An affine reconstructed algorithm for diffusion on triangular grids using the nodal discontinuous Galerkin method

Yang Song\textsuperscript{a}, Bhuvana Srinivasan\textsuperscript{a,}\textsuperscript{*}

\textsuperscript{a}Virginia Tech, Blacksburg, VA 24060

Abstract
In this paper, an affine reconstructed discontinuous Galerkin (aRDG) method is described for solving the diffusion operator in convection-diffusion equations. The proposed numerical approach reconstructs a smooth solution in a parallelogram that is enclosed by the quadrilateral formed by two adjacent triangle elements. The interface between these two triangles is the diagonal of the enclosed parallelogram. Similar to triangles, the mapping of parallelograms from a physical domain to a reference domain is also an affine mapping. Thus, all computations can still be performed on the reference domain, which promotes efficiency in computation and storage. This reconstruction does not make assumptions on choice of polynomial basis. Reconstructed DG algorithms have previously been developed for modal implementations of the convection-diffusion equations. However, to the best of the authors’ knowledge, this is the first practical guideline that has been proposed for applying the reconstructed algorithm on a nodal discontinuous Galerkin method.

Keywords: nodal discontinuous Galerkin method, reconstruction, convection diffusion equation, computational efficiency, unstructured, triangle elements

1. Introduction
In recent years, the Discontinuous Galerkin (DG) method has been successfully applied to hyperbolic conservation laws\cite{1,2,3,4,5}. Due to its

\textsuperscript{*}Corresponding author
URL: srinbhu@vt.edu (Bhuvana Srinivasan)
compactness, high order accuracy, and versatility, the DG algorithm is fa-
vorable for applications to convection-diffusion problems,
\[
\frac{\partial u}{\partial t} + \nabla \cdot (\vec{v} u) - \nabla \cdot (D \nabla u) = s
\]
(1)

where \( u \) represents conservative variables, \( \vec{v} \) is the velocity field, \( D \) is the diffusion coefficient and \( s \) represents source terms. A significant amount of literature exists on accurate and efficient DG implementations for the convection terms.

However, solving the diffusion term in DG is non-trivial. The diffusive flux is not defined on the interface of elements as DG solution representations are only piecewise continuous. Approximating the diffusive flux as a simple arithmetic mean from both sides of the interface is not appropriate as it ignores the possible jump of the solutions. A number of numerical algorithms have been proposed in the DG community to approximate the diffusion operator with high order accuracy, for example, Douglas and Dupont [6], Arnold [7], Cockburn and Shu [8], Peraire and Persson [9], Liu and Yan [10], and others. However, all the above methods require large computational effort relative to the algorithm presented here.

In 2005, Van Leer proposed a recovery-based DG algorithm to solve the diffusion operator, where a new polynomial that is smoothly defined across two adjacent elements is recovered from the two original polynomials with order of \( P \) [11]. The new polynomial is of order \( 2P + 1 \) and is indistinguishable from the original solutions defined across two cells in a weak sense. This recovery-based method is a more natural and accurate way of calculating the diffusive flux. This algorithm is further developed and applied on a two dimensional structured mesh [12]. However, the accuracy of the scheme is affected not only by the diffusive part but also the hyperbolic parts in the system. In fact, the order of accuracy is determined by the least accurate component in the system. Hence, a highly accurate diffusion solver does not increase the overall accuracy of the scheme in solving convection-diffusion problems. Also, constructing an appropriate basis function defined on the combination of two elements is an involved process. More recently, a reconstruction-based DG algorithm using Taylor basis functions is proposed in [13]. In this algorithm, similar to the recovery DG algorithm, a smooth solution is reconstructed across two adjacent elements. Unlike the recovery DG algorithm, the reconstructed solution has the same polynomial order as the original solutions and is not indistinguishable from the original solutions in a
weak sense. The reconstruction-based DG algorithm can solve the diffusion term with the same order of accuracy as the hyperbolic solver, making the scheme computationally efficient. Also, since the reconstructed polynomial has the same order as the underlying DG solution, it is not necessary to carefully construct a basis function that is well conditioned across two elements. The choice of Taylor basis simplifies the reconstruction process significantly although it suffers from ill-conditioning.

Storage management and computational efficiency are playing increasingly significant roles in modern computational software especially for large-scale high fidelity simulations. Conventional DG algorithms solve hyperbolic terms on a reference element, then transform the solution to physical elements. There are advantages with respect to computational efficiency and memory management if the reconstructed DG algorithm could be solved on a reference domain. Depending on the shape of the elements (triangle, quadrilateral, etc.), different memory requirements are dictated by the need to store the transformation Jacobians. Without careful treatment, this could result in higher cost of either memory or computation for recovery or reconstruction methods. Thus, solving the diffusion operator using DG in a stable, efficient, and accurate manner is still an open question. It is worth mentioning that recent developments have been made in the the reconstructed DG algorithm to couple the direct DG method [14] with a first-order hyperbolic system (FOHS) [15]. However, the primary focus of this paper is on memory and computational efficiency while solving the diffusion term. What is more, there is no guideline currently available on how to apply the reconstruction technique directly on a nodal DG method. This work proposes a new reconstructed DG method that is both storage- and computationally-efficient, and couples naturally with the widely-used nodal DG algorithm described by Hesthaven and Warburton [16]. This algorithm ensures that the reconstruction is performed on affine elements, where the transformation Jacobian is constant, thus ensuring efficiency.

2. Governing equation and discretization

2.1. Governing equation

This work focuses on solving the diffusion operator using a reconstructed DG method. The governing equation is the diffusion equation,

\[
\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u) \tag{2}
\]
where $D$ is the diffusion coefficient. Without losing generality, $D$ is assumed to be a positive constant in space and time.

### 2.2. Discretization

In DG, the numerical solution can be expressed as a direct sum of local piecewise polynomials as

$$ u(x, t) \simeq u_h(x, t) = \bigoplus_{k=1}^{K} u_h^k(x, t). $$

Replacing $u$ in equation 2 with $u_h$ and multiplying a test function $\phi_i$ and integrating over non-overlapping cells $\Omega_k$, where $k = 1, ..., K$, will give a typical DG treatment,

$$ \int_{\Omega_k} \left( \frac{\partial u_h^k}{\partial t} \phi_i^k - D(\nabla^2 u_h^k)\phi_i^k \right) d\Omega = 0. \quad (4) $$

A DG scheme can be obtained by integrating the second term in equation 4 by parts,

$$ \int_{\Omega_k} \left( \frac{\partial u_h^k}{\partial t} \phi_i^k + D\nabla u_h^k \cdot \nabla \phi_i^k \right) d\Omega - D \int_{\partial\Omega_k} (\phi_i^k \hat{n} \cdot \nabla \tilde{u}^k) d\partial\Omega = 0. \quad (5) $$

Since $u_h^k$ is discontinuous at the cell interface, the diffusive flux $\nabla u_h^k$ in the surface integration is not directly available on the boundary of $\Omega_k$ and cannot be treated as an advective flux, thus it cannot be simply approximated by a Riemann flux solver [17, 13]. Hence, $\nabla u_h^k$ is replaced by a reconstructed solution $\nabla \tilde{u}^k$ that is smoothly defined at the interface. The details of this reconstruction algorithm will be discussed in section 4.4.

### 3. Nodal discontinuous Galerkin method

Following the nodal DG algorithm from [16], the test function and basis function are chosen to be Lagrange polynomials, $\ell_i$. For the sake of simplicity, the subscript $h$ is dropped from now on. Then equation 5 can be rewritten as

$$ \int_{\Omega_k} \left( \frac{\partial u^k}{\partial t} \ell_i^k + D\nabla u^k \cdot \nabla \ell_i^k \right) d\Omega - D \int_{\partial\Omega_k} (\ell_i^k \hat{n} \cdot \nabla \tilde{u}^k) d\partial\Omega = 0. \quad (6) $$
Solutions on Legendre-Gauss–Lobatto (LGL) nodes [18] are chosen to be the expansion coefficients. Assume the polynomial order is $P$ and $x^k_j$ are the LGL nodes defined on $\Omega_k$, then the solution in $\Omega_k$ can be represented as the nodal expansion

$$u^k(x, t) = \sum_{j=1}^{N_p} u^k(x^k_j, t) \ell^k_j(x),$$

(7)

where $N_p = (P+1)(P+2)/2$ is the total number of nodes or unknowns in $\Omega_k$ and $u^k = [u^k(x^k_1, t), \ldots, u^k(x^k_{N_p}, t)]^T$. The modal expansion of the solution is introduced,

$$u^k(x, t) = \sum_{j=1}^{N_p} \hat{u}^k_j(t) \psi^k_j(x),$$

(8)

where $\hat{u}^k = [\hat{u}^k_1(t), \ldots, \hat{u}^k_{N_p}(t)]^T$ are the modal expansion coefficients and $\psi^k_j(x)$ are the orthonormal modal polynomial basis in $\Omega_k$. For more details of how to construct $\psi_j$ in triangular element, please refer to [16]. The Vandermonde matrix $\Psi^k$ is defined as

$$\Psi^k_{ij} = \psi^k_j(x_i),$$

(9)

such that

$$u^k = \Psi^k \hat{u}^k.$$

(10)

In the nodal DG method [16], all computations can be performed on the reference triangle $I = \{(r, s) | (r, s) \geq -1; r + s \leq 0\}$. Since the mapping for triangular elements is an affine transformation [19, 20], the Jacobians of this mapping are constant in a triangle. This mapping is shown in Figure 1 and described in equations 11 and 12,

$$x = -\frac{r + s}{2} v^1 + \frac{r + 1}{2} v^2 + \frac{s + 1}{2} v^3,$$

(11)

$$(x_r, y_r) = \frac{v^2 - v^1}{2}, (x_s, y_s) = \frac{v^3 - v^1}{2}.$$

(12)

The Jacobians of this mapping are described in equations 13 and 14,

$$r_x = \frac{y_s}{J}, r_y = -\frac{x_s}{J}, s_x = -\frac{y_r}{J}, s_y = \frac{x_r}{J},$$

(13)

$$J = x_r y_s - x_s y_r.$$

(14)
For the remainder of this paper, any variable or matrix without the element index superscript $k$ is defined on $I$. Now, equation 6 can be written as

$$\frac{\partial u^k}{\partial t} + D \left( M^{k-1} S^k \cdot \nabla u^k \right) - D \sum_{f=1}^{3} \text{LIFT}_f^k \left( \hat{n}_f^k \cdot \nabla \tilde{u}_f^k \right) = 0,$$

(15)

where the mass matrix and stiffness matrix are defined as

$$M_{ij}^k = \int_{\Omega_k} \ell_i^k \ell_j^k d\Omega = J_k \int_I \ell_i \ell_j dI = J_k M,$$

(16)

$$S_{ij}^k = \int_{\Omega_k} \ell_i^k \nabla \ell_j^k d\Omega$$

$$= J_k \int_I \ell_i \begin{bmatrix} r_x & s_x \\ r_y & s_y \end{bmatrix}^k \begin{bmatrix} \frac{\partial \ell_j}{\partial r} \\ \frac{\partial \ell_j}{\partial s} \end{bmatrix} dI$$

(17)

$$= J_k r_x^k \int_I \ell_i \nabla \ell_j dI$$

$$= J_k r_x^k S,$$

respectively. Only reference mass, stiffness matrices, and geometric factors need to be stored. The lift operator is defined as

$$\text{LIFT}_f^k \left( \hat{n}_f^k \cdot \nabla \tilde{u}_f^k \right) = M^{k-1} \int_{\partial \Omega_k}^{f} \ell_i^k \hat{n}_f^k \cdot \nabla \tilde{u}_f^k d\partial \Omega.$$  

(18)

Here, the surface integration cannot be easily transformed to the reference domain, as the reconstructed element is not guaranteed to share the same mapping transformation of triangular elements as described in equation 11.
and 12. This means that this surface integration needs to be precalculated and stored on all elements, which is computationally inefficient. This will be discussed in the following section.

Now, equation 15 can be written as

\[
\frac{\partial u^k}{\partial t} + D \left( M^{-1} S^T \cdot \nabla u^k \right) - D \sum_{j=1}^{3} \text{LIFT}_j^k \left( \hat{n}_j \cdot \nabla \tilde{u}_j \right) = 0.
\] (19)

If \( D \) is not a constant, but a function of space and time, and also not isotropic (i.e. \( D = (D_x, D_y) \)) then equation 19 can be rewritten as

\[
\frac{\partial u^k}{\partial t} + (M^{-1} S^T \cdot D \nabla u^k) - \sum_{j=1}^{3} \text{LIFT}_j^k \left( \hat{n}_j \cdot D \nabla \tilde{u}_j \right) = 0.
\] (20)

An alternative way of calculating the reconstructed solution for the surface term is,

\[
\frac{\partial u^k}{\partial t} + (M^{-1} S^T \cdot D \nabla u^k) - \sum_{j=1}^{3} \text{LIFT}_j^k \left( \hat{n}_j \cdot \hat{D} \nabla \tilde{u}_j \right) = 0.
\] (21)

Test results indicate minimal differences between the two reconstructed formulations described in equations 20 and 21.

4. Affine reconstructed algorithm

4.1. Non-affine mapping in quadrilaterals

To obtain a reconstructed solution that is smoothly defined at the interface, the reconstruction needs to be performed on the combination of two triangles, which is a quadrilateral. Hence, it is important to consider the mapping transformation between a quadrilateral element \( \Omega^q \) and a reference square element \( I^q = \{ R = (R, S)| -1 \leq (R, S) \leq 1 \} \). Here superscript \( q \) refers to quadrilateral. This mapping is described in equation 22

\[
X = \frac{1}{4}(1-R)(1-S)v^1 + \frac{1}{4}(1+R)(1-S)v^2 + \frac{1}{4}(1+R)(1+S)v^3 + \frac{1}{4}(1-R)(1+S)v^4,
\] (22)

which is not always an affine mapping. Thus, assuming \( I(\mathbf{r}) \) for the reference triangle and \( I^q(\mathbf{R}) \) for the reference square element share the same coordinate
system, then $\Omega(\mathbf{x})$ and $\Omega^q(\mathbf{X})$ are not in the same physical coordinate system. To demonstrate this, $P4$ ($Pn$ denotes polynomial order $n$) tensor product nodal points in $I^q$, as shown in Figure 2-a, are mapped to an arbitrary quadrilateral element $\Omega^q_1$ through equation 22 as shown in Figure 2-b. Note that the nodes on the diagonal of $\Omega^q_1$ are curved and do not represent the straight interface between the two triangles. Figure 2-c provides another example where the diagonal of the quadrilateral in $\Omega^q_2$ is not curved but the nodes on diagonal are not symmetric. This shows that the diagonal of $\Omega^q$ does not represent the interface between two triangular elements. This makes the reconstruction unfavorable as the surface integration described in equation 18 can then only be evaluated on the physical domain, which is inefficient for both computation and storage management.

![Figure 2: Mapping transformation in quadrilaterals. (a) tensor product of LGL nodes on $I^q$; (b) transformation from $I^q$ to $\Omega^q_1$ that has a curved diagonal; (c) transformation from $I^q$ to $\Omega^q_2$ that has a straight diagonal but with asymmetric nodes along the diagonal.](image)

4.2. Enclosed parallelogram

The mapping from equation 22 can be reduced to affine mapping when the physical quadrilateral $\Omega^q$ is a parallelogram, which is shown in Figure 3. For any quadrilateral $\Omega^q$ formed by two adjacent triangles $\Omega_1$ and $\Omega_2$, one can always find an enclosed parallelogram $\Omega^p$ that shares the same diagonal with $\Omega^q$, which is also the interface between two triangles. This is demonstrated in Figure 4. Once $\Omega^p$ is found, the solution from $\Omega_1$ and $\Omega_2$ is projected onto the two smaller triangles $\Omega^p_1$ and $\Omega^p_2$ that form the parallelogram. Then the solution from these two triangles can be used to reconstruct a polynomial $\tilde{u}$.
that is continuously defined in the parallelogram. This reconstruction can be done in the logical element $I^q = I + I^{-1}$, where $I^{-1} = \{r = (r, s)| (r, s) \leq 1; r + s \geq 0\}$, with solution of $\Omega'_1$ projected on $I$ and solution $\Omega'_2$ projected on $I^{-1}$, when the shared interface in $\Omega'_1$ and $\Omega'_2$ is the hypotenuse in $I$ and $I^q$. This is because the nodes on the diagonal of $\Omega^p$ are located exactly at the nodes on the interface of $\Omega'_1$ and $\Omega'_2$. In other words, the mapping transformation between $\Omega^p$ and $I^q$ is identical to the mapping transformation between $\Omega'$ and $I$. The formula for the projection is provided here but the reconstruction procedure will be discussed in detail in section 4.4. Once the new vertices are found for $\Omega'_1$ and $\Omega'_2$, one can easily construct a projection Vandermonde matrix $\mathcal{V}_p$ that projects the modal expansion coefficients $\hat{\mathbf{u}}$ on $\Omega$ to the nodal solution $\mathbf{u}'$ on $\Omega'$, as described in equation (23):

$$\mathbf{u}' = \mathcal{V}_p \hat{\mathbf{u}}.$$  

(23)

Now equation (18) can be rewritten as

$$\text{LIFT}_f (\hat{n}_f^k \cdot \nabla \hat{u}_f^k) = M^{k-1} \int_{\partial \Omega'_k} \ell_i^k \hat{n}_f^k \cdot \nabla \hat{u}_f^k d\partial \Omega$$

$$= J^k M^{-1} \left( \int_{\partial \Omega'_k} \ell_i^k \ell_p^f d\partial \Omega \right) \hat{n}_f^k \cdot \nabla \hat{u}_f^k$$

$$= \frac{J_f}{J_k} M^{-1} \left( \int_{\partial I} \ell_i \ell_p^f d\partial I \right) \hat{n}_f^k \cdot \nabla \hat{u}_f^k$$

$$= \frac{J_f}{J_k} \text{LIFT}_f (\hat{n}_f^k \cdot \nabla \hat{u}_f^k),$$

(24)
where $\tilde{\ell}_{k,f}$ is the basis function defined on the diagonal of the reconstructed enclosed parallelogram element, which is the same as the basis function defined on the edge of the triangle. $\ell_{f}$ is the basis function defined on edge $f$ in $I$. $J_{f}$ is the transformation Jacobian along edge $f$ of $\Omega_{k}$. $J_{f}$ can also be seen as the ratio between the length of $\Omega_{k}^{f}$ and $I^{f}$. $\nabla \tilde{u}_{k}^{f}(x)$ are $N_{fp} \times 1$ arrays of the gradients of the reconstructed nodal solutions on the three edges of element $\Omega_{k}$. $N_{fp} = P + 1$ is the total number of nodes on one edge. $\nabla \tilde{u}_{j}^{k}(x)$ can be calculated as,

$$\nabla \tilde{u}(x)_{f}^{k} = \begin{bmatrix} r'_{x} & s'_{x} \\ r'_{y} & s'_{y} \end{bmatrix}^{k,f} \begin{bmatrix} \frac{\partial \tilde{u}^{k}_{f}}{\partial r} \\ \frac{\partial \tilde{u}^{k}_{f}}{\partial s} \end{bmatrix},$$

(25)

where the geometric factors are constant in a parallelogram, which requires much less storage compared to quadrilateral elements. Equation 19 now can be written as,

$$\frac{\partial u^{k}}{\partial t} + D \left( M^{-1} S^{T} \cdot \nabla u^{k} \right) - D \sum_{f=1}^{3} \frac{J_{f}^{k}}{J_{f}^{LIFT}} \left( \hat{n}_{f}^{k} \cdot \nabla \tilde{u}_{f}^{k} \right) = 0,$$

(26)

in which all matrices are defined in $I$. This form has advantages for numerical implementation as the matrices can be precalculated while also using minimal storage.

4.3. Reordering nodes in the reference domain $(r,s)$

Every edge of $\Omega_{k}$ that has a neighboring element will need to be the hypotenuse in $I$ for the reconstruction. An immediate solution to this would be changing the ordering of the vertices $[v^{1}, v^{2}, v^{3}]$ in equation 11 to change the ordering of the nodes in $\Omega_{k}$, so that the target edge of $\Omega_{k}$ can be remapped to the hypotenuse of $I$. However, this needs to be done for two other edges of each element, and requires either large computational effort if it is calculated during run-time or duplicated large storage if it is precalculated. This breaks the simplicity and efficiency of this scheme. A more efficient way to solve this is to change the ordering of nodes in $I$ to map its hypotenuse to the target edge in $\Omega$, without changing the ordering of nodes in $\Omega$.

There are three orderings of $(r, s)$ in $I$ that can be used for performing the aRDG treatment on three edges of $\Omega$. Accordingly, three Vandermonde matrices $[V_{r1}, V_{r2}, V_{r3}]$ can be generated to project the original nodal solutions
\[ \hat{u}_f = V_{r_f}^{-1} u. \]  

(27)

Combining equations 10, 23, and 27, the modal expansion coefficient \( \hat{u}'_f \) is calculated in \( \Omega' \), where the edge \( f \) in \( \Omega \) (or \( \Omega' \)) is the hypotenuse in \( I \), from the nodal solution \( u \) in \( \Omega \), as

\[ \hat{u}'_f = V^{-1} V_p V_{r_f}^{-1} u. \]  

(28)

This expression can also be precomputed using any symbolic solver.

4.4. Reconstruction

The components necessary for the reconstruction have been described to this point. The reconstruction process is performed using the modal solution, which is computed from the Vandermonde matrix and the nodal solution in the two smaller triangles that form the enclosed parallelogram. Similar to the recovery [11] and the reconstruction [13] methods, a new polynomial is
constructed that is smoothly defined across two adjacent cells,

\[
\int_{\Omega_1} \sum_{r=1}^{M_p} \tilde{u}_r \tilde{\psi}_r \psi_m d\Omega = \int_{\Omega_1} \sum_{r=1}^{N_p} \hat{u}_r^1 \psi_r \psi_m d\Omega, \\
\int_{\Omega_2} \sum_{r=1}^{M_p} \tilde{u}_r \tilde{\psi}_r \psi_m d\Omega = \int_{\Omega_2} \sum_{r=1}^{N_p} \hat{u}_r^2 \psi_r \psi_m d\Omega,
\]

where \( N_p = (P + 1)(P + 2)/2 \) is the number of modes in a triangle and \( M_p \) is the number of modes in the parallelogram, respectively. \( \hat{u}_r^1 \) and \( \hat{u}_r^2 \) are the modal solutions on the two smaller triangles \( \Omega_1 \) and \( \Omega_2 \). \( \tilde{u}_r \) is the reconstructed modal solution on the parallelogram. Using tensor product of Gauss-Legendre polynomial basis for the parallelogram, \( M_p = (P+1)(P+1) \).

This system has \( 2N_p \) equations and \( M_p \) unknowns. This affine reconstruction method solves \( (P + 1)^2 \) unknowns from \( (P + 1)(P + 2) \) equations which differs from the \( (P+1)(P+2)/2 \) unknowns (potentially with additional higher order correction terms) in the work of [13]. This system is solved using a least squares method described in [13].

5. Results

Numerical tests are performed on multiple linear and non-linear scalar equations with diffusion and the Navier-Stokes equations using \( P1, P2, \) and \( P3 \) nodal DG algorithms with the aRDG method. Three types of grids, as shown in Figure [5], are tested. Grid-a and -b are \( 0 \leq x \leq 10 \). Grid-b has the bottom-left corner moved to \((1.5, -3.5)\), the top-right corner moved to \((11.5, 6.5)\), and the center moved to \((6.5, 1.5)\). In grid-a, each quadrilateral combined by two adjacent triangles is a parallelogram, thus no error associated with area truncation will be generated through the reconstruction process. In grid-b, large area truncation will occur on the diagonals of the domain, where the combination of two adjacent triangles forms a larger triangle with a larger area than the enclosed parallelogram on which the reconstruction is performed. In grid-c, the bottom-left and top-right corners are moved so that larger area truncation to obtain an enclosed parallelogram for reconstruction will occur along the top-left, top-right, and bottom-right half of the diagonals. However, the size of each element is the same even though the shape is different. Among the four sections of the diagonals in grid c, the top-right section has the largest truncated area when obtaining
an enclosed parallelogram for reconstruction. Convergence studies are performed on a series of systematic refinements of these three grids. Series of grid- \(a\) has 32, 128, 512, 2048, and 8192 elements, while series of grid- \(b\) and -c have 16, 64, 256, 1024, 4096 elements.

Figure 5: Three types of grids used in the tests.

In this section, the global \(L_2\) and \(L\infty\) norms of the error are calculated as follows,

\[
L_2 = \sqrt{\frac{\sum_{k=1}^{K} \int_{\Omega_k} [u^k - u_e]^2 \, d\Omega}{\sum_{k=1}^{K} |\Omega_k|}},
\]

(30)

\[
L_{\text{inf}} = \max_{k=1}^{K} \frac{\int_{\Omega_k} |u^k - u_e| \, d\Omega}{|\Omega_k|},
\]

(31)

where \(u_e\) is the analytical solution. It is important to point out that the errors calculated in this section contain both the spatial and temporal discretization errors. Based on [21], the error norms are,

\[
\|\varepsilon_{h_t}^h\| = g_x h_x^\phi + g_t h_t^\phi
\]

(32)

where \(g_x\) and \(g_t\) are constants. \(h_x\) is the spatial grid size and \(h_t\) is time-step size. For all the simulations presented in this section, the five-stage fourth-order Runge-Kutta scheme [22] is used. The time step \(h_t\) is calculated from the most restrictive mesh refinement level and is fixed for all meshes. When \(h_t\) is fixed, equation (32) becomes

\[
\|\varepsilon_{h_t}^h\| = g_x h_x^\phi + \phi,
\]

(33)
where $\phi$ is constant and has been verified to be negligible compared to the spatial errors in all the calculations in this section. Thus, the convergence rates presented in this section represent the observed spatial order of accuracy.

5.1. Diffusion equation

The diffusion equation described in equation (2) is solved on the three grids presented in Figure 5. At $t = -D_0/D$, a solute of mass $M = 1$ is loaded at $(x_0, y_0)$, where $(x_0, y_0) = (5, 0)$ for grid-a and -b, and $(x_0, y_0) = (6.5, 1.5)$ for grid-c. The analytical solution is provided as

$$u_e = \left( \frac{M}{4\pi(Dt + D_0)} \right) e^{-\frac{(x-x_0)^2+(y-y_0)^2}{4(Dt+D_0)}},$$

where $D = 1$, and $D_0$ is set to be 2 to make it numerically feasible at $t = 0$. This reconstruction follows equation (19) as the diffusion coefficient is a constant. The initial condition at $t = 0$ and final solution of $t = 0.5$ are presented in Figure 6.

Results of the convergence study are presented in Figure 7. Both the convergence rates of the $L_2$ and $L_\infty$ of errors for all three types of grids are close to the formal order of accuracy $P = P + 1$ for $P1$, $P2$, and $P3$ tests. The fact that convergence lines of grid-a, -b, and -c are close to each other also indicates that the area truncation in the aRDG process has minor impact on the accuracy of the scheme. When two triangles form a parallelogram, the density of degrees of freedom of the reconstructed solution remains the same. When the enclosed parallelogram truncates a large area from the original adjacent triangles that form a quadrilateral, the density of degrees of freedom in the enclosed parallelogram is increased, which could compensate for errors associated with the area truncation.

5.2. Scalar advection-diffusion equation

In order to test how well the aRDG diffusion solver couples with the well-benchmarked NDG hyperbolic solver, this test focuses on the scalar advection-diffusion equation,

$$\frac{\partial u}{\partial t} + \vec{a} \cdot \nabla u - D\nabla^2 u = 0.$$  

The analytical solution is given by,

$$u_e = \left( \frac{M}{4\pi(Dt + D_0)} \right) e^{-\frac{(x-axt-x_0)^2+(y-ayt-y_0)^2}{4(Dt+D_0)}},$$
Figure 6: Initial condition at \( t = 0 \) and final solution at \( t = 0.5 \) for the diffusion test. \( P3 \) test on grid-\( b \) with 4096 elements is presented here.

Figure 7: Convergence tests of the diffusion equation on three types of grids (Figure 5) using (a) \( P1 \), (b) \( P2 \), and (c) \( P3 \) NDG algorithms. Formal orders of accuracy are indicated by the slopes with magenta lines.

Similar to the diffusion test, equation 19 is applied for the reconstruction of the diffusion term here. A solute of mass is loaded at \((x_0, y_0)\) at \( t = -D_0/D \), with \( D = 1 \) and \( D_0 = 2 \). However, \((x_0, y_0)\) is set to be \((4, -1.0)\) for all three types of grids (Figure 5), and a constant advection speed \( \vec{a} = (6, 6) \) is chosen so that the diffusive mass is traveling along the diagonal of the domain where truncation of area occurs in aRDG for grid-\( b \) and -\( c \). This way, the \( L_\infty \) of the error captures the error associated with area truncation in aRDG, if any.

The initial condition at \( t = 0 \) and the final solution at \( t = 0.5 \) are pre-
sented in Figure 8. Convergence tests are shown in Figure 9. Similar to the pure diffusion test case, the optimal convergence is achieved for all types of meshes and polynomial orders that are tested. Again, the convergence lines for all three grids are close to each other.

Figure 8: Initial condition at $t=0$ and final solution at $t=0.5$ for the advection-diffusion test. $P3$ test on grid-$b$ with 4096 elements is shown here.

Figure 9: Convergence tests of advection-diffusion equation on three types of grids (Figure 5) using (a) $P1$, (b) $P2$, and (c) $P3$ NDG algorithms. Formal orders of accuracy are indicated by the slopes with magenta lines.
5.3. Convection-diffusion equation with non-constant coefficients

In order to test the robustness of the aRDG scheme on non-linear equations, a scalar convection-diffusion equation with spatially- and temporally-varying coefficients is employed here,

\[
\frac{\partial C}{\partial t} + \frac{1}{2} \frac{\partial}{\partial x} \left( a_{0x} C \right) + \frac{1}{2} \frac{\partial}{\partial y} \left( a_{0y} C \right) - \frac{\partial}{\partial x} \left( D_{0x} C \frac{\partial C}{\partial x} \right) - \frac{\partial}{\partial y} \left( D_{0y} C \frac{\partial C}{\partial y} \right) = S_{\text{MMS}},
\]

where \((a_{0x}, a_{0y})\) and \((D_{0x}, D_{0y})\) are constants. The advection and diffusion coefficients are non-constant and do not assume isotropy. Equation 20 is applied here for the reconstruction of the diffusion terms. The analytical solution is constructed by method of manufactured solutions (MMS) 21, a standard method used for code verification.

The results of the convergence tests are presented in Figure 10. The convergence rates agree with the theoretical rates except for \(P_2\), where the observed rate is slightly lower than the theoretical rates. This behavior is consistent with previous results 22. Similar to the linear test cases presented, no significant difference is found between the results on different grids, which indicates that the truncation of the area to obtain an enclosed parallelogram for reconstruction does not introduce noticeable error into this system.

![Figure 10: Convergence tests of scalar convection-diffusion equation with spatially- and temporally-varying coefficients on three types of grids (Figure 5) using (a) \(P_1\), (b) \(P_2\), and (c) \(P_3\) NDG algorithms. Formal orders of accuracy are indicated by the slopes with magenta lines.](image-url)
5.4. Shear diffusion equation with non-constant coefficients

Tests are performed on three types of grids (described in Figure 5) using the shear term in the diffusion equation to further benchmark the robustness of aRDG algorithm. Following the work of [24], the shear diffusion equation is described as,

\[
\frac{\partial C}{\partial t} - \frac{\partial}{\partial x} \left( D_0 C \frac{\partial C}{\partial x} \right) - \frac{\partial}{\partial y} \left( D_0 C \frac{\partial C}{\partial y} \right) - \theta D_0 \left[ \frac{\partial}{\partial x} \left( C \frac{\partial C}{\partial y} \right) + \frac{\partial}{\partial y} \left( C \frac{\partial C}{\partial x} \right) \right] = S_{MMS},
\]

where \( \theta = \frac{1}{6} \). Equation 20 is applied here for the reconstruction of the diffusion term, and the convergence results are presented in Figure 11. In this study, noticeable differences in the convergence errors from three types of grids can be observed on \( P1 \) and \( P2 \) tests. Convergences rates agree well with theory, except in \( P2 \) tests, where the convergence rates on grid-\( b \) and grid-\( c \) are slower than the theoretical rate. The accuracy of aRDG appears to be more sensitive to area truncation necessary to obtain the enclosed parallelogram for \( P2 \) shear diffusion problems. However, no significant difference can be observed on different grids for \( P3 \) tests, and the computed convergence rates successfully predict the theory.

![Figure 11](image.png)

Figure 11: Convergence tests of scalar shear-diffusion equation with spatial and temporal varying coefficients on three types of grids (Figure 5) using (a) \( P1 \), (b) \( P2 \), and (c) \( P3 \) NDG algorithms. Formal orders of accuracy are indicated by the slopes with magenta lines.

The algorithms presented here and in [24] exclusively use face neighbors of the element to perform the reconstruction and recovery, respectively. For complete consistency with accurately resolving the shear term in the diffusion
equation, particularly as the shear term becomes significant, it is necessary to account for all face and vertex neighbors of the elements. However, a practical implementation including all vertex neighbors while maintaining computational and storage efficiency is non-trivial for unstructured grids and is a subject of future work. The likely reason that the shear term here still produces sufficient order of accuracy is due to (i) the normal stresses being dominant as is the case in most physical systems and (ii) the fourth-order Runge-Kutta time-integration scheme sufficiently resolving the cross derivatives over the five stages for the problems tested.

5.5. Navier-Stokes equations

This section applies the aRDG algorithm to the compressible Navier-Stokes equations,

\[
\frac{\partial Q}{\partial t} + \frac{\partial F_i}{\partial x_i} + \frac{\partial G_i}{\partial x_i} = 0, \quad i = 1, \ldots, N_d, \tag{39}
\]

where

\[
Q = \begin{pmatrix}
\rho \\
\rho u_j \\
\epsilon
\end{pmatrix}, \quad F_i = \begin{pmatrix}
\rho u_i \\
\rho u_i u_j + p \delta_{ij} \\
(\epsilon + p) u_i
\end{pmatrix}, \quad G_i = \begin{pmatrix}
0 \\
-\Pi_{ij} \\
-u_j \Pi_{ij} + q_i
\end{pmatrix}, \tag{40}
\]

and the viscous stress tensor \( \Pi \) and heat flux \( q \) are given by

\[
\Pi_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \nabla \cdot \mathbf{u} \delta_{ij}, \tag{41}
\]

\[
q_i = -\kappa \frac{\partial T}{\partial x_i}. \tag{42}
\]

The molecular viscosity \( \mu \) is calculated through Sutherland’s law \[25\] and thermal conductivity \( \kappa \) is calculated as

\[
\kappa = \frac{C_p \mu}{Pr}, \tag{43}
\]

where the Prandtl number \( Pr \) is 0.7.

Two sets of tests are performed. The first one is a code verification test and the second one is a model validation test.
5.5.1. Method of Manufactured Solutions (MMS)

Code verification is performed on grid-b (Figure 5) using MMS. Lax-Friedrichs [26] flux is applied here for the hyperbolic terms. According to [16], the optimal order of accuracy of the NDG algorithm for a system is $P + 1/2$, when a general monotone flux is used. The results are presented in Figure 12. The observed orders of accuracy for all three variables in $P1$ tests are slightly higher than the optimal rate. Results of $P2$ and $P3$ tests show good agreement with theory.

![Figure 12: Convergence tests of compressible Navier-Stokes equations on grid-b (Figure 5) using (a) $P1$, (b) $P2$, and (c) $P3$ NDG algorithms. Convergence rates for mass, momentum, and total energy are presented. Formal orders of accuracy are indicated by the slopes with magenta lines.](image)

5.5.2. Flow over cylinder

Model validation is performed on an subsonic flow over cylinder case with $Re = 40$. A circular cylinder with a diameter of $D$ is placed at the center of a domain of size $32D \times 16D$. The computed Mach number is plotted in Figure 13 with streamlines indicating the recirculation. The drag coefficient and the length of the recirculation region are calculated and presented in Table 1, which agree well with [27].

6. Conclusion

In this paper, an affine reconstructed discontinuous Galerkin method has been described to solve the diffusion operator accurately and efficiently on unstructured grids of triangles. A practical guideline on how to apply this
Table 1: Drag coefficient and length of recirculation for subsonic flow over circular cylinder with $Re = 40$

| $Re = 40$ | Drag coefficient | Length of recirculation |
|-----------|-------------------|-------------------------|
| Current study | 1.47             | 2.26D                   |
| Tseng and Ferziger [27] | 1.53             | 2.21D                   |

Figure 13: Mach number plot with streamlines of subsonic flow over circular cylinder with $Re = 40$

algorithm to the nodal discontinuous Galerkin method has been provided. All computations can be done on the reference domain, which couples well with the notable nodal discontinuous Galerkin scheme from [16]. Benchmark tests are performed on three types of grids with different refinement levels using $P_1$, $P_2$, and $P_3$ NDG schemes for linear and non-linear scalar equations with diffusion and the Navier-Stokes equations. The observed orders of accuracy generally agree with the formal orders of accuracy for all tests. Some $P_2$ results have a $\mathcal{O}(h^2)$ convergence as described in [23] which shows that the optimal order of accuracy of DG for diffusion is $\mathcal{O}(h^{P+1})$ for odd $P$ and $\mathcal{O}(h^P)$ for even $P$. By maintaining the same polynomial order for the described reconstruction method, the density of nodes in the reconstructed element on the physical domain is not decreased, which means discretization error is not increasing through this reconstruction. When two triangles form a parallelogram, the density of degrees of freedom of the reconstructed solution remains the same. When the enclosed parallelogram truncates a large area
from the original adjacent triangles that form a quadrilateral, the density of degrees of freedom in the enclosed parallelogram is increased, which could compensate for errors associated with the area truncation. This may explain why the errors associated with all three types of grids are very close to each other for most of the tests presented, except for when the shear term is included in the diffusion. It is also straightforward to extend the aRDG algorithm to other types of elements as long as an enclosed parallelogram can be found in adjacent elements. Future work will focus on extending the aRDG algorithm to three dimensional unstructured grids.

**Funding Sources**

This work was supported by the US Department of Energy under grant number DE-SC0016515.

The author acknowledges Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this work. URL: http://www.arc.vt.edu

**References**

References

[1] F. Bassi, S. Rebay, High–order accurate discontinuous finite element solution of the 2D Euler equations, Journal of computational physics 138 (2) (1997) 251–285.

[2] B. Cockburn, C.-W. Shu, The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems, Journal of Computational Physics 141 (2) (1998) 199–224.

[3] B. Srinivasan, A comparison between the discontinuous Galerkin method and the high resolution wave propagation algorithm for the full two–fluid plasma model, Ph.D. thesis, University of Washington (2006).

[4] B. Srinivasan, A. Hakim, U. Shumlak, Numerical methods for two–fluid dispersive fast MHD phenomena, Communications in Computational Physics 10 (1) (2011) 183–215.

[5] B. Cockburn, G. E. Karniadakis, C.-W. Shu, Discontinuous Galerkin methods: theory, computation and applications, Vol. 11, Springer Science & Business Media, 2012.
[6] J. Douglas, T. Dupont, Interior penalty procedures for elliptic and parabolic Galerkin methods, in: Computing methods in applied sciences, Springer, 1976, pp. 207–216.

[7] D. N. Arnold, An interior penalty finite element method with discontinuous elements, SIAM journal on numerical analysis 19 (4) (1982) 742–760.

[8] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin method for time–dependent convection–diffusion systems, SIAM Journal on Numerical Analysis 35 (6) (1998) 2440–2463.

[9] J. Peraire, P.-O. Persson, The compact discontinuous Galerkin (cdg) method for elliptic problems, SIAM Journal on Scientific Computing 30 (4) (2008) 1806–1824.

[10] H. Liu, J. Yan, The direct discontinuous Galerkin (ddg) methods for diffusion problems, SIAM Journal on Numerical Analysis 47 (1) (2009) 675–698.

[11] B. van Leer, S. Nomura, Discontinuous Galerkin for diffusion, AIAA Paper 2005-5108.

[12] R. Nourgaliev, H. Park, V. Mousseau, Recovery discontinuous Galerkin Jacobian–free Newton–Krylov method for multiphysics problems, in: Computational Fluid Dynamics Review 2010, World Scientific, 2010, pp. 71–90.

[13] H. Luo, L. Luo, R. Nourgaliev, V. Mousseau, N. Dinh, A reconstructed discontinuous Galerkin method for the compressible Navier–Stokes equations on arbitrary grids, J. Comput. Phys. 229 (2010) 6961–6978.

[14] X. Yang, J. Cheng, H. Luo, Q. Zhao, A reconstructed direct discontinuous Galerkin method for simulating the compressible laminar and turbulent flows on hybrid grids, Computers & Fluids 168 (2018) 216–231.

[15] J. Lou, L. Li, H. Luo, H. Nishikawa, Reconstructed discontinuous Galerkin methods for linear advection–diffusion equations based on first–order hyperbolic system, Journal of Computational Physics 369 (2018) 103–124.
[16] J. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods, Algorithms, Analysis, and Applications, Springer, 2007.

[17] B. van Leer, M. Lo, A discontinuous Galerkin method for diffusion based on recovery, AIAA Paper 2007-4083.

[18] M. Abramowitz, I. A. Stegun, Handbook of mathematical functions with formulas, graphs, and mathematical tables. national bureau of standards applied mathematics series 55. Tenth Printing.

[19] O. Veblen, J. W. Young, Projective geometry, Vol. 2, Ginn, 1918.

[20] M. Berger, Geometry, Springer, 1987.

[21] W. L. Oberkampf, C. J. Roy, Verification and validation in scientific computing, Cambridge University Press, 2010.

[22] M. H. Carpenter, C. A. Kennedy, Fourth–order 2N–storage Runge–Kutta schemes.

[23] J. T. Oden, I. Babuška, C. E. Baumann, A discontinuous hp finite element method for diffusion problems, Journal of computational physics 146 (2) (1998) 491–519.

[24] P. E. Johnson, E. Johnsen, The compact gradient recovery discontinuous Galerkin method for diffusion problems, Journal of Computational Physics 398 (2019) 108872.

[25] W. Sutherland, LII. The viscosity of gases and molecular force, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 36 (223) (1893) 507–531.

[26] E. F. Toro, Riemann solvers and numerical methods for fluid dynamics: a practical introduction, Springer Science & Business Media, 2013.

[27] Y.-H. Tseng, J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of computational physics 192 (2) (2003) 593–623.