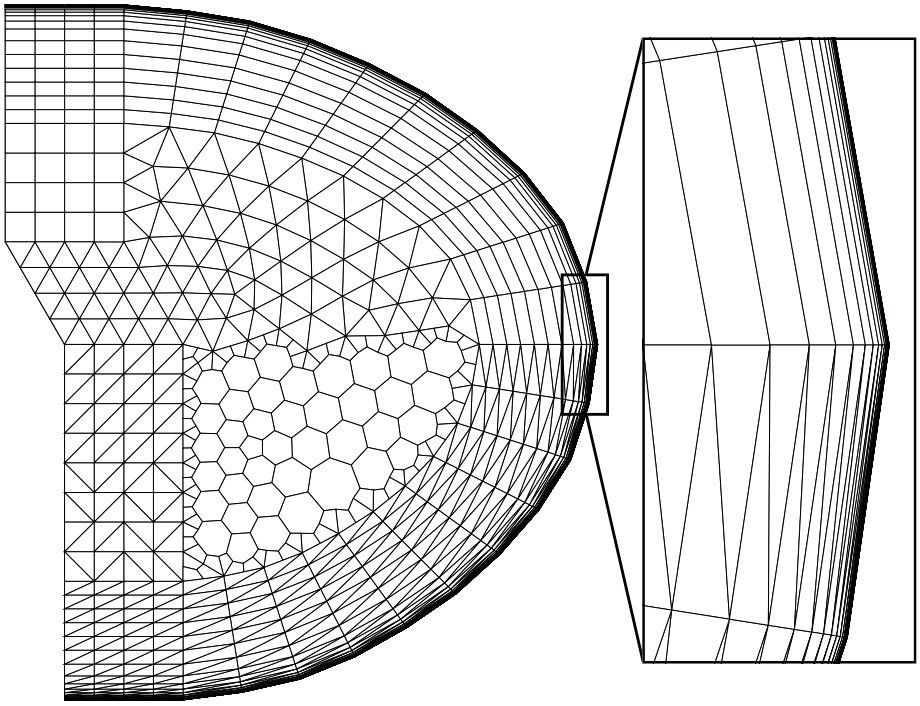


Figure 11: Test mesh for gradient order of accuracy assessment.

The gradient calculation methods outlined in the previous section are assessed on a test mesh (see Figure 11) that contains various cell types.

Table 1. Test mesh cell types.

Quadrilateral	Triangular	Polyhedral
Square	Equilateral	Random
Stretched	Regular patterned right	
Stretched with curvature	Random patterned right	
	Stretched	
	Stretched with curvature	
	Random	

Maximum aspect ratio of the stretched cells is around 2000.

In order to assess the order of accuracy (OOA), the numerical gradient of a known test function is compared to its analytical value on consecutively refined meshes. This requires consistent refinement, which is not easily attained for unstructured mesh types. Instead of global refinement, we adopt a method where each cell and its immediate neighbors that form its gradient stencil are locally scaled down for each refinement level k. In the implementation, this is effectively achieved by scaling the

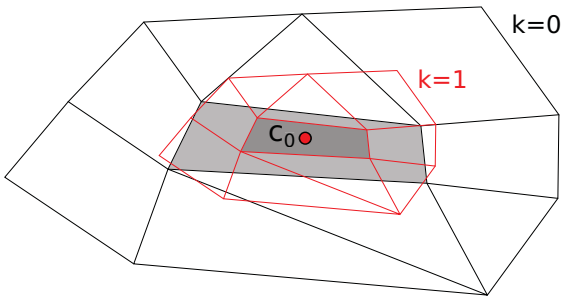


Figure 12: Local scaling of a cell and its gradient stencil.

distribution of the test function:

$$\phi_i = \left(1 + x^*_{1,i} + x^*_{2,i} + x^*_{1,i}x^*_{2,i}\right)\left(\sin\left(\frac{x^*_{1,i}}{4\pi}\right) + \sin\left(\frac{x^*_{2,i}}{4\pi}\right)\right) \tag{35}$$

where

$$x^*_{1,i} = \frac{x_{1,i}}{2^k|\vec{e}_{min}|} \tag{36}$$

$$x^*_{2,i} = \frac{x_{2,i}}{2^k|\vec{e}_{max}|} \tag{37}$$

$$x_{1,i} = (\vec{r}_i - \vec{r}_0)\cdot\vec{d}_{x_1} \tag{38}$$

$$x_{2,i} = (\vec{r}_i - \vec{r}_0)\cdot\vec{d}_{x_2} \tag{39}$$

$$\vec{d}_{x_1} = \frac{\vec{e}_{min}}{|\vec{e}_{min}|} \quad \text{and} \quad \vec{d}_{x_1}\times\vec{d}_{x_2} = 0 \tag{40}$$

where i is the stencil point (neighbor cell center), k is the refinement level, $\vec{e}_{min}$ and $\vec{e}_{max}$ are the shortest and longest edge vectors respectively. The function ϕ is evaluated on a local coordinate system with an origin at the center c_0 (see Figure 12) and aligned with the direction of the shortest edge $\vec{e}_{min}$ of the cell in question. The distribution is also anisotropic, i.e. the function changes more rapidly in the direction of the shortest edge. The function is evaluated for each cell (and its stencil) repeatedly through several refinement levels and the error in the resulting numerical gradient is calculated,

$$\epsilon^k_0 = \left|\nabla\phi^k_{0,numerical} - \nabla\phi_{0,exact}\right| \tag{41}$$

enabling estimation of gradient order of accuracy for cell 0 as

$$\mathcal{O}_0 = \frac{\log\left(\epsilon^{k-1}_0/\epsilon^k_0\right)}{\log 2} \tag{42}$$

IV.B. Results

The gradient order of accuracy (OOA) for various aforementioned methodologies are presented in Figures 14-20. The OOA, calculated between refinement levels 19 and 20, are shown separately in x and y directions. Note that, for a 2^{nd} order scheme, a gradient operator OOA of at least 1 is needed.

Figure 14 shows the distribution of OOA for the LSQR scheme with compact stencil. The operator is able to produce at least 1^{st} order accuracy in the entire field. The cells with a uniformly spaced stencil that are aligned with x or y directions exhibit 2^{nd} order accuracy in the corresponding directions. This is due to perfect cancellation of 1^{st} order errors and it breaks down as soon as mesh uniformity is lost.

LSQR with the extended stencil yields similar results (see Figure 15) to that of the compact stencil, with the exception that the 2^{nd} order accurate region is narrower as expected. The cancellation leading to the increased order now needs to be satisfied in a wider stencil.

The Green-Gauss method with simple averaging as well as IDW fails to achieve 1^{st} order accuracy as seen in Figures 16 and 17 respectively. Both of these methods neglect to incorporate directionality of their stencils and neither is linear-exact, i.e. can't reproduce gradient of a linear function exactly. Even a 1D scenario with non-uniform spacing (see Figure 13) is sufficient to show that their leading error term is 0^{th} order. This is demonstrated below for the case of the simple averaging.

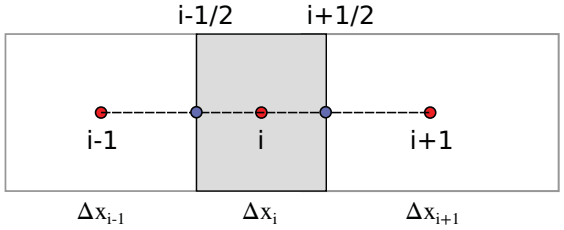


Figure 13: A 1D stencil with non-uniform spacing.

The simple face averages,

$$\phi_{i+1/2} \approx \frac{\phi_i + \phi_{i+1}}{2} \tag{43}$$

$$\phi_{i-1/2} \approx \frac{\phi_{i-1} + \phi_i}{2} \tag{44}$$

are used to calculate the gradient via the Green-Gauss method as

$$\nabla^{GG}\phi_i = \frac{\phi_{i+1/2} - \phi_{i-1/2}}{\Delta x_i} = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x_i} \tag{45}$$

Taylor series expansions around point i in forward and backward directions,

$$\phi_{i+1} = \phi_i + \nabla\phi_i \frac{\Delta x_i + \Delta x_{i+1}}{2} + \nabla^2\phi_i \frac{(\Delta x_i + \Delta x_{i+1})^2}{8} + \mathcal{O}(\Delta x^3) \tag{46}$$

$$\phi_{i-1} = \phi_i - \nabla\phi_i \frac{\Delta x_i + \Delta x_{i-1}}{2} + \nabla^2\phi_i \frac{(\Delta x_i + \Delta x_{i-1})^2}{8} + \mathcal{O}(\Delta x^3) \tag{47}$$

are substituted in Equation 45 to yield:

$$\begin{aligned}
\nabla^{GG}\phi_i &= \frac{1}{2\Delta x_i}\left[\nabla\phi_i\left(\Delta x_i + \frac{\Delta x_{i+1} + \Delta x_{i-1}}{2}\right)\right. \\
&\quad \left. + \nabla^2\phi_i \frac{2\Delta x_i(\Delta x_{i+1} - \Delta x_{i-1}) + \Delta x_{i+1}^2 - \Delta x_{i-1}^2}{8} + \mathcal{O}(\Delta x^3)\right] \\
&= \nabla\phi_i\left(\frac{1}{2} + \frac{\Delta x_{i+1} + \Delta x_{i-1}}{4\Delta x_i}\right) + \nabla^2\phi_i\left(\frac{\Delta x_{i+1} - \Delta x_{i-1}}{8} + \frac{\Delta x_{i+1}^2 - \Delta x_{i-1}^2}{16\Delta x_i}\right) + \mathcal{O}(\Delta x^2) \\
&= \nabla\phi_i + \nabla\phi_i\left(-\frac{1}{2} + \frac{\Delta x_{i+1} + \Delta x_{i-1}}{4\Delta x_i}\right) + \mathcal{O}(\Delta x)
\end{aligned} \tag{48}$$

Equation 48 shows that the leading error term in the simple face averaged Green-Gauss gradient operator is 0^{th} order. We also observe that when $\Delta x_i = \Delta x_{i-1} = \Delta x_{i+1}$, 2^{nd} order accuracy is attained. The equation can also be cast as

$$\nabla^{GG}\phi_i = \mathcal{C}\nabla\phi_i + \mathcal{O}(\Delta x) \tag{49}$$

where $\mathcal{C}$ is a constant (with uniform refinement):

$$\mathcal{C} = \frac{1}{2} + \frac{\Delta x_{i+1} + \Delta x_{i-1}}{4\Delta x_i} \tag{50}$$

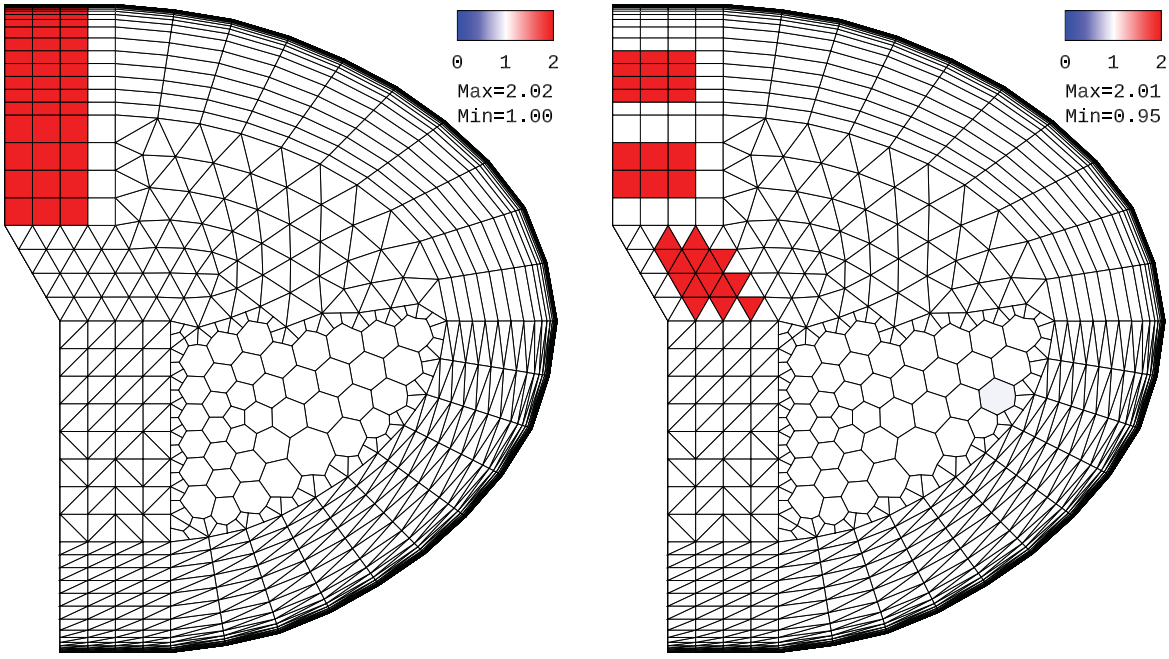
This implies that the estimated gradient will converge to a value in a 1^{st} order accurate manner. However, the converged value will be different from the exact value by a factor of $\mathcal{C}$. In other words, the method is intrinsically inconsistent. This is easy to overlook in practice, since the value of $\mathcal{C}$ is regularly close to 1. This, although, should not be taken as granted since a general unstructured mesh for a complicated geometry can contain cell groups leading to a gross violation of this property, hence producing large local errors.

A similar analysis can easily be performed for the IDW method, with the same conclusions. These two methods (Green-Gauss with simple face averaging or IDW), although commonly applied in practice, are inconsistent and should not be preferred.

The WTLI and LSQR face interpolation methods for the Green-Gauss gradient yields very similar results (see Figures 18 and 19). Thanks to their linear-exact property, they are able to achieve 1^{st} order accuracy, with the exception of a limited number of cells. Recall that the plotted OOA distributions are generated using the error drop between refinement levels 19 and 20. The cells that appear to fail the test here were indeed 1^{st} order accurate at around 15^{th} refinement level. Also observing that this group of cells exhibit a rather greater variation of length scales in their stencils, it is not far-fetched to assume that the ailment is an early onset of the round-off error. Nonetheless, this still exposes a poor quality of these schemes compared to the pure LSQR gradient.

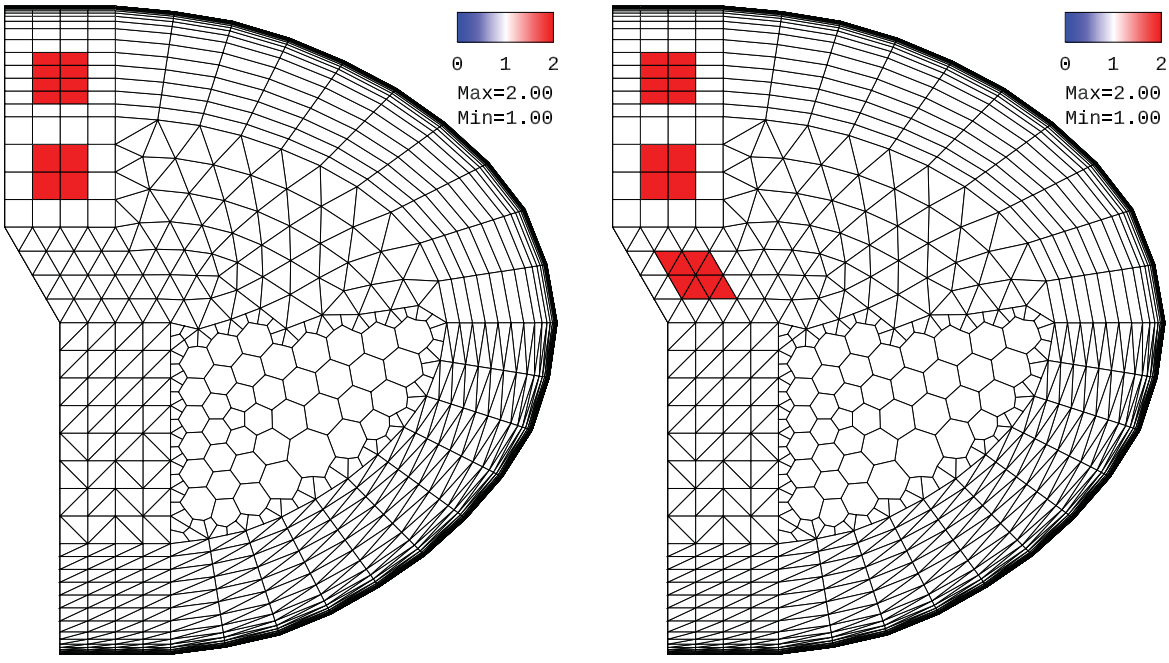
The curvilinear gradient results, as shown in Figure 20, are much like those of the LSQR with compact stencil. This is not surprising as both methods are linear-exact and both utilize compact stencils. Note that the curvilinear gradient operator has the most compact stencil of the alternative methods in scope here, utilizing at most 4 points (for 2D).

The Figures 14-20 also report the minimum and maximum order of accuracies observed in the entire domain. For the LSQR and curvilinear methods, the min/max order of accuracies are bounded between 1 and 2, indicating that all the cells in the domain, including the thin cells next to the boundary, achieve at least 1^{st} order. The other methods produce at least a few cells that violate this range and exhibit a large positive or even negative order of accuracy. Note that these min/max values are obtained from the error comparisons between 19^{th} and 20^{th} refinement levels (see Equations 35-40). The cells that stall in error convergence earlier exhibit fluctuations in error magnitudes with further refinement levels. These fluctuation magnitudes can be relatively large compared to the expected error corresponding to 1^{st} order accuracy at level 20 , hence resulting in the seemingly random order of accuracies greatly deviating from the reasonable range of 0 to 2.



(x-direction) (y-direction)

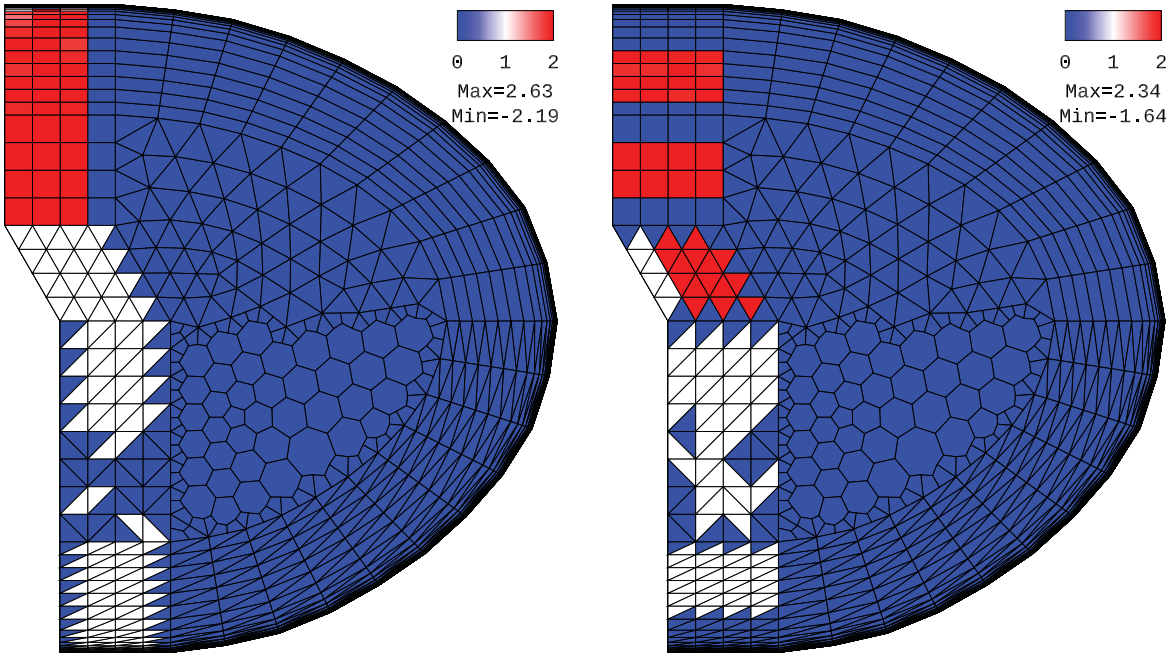
Figure 14: Gradient order of accuracy distribution for the LSQR Compact method.



(x-direction) (y-direction)

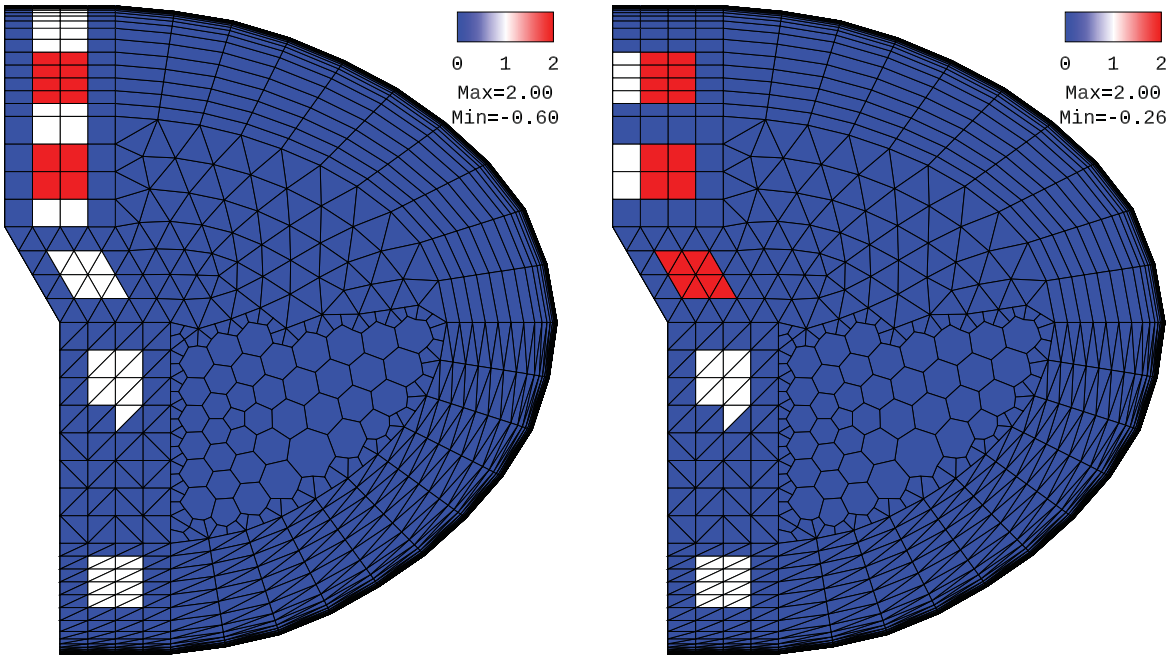
Figure 15: Gradient order of accuracy distribution for the LSQR Extended method.

While order of accuracy is a crucial property to inspect, it is pertinent to look at the actual error



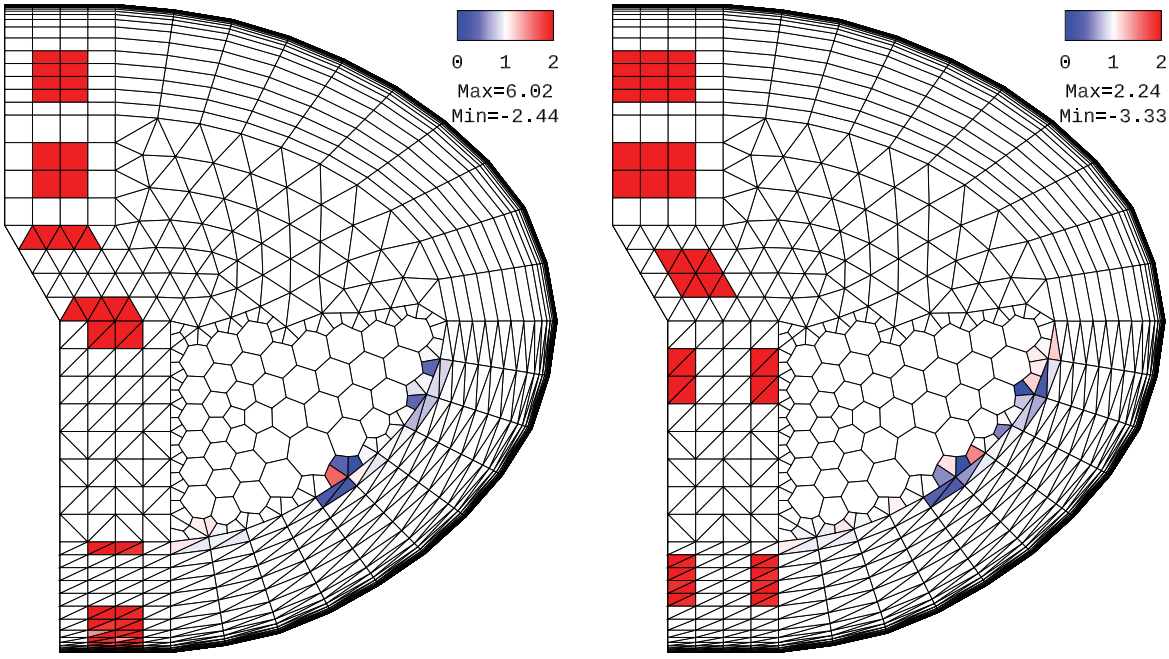
(x-direction) (y-direction)

Figure 16: Gradient order of accuracy distribution for the Green-Gauss Simple method.



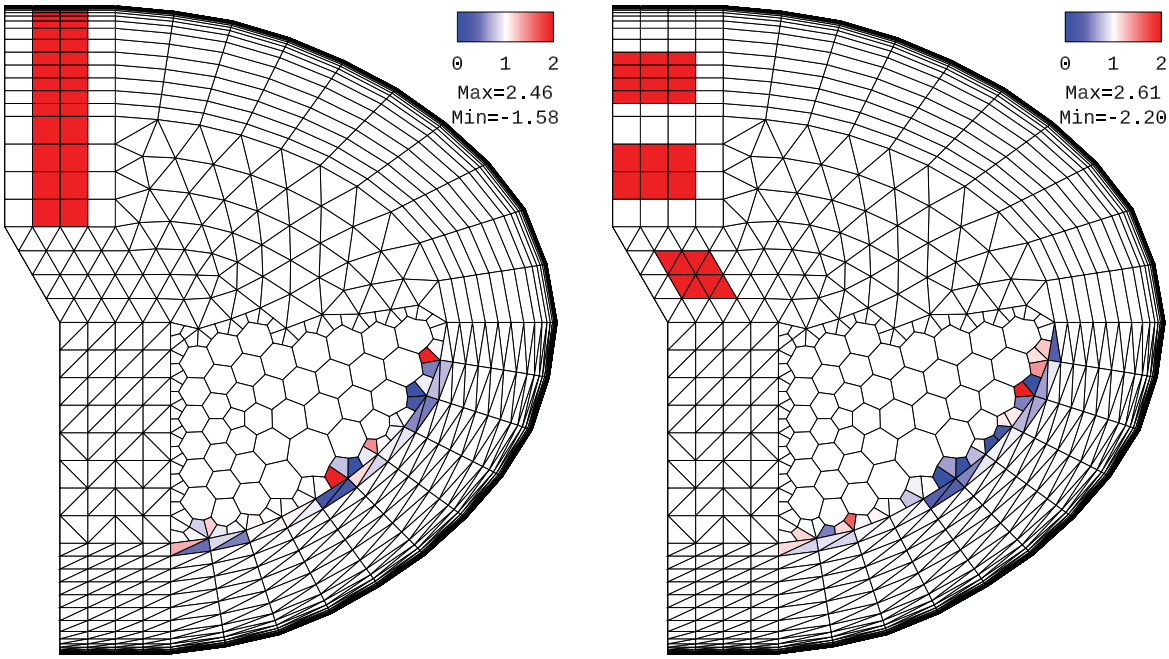
(x-direction) (y-direction)

Figure 17: Gradient order of accuracy distribution for the Green-Gauss IDW method.



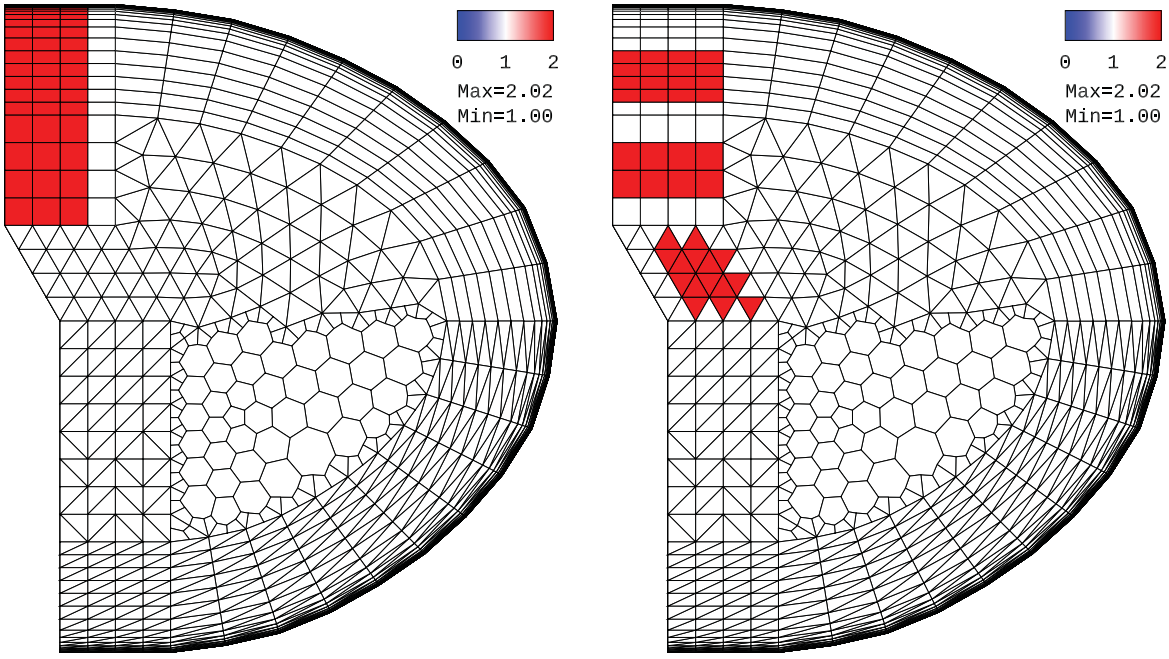
(x-direction) (y-direction)

Figure 18: Gradient order of accuracy distribution for the Green-Gauss WTLI method.



(x-direction) (y-direction)

Figure 19: Gradient order of accuracy distribution for the Green-Gauss LSQR method.



(x-direction) (y-direction)

Figure 20: Gradient order of accuracy distribution for the Curvilinear method.

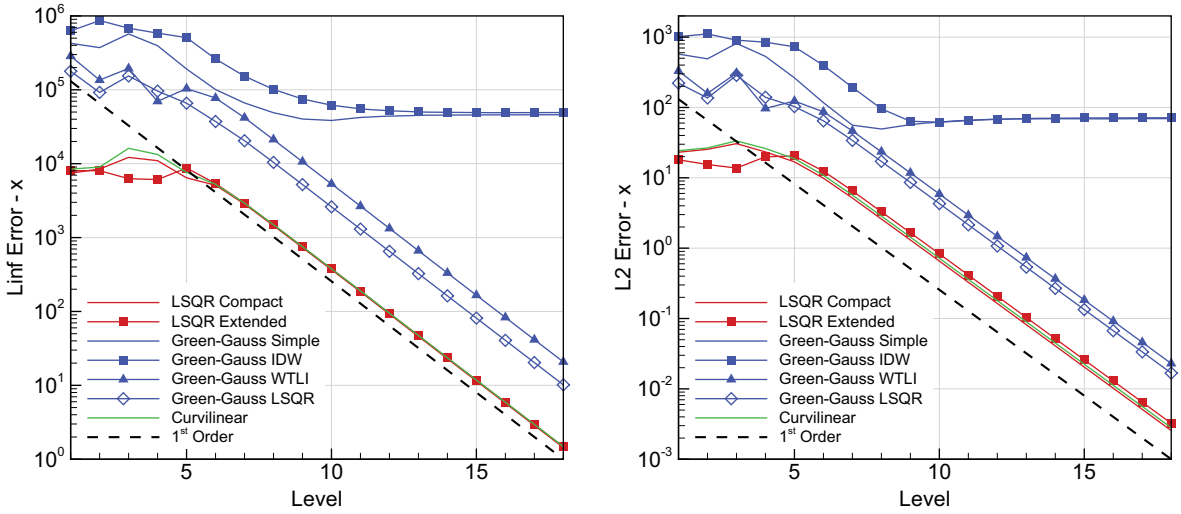


Figure 21: Global error norms for x-direction gradient for various gradient methods.

levels as several of the gradient operator choices were demonstrated to satisfy 1^{st} order accuracy. Figure 21 shows the L_∞ and L_2 error norms with respect to the refinement level. First we would like to clear the peculiar behavior of the Green-Gauss method with simple and IDW face averaging. Both seem to approach a 1^{st} order convergence rate before stalling at a fixed error level. This is due to the aforementioned inconsistency as they converge, in a 1^{st} order manner, to a gradient value that is not consistent with the exact value. Note here that without a deep enough convergence study,

this issue could have been overlooked, leading to a false conclusion that these methods are 1^{st} order accurate.

The rest of the operators are all linear-exact, and consequently they all consistently exhibit 1^{st} order accuracy as it was apparent from the OOA distributions shown earlier on the test mesh. The error norms shown in Figure 21 now reveal that the Green-Gauss methods (with WTLI or LSQR face averaging) yield significantly larger errors compared to the curvilinear or the LSQR methods. Within the latter group, the LSQR compact has slightly lower error then the LSQR extended while the curvilinear method places in between.

Further drilling down on the results, we inspect errors for individual cells of various cell types as seen in Figures 22 and 23. Note that each cell type is sampled from the test mesh (Figure 11) so that their stencils are also the same type.

For regular cell types (square, equilateral triangle and right triangle), all the gradient operators are able to produce at least 1^{st} order accuracy. In fact, the square cell type stencil yields 2^{nd} order accuracy for each method. The curvilinear method produces a notably smaller error for this case. For irregular stencils, which are of greater practical interest, we start observing the familiar result of convergence stalling for the inconsistent schemes, namely the Green-Gauss method with simple or IDW face averaging.

Discarding the special case of the square stencil, the LSQR gradient operator consistently produces the lowest errors except for the cases of thin triangles and thin quadrilaterals (commonly encountered in boundary layer regions of CFD meshes). For the thin cells, the trend reverses and the LSQR method yields the largest errors while the consistent Green-Gauss methods perform the best. Note that the thin cells mentioned here were sampled near the curved boundary region of the test mesh. Whereas the Green-Gauss method exhibited mediocre performance elsewhere, its favorable behavior in the crucial boundary layer type meshes demonstrates its appeal.

The errors associated with the curvilinear method were erratic, yielding the best result for the square cells and placing among the lower error range elsewhere with two exceptions; the thin quadrilateral and the arbitrary polyhedral where it exhibited the largest errors. This suggests that a smarter logic for stencil reduction (in 2D, down-selection of 4 stencil points) needs to be developed. Otherwise, we consider this method promising, considering that it has the most compact stencil, hence the lowest computational cost.

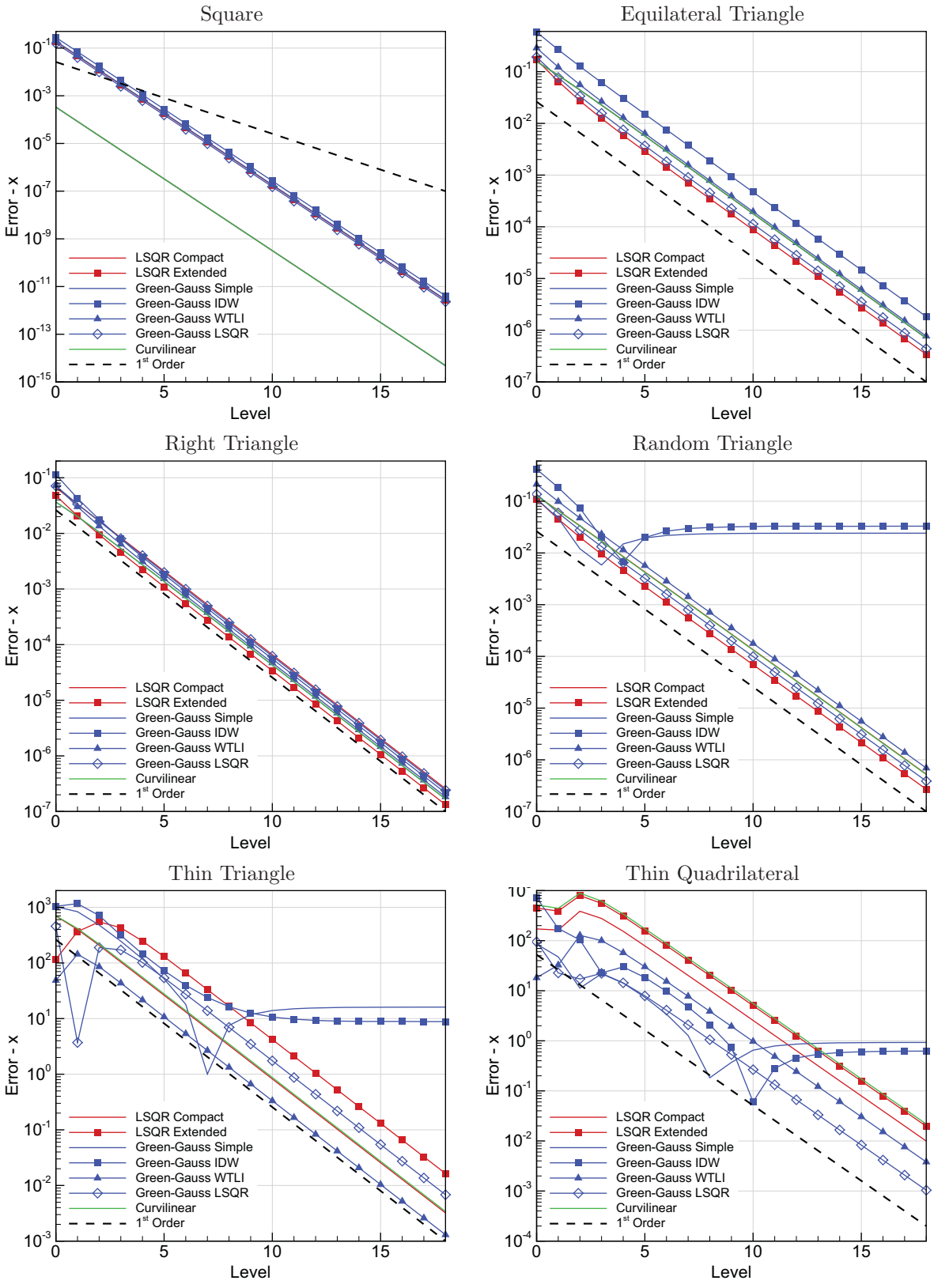


Figure 22: X-gradient error vs. levels of refinements for various cell types in the domain.

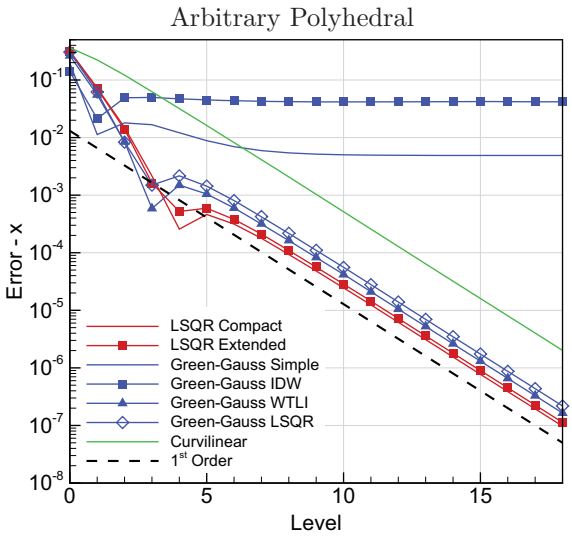


Figure 23: X-gradient error vs. levels of refinements for various cell types in the domain. (Continued)

V. Decay of an Inviscid Vortex

The previous section examined the accuracy property of various gradient operator choices. In this section, we look into some practical implications in terms of the discretization error of a CFD solver.

The cell-centered, 2^{nd} order accurate MUSCL[13] scheme of LAVA uses a piecewise linear reconstruction to cell faces as

$$u_f = u_c + \nabla u_c \cdot (\vec{r}_f - \vec{r}_c) \tag{51}$$

While the gradient operators, as explained in the previous sections, are not upwinded, the fluxes at the faces are evaluated using the AUSMPW+[14] upwind flux function.

In order to compare the influence of each gradient operator on the scheme's discretization error, a 2D inviscid standing vortex test case was adopted. The initial conditions were chosen to match those published by Pulliam[15] as

$$\begin{aligned} T &= T_\infty - \frac{V_s^2(\gamma-1)}{16 G_s \gamma \pi^2} e^{2G_s(1-r^2)} \\ \rho &= T^{\frac{1}{(\gamma-1)}} \\ u &= M_\infty - \frac{V_s}{2\pi}(y-y_0) e^{G_s(1-r^2)} \\ v &= \frac{V_s}{2\pi}(x-x_0) e^{G_s(1-r^2)} \end{aligned} \tag{52}$$

with

$$M_\infty = 0,\ T_\infty = 1,\ \text{and}\ \gamma = 1.4 \tag{53}$$

The vortex strength is taken as $V_s = 5$ and the Gaussian width scale is $G_s = 0.5$. The vortex core origin is at $(x_0, y_0) = (0, 0)$ and the radial coordinate is defines as

$$r = \sqrt{(x-x_0)^2 + (y-y_0)^2} \tag{54}$$

The initial pressure distribution for the vortex is plotted in Figure 24. Time integration was carried out using an implicit, 2^{nd} order accurate dual time stepping scheme with a step size of

$\Delta t = 0.05$ for 10000 steps. A dual iteration residual drop of at least 8 orders of magnitude was enforced for each time step (L_2 norm of the continuity equation residual was used).

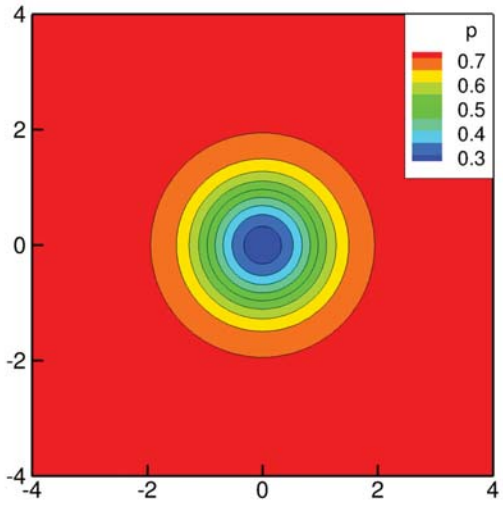


Figure 24: Initial pressure distribution for the 2D inviscid vortex.

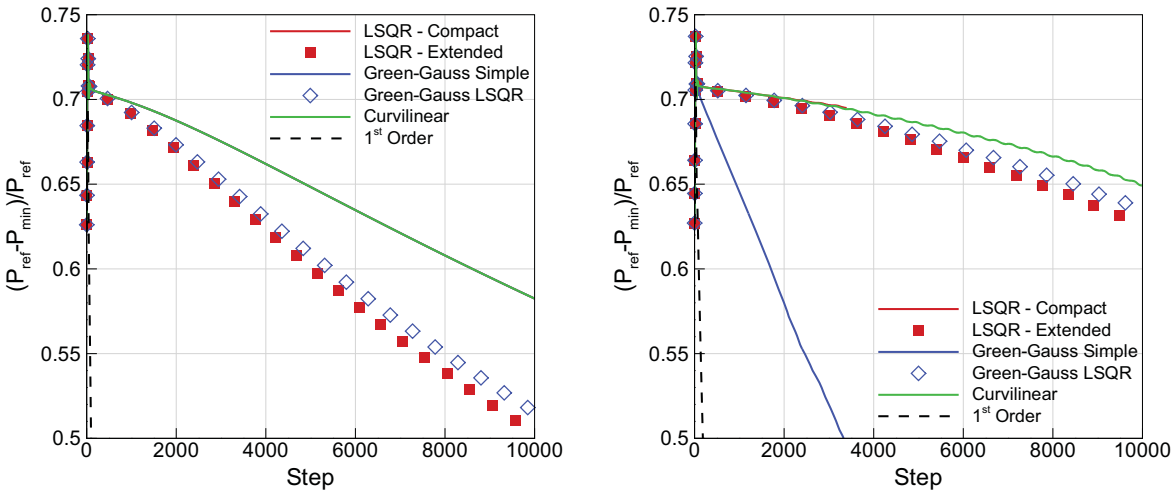
Two different meshes were utilized; a square cell type and a random triangulation using a uniform cell spacing of $\Delta x = 0.1$. Both meshes spanned a range 20 unit lengths around the vortex core. Although both meshes use the same cell spacing, the triangular mesh naturally ended up having about twice as many cells, hence, degrees of freedom.

Note that in the ideal case, the vortex is expected to stay stationary and retain its initial condition indefinitely since there is no physical mechanism to dissipate it. However, it gradually loses its strength over time due to the numerical dissipation. As an indicator of the vortex strength, the minimum pressure in the domain was tracked throughout the time integration. The test is repeated for several choices of gradient operators on both meshes, while everything else is held identical. Figure 25 shows the results for both the square and the random triangular meshes. The ideal behavior is for the quantity $(P_{ref} - P_{min})/P_{ref}$ (plotted on the vertical axis) to remain constant. The observed decay is the result of the numerical scheme's discretization error which seems to be quite sensitive to the particular gradient operator choice.

An immediate observation is that there is a stark difference between the decay rates of the 1^{st} and 2^{nd} order accurate schemes (the former assuming a constant distribution of variables within each cell, hence simply assuming a zero gradient).

On the square mesh, all the compact stencil operators (LSQR compact, Green-Gauss simple and curvilinear) yielded relatively lower decay rates. In fact, they delivered identical results. The other two gradient methods (LSQR extended and Green-Gauss LSQR) which operate on larger stencils, were comparable to each other while having a relatively higher decay rates.

On the random triangular mesh, similar observations with regards to the stencil compactness hold true. However, the inconsistent Green-Gauss simple averaging method exhibited a severely large amount of dissipation, coming closer to the 1^{st} order scheme than the family of 2^{nd} order results. Another notable issue is that the LSQR compact method, initially exhibiting a slightly lower dissipation than the curvilinear method, was unstable on this mesh and diverged at around 3500 steps. We should note here that no limiters were used in these simulations and that the robustness, while a crucial property, is not in the scope of the current paper but left for a future study.



(Square Mesh) (Triangular Mesh)

Figure 25: Dissipation of an inviscid standing vortex.

VI. Conclusions

A detailed accuracy study of gradient calculation methods for cell-centered unstructured data is presented. Necessity of the linear-exactness property for 1^{st} order gradient accuracy, and consequently a 2^{nd} order scheme, is emphasized. A straightforward, yet novel, approach utilizing local curvilinear transformation is proposed. The curvilinear method offers the most compact gradient stencil among those studied here.

Two commonly used methods, namely the Green-Gauss with either simple or IDW face averaging (neither are linear-exact), was shown to be inconsistent and 0^{th} order accurate. This was only observed for irregular meshes and was revealed through a deep refinement study. Depending on the combination of the test function and the test mesh scale, this handicap may well be overlooked if only limited levels of refinement were applied.

No clear "best" method emerged but strengths and shortcomings of the investigated methodologies for different cell types are exposed. Gradient operators with compact stencils, namely LSQR compact and curvilinear, generally exhibited lower errors. LSQR compact scheme caused stability issues for the solution of the inviscid standing vortex problem on the random triangulated mesh. The curvilinear scheme, on the other hand, had an erratic behavior for different cell types, yielding overall low error levels but exhibiting a large error for a sample arbitrary polyhedral cell. This suggests that the method could benefit from development of a smarter stencil reduction logic (to down-select 4 points from the available stencil in 2D).

The Green-Gauss method stood out with lower errors for thin triangular or quadrilateral cell types, such as those found in typical boundary layer meshes. While this sounds like a niche property, it is a very attractive quality for CFD solvers.

The current study is planned to be followed up by a systematic evaluation of robustness and consideration of a hybrid approach whereby strengths of different gradient operators are leveraged based on the local cell types.

References

[1]Diskin, B., Thomas, J., Nielsen, E., Nishiwaka, H., and White, J., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Viscous Fluxes," *AIAA Journal, Vol. 48, No. 7*, 2010, doi: 10.2514/1.44940.

[2]Diskin, B. and Thomas, J., "Comparison of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Inviscid Fluxes," *AIAA Journal, Vol. 49, No. 4*, 2011, doi: 10.2514/1.J050897.

[3]Kiris, C., Barad, M., Housman, J., Sozer, E., Brehm, C., and Moini-Yekta, S., "The LAVA Computational Fluid Dynamics Solver," *AIAA SciTech, Jan 13-17, National Harbor, Maryland*, 2014.

[4]Moini-Yekta, S., Barad, M., Sozer, E., Brehm, C., Housman, J., and Kiris, C., "Verification and Validation Studies for the LAVA CFD Solver," 31^{st} *AIAA Applied Aerodynamics Conference, June 24-27, San Diego, CA*, 2013.

[5]Moini-Yekta, S., Barad, M., Sozer, E., Brehm, C., Housman, J., and Kiris, C., "Towards Hybrid Grid Simulations of the Launch Environment," *ICCFD7-3102, July 9-13, Big Island, Hawaii*, 2012.

[6]Aftosmis, M., Gaitonde, D., and Tavares, T., "Behavior of Linear Reconstruction Techniques on Unstructured Meshes," *AIAA J., vol. 33, no. 11, pp. 2038-2049*, 1995.

[7]Mavriplis, D., "Revisiting the Least-Squares Procedure for Gradient Reconstruction on Unstructured Meshes," *AIAA Paper 2003-3986*, 2003.

[8]Shima, E., Kitamura, K., and Fujimoto, K., "New Gradient Calculation Method for MUSCL Type CFD Schemes in Arbitrary Polyhedra," 48^{th} *AIAA Aerospace Sciences Meeting, Jan 4-7, Orlando, FL*, 2010.

[9]Shima, E., Kitamura, K., and Haga, T., "Green-Gauss/Weighted-Least-Squares Hybrid Gradient Reconstruction for Arbitrary Polyhedra Unstructured Grids," *AIAA Journal, Vol. 51, No. 11*, 2013, doi: 10.2514/1.J052095.

[10]Correa, C., R., H., and K., M., "A Comparison of Gradient Estimation Methods for Volume Rendering on Unstructured Meshes," *IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 3, March*, 2011.

[11]Barth, T. and Jespersen, D., "The Design and Application of Upwind Schemes in Unstructured Meshes," 27^{th} *AIAA Aerospace Sciences Meeting, Jan 9-12, Reno, NV*, 1989.

[12]Barth, T., "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations," *NATO AGARD-R-787*, 1992.

[13]van Leer, B., "Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov's Method," *J. Comp. Phys., 87, 408-463*, 1979.

[14]Kim, K. H., Kim, C., and Rho, O.-H., "Methods for the Accurate Computations of Hypersonic Flows: I. AUSMPW+ Scheme," *Journal of Computational Physics*, Vol. 174, 2001, pp. 38–80.

[15]Pulliam, T., "High Order Accurate Finite-Difference Methods: as seen in OVERFLOW," *AIAA Journal*, June 2011, AIAA-2011-3851.