SCIENTIFIC REPORTS
natureresearch

Corrected: Author Correction

OPEN

Received: 25 March 2019
Accepted: 10 September 2019
Published online: 08 October 2019

Supersonic turbulent flow simulation using a scalable parallel modal discontinuous Galerkin numerical method

Tomas Houba[1], Arnob Dasgupta[2], Shivasubramanian Gopalakrishnan[3], Ryan Gosse[4] & Subrata Roy[1,2]

The scalability and efficiency of numerical methods on parallel computer architectures is of prime importance as we march towards exascale computing. Classical methods like finite difference schemes and finite volume methods have inherent roadblocks in their mathematical construction to achieve good scalability. These methods are popularly used to solve the Navier-Stokes equations for fluid flow simulations. The discontinuous Galerkin family of methods for solving continuum partial differential equations has shown promise in realizing parallel efficiency and scalability when approaching petascale computations. In this paper an explicit modal discontinuous Galerkin (DG) method utilizing Implicit Large Eddy Simulation (ILES) is proposed for unsteady turbulent flow simulations involving the three-dimensional Navier-Stokes equations. A study of the method was performed for the Taylor-Green vortex case at a Reynolds number ranging from 100 to 1600. The polynomial order $P = 2$ (third order accurate) was found to closely match the Direct Navier-Stokes (DNS) results for all Reynolds numbers tested outside of Re = 1600, which had a normalized RMS error of 3.43×10^{-4} in the dissipation rate for a 60^3 element mesh. The scalability and performance study of the method was then conducted for a Reynolds number of 1600 for polynomials orders from $P = 2$ to $P = 6$. The highest order polynomial that was tested ($P = 6$) was found to have the most efficient scalability using both the MPI and OpenMP implementations.

Advances in modern computer hardware have enabled numerical computations to reach progressively larger scales. To handle the challenging and costly simulations, parallel computations have become widespread in both research and production Computational Fluid Dynamics (CFD) and other Computer-Aided Engineering (CAE) codes. To keep up with the demand for increasingly larger and more complex numerical models, the scalability and efficiency of a parallel implementation for a numerical discretization is an important factor. One way to improve the parallel efficiency of CFD software is to optimize the underlying code. Examples of these types of optimizations are the improvement of loop-level parallelism, serial efficiency of the code[1], reducing the number of cache misses and optimizing the achievable memory bandwidth[2].

Another possible path to improve the parallel efficiency is to consider the numerical method implemented in the software. One promising direction is the application of high-order methods for massively parallel CFD. In the CFD community, high-order methods are considered to be those which are third order and higher[3]. Low-order schemes are widely used in CFD, but there are applications for which they are considered insufficient, including turbulence, aeroacoustics, boundary layer flows, vortical flows, shock-boundary layer interactions and others[4]. For these types of flows, low-order methods require extremely small discretization scale lengths to accurately resolve the unsteady vortices over relevant length and time scales. This has led to a large amount of research in high-order methods aimed at solving physics problems which are not well suited to low-order methods. Outside of this "physics" argument for use of high-order methods, there is the issue of parallel scalability.

[1]SurfPlasma Inc, Gainesville, FL, 32601, USA. [2]Applied Physics Research Group, Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, 32601, USA. [3]Department of Mechanical Engineering, Indian Institute of Technology Bombay, Mumbai, 400076, India. [4]Vanilla Peak Ct, Vail, AZ, 85641, USA. Correspondence and requests for materials should be addressed to S.R. (email: roy@ufl.edu)

Since higher order polynomial approximations require more calculations to be carried out per element, it is expected that the scheme will exhibit a higher efficiency when higher order polynomials are used. The parallel algorithm requires a finite setup and communication time, which decreases its efficiency below the ideal linear speed-up. This overhead time depends on the scale of the parallel simulation, i.e. the number of parallel tasks or threads used. Since the higher order polynomials spend a longer time calculating the solution on a per degree of freedom basis, it is expected that the overhead time would be more negligible in comparison.

The objective of this paper is to demonstrate a scalable, parallel, high-order description of modal Discontinuous Galerkin (DG) elements for supersonic, turbulent boundary layer flows using Runge-Kutta explicit time marching. The spatial discretization scheme considered in the DG method can be made high-order by increasing the approximation order P of the interpolating polynomial. Polynomial approximations ranging from $P=2$ to $P=6$ are compared for a canonical problem of isotropic turbulence to study their parallel efficiency. In addition, the computational cost required to reach the same error as a lower-order polynomial is considered. This is an important metric to obtain the full picture of the computational cost of the different polynomial orders. Other authors have proposed using operation count instead of runtime comparisons, and found that for implicit solvers, high-order methods were more efficient than low-order ones[5]. Parallel scalability is important, but only if the underlying serial computational cost is not prohibitively expensive to the point where the benefit gained from a better scalability is lost. In addition to the isotropic turbulence, the method was also validated on a zero-pressure gradient supersonic Mach 2.25 turbulent boundary layer flow over a flat plate.

This paper is organized as follows. The next section describes the governing equations used in the study. Then an overview of the numerical method, including the DG spatial discretization, the numerical fluxes and the time integration is given. The ensuing section provides the background and results for the isotropic turbulence (Taylor-Green vortex) test case. Following that the results of the parallel scalability studies and performance comparisons of different polynomial orders are presented. Then, the turbulent boundary layer flow solution for a supersonic flow over a flat plate is documented. Finally the conclusions from these studies are summarized.

Governing Equations

To understand the fluid mechanics, one must appreciate the partial differential equations which govern fluid flow. This section describes these governing equations as well as other equations involved in this study.

Compressible Navier-Stokes equations.

For a compressible Newtonian fluid, the multi-dimensional N-S equations in normalized conservative form can be written as

$$\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{t}} + \nabla \cdot (\boldsymbol{\rho v}) = 0 \tag{1}$$

$$\frac{\partial(\boldsymbol{\rho v})}{\partial \boldsymbol{t}} + \nabla \cdot (\boldsymbol{\rho v v}^{\top} + p\mathbf{I} - \bar{\boldsymbol{\tau}}) = 0 \tag{2}$$

$$\frac{\partial(\boldsymbol{\rho} E)}{\partial \boldsymbol{t}} + \nabla \cdot [(\boldsymbol{\rho} E + p)\boldsymbol{v} - k\nabla T - \boldsymbol{v} \cdot \bar{\boldsymbol{\tau}}] = 0 \tag{3}$$

$$\bar{\boldsymbol{\tau}} = [\begin{matrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{matrix}];\ \tau_{ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right);\ i, j, k = x, y, z \tag{4}$$

$$p = (\gamma - 1)\left[\rho E - \frac{1}{2}\rho\,|\mathbf{v}|^2\right];\ T = \frac{p}{\rho R};\ R = \frac{1}{\gamma M^2};\ k = \frac{\mu c_p}{\mathrm{Pr}} \tag{5}$$

Here $\bar{\boldsymbol{\tau}}$ denotes the viscous stress tensor which is given by Eq. (4). The term μ in the viscous stresses is the dynamic viscosity of the fluid and Sutherland's law is used to define it. The term k denotes the thermal conductivity of the fluid with T being its temperature. This term comes from the Fourier's Law of heat conduction. The thermal conductivity is obtained using the dynamic viscosity μ, Prandtl number (Pr) and specific heat (c_p) of the fluid given by Eq. (5). The velocity vector is denoted by $\mathbf{v}$, which includes the three components, u, v and w in streamwise (x), wall normal (y) and spanwise (z) directions, respectively.

Numerical Method

DG finite element was first presented by Reed and Hill[6] to solve the neutron transport equations. Due to its inherent advantage of solving linear equation systems on an element-by-element basis, it has become one of the most promising computational technique to solve large equation systems with high parallel efficiency, even allowing the numerical formulation to approach an "embarrassingly parallel problem". However, the next challenge was to solve the nonlinear systems of equations such as the hyperbolic conservation laws, which are prominent in most physical systems. For this, an explicit version of this method was devised[7] which employed the use of Runge – Kutta time discretization with a Total Variation Diminishing in the Means (TVDM) and Total Variation Bounded (TVB) slope limiter. This method was called the RKDG method. This was extended to high order RKDG methods[8] which showed $P+1$ order of convergence for P order space discretization.

The development of DG methods for nonlinear hyperbolic systems occurred rapidly over the past two decades. The improvement of the computer architecture (for example, the advent of petascale computing machines)

Order	$\varphi_l(\boldsymbol{x})$	$\varphi_l(\boldsymbol{x}, \boldsymbol{y})$	$\varphi_l(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$
0	1	1	1
1	x	x, y	x, y, z
2	$3x^2 - 1$	$3x^2 - 1, 3y^2 - 1, xy$	$3x^2 - 1, 3y^2 - 1, 3z^2 - 1, xy, yz, xz$
3	$5x^3 - 3x$	$5x^3 - 3x, 5y^3 - 3y, (3x^2 - 1)y, (3y^2 - 1)x$	$5x^3 - 3x, 5x^3 - 3, 5x^3 - 3, (3x^2 - 1)y, (3y^2 - 1)z, (3y^2 - 1)x, (3y^2 - 1)z, (3z^2 - 1)x, (3z^2 - 1)y$

Table 1. Basis functions.

combined with the need to solve both hyperbolic and elliptic problems led to the extension of this method to convection-diffusion type problems. The first study of this form of equations was conducted on hydrodynamic models for semiconductor device simulations[9,10]. This was further studied for compressible Navier Stokes equations[11] to achieve higher order of accuracy. It involved the simple breakdown of the second order equation into two first order equations with U and dU as independent variables and then solving the system using the original RKDG method. This method also known as the first Bassi – Rebay (BR1) method[11] was further extended to achieve higher stability. This incorporated the explicit evaluation of the term dU without making it a new variable. This is also known as the second Bassi – Rebay (BR2) method[12]. There are numerous other methods[13] to tackle these type of equation systems and can also be generalized as the Local Discontinuous Galerkin (LDG) methods[14]. It should also be noted that different methods have been implemented on the DG framework. Some of these methods include Spectral DG method and *hp* – adaptive methods. The first DG spectral method was conducted for elliptic problems[15] and linear hyperbolic problems[16]. It was further studied for advection diffusion problems, compressible flow and complex geometries[17–19]. Implementation of adaptive methods in DG is straight forward. This is because there is no inter – element continuity requirement, which allows for simple changes of the element order based on the gradient. Lower orders are achieved by making the higher order terms zero. This method has been applied to both hyperbolic conservation laws[20] and convection diffusion problems[21,22].

The entire DG framework was implemented in an in-house code called the Multiscale Ionized Gas (MIG) flow code. This is a FORTRAN 90 modular code, which can be used to solve various problems like plasma drift diffusion equations[23], hypersonic non-equilibrium flow[24], magnetohydrodynamic equations[25], and subsonic turbulent flow control[26]. The framework is parallelized via the message passing interface (MPI), which enables it to perform computations on multiple nodes on conventional supercomputing clusters. The sections ahead, will describe the space and time discretization for the Discontinuous Galerkin finite element framework, convergence study, implementation of slope limiters, and parallelization of the code.

Discontinuous Galerkin space discretization. To understand the discretization process for convection – diffusion problems, a generic scalar equation is chosen which can be extended to any equation system. This is given by

$$\frac{\partial U}{\partial t} + \nabla \cdot \overrightarrow{F}^{inv}(U) - \nabla \cdot \overrightarrow{F}^{v}(U, \nabla U) = 0 \tag{6}$$

$$U(\boldsymbol{x}, 0) = U_0(\boldsymbol{x}) \tag{7}$$

Where U denotes the conserved scalar variable, F^{inv} and F^v denote the inviscid and viscous fluxes respectively and $\boldsymbol{x} \in \Omega$, which is the multidimensional domain. All the boundaries are considered periodic in this section. For an element, the approximate solution U_h $(\boldsymbol{x}, t)$ is represented by Eq. (8).

$$U_h(x, t) = \sum_{l=0}^{p} U_K^l(t)\varphi_l(x) \tag{8}$$

Where subscript K denotes the element, U_K^l denotes the modal degrees of freedom of that element, φ_l denotes the basis function. Legendre polynomials are chosen as the local basis functions because of their property of L^2 – orthogonality, which leads to a diagonal mass matrix and is beneficial when performing explicit calculations. The list of basis functions for a transformed coordinate system of $x,\ y,\ z \in [-1,\ 1]$ is provided in Table 1.

To obtain the weak form of the equation, the variable U is replaced by U_h and Eq. (6) is multiplied with the basis function φ_l. After integration by parts, Eq. (9) is obtained.

$$\begin{aligned} &\frac{d}{dt}\int_K U_h\varphi(x)\ dx - \int_K \overrightarrow{F}^{inv} \cdot \nabla\varphi(x)dx + \sum_{e\in\Gamma}\int_e \overrightarrow{F}^{inv} \cdot \overrightarrow{n}_{e,K}\varphi(x)d\Gamma \\ &+ \int_K \overrightarrow{F}^{v} \cdot \nabla\varphi(x)dx - \sum_{e\in\Gamma}\int_e \overrightarrow{F}^{v} \cdot \overrightarrow{n}_{e,K}\varphi(x)d\Gamma = 0 \end{aligned} \tag{9}$$

In Eq. (9), $n_{e,K}$ denotes the outward unit normal for the edge e (it can be a face or an edge) of element K. Figure 1 shows a representation of these elements. The element boundary space is denoted by Γ. For the terms in summation, where fluxes are to be evaluated at the element interfaces, the solution U_h is discontinuous and cannot be uniquely defined. Thus, the terms must be replaced by a locally Lipschitz, consistent, monotone flux to maintain the stability and convergence properties of the scheme with higher order of accuracy[8]. In Eq. (9), $\overrightarrow{F}^{v}$ is

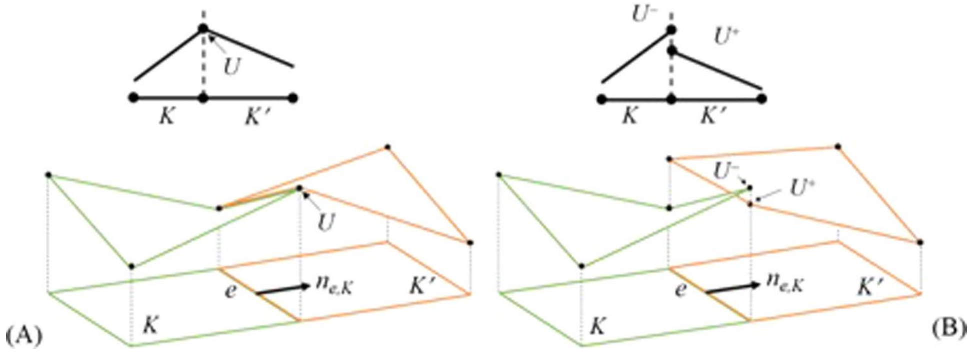

Figure 1. Comparison between continuous and discontinuous Galerkin method. (**A**) Continuous element with interface solution U for element K and K' and (**B**) discontinuous element with interface solutions U^- and U^+ for element K and K' respectively sharing the edge e with an outward unit normal $n_{e,K}$.

n	**1**	**2**	**3**		**4**		**5**		
w_n	2	1	$\frac{8}{9}$	$\frac{5}{9}$	$\frac{18+\sqrt{30}}{36}$	$\frac{18-\sqrt{30}}{36}$	$\frac{128}{225}$	$\frac{322+13\sqrt{70}}{900}$	$\frac{322-13\sqrt{70}}{900}$
x'_n	0	$\pm\sqrt{\frac{1}{3}}$	0	$\pm\sqrt{\frac{3}{5}}$	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	0	$\pm\sqrt{\frac{5}{9}-\frac{2}{9}\sqrt{\frac{10}{7}}}$	$\pm\sqrt{\frac{5}{9}+\frac{2}{9}\sqrt{\frac{10}{7}}}$

Table 2. Gauss – Legendre Quadrature.

a function of both U and ∇U, which implies that either ∇U needs to be evaluated as a new variable or treated explicitly. Detailed descriptions of the numerical integration, fluxes and terms are provided in the next two sections.

Numerical integration. All the integrals can be written in discrete form using Gauss – Legendre quadrature rules.

$$\int_K f(x)dx = jac\int_{-1}^{1} f(x')dx' = jac\sum_{n=1}^{N} w_n f(x'_n)dx' \tag{10}$$

In Eq. (10) *jac* is obtained when transforming from global coordinate system to local coordinate system. Also for all the integrals shown, the basis functions vary with space, while the degrees of freedom vary in time as shown in Eq. (8). Since the basis functions are already in transformed space x'_n are the Gauss – Legendre points provided in Table 2. One should note that for multidimensional integration the single summation becomes multiple summations with quadrature points x'_n and weights w_n being obtained via tensor product of one-dimensional weights and points.

Inviscid fluxes. As mentioned earlier, the discontinuity at the element interfaces requires the use of numerical fluxes. There are wide variety of numerical fluxes which satisfy the locally Lipschitz, monotone and consistent criteria[27]. However, the present work uses either Godunov flux or Local Lax-Friedrichs flux[28]. The later, also known as ENO-LLF, provides better shock capturing with improved accuracy. Although it is more diffusive than the Roe flux and the Godunov flux, its impact on the solution is insignificant for higher order approximations[27]. After replacing the inviscid flux in Eq. (9) with the numerical flux $h_{e,K}^{inv}$, the first summation term can be written as

$$\sum_{e\in\Gamma}\int_e \overrightarrow{F}^{inv}\cdot\overrightarrow{n}_{e,K}\varphi(x)d\Gamma = \sum_{e\in\Gamma}\int_e h_{e,K}^{inv}(U_h^-, U_h^+)\varphi(x)d\Gamma \tag{11}$$

The $+$ and $-$ states of the solution refer to the outside and inside solution along edge e as depicted in Fig. 1. The Godunov flux is given by

$$h_{e,K}^{inv}(U^-, U^+) = \begin{Bmatrix} \min_{U^-\le U\le U^+}F^{inv}(U), & \text{if } U^- \le U^+ \\ \max_{U^+\le U\le U^-}F^{inv}(U), & \text{otherwise} \end{Bmatrix} \tag{12}$$

Eq. (12) can be interpreted as, if the neighbouring solution U^+ is larger than the inside solution U^- then choose the minimum flux (min $[F^{inv}(U^+),\ F^{inv}(U^-)]$) otherwise choose the maximum of the two.

The Lax – Friedrichs flux is given by

$$h_{e,K}^{inv}(U_h^+, U_h^-) = \frac{1}{2}\left[\overrightarrow{F}^{inv}(U_h^-)\cdot\overrightarrow{n}_{e,K} + \overrightarrow{F}^{inv}(U_h^+)\cdot\overrightarrow{n}_{e,K} - \alpha_{e,K}(U_h^+ - U_h^-)\right] \tag{13}$$

In Eq. (13) $\alpha_{e,K}$ is obtained by evaluating the largest absolute eigenvalue of the jacobian matrices for the outside and inside elements.

$$\begin{aligned}\alpha_{e,K} &= \max[\{abs(\boldsymbol{\lambda}^{+})\}, \{abs(\boldsymbol{\lambda}^{-})\}]\\ \boldsymbol{\lambda}^{+} &= eigenvalue\left\{\frac{\partial F(\overline{U}^{+})}{\partial U}\cdot n_{e,K}\right\}\\ \boldsymbol{\lambda}^{-} &= eigenvalue\left\{\frac{\partial F(\overline{U}^{-})}{\partial U}\cdot n_{e,K}\right\}\end{aligned} \tag{14}$$

For Euler equations or Navier-Stokes equations the eigenvalues are $u+a$, $u-a$ and u, where a is the speed of sound. In Eq. (14), $\overline{U}$ is the mean solution of the inside or outside element depending on the λ being evaluated.

Viscous fluxes. The viscous terms in Eq. (9) can be modelled in numerous ways. Some of the common methods are LDG[14], Bassi – Rebay (BR1 and BR2)[11,12], Interior Penalty (IP)[29], Baumann – Oden[21] etc. type methods. A detailed comparison and insight on these methods can be found in Arnold *et al.*[13]. However, for brevity only the LDG, BR1 and BR2 schemes are described here.

The viscous fluxes include ∇U as an unknown which must be evaluated either a priori or along with the equation system. To evaluate ∇U, Eq. (6) is first changed to Eq. (15) and Eq. (16).

$$\frac{\partial U}{\partial t} + \nabla\cdot\overrightarrow{F}^{inv}(U) - \nabla\cdot\overrightarrow{F}^{v}(U,\theta) = 0 \tag{15}$$

$$\theta = \nabla U \tag{16}$$

The same procedure as mentioned before is followed and finally equations like Eq. (9) are obtained.

$$\begin{aligned}&\frac{d}{dt}\int_K U_h\varphi(x)dx - \int_K \overrightarrow{F}^{inv}(U_h)\cdot\nabla\varphi(x)dx + \sum_{e\in\Gamma}\int_e \overrightarrow{F}^{inv}(U_h)\cdot\overrightarrow{n}_{e,K}\varphi(x)d\Gamma\\ &+\int_K \overrightarrow{F}^{v}(U_h,\theta_h)\cdot\nabla\varphi(x)dx - \sum_{e\in\Gamma}\int_e \overrightarrow{F}^{v}(U_h,\theta_h)\cdot\overrightarrow{n}_{e,K}\varphi(x)d\Gamma = 0\end{aligned} \tag{17}$$

$$\int_K \theta_h\varphi(x)dx + \int_K U_h\nabla\varphi(x)dx - \sum_{e\in\Gamma}\int_e U_h\overrightarrow{n}_{e,K}\varphi(x)d\Gamma = 0 \tag{18}$$

It should be noted that in Eq. (17) and Eq. (18) θ_h denotes the approximate solution of the auxiliary variable θ as in the definition given in Eq. (8). As discussed earlier, the discontinuous interface requires the fluxes in the summation terms to be evaluated using a locally Lipschitz, consistent and monotone flux. Therefore the last terms in Eq. (17) and Eq. (18) are represented as Eq. (19) and Eq. (20).

$$\sum_{e\in\Gamma}\int_e \overrightarrow{F}^{v}(U_h,\theta_h)\cdot\overrightarrow{n}_{e,K}\varphi(x)d\Gamma = \sum_{e\in\Gamma}\int_e h^{v}_{e,K}(U_h^{+},U_h^{-},\theta_h^{+},\theta_h^{-})\cdot\overrightarrow{n}_{e,K}\varphi(x)d\Gamma \tag{19}$$

$$\sum_{e\in\Gamma}\int_e U_h\overrightarrow{n}_{e,K}\varphi(x)\,d\Gamma = \sum_{e\in\Gamma}\int_e h^{\theta}_{e,K}(U_h^{+},U_h^{-},\theta_h^{+},\theta_h^{-})\overrightarrow{n}_{e,K}\varphi(x)d\Gamma \tag{20}$$

The choice of numerical fluxes $h^{v}_{e,K}$ and $h^{\theta}_{e,K}$ gives rise to different methods.

Local discontinuous Galerkin method. The viscous numerical fluxes for this method can be written as

$$\begin{aligned}&h(U_h^{+},U_h^{-},\theta_h^{+},\theta_h^{-}) = \frac{1}{2}[\overrightarrow{F}^{v}(U_h^{+},\theta_h^{+}) + \overrightarrow{F}^{v}(U_h^{-},\theta_h^{-}) + \boldsymbol{C}(\boldsymbol{U}^{+}-\boldsymbol{U}^{-})]\\ &\boldsymbol{C} = \begin{bmatrix} c_{11} & c_{12}\\ -c_{12} & 0\end{bmatrix},\ \boldsymbol{U} = \{U,\theta\}\end{aligned} \tag{21}$$

Using Eq. (21) and since $\overrightarrow{F}^{v}$ for Eq. (16) is U, obtain the expressions for $h^{v}_{e,K}$ and $h^{\theta}_{e,K}$ are

$$\begin{aligned}&h^{v}_{e,K}(U_h^{+},U_h^{-},\theta_h^{+},\theta_h^{-})\\ &\quad= \frac{1}{2}[\overrightarrow{F}^{v}(U_h^{+},\theta_h^{+}) + \overrightarrow{F}^{v}(U_h^{-},\theta_h^{-})] + c_{11}(U_h^{+}-U_h^{-}) + c_{12}(\theta_h^{+}-\theta_h^{-})\\ &h^{\theta}_{e,K}(U_h^{+},U_h^{-},\theta_h^{+},\theta_h^{-}) = \frac{1}{2}(U_h^{+}+U_h^{-}) - c_{12}(U_h^{+}-U_h^{-})\end{aligned} \tag{22}$$

A detailed discussion about the choice of constants c_{11} and c_{12}, as well as the extension to multidimensional problems have been described by Cockburn and Shu[14].

Bassi – Rebay method I. The numerical fluxes $h_{e,K}^{\nu}$ and $h_{e,K}^{\theta}$ are obtained by averaging the fluxes at the edge of the element and its neighbor. This is provided in Eq. (23) and Eq. (24)

$$h_{e,K}^{\nu}(U_h^+, U_h^-, \theta_h^+, \theta_h^-) = \frac{1}{2}[\vec{F}^{\nu}(U_h^+, \theta_h^+) + \vec{F}^{\nu}(U_h^-, \theta_h^-)] \tag{23}$$

$$h_{e,K}^{\theta}(U_h^+, U_h^-, \theta_h^+, \theta_h^-) = \frac{1}{2}[U_h^+ + U_h^-] \tag{24}$$

The above method describes the BR1 scheme. However due to the method's deficiencies, such as non – optimal accuracy for purely elliptic problems, spread stencil and increase in the number of degrees of freedom per element (specially for implicit algorithm)[12], lead to the implementation of BR2 scheme.

Bassi – Rebay method II. This scheme uses the property that, the evaluation of a solution gradient inside the element is trivial and can be obtained using the gradients of the basis functions. However, for $P = 0$ elements and at interface discontinuities it is not trivial. To obtain ∇U without adding an extra equation a correction term R is added. This is known as the lift operator. After few mathematical manipulations[12] Eq. (18) can be rewritten as Eq. (25).

$$\int_K \theta_h \varphi(x)dx = \int_K \varphi(x)\nabla U_h dx + \sum_{e\in\Gamma}\int_e \frac{1}{2}(U_h^+ + U_h^-)\vec{n}_{e,K}\varphi(x)d\Gamma \tag{25}$$

Thus, we can write $\theta_h = \nabla U_h + R_h$, where R_h is defined like Eq. (8) and can be obtained using Eq. (26).

$$\int_K R_h\varphi(x)dx = \sum_{e\in\Gamma}\int_e \frac{1}{2}(U_h^+ + U_h^-)\vec{n}_{e,K}\varphi(x)d\Gamma \tag{26}$$

Using the global lifting operator leads to a non-compact stencil which can be avoided by using local lift operators r_h. This is defined by

$$\int_K r_h\varphi\ (x)dx = \int_e \frac{1}{2}(U_h^+ + U_h^-)\vec{n}_{e,K}\varphi(x)d\Gamma,\ \ R_h = \sum_{e\in K} r_h \tag{27}$$

When performing volume integrals, global lift operators are used and for element boundary integrals, local lift operators are used. Using this scheme leads to a reduction in the number of degrees of freedom. The information from immediate neighbors is only required producing a compact stencil. This minimization of information needed from the local region means that the method spends most of its time computing local integrals, and the communication workload is far smaller than the computational workload. A scenario then arises where most of the calculations in each individual element are independent and thus almost "embarrassingly parallel" making them amenable to exploit maximum parallel efficiencies.

Temporal discretization. The choice of time integration depends on the problem in hand. For transient accuracy, high order time accurate schemes need to be implemented. Problems involving acoustic wave propagation fall in this category. This section will describe some of the common time integration methods implemented and their advantages and disadvantages.

Explicit time integration. To solve the nonlinear hyperbolic conservation laws in a DG framework an explicit implementation of the method was introduced[30]. This overcame the issue of solving nonlinear problems on an element by element basis. However, an explicit method is restricted by the CFL condition. To improve the stability of the scheme a TVDM slope limiter was implemented[31]. However, this method was only first order accurate in time and the slope limiter affected the smooth regions of the solution reducing the spatial accuracy. This was finally overcome by using the RKDG method and a modified slope limiter which was second order in time and maintained the accuracy of the scheme in smooth regions[7]. This made the scheme stable for CFL $\leq 1/3$. To show the explicit time integration Eq. (9) is written in a modified form given by Eq. (29).

$$\frac{d}{dt}\int_K U_h(x, t^n)\varphi(x)dx = L_h[U_h(x, t^n)]$$

$$\frac{d}{dt}\int_K U_K^l(t^n)\varphi_l(x)\varphi_l^T(x)dx = L_h[U_h(x, t^n)]$$

$$\frac{d}{dt}[U_K^l(t^n)]\int_K \varphi_l(x)\varphi_l^T(x)dx = \frac{d}{dt}[U_K^l(t^n)][M] = L_h[U_h(x, t^n)] \tag{28}$$

$$\frac{d}{dt}[U_K^l(t^n)] = L_h[U_h(x, t^n)][M]^{-1} \tag{29}$$

The mass matrix $[M]$, is diagonal for the present choice of basis functions. For the simple Euler explicit case, Eq. (29) can be written as Eq. (30) which yields first order accuracy in time.

$$[U_K^l(t^{n+1}) - U_K^l(t^n)] = (\Delta t)L_h[U_h(x, t^n)][M]^{-1} \tag{30}$$

Using the second order RKDG method the solution can be more time accurate. This is described in Eq. (31)

$$[U_K^l(t^m)] = [U_K^l(t^n)] + (\Delta t)L_h[U_h(x, t^n)][M]^{-1}$$

$$[U_K^l(t^{m+1})] = [U_K^l(t^m)] + (\Delta t)L_h[U_h(x, t^m)][M]^{-1}$$

$$[U_K^l(t^{n+1})] = \frac{1}{2}([U_K^l(t^{m+1}) + U_K^l(t^n)]) \tag{31}$$

The RKDG method has been proven to give CFL ≤ 1/3 for $P = 1$ and CFL ≤ 1/5 for $P = 2$ case[7]. Although RKDG scheme has high parallelizability, being an explicit scheme it has CFL restrictions.

Implicit time integration. Since the problems studied are nonlinear in nature, the Newton's method is employed to solve for the equation system. The goal here is to find a value iteratively, which would be closest to the actual solution. Thus, Eq. (29) is written as Eq. (32) for iteration q

$$f(U_K^l(t^{n,q})) = \frac{d}{dt}[U_K^l(t^{n,q})] - L_h[U_h(x, t^{n,q})][M]^{-1} \approx 0 \tag{32}$$

To get the next time step solution Eq. (32) is discretized in time using the Euler Implicit algorithm to obtain Eq. (33).

$$f(U_K^l(t^{n+1,q}), U_K^l(t^{n+1,q-1})) = [U_K^l(t^{n+1,q}) - U_K^l(t^{n+1,q-1})] - \Delta t L_h[U_h(x, t^{n+1,q})][M]^{-1} \tag{33}$$

Therefore, for $q \geq 1$, Newton's method can be applied to Eq. (33). It should be noted that when $q = 1$ in Eq. (33), $U_K^l(t^{n+1,q-1}) = U_K^l(t^n)$.

$$\left[\frac{\partial\{f(U_K^l(t^{n+1,q}), U_K^l(t^{n,q}))\}}{\partial U_K^l(t^{n+1,q})}\right][U_K^l(t^{n+1,q+1}) - U_K^l(t^{n+1,q})] = -f(U_K^l(t^{n+1,q+1}), U_K^l(t^{n+1,q})) \tag{34}$$

Test Cases

Taylor green vortex.

Background. This is one of the canonical problems studied for hydrodynamic turbulence. This has been extensively studied in literature to derive empirical and analytical relations in turbulent flow physics. Early in depth numerical investigation of this problem was done by Orszag[32] and Brachet *et al.*[33,34]. This problem was also studied by Comte-Bellot and Corrsin[35] experimentally as a grid turbulence problem. These studies have become the benchmark for turbulent code validation. Since then, different numerical methods[36–39] have been used to improve or validate these studies. Results for different Reynolds number, mesh and spatial order of accuracy are compared and investigated. The domain size $\Omega = (2\pi \times 2\pi \times 2\pi)$ with periodic boundaries on all faces. The initial conditions for this problem are

$$u_0 = \sin(x)\cos(y)\cos(z), \quad v_0 = \sin(y)\cos(x)\cos(z), \quad w_0 = 0,$$

$$p_0 = 100 + \frac{1}{16}(\cos(2x) + \cos(2y))(\cos(2z) + 2), \; \rho_0 = 1 \tag{35}$$

This problem is solved using RKDG method, which involves RK2 time marching and LDG scheme for viscous flux. Two types of inviscid fluxes are tested, namely Godunov flux and LLF flux. The mesh is uniform in all directions and the DOFs for an N^3 mesh corresponds to $N^3 \times (P + 1)^3$. Although the cases can be run at different time step Δt, the solutions are obtained using $\Delta t = 2.5 \times 10^{-4}$, to have similar time diffusion. The time step is kept low since the Godunov flux requires more restrictive time stepping than the LLF flux. The simulations are run till $t = 10$. Three main parameters are used to study this case. These include the integrated kinetic energy E_k, kinetic energy dissipation rate ε and integrated enstrophy ζ. These parameters are given in Eq. (36). For incompressible flows ε and ζ can be related using the relation given in Eq. (37). It should be noted that evaluation of ε (ζ) requires additional degrees of freedom to reach the correct ε levels when compared to ε (E_k).

$$E_k = \frac{1}{\rho_0\Omega}\int_\Omega \rho\frac{\mathbf{v}\cdot\mathbf{v}}{2}d\Omega \; ; \; \varepsilon(E_k) = -\frac{\partial E_k}{\partial t}; \; \zeta = \frac{1}{\rho_0\Omega}\int_\Omega \rho\frac{\boldsymbol{\omega}\cdot\boldsymbol{\omega}}{2}d\Omega \tag{36}$$

$$\varepsilon(\zeta) = \frac{2\zeta}{\text{Re}} \tag{37}$$

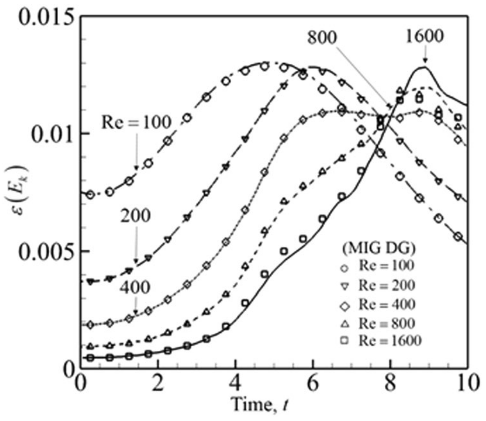

Figure 2. Energy dissipation rate at different Reynolds number using third order accurate DG solution on a 60^3 mesh compared with DNS results[34].

Re	Norm RMS Error
100	2.25×10^{-6}
200	2.85×10^{-6}
400	2.62×10^{-6}
800	3.14×10^{-5}
1600	3.43×10^{-4}

Table 3. Norm RMS Error in dissipation rate at different Reynolds number.

Order	Godunov Flux	Local Lax – Friedrichs flux
2	3.43×10^{-4}	7.35×10^{-4}
3	9.38×10^{-5}	3.36×10^{-4}
4	7.83×10^{-5}	1.93×10^{-5}

Table 4. Norm RMS Error in dissipation rate for Godunov and LLF fluxes.

Effect of Reynolds number. To study the effect of Reynolds number (Re), the inviscid flux is kept as Godunov flux and a 60^3 (180^3 degrees of freedom) mesh size is used. The third order accurate ($P=2$) spatial accuracy is chosen. The Reynolds numbers tested are 100, 200, 400, 800 and 1600. The normalized Root Mean Square (RMS) error of $\varepsilon(E_k)$ in comparison with DNS data is given in Table 3. The norm error is evaluated using Eq. (38). The timestep is 10^{-3} sec and data is printed at every 250 steps within the 10 sec interval ($N=40$) for all cases considered for Eq. (38). Except Re = 1600 all the other Reynolds number have results that are comparative to DNS results[34]. The profile of kinetic energy dissipation rate ε (E_k) is shown in Fig. 2. The dissipation rate is captured accurately by MIG DG ILES. However, in the next section it will be seen that using LLF inviscid flux has slightly more error than the Godunov flux due to its higher dissipation.

$$\text{Norm RMS Error} = \sqrt{\frac{\sum_{i=1}^{N}(\varepsilon_i - \varepsilon_{\text{DNS}})^2}{N}} \tag{38}$$

Effect of inviscid numerical flux. To study the effect of numerical fluxes, the Godunov flux and LLF flux are tested for a 60^3 with $P=2$ (180^3 DOF), 45^3 with $P=3$ (180^3 DOF) and 36^3 with $P=4$ (180^3 DOF) mesh sizes. It should be noted that the total DOF is calculated by $N^3 \times (P+1)^3$. The Reynolds number for the cases here is kept at Re = 1600. The normalized RMS error of ε (E_k) in comparison with DNS data[36] is given in Table 4. The dissipation rate has higher errors when the LLF scheme is used. The greater diffusive nature of LLF flux was also observed by Beck *et al.*[40] when comparing with the Roe scheme. However, the differences are very low as the errors are two orders of magnitude lower than the variable value. It should be noted that although Godunov flux is more accurate due to its least dissipative nature, it creates larger oscillations which can result in backscatter and also requires a lower time step. Therefore, although LLF has higher numerical dispersion, it is preferable to be used with slightly higher degrees of freedom. For this problem using around 1.4 times the number of DOF in each direction matches the solutions for both the fluxes at $P=2$. For higher orders, the differences in dissipation rate due to fluxes become negligible. This can be observed in Fig. 3 which depicts the similarity in solutions for the two fluxes at different degrees of freedom for a $P=2$ and $P=4$ case.

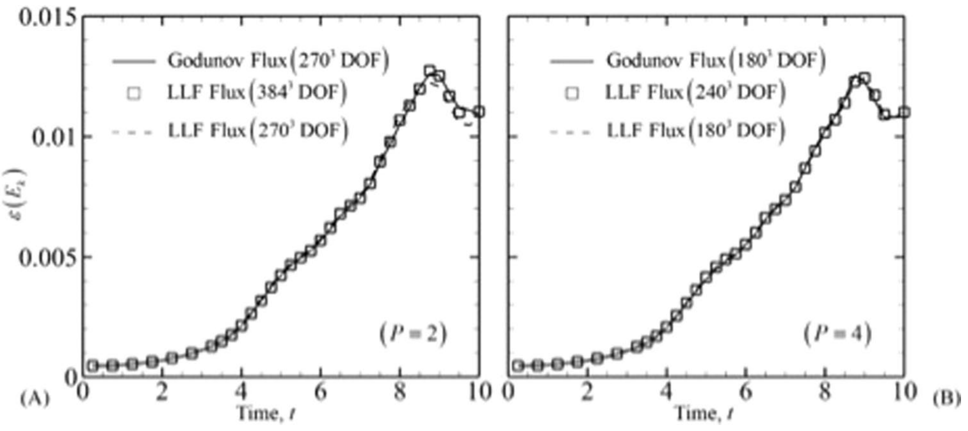

Figure 3. Comparison of energy dissipation rate for different inviscid numerical fluxes at different degrees of freedom and polynomial order. Dissipation rate for (**A**) $P=2$ and (**B**) $P=4$.

Effect of spatial order of accuracy. To study the effect of spatial order of accuracy LLF flux is chosen as the inviscid numerical flux. The problem is studied using orders $P=2$, $P=3$ and $P=4$. The Reynolds number for the cases here is kept at $\mathrm{Re}=1600$. All the parameters mentioned in Eq. (36) and Eq. (37) are depicted in Fig. 4. Both $\varepsilon\ (\zeta)$ and $\varepsilon\ (E_k)$ are compared to highlight the differences between ILES results and DNS results[36] as well as to show that, capturing gradients in ILES requires more degrees of freedom. The DNS results are obtained using 13-point DRP scheme with 512^3 grid. The solutions obtained using $P=2$ have the largest error for the same DOF. This is a known property which is utilized in turbulent flow simulations using higher order methods. However, as shown in the previous paragraph, the differences between the fluxes are negligible.

To see if the solution converges, higher DOFs were compared to the DNS solution. This is depicted in Fig. 5. Although $\varepsilon\ (E_k)$ has converged to the DNS solution, $\varepsilon\ (\zeta)$ has not converged yet. This behavior was also observed by DeBonis[36] who performed a comparison between 4th, 8th and 12th order central finite difference schemes with a 13-point DRP scheme (DNS). Similar behavior has been found for DNS[41] solutions using DG methods.

Energy spectrum. The kinetic energy spectrum for all the cases is plotted at $t=10$ in Fig. 6. All the curves follow the standard turbulent spectrum of $-5/3$ slope. The differences between the spectrums for different order polynomials depicted in Fig. 6(A) are negligible. Note, the effect of flux is not significant on the energy spectrum.

Flow structures. The instantaneous iso-surface of Q – criterion (positive second scalar invariant of ∇u) colored with velocity magnitude is depicted in Fig. 7. The equation defining Q – criterion is provided in Eq. (39). The data corresponds to the simulation with $P=3$ (DOF $=320^3$). The coherent structures keep breaking down into smaller structures as the time progresses and finally around $t=9$ the flow becomes fully turbulent.

$$Q=\frac{1}{2}[|\Omega|^2-|\mathbf{S}|^2];\ \ \Omega=\frac{1}{2}[\nabla\mathbf{v}-(\nabla\mathbf{v})^T];\ \ \mathbf{S}=\frac{1}{2}[\nabla\mathbf{v}+(\nabla\mathbf{v})^T] \tag{39}$$

Parallel Algorithms

To parallelize the MIG code, open MPI was used and the code was tested at the University of Florida high performance computing center. All the tests were run on servers with Intel E5-2698 v3 processors with the capability to achieve HPL R_{max} of 7.381×10^5 GFlops. The domain was decomposed lexicographically with equal elements in each processor. The solution time for N-S equations was studied for processor counts of 1, 8, 16, 32, 64, 128, 256 and 512. The parallel performance is studied by solving the time explicit N-S equations for the Taylor Green vortex isotropic turbulence problem. Two cases were tested with total number of elements, 32^3 (DOF $=5570560$) and 64^3 (DOF $=44545480$). A small number of elements was chosen to result in a partitioning scheme with significant communication time with respect to the calculations performed. The problem is run for 100 time steps to average out the total time duration and the all the tests are repeated three times.

Figure 8(A) shows that the speedup on a log-log plot is similar for both 32^3 and 64^3 cases up to 512 processors. The power data fit to 32^3 case shows a speedup slope of 0.94 while for 64^3 it shows 0.95. Based on the data fit the parallel speedup (speedup/ideal) efficiency ranges from 99% for 8 processors to 63% for 512 processors. In Fig. 8(B) the speedup is plotted on a linear scale and the 32^3 case starts to plateau due to increase in communication time between processors while the 64^3 case maintains a linear slope. The processors show different performances for different runs since each case is not run on the same server, which gives a deviation in speedup of up to 5%. The initial higher speedup for the 32^3 case compared to the 64^3 case is within this tolerance limit. Further improvements can be made by using non – blocking instead of blocking MPI send and receive commands. Also optimizing the domain decomposition can lower communication time.

The convergence study is shown in Fig. 9 compares the convergence rates for different orders of polynomials. As evident from the plots, the higher order methods show the higher convergence rates in agreement with theory.

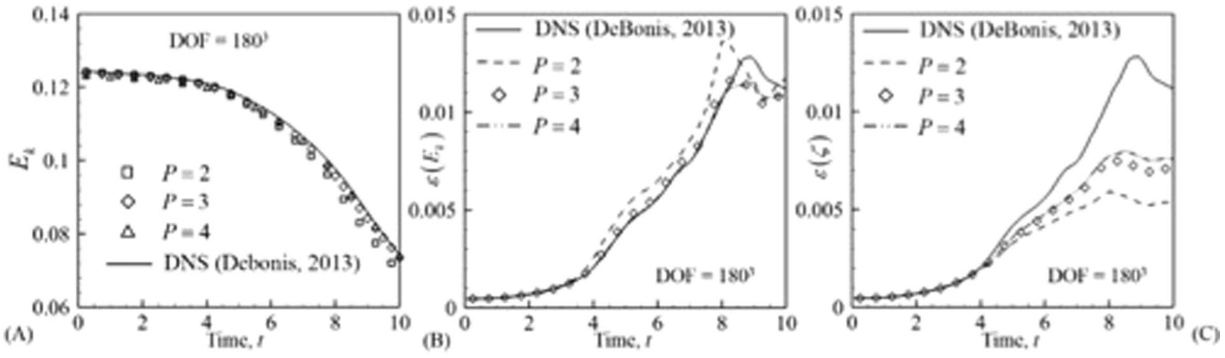

Figure 4. Comparison of the MIG DG solution with published DNS results[36]. (**A**) Turbulent kinetic energy, (**B**) energy dissipation rate based on integral kinetic energy and (**C**) energy dissipation rate based on enstrophy.

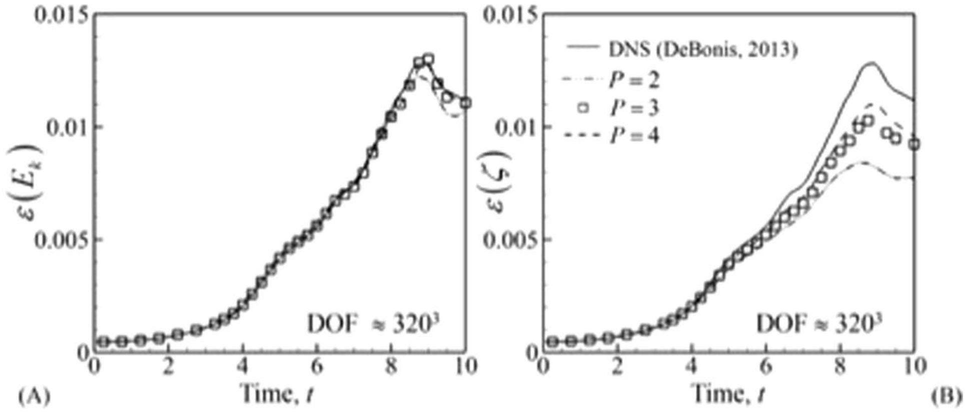

Figure 5. Comparison of turbulent kinetic energy dissipation rate for different order of spatial accuracy at approximately 320³ DOF. (**A**) Dissipation rate based on integral kinetic energy and (**B**) enstrophy.

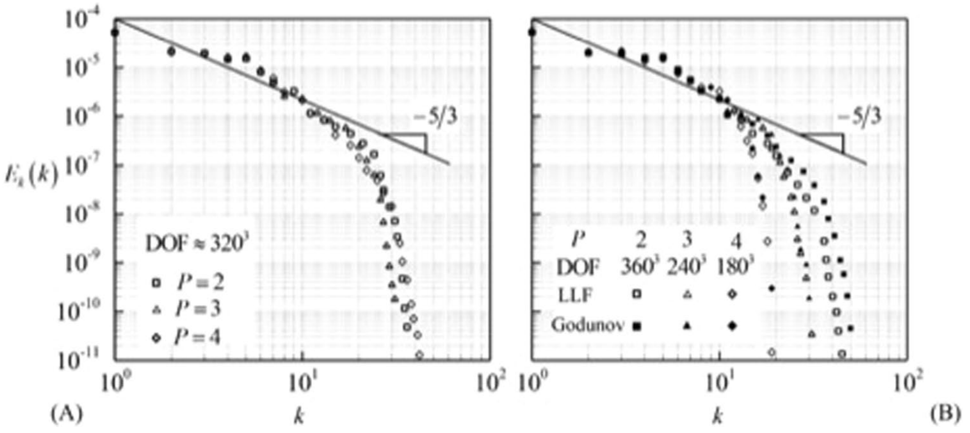

Figure 6. Kinetic energy spectrum for Taylor Green vortex problem at $t = 10$. (**A**) Effect of polynomial order and (**B**) effect of inviscid flux on energy spectrum.

A similar scaling study was performed on the Stampede2 machine at the Texas Advanced Computing Center. The study was performed on the new Intel Knights Landing architecture. This new hardware promises much better performance through improved memory bandwidth and larger cache memory per computational core.

The Knights Landing processor from intel consists of 36 active tiles each consisting of 2 processing computational cores, hence having a maximum total of 72 processing cores per compute node (Fig. 10). The cores are connected to each other via a two dimensional on-die ring type interconnect which can deliver an aggregate data bandwidth in excess of 700 gigabytes per second. Each tile containing 2 processing cores shares a 1-megabyte level 2 cache and each compute core has its individual L1 instruction and data caches of 32 kilobytes respectively.

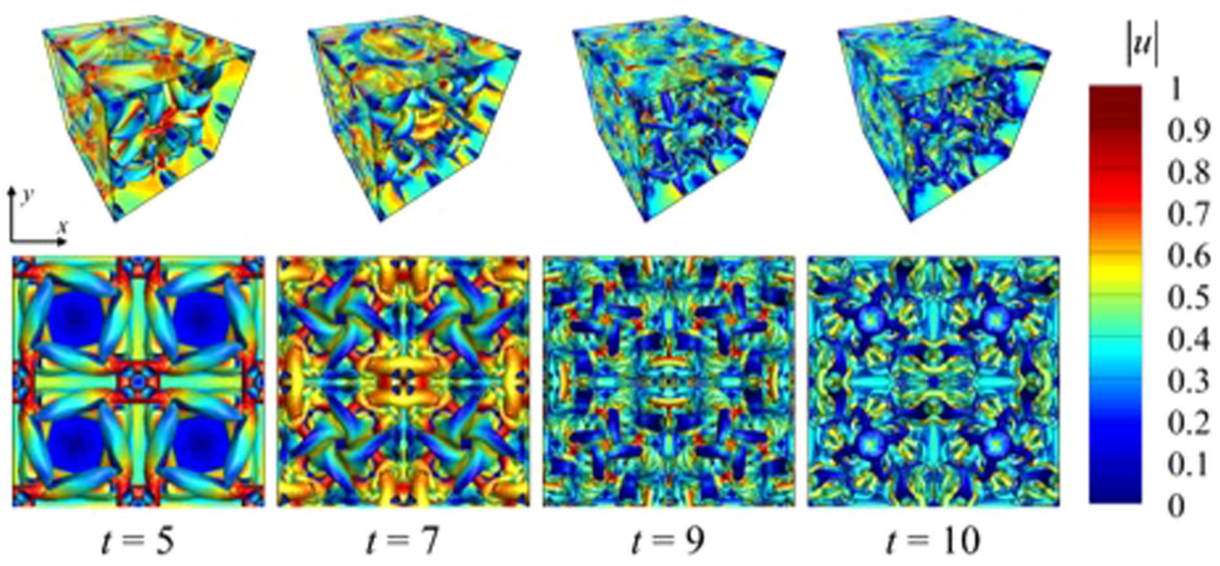

Figure 7. Instantaneous Q – criterion colored with velocity magnitude showing breakdown of coherent structures with time for a Taylor Green vortex problem.

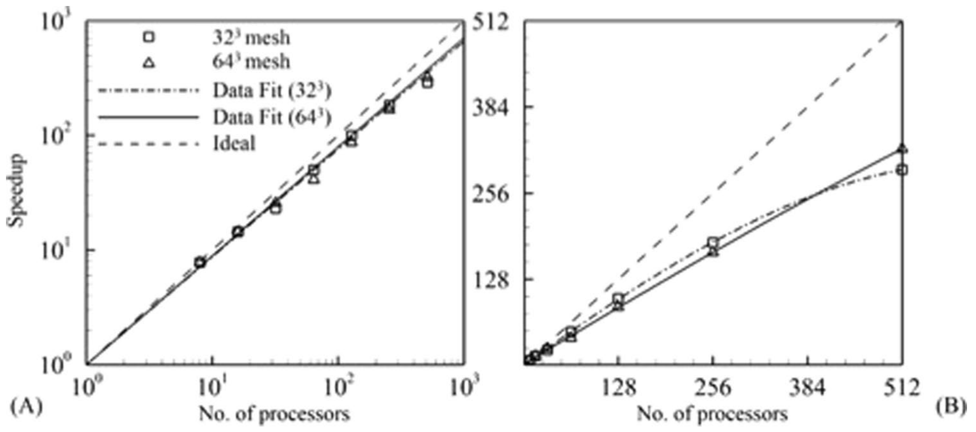

Figure 8. Parallel performance for different number of elements. (**A**) Comparison of speedup on a log-log plot with data fit using power curves and (**B**) speedup on a linear scale plot with data fit using quadratic polynomial.

Additionally, each core has two vector processing units (VPUs) which allows for very fast floating-point arithmetic operations in parallel.

The local nature of the discontinuous Galerkin method makes it a prime candidate for peak performance on such a vectorized architecture. The fundamental idea behind the Galerkin method is the repeated interpolation and integral computations within each discretized element. As higher orders of the approximation polynomials are employed the amount of computations being performed on the data increases exponentially. This ideally suits the kind of architecture possessed by the Knights Landing processor which relies on bringing in chunks of data from higher level, slow memory like conventional RAM (Random Access Memory) to large caches less frequently and then working on them for longer periods of time.

Figures 11 and 12 show the scaling of different orders of polynomials with numerical accuracy via MPI and OpenMP implementations of the MIG code. It is quite evident from the initial results that the higher order polynomial solutions show the best promise of parallel scaling. The MPI scaling showed the most dramatic difference between the higher order polynomials. For 100 MPI tasks, the parallel efficiency was 56.6% for $P = 2$, 79.5% for $P = 4$ and 89.0% for $P = 6$. At 400 MPI tasks, the efficiency fell to 55.7% for $P = 2$ and 77.4% for $P = 4$. The OpenMP scaling showed much less of a difference between the three polynomial orders. The code was run for as many as 64 OpenMP threads on the Knights Landing processor. The highest order tested ($P = 6$) was marginally more efficient, with the biggest difference occurring as the number of threads was increased. For 64 threads, the efficiencies were 90.6%, 92.0% and 92.5% for $P = 2$, 4, and 6, respectively. Overall, the OpenMP efficiency was greater than the MPI efficiency for equivalent number of threads and tasks. As described earlier, improvements to the MPI implementation and domain decomposition method can improve the efficiency of the MPI scaling.

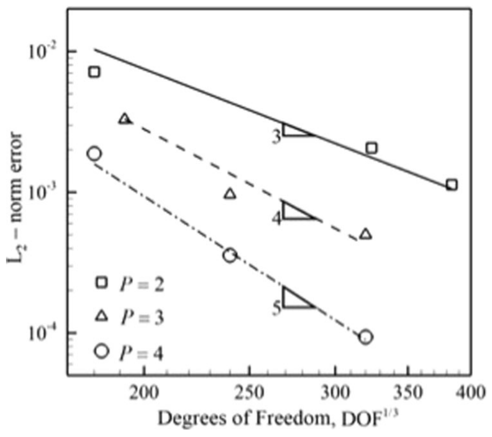

Figure 9. Comparison of rate of convergence for $P=2$, $P=3$ and $P=4$ uniform rectangular elements using the LDG scheme to solve Navier-Stokes equations for isotropic turbulence problem.

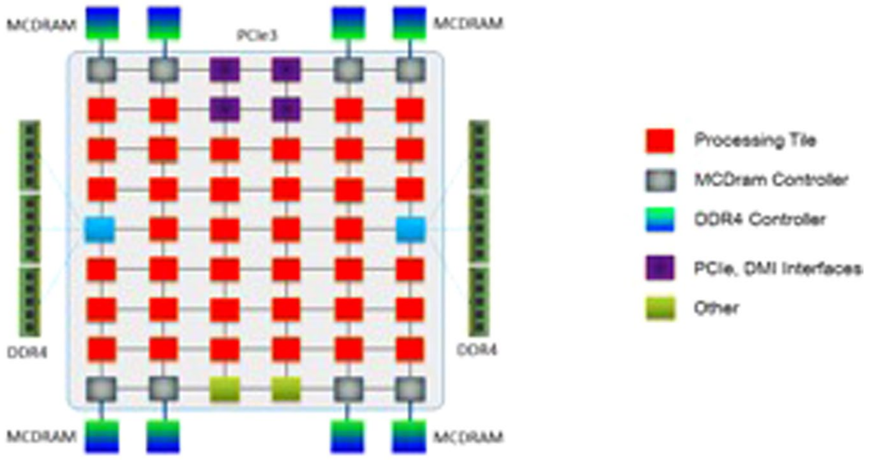

Figure 10. Block Diagram of Intel Knights Landing Processor Architecture (from Intel Website).

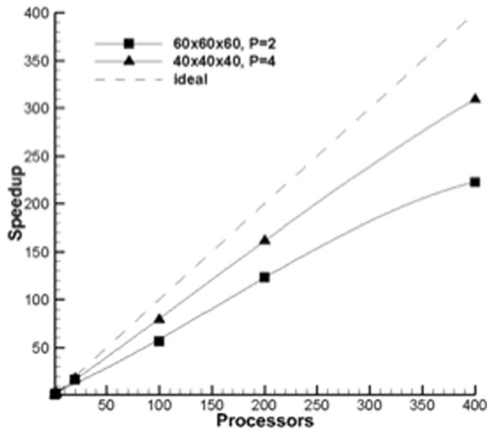

Figure 11. Parallel scaling of MIG code via MPI.

For scientific or an engineering problems one is interested in the eventual computational cost for a pre-specified level of acceptable error in the simulations. The combination of a better error convergence rate and superior scaling for higher order methods leads to the evaluation that for reasonably large numerical problem the computational cost for a specific level of numerical error will be the lowest for higher order methods.

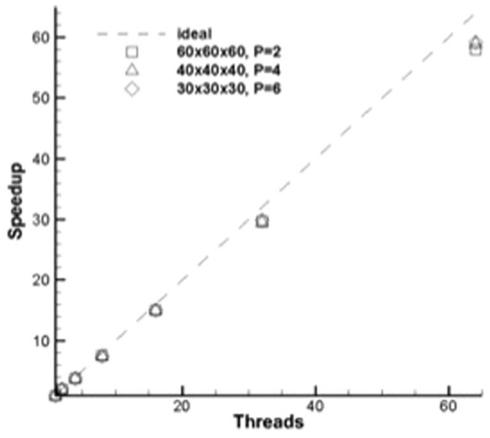

Figure 12. Parallel scaling of MIG code via OpenMP.

Parameter	Value
u_∞	745.2 m/s
p_∞	10.13 kPa
T_∞	273 K
T_∞	2.25
Re_∞	400–1350

Table 5. Freestream conditions for the turbulent boundary layer flow.

Mach 2.25 Turbulent Boundary Layer Flow

The scheme described in the Numerical Method was used to simulate the zero pressure gradient developing turbulent boundary layer at Mach 2.25. This case serves as a good stepping stone for high speed turbulent flow simulations with higher freestream Mach numbers. The primary challenge in simulating high speed turbulent boundary layers is tied to the large temperature gradient that develops across the boundary layer caused by the increased influence of viscous dissipation (Spina *et al.* 1994).

The freestream conditions and the corresponding range of the momentum thickness-based Reynolds number $\mathrm{Re}_\theta = u_\infty \theta / \nu_\infty$ (based on the freestream viscosity as opposed to the wall viscosity) for the present case are listed in Table 5. In order to transition the flow to turbulence, the flow is tripped using the method of Schlatter and Örlü (2012) through bypass transition. The tripping body force is in the wall-normal direction, and it is given by

$$f_y = \exp\left[[(x - x_0)/\ell_x]^2 - \left[y/\ell_y\right]^2\right] g(z, t) \tag{40}$$

The forcing function g(z,t) in Eq. (12) fluctuates as a function of time, and it also contains a random coefficient which varies in the spanwise direction. The full form of the forcing function is given in Schlatter and Örlü (2012).

The computational domain consists of $N_x \times N_y \times N_z = 900 \times 64 \times 64$ finite elements. Within each element, a modal basis function representation is employed. Quadratic Legendre basis functions (p = 2) with third-order spatial accuracy are used to interpolate the solution. The grid in the direction normal to the wall is stretched, with the smallest grid spacing (based on the inner wall units) of $\Delta y^+ = 0.5$ at the wall. In the spanwise and streamwise directions, the grid is uniform. The boundary conditions for the computational domain are as follows. The laminar boundary layer solution is used as an inflow boundary condition for the simulation. At the wall, the adiabatic and no-slip boundary conditions are prescribed. In the spanwise direction, symmetric boundary conditions are enforced. In the streamwise direction, a fringe region is added at the outlet to eliminate reflections from the outlet boundary. This concept has been successfully used in simulations of turbulent boundary layers in the past, *e.g.* Spalart and Watmuff[42].

Instantaneous flow field. The features of the instantaneous flow are studied in Fig. 13, which gives the plot of the Q criterion iso-surfaces at the value of Q = 3. The iso-surfaces are colored by the magnitude of the streamwise velocity for a momentum thickness Reynolds number Re_θ in the range of 400-1350. The plane below the iso-surfaces represents the flat plate. The flow is tripped and the initial coherent structures quickly break down into a fully turbulent flow. Asymmetric one-legged hairpin vortices can be observed along with the more typical structures. The flow in Fig. 13 is plotted after both the initial transients disappeared and the mean flow calculations were carried out, corresponding roughly to three flow-through times (the fluid convecting three times over the streamwise length of the plate).

Figure 14 shows the instantaneous normalized density, normalized streamwise velocity and temperature fields for the flow after two flow-through times. The slice location of the xy-plane corresponds to the center of the

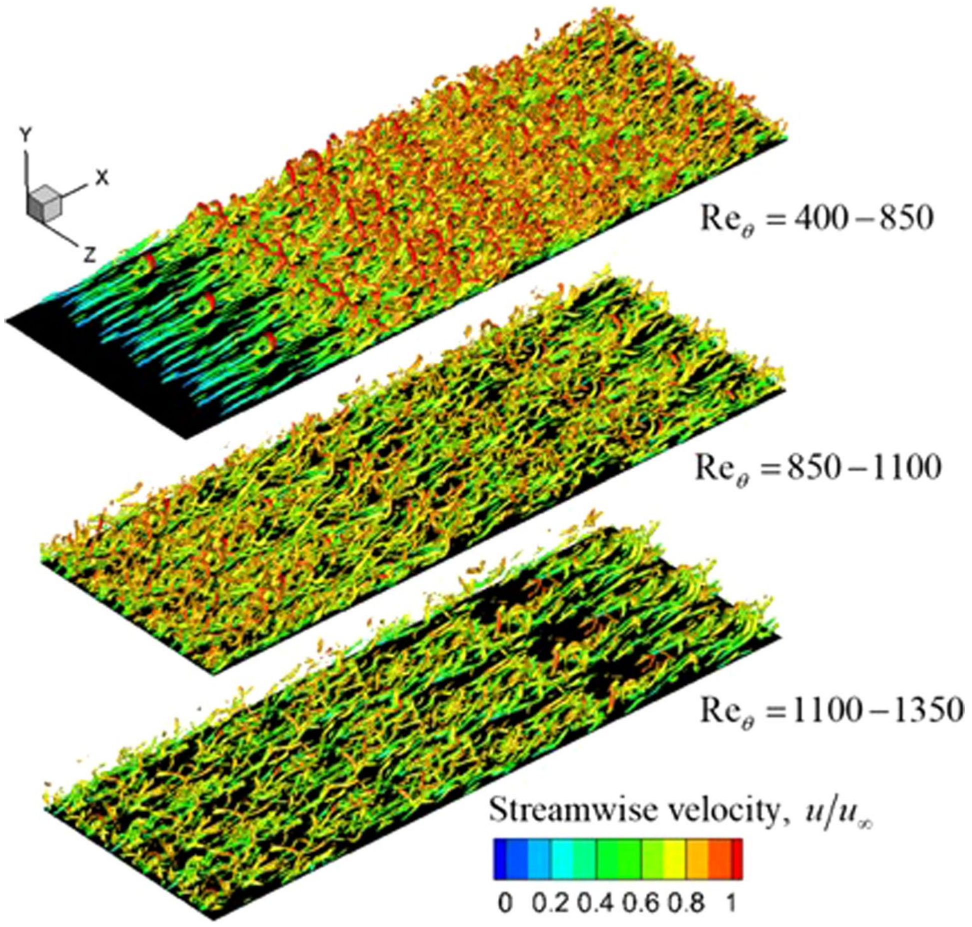

Figure 13. Iso-surfaces of the instantaneous normalized Q criterion (Q = 3). The domain is duplicated three times for visualization purposes.

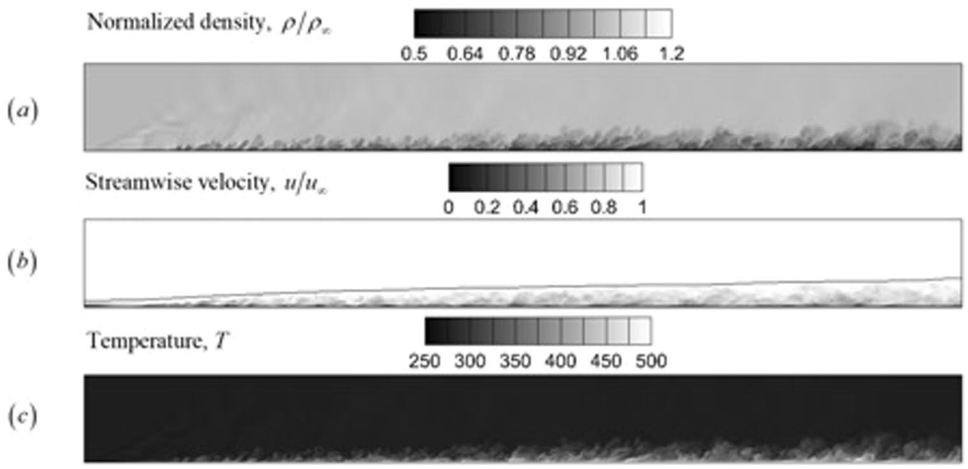

Figure 14. Instantaneous contours for the turbulent boundary layer at M = 2.25. (**a**) Normalized density field, (**b**) normalized instantenous velocity, and (**c**) temperature (K) are plotted.

domain in the spanwise direction (z = 1.5 mm and k = 32). The height of the domain in the wall normal direction was found to be sufficient to avoid the reflections from the top boundary destroying the rest of the solution. A large-scale structure angle of about 45 degrees has been observed for this problem, along with a shallower 10 degree angle for structures closer to the wall. These angles can also roughly be seen in Fig. 14, despite the lesser level of resolution in this study in comparison with that of Poggie[43].

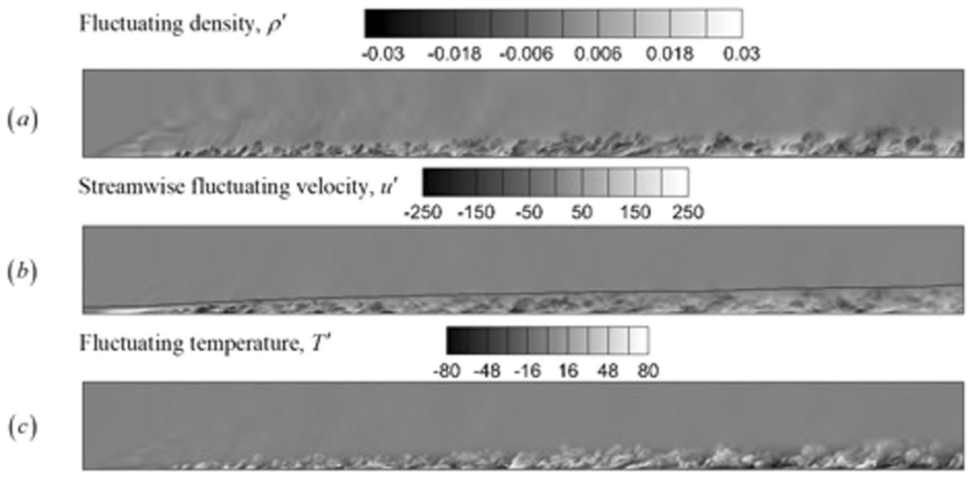

Figure 15. Instantaneous contours for the turbulent boundary layer at $M = 2.25$. Fluctuations of the (**a**) normalized density field, (**b**) normalized instantenous streamwise velocity, and (**c**) temperature (K) are plotted.

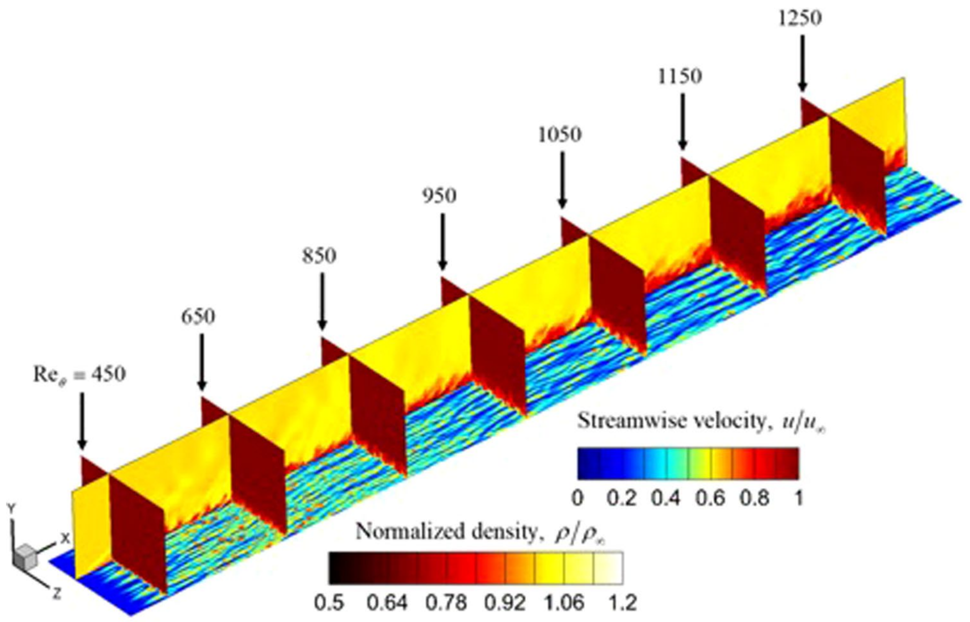

Figure 16. Streamwise velocity plotted at the wall-normal direction plane corresponding to $y \approx 6 \times 10^{-5}$ ($j = 15$) and several planes along the streamwise direction. Normalized density is plotted for a spanwise plane to illustrate the boundary layer growth and the flow structures.

The ratio of the freestream temperature to the wall temperature matches the expected value that can be obtained from a similarity solution of a compressible laminar boundary layer with the adiabatic wall boundary condition. The growth of the boundary layer is shown in Fig. 14(b) by plotting the boundary layer thickness.

Figure 15 shows the fluctuations of the normalized density, normalized streamwise velocity and temperature fields. The xy-plane slice location is identical to that in Fig. 14. The fluctuations are plotted to offer additional visualization of the turbulent flow field. The high intensity fluctuations in the boundary layer show packets of fluid which are hotter and lighter than the freestream fluid and which are pushed upward as the boundary layer grows.

In Fig. 16, the flow structures along a wall normal plane are visualized by plotting the velocity contours at the $j = 15$ plane ($y \approx 6 \times 10^{-2}$ mm). The vortices are seen to produce low speed streaks which spread out as the flow becomes fully turbulent. The figure also shows that the fluid in the boundary layer is drawn upward away from the wall. The normalized density profile is plotted along various spanwise planes and shows the growth of the flow structures as the thickness of the boundary layer increases.

Mean Flow Field. Figure 17 shows the skin friction coefficient plotted against the momentum thickness Reynolds number in the range of $Re_\theta = 700–1200$. This range is chosen because the flow has already become fully turbulent. The skin friction decreases with increasing momentum thickness Reynolds number.

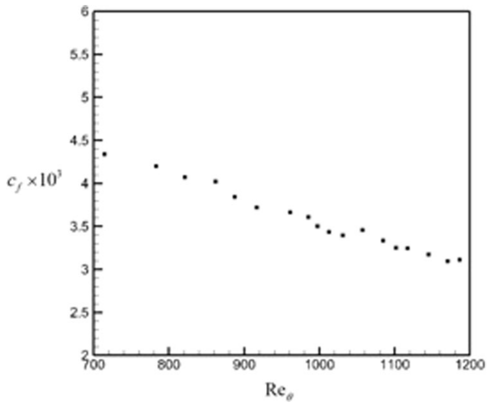

Figure 17. Skin friction coefficient plotted versus the momentum thickness-based Reynolds number in the range 700–1200.

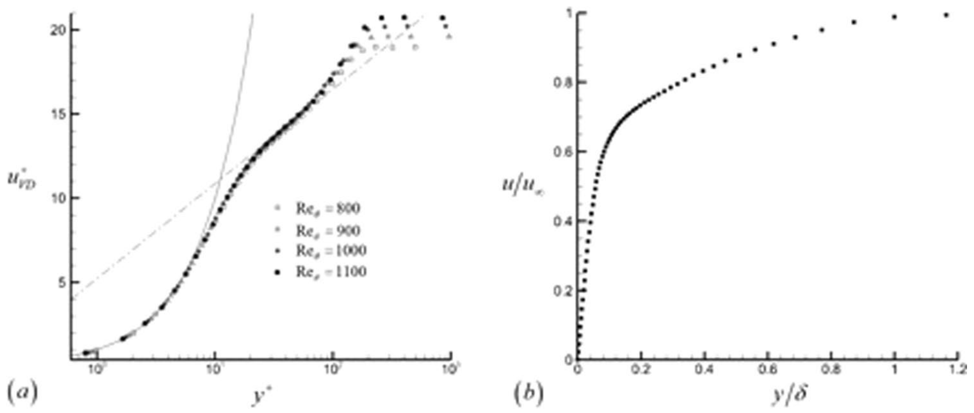

Figure 18. (**a**) The van Driest transformed velocity plotted in the inner wall coordinates for multiple downstream locations, and (**b**) the mean streamwise velocity plotted in the outer coordinatesat the downstream location corresponding to $Re_\theta = 1000$.

Figure 18 plots the mean velocity profiles for the turbulent boundary layer, scaled in both the inner and outer coordinates. For the inner wall coordinates, the mean velocity is plotted in the van Driest-transformed form[26]. The transformed velocity is given by

$$u^+_{VD} = \int_0^{u^+} \left(\frac{\overline{\rho}}{\overline{\rho}_w}\right)^{1/2} du^+ \tag{41}$$

The transformed velocity is plotted at downstream locations of $Re_\theta = 800–1100$. The streamwise velocity profiles are seen to collapse reasonably well to the law of the wall in the viscous sublayer (indicated by a solid line in Fig. 18) and the buffer layer (dashed and dotted line in Fig. 18) for this particular case.

Figures 19 and 20 give the Reynolds stresses scaled by the wall shear stress. The transformed Reynolds stress is plotted using the inner coordinates in Fig. 19 and the outer coordinates in Fig. 20. The transformed Reynolds stress is calculated as

$$R^+_{ij} = \frac{\overline{\rho}\overline{u_i' u_j'}}{\tau_w} = \frac{\overline{\rho}\overline{u_i' u_j'}}{\overline{\rho}_w u_\tau^2} \tag{42}$$

The Reynolds stresses are plotted at the downstream location of $Re_\theta = 1000$, which is not in close proximity to the outflow boundary but still sufficiently far from the flow tripping region. At this location, the peak value of the normal Reynolds stress occurs at approximately $y^+ = 11$ ($y/\delta = 0.06$). The wall normal and spanwise components of the Reynolds stress are smaller in comparison, and their peak values occur further away from the wall. After reaching their peak values, all the stresses decay with increasing y/δ sufficiently far away from the wall. The Reynolds stresses indicate that the majority of the turbulence is produced in the region of $y^+ = 10–100$, fairly close to the wall.

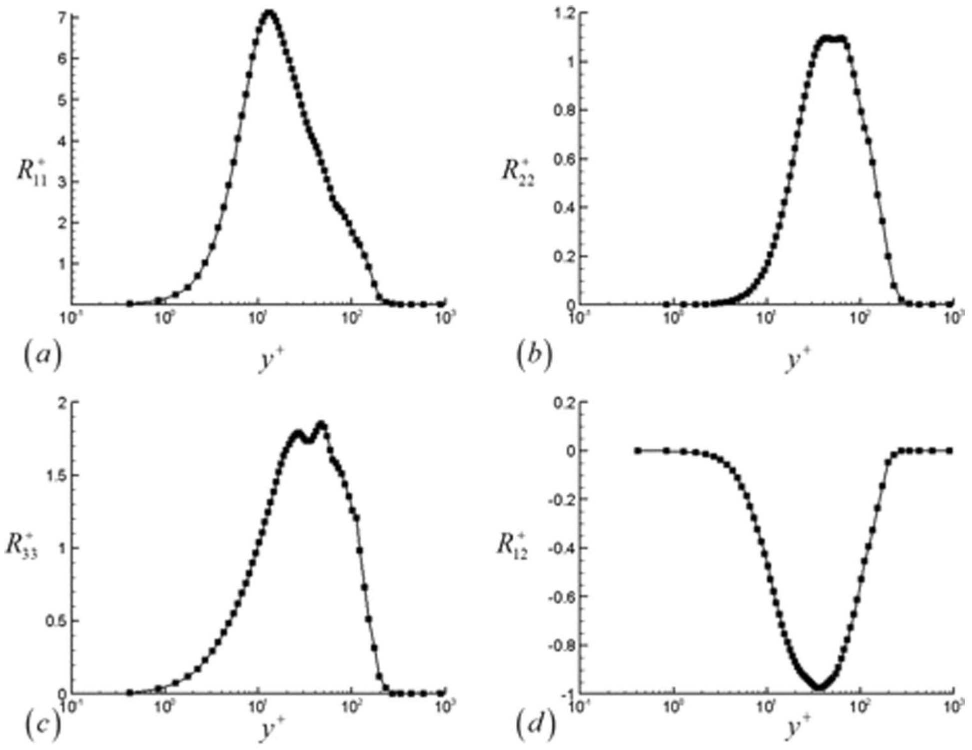

Figure 19. Transformed Reynolds stresses plotted in the inner wall coordinates at the downstream location corresponding to $\mathrm{Re}_\theta = 1000$. (**a**) Streamwise, (**b**) wall normal, (**c**) spanwise and (**d**) u′v′ components are shown.

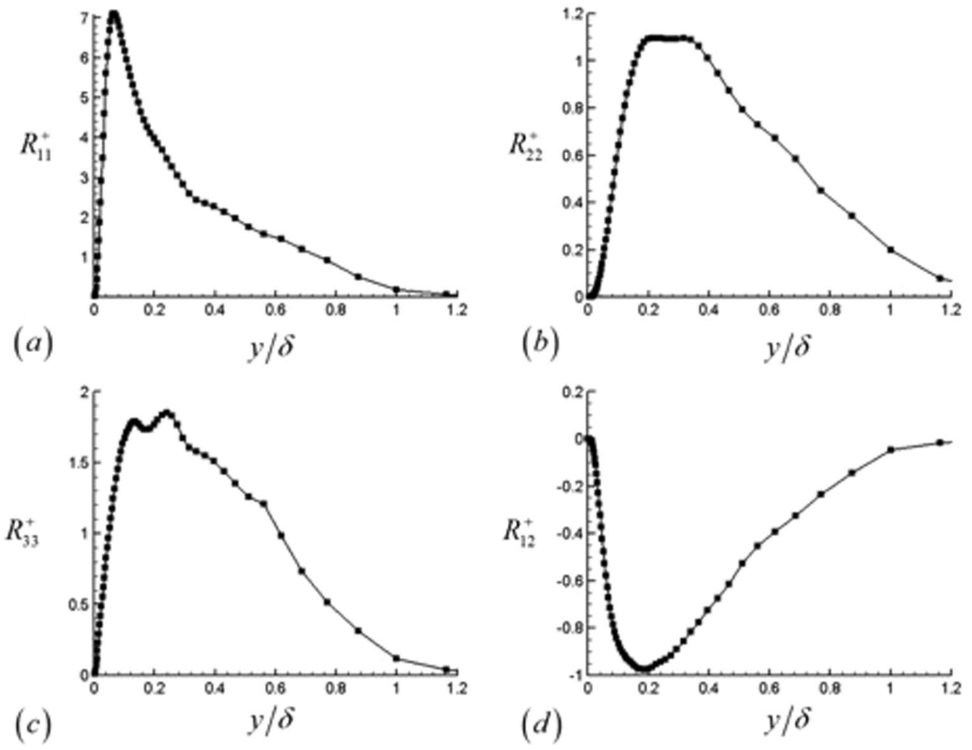

Figure 20. Transformed Reynolds stresses plotted in the outer coordinates at the downstream location corresponding to $\mathrm{Re}_\theta = 1000$. (**a**) Streamwise, (**b**) wall normal, (**c**) spanwise and (**d**) u′v′ components are shown.

Conclusions

A scalable, parallel, high-fidelity DG formulation was demonstrated for the test case of isotropic turbulent flow for a Taylor-Green vortex problem for Reynolds numbers ranging between 100–1600. The solutions for the DG ILES method were found to match up well with DNS results up to the Reynolds number of 1600, for which a higher resolution is necessary. Tests using the Godunov and LLF numerical fluxes showed negligible differences in the dissipation rates for polynomial orders beyond $P = 2$. The kinetic energy dissipation rate was found to converge to the DNS solution when a higher number of DOFs was used.

A Reynolds number of 1600 was then used to study the performance of different polynomial orders from $P = 2$ to $P = 6$ with regard to computational cost and scalability in parallel. The DG method demonstrated the advantages of the higher-order polynomials for parallel implementation. Namely, the higher-order polynomials showcased superior scalability and performance to achieve a given level of error over the range that was tested. The studies showed that increasing the order of the interpolating polynomial increased the parallel efficiency using both the MPI and OpenMP parallel implementations. The improvement in parallel efficiency was larger for the MPI implementation than for the OpenMP implementation. The MPI implementation showed improvements of 20–30% in parallel efficiency between $P = 2$ and $P = 6$ depending on the number of tasks. The difference between $P = 2$ and $P = 6$ for the OpenMP implementation was as small as 2%. It is not conclusive from this study whether this behaviour continues into higher order polynomials or if the efficiency saturates.

The same parallel framework was used to compute simulations of the development of a supersonic turbulent boundary layer at Mach 2.25. These computations employed nearly 33 million spatial degrees of freedom with the solution domain being approximated with quadratic Legendre polynomials. Numerical investigations into the physics such as the variation of the skin friction coefficient with the Reynolds number and characteristics of the Reynolds stress in the boundary layer are presented. These computations provide confidence in the capabilities of the numerical framework to perform more investigations to provide deeper physical insights in such phenomena in future work. The parallel scalability and efficiency of this framework suggests that such large studies can now be attainable in reasonable amount of time.

The focus of future studies is exploring polynomial orders beyond $P = 6$ to determine the diminishing returns on scaling when continuing to increase the polynomial order. In addition, domain decomposition will be performed by minimizing the surface area over which separate MPI tasks are required to communicate with each other. These changes should improve the scaling of the MPI implementation, which is not as optimized as the OpenMP scaling at the present time.

Data Availability

The data for the presented results is available and will be posted on a public link at APRG website.

References

1. Behr, M., Pressel, D. M. & Sturek, W. B. Comments on CFD code performance on scalable architectures. *Comput. Methods Appl. Mech. Engrg*. 263–277 (2000).
2. Gropp, W. D., Kaushik, D. K., Keyes, D. E. & Smith, B. F. High-performance parallel implicit CFD. *Parallel Computing*, 337–362 (2001).
3. Wang, Z. *et al*. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 811–845 (2013).
4. Deng, X., Mao, M., Tu, G., Zhang, H. & Zhang, Y. High-Order and High Accurate CFD Methods and Their Applications for Complex Grid Problems. *Commun. Comput. Phys*. 1081–1102 (2012).
5. Huerta, A., Angeloski, A., Roca, X. & Peraire, J. Efficiency of high-order elements for continuous and discontinuous Galerkin methods. *International Journal for Numerical Methods in Engineering*, 529–560 (2013).
6. Reed, W. H., & Hill, T. R. *Triangular mesh methods for the Neutron Transport equation*. Los Alamos, NM: LA-UR-73-479 (1973).
7. Cockburn, B. & Shu, C. W. The Runge-Kutta local projection P1 - discontinuous Galerkin method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis* **25**(3), 337–361 (1991).
8. Cockburn, B. & Shu, C. W. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Mathematics of Computation* **52**(186), 411–435 (1989).
9. Chen, Z. & Cockburn, B. Analysis of a finite element method for the drift-diffusion semiconductor device equations. *Numerische Mathematik* **71**(1), 1–28 (1995).
10. Chen, Z. A Finite element method for the quantum hydrodynamic model for semiconductor devices. *Computers & Mathematics with Applications* **31**(7), 17–26 (1996).
11. Bassi, F. & Rebay, S. A high-order accurate discontinuous finite element method for numerical solution of the compressible Navier-Stokes equations. *Journal of Computational Physics* **131**(2), 267–279 (1997).
12. Bassi, F., Crivellini, A., Rebay, S. & Savini, M. Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k–ω turbulence model equations. *Computer and Fluids* **34**(4), 507–540 (2005).
13. Arnold, D. N., Brezzi, F., Cockburn, B. & Marini, L. D. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis* **39**(5), 1749–1779 (2002).
14. Cockburn, B. & Shu, C. W. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis* **35**(6), 2440–2463 (1998).
15. Funaro, D. A multidomain spectral approximation of elliptic equations. *Numerical Methods for Partial Differential Equations* **2**(3), 187–205 (1986).
16. Funaro, D. & Gottlieb, D. Convergence results for pseudospectral approximations of hyperbolic systems by a penalty-type boundary treatment. *Mathematics of Computation* **57**(196), 585–596 (1991).
17. Hesthaven, J. S. & Gottlieb, D. A stable penalty method for the compressible Navier–Stokes equations: I. Open boundary conditions. *SIAM Journal on Scientific Computing* **17**(3), 579–612 (1996).
18. Hesthaven, J. S. A Stable Penalty Method for the Compressible Navier–Stokes Equations: II. One-Dimensional Domain Decomposition Schemes. *SIAM Journal on Scientific Computing* **18**(3), 658–685 (1997).
19. Hesthaven, J. S. A Stable Penalty Method for the Compressible Navier–Stokes Equations: III. Multidimensional Domain Decomposition Schemes. *SIAM Journal on Scientific Computing* **20**(1), 62–93 (1998).
20. Bey, K. S., Oden, T. J. & Patra, A. A parallel hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws. *Applied Numerical Mathematics* **20**(4), 321–336 (1996).

21. Baumann, C. E. & Oden, T. J. A discontinuous hp finite element method for convection—diffusion problems. *Computer Methods in Applied Mechanics and Engineering* **175**(3–4), 311–341 (1999).
22. Houston, P., Schwab, C. & Suli, E. Discontinuous hp-finite element methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis* **39**(6), 2133–2163 (2002).
23. Roy, S. & Gaitonde, D. Radio Frequency induced ionized collisional flow model for application at atmospheric pressures. *Journal of Applied Physics* **96**(5), 2476–2481 (2004).
24. Bhatia, A., Roy, S. & Gosse, R. Effect of dielectric barrier discharge plasma actuators on non-equilibrium hypersonic flows. *Journal of Applied Physics* **116**(16), 164904 (2014).
25. Das Gupta, A. & Roy, S. Discontinuous Galerkin Method for Solving Magnetohydrodynamic Equations. *53rd AIAA Aerospace Sciences Meeting*. Kissimmee, Florida: AIAA (2015).
26. Das Gupta, A. & Roy, S. Three-dimensional plasma actuation for faster transition to turbulence. *J. Phys. D: Appl. Phys.* **50**, 425201 (2017).
27. Cockburn, B. Discontinuous Galerkin Methods for Convection-Diffusion Problems. In T. J. Barth, & H. Deconinck (Eds), *High-Order Methods for Computational Physics* (pp. 69–224). Springer (2001).
28. Shu, C. W. & Osher, S. Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes II. *Journal of Computational Physics* **83**(1), 32–78 (1989).
29. Douglas, J. Jr. & Dupont, T. Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods. *Computing Methods in Applied Sciences. Lecture Notes in Phys* **58**, 207–216 (1976).
30. Chavent, G. & Salzano, G. A finite element method for the 1D water flooding problem with gravity. *Journal of Computational Physics* **45**(3), 307–344 (1982).
31. Chavent, G. & Cockburn, B. The local projection P0 P1 discontinuous-Galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis* **23**(4), 565–592 (1989).
32. Orszag, S. A. Numerical simulation of the Taylor-Green vortex. In R. Glowinski, & J. L. Lions, *Computing Methods in Applied Sciences and Engineering Part 2. Lecture Notes in Computer Science* (Vol. 11, pp. 50–64). (Berlin, Heidelberg: Springer, 1974).
33. Brachet, M. E. Direct simulation of three-dimensional turbulence in the Taylor—Green vortex. *Fluid dynamics research* **8**(1–4), 1–8 (1991).
34. Brachet, M. E. *et al*. Small-scale structure of the Taylor–Green vortex. *Journal of Fluid Mechanics* **130**, 411–452 (1983).
35. Comte-Bellot, G. & Corrsin, S. Simple Eulerian time correlation of full-and narrow-band velocity signals in grid-generated, 'isotropic' turbulence. *Journal of Fluid Mechanics* **48**(2), 273–337 (1971).
36. DeBonis, J. Solutions of the Taylor-Green vortex problem using high-resolution explicit finite difference methods. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. Grapewine, Texas (2013).
37. Fureby, C. & Grinstein, F. F. Large eddy simulation of high-Reynolds-number free and wall-bounded flows. *Journal of Computational Physics* **181**(1), 68–97 (2002).
38. Gassner, G. J. & Beck, A. D. On the accuracy of high-order discretizations for underresolved turbulence simulations. *Theoretical and Computational Fluid Dynamics* **27**(3–4), 221–237 (2013).
39. Hickel, S., Adams, N. A. & Domaradzki, J. A. An adaptive local deconvolution method for implicit LES. *Journal of Computational Physics* **213**(1), 413–436 (2006).
40. Beck, A. D., Gassner, G. J. & Munz, C. D. On the effect of flux functions in discontinuous Galerkin simulations of underresolved turbulence. In M. Azaïez, H. El Fekih, & J. Hesthaven, *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM 2012. Lecture Notes in Computational Science and Engineering* (Vol. 95, pp. 145–155). Springer, Cham (2014).
41. Chapelier, J. B., De La Llave Plata, M., Renac, F. & Lamballais, E. Evaluation of a high-order discontinuous Galerkin method for the DNS of turbulent flows. *Computers & Fluids* **95**, 210–226 (2014).
42. Spalart, P. R. & Watmuff, J. H. Experimental and numerical study of a turbulent boundary layer with pressure gradients. *Journal of Fluid Mechanics* **249**, 337–371 (1993).
43. Poggie, J. Compressible Turbulent Boundary Layer Simulations: Resolution Effects and Turbulence Modeling. AIAA Paper 2015–1983 (2015).

Acknowledgements
This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. TH and SR acknowledge the support of this work by the AFRL Computational Sciences Center and the DoD HPCMP's User Productivity Enhancement, Technology Transfer, and Training (PETTT) Program (Contract No: GS04T09DBC0017 through Engility Corporation).

Author Contributions
R.G. and S.R. conceived the research; T.H. and A.D. implemented the formulation and performed the numerical experiments; T.H., A.D. and S.G. made the figures; T.H., A.D.G., S.G. and S.R. analyzed the data and wrote the paper; R.G. and S.R. supervised the project.

Additional Information
Competing Interests: The authors declare no competing interests.