Overview of the NASA Glenn Flux Reconstruction Based High-Order Unstructured Grid Code

Seth C. Spiegel,* James R. DeBonis† and H. T. Huynh,‡
NASA Glenn Research Center, Cleveland, OH, 44135, USA

A computational fluid dynamics code based on the flux reconstruction (FR) method is currently being developed at NASA Glenn Research Center to ultimately provide a large-eddy simulation capability that is both accurate and efficient for complex aeropropulsion flows. The FR approach offers a simple and efficient method that is easy to implement and accurate to an arbitrary order on common grid cell geometries. The governing compressible Navier-Stokes equations are discretized in time using various explicit Runge-Kutta schemes, with the default being the 3-stage/3rd-order strong stability preserving scheme. The code is written in modern Fortran (i.e., Fortran 2008) and parallelization is attained through MPI for execution on distributed-memory high-performance computing systems. An h-refinement study of the isentropic Euler vortex problem is able to empirically demonstrate the capability of the FR method to achieve super-accuracy for inviscid flows. Additionally, the code is applied to the Taylor-Green vortex problem, performing numerous implicit large-eddy simulations across a range of grid resolutions and solution orders. The solution found by a pseudo-spectral code is commonly used as a reference solution to this problem, and the FR code is able to reproduce this solution using approximately the same grid resolution. Finally, an examination of the code's performance demonstrates good parallel scaling, as well as an implementation of the FR method with a computational cost/degree-of-freedom/time-step that is essentially independent of the solution order of accuracy for structured geometries.

I. Introduction

Accurate and efficient numerical methods for the simulation of complex aerodynamic flows are desired to help reduce the design and development costs of aerodynamic vehicles and aeropropulsion systems. The primary challenge lies in the prediction of the turbulent motion within the flow and its effect on the overall motion of the fluid. Advanced methods such as large-eddy simulation (LES) and direct numerical simulation (DNS) offer great advancements in predicting turbulence, but these methods require significant improvements in accuracy and efficiency in order to become widely used. Focusing on either the accuracy or efficiency aspect of the underlying numerical algorithms, when developing a computational fluid dynamics (CFD) code, usually demands concessions from the other parameter.

Promising results have been found applying LES in the realm of aeropropulsion, but the spatial and temporal resolution needed for LES requires high-order (higher than second) numerical schemes historically only found in structured grid methods with large stencils. These methods have numerous drawbacks: difficulty in grid generation, high sensitivity to grid quality, complex boundary conditions, and poor scalability for parallel computing. These drawbacks severely limit the ability of these codes to compute complex aeropropulsion configurations, e.g., noise suppressing nozzles.[1]

In an attempt to alleviate the difficulties of structured grid generation for complex geometries, researchers have applied LES methods to unstructured grids.[2,3] However, the limited second-order accuracy of current unstructured methods requires an exorbitant number of grid points and offers little incentive over the traditional structured methods.[4,5] The flux reconstruction (FR) method[6–8] provides a simple, efficient, and

*NASA Postdoctoral Fellow, Inlets and Nozzles Branch, 21000 Brookpark Road, and AIAA Member.
†Aerospace Engineer, Inlets and Nozzles Branch, 21000 Brookpark Road, and AIAA Associate Fellow.
‡Aerospace Engineer, Inlets and Nozzles Branch, 21000 Brookpark Road, and AIAA Member.

easy to implement method for solving the Navier-Stokes equations on unstructured grids and is accurate to an arbitrary order. By employing the high-order FR method with LES on unstructured grids, limitations related to structured grid generation, grid resolution, and computational efficiency can be overcome.[9–12]

Section II begins by providing an overview of the FR code that is the subject of this work. This overview includes a brief review of the governing equations in section II.A, brief descriptions of the numerical methods used to discretize the governing equations in space and time in section II.B, and a few details regarding the code development in sections II.C and II.D. Two canonical problems are then used in section III to verify the capabilities of this new code. The first problem consists of modeling a stationary isentropic vortex for an indefinite period of time, and the results are presented in section III.A. This version of the isentropic vortex is unchanging with time; therefore, an h-refinement study allows for a quantifiable measurement of the code's order of accuracy since any deviation in the solution from the initial conditions is from numerical error. In section III.B, the Taylor-Green vortex problem is used to verify the implicit LES capabilities of this code for varying combinations of grid resolution and solution order. The results of these simulations are compared to those found by a pseudo-spectral code commonly used as a reference solution to this problem. Finally, a summary of the code and the numerical results are provided in section IV.

II. Code Description

A new CFD code named GFR (Glenn FR) is currently being developed at NASA Glenn Research Center to ultimately provide efficient LES capabilities using the FR method. This section describes some of the features and capabilities currently found within this new code, and begins with a brief review of the differential form of the compressible Navier-Stokes equations solved by GFR. Brief overviews of the FR method for discretizing the advection and diffusion terms of the governing equations are presented for the analogous 1D advection and diffusion equations, followed by the current methods available in GFR for advancing the solution in time. Finally, some details regarding the code development that separate the FR method (and related methods) from traditional finite-volume methods are discussed in the remaining sections.

II.A. Governing Equations

The 3D Navier-Stokes equations for a calorically perfect, compressible fluid can be written in differential form as

$$\frac{\partial}{\partial t}\left(\rho\right) + \frac{\partial}{\partial x_i}\left(\rho v_i\right) = 0 \tag{1a}$$

$$\frac{\partial}{\partial t}\left(\rho v_i\right) + \frac{\partial}{\partial x_j}\left(\rho v_i v_j + \delta_{ij} p\right) = \frac{\partial}{\partial x_j}\left(\tau_{ij}\right) \tag{1b}$$

$$\frac{\partial}{\partial t}\left(\rho e_t\right) + \frac{\partial}{\partial x_i}\left(\rho v_i e_t + v_i p\right) = \frac{\partial}{\partial x_i}\left(v_j \tau_{ij} - q_i\right) \tag{1c}$$

where ρ, v_i, p, and e_t respectively refer to the density, velocity in coordinate direction i, pressure, and the total energy per unit mass of the system, and δ_{ij} is the Kronecker delta function. Thermodynamic closure is found through the total energy equation

$$\rho e_t = \frac{p}{\gamma - 1} + \tfrac{1}{2}\rho v_i v_i \tag{2}$$

where γ is the ratio of specific heats for the fluid in question. The viscous stress tensor, τ_{ij}, is based on Stokes' hypothesis and given by the equation

$$\tau_{ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\frac{\partial v_k}{\partial x_k}\delta_{ij}\right) \tag{3}$$

Above, μ is the molecular viscosity and its relation to the temperature of the fluid, T, is modeled by Sutherland's law

$$\mu = \mu_0\left(\frac{T}{T_0}\right)^{\frac{3}{2}}\frac{T_0 + T_{\text{Suth}}}{T + T_{\text{Suth}}} \tag{4}$$

where μ_0 is the reference viscosity at the reference temperature T_0 and $T_{\text{Suth}} = 110.4\,\text{K}$. The heat flux vector, q_j, is modeled by Fourier's law

$$q_j = -\lambda \frac{\partial T}{\partial x_j} \tag{5}$$

where λ is the thermal conductivity which is related to the specific heat at constant temperature, c_p, and Prandtl number, Pr, through the relation

$$\lambda = \frac{c_p \mu}{\text{Pr}} \tag{6}$$

Additionally, with the specific gas constant denoted as R_{gas}, the temperature, pressure, and density of the fluid are related through the ideal gas law

$$p = \rho R_{\text{gas}} T \tag{7}$$

and the speed of sound, a, is given by the relations

$$a = \sqrt{\frac{\gamma p}{\rho}} = \sqrt{\gamma R_{\text{gas}} T} \tag{8}$$

In the present work, the fluid is assumed to be calorically perfect air with the following quantities held constant: the Prandtl number is $\text{Pr} = 0.72$, the ratio of specific heats is $\gamma = 1.4$, and the dimensionalized specific gas constant is $R_{\text{gas}} = 287.15\,\text{J}/(\text{kg}\,\text{K})$.

II.B. Numerical Methods

The goal of this section is to present the base concepts behind the FR method and how it is implemented within GFR. We first summarize the discretization of the advection terms in equation (1) using the FR method initially presented by Huynh in Ref. 6. Next, the extension of FR for the diffusion terms is briefly discussed. Finally, we outline the methods for advancing the solution in time that are currently implemented within GFR.

The concepts of the FR method presented in this section are based on solving the 1D differential equations analogous to the advection and diffusion terms of the governing equations. This simplification is intended to give the reader a feel for how the method works, instead of a thorough and complex derivation of the method for a multidimensional system of coupled equations, e.g., the 3D Navier-Stokes equations. Tensor products are used to extend these ideas to higher dimensions for structured geometries, i.e., quadrilaterals and hexahedra, and detailed descriptions of the FR method for these geometries can be found in Ref. 6 and 7. Furthermore, details for extending the FR method to unstructured geometries can be found in Ref. 8 and 13, and Ref. 14 presents a matrix-oriented reformulation of the FR method that is generalized for all geometries.

II.B.1. Spatial Discretization of Advection Terms

To illustrate the idea of the FR method for the advection terms of equation (1), consider the 1D conservation law

$$u_t + f_x = 0 \tag{9}$$

where the subscripts t and x denote partial differentiation with respect to time and space, respectively. Let the computational domain $\mathbf{\Omega}$ be divided into an arbitrary number of nonoverlapping cells, and a mapping function, $r = \chi(x)$, is used to transform each cell from physical space to a common reference space/element. With r denoting the local coordinate variable that varies on the reference element $\mathbf{I} = [-1, 1]$, the inverse function $x = \chi^{-1}(r)$ similarly maps the reference element back to physical space. The details of such mappings, especially for higher dimensions and curved boundaries, are beyond the scope of this paper; the interested reader can find additional information in many texts on spectral-/finite-element type methods, including but not limited to Ref. 15–18. The remainder of this section focuses on using the FR method to solve the transformed version of equation (9) within each reference element.

At a given time t, the exact solution to equation (9) is approximated locally within each cell as a polynomial of degree P, and this polynomial approximation is generally discontinuous across cell interfaces. Let $\boldsymbol{\xi}$, with ξ_n, $n = 1, 2, \ldots, N_P$, denote a set of coordinate values on $\mathbf{I}$ that give the location of the $N_P = P + 1$ solution points within the reference element; the Gauss points are used as the solution points

for all simulations in this work. Let u_n denote the known value of the solution polynomial at each solution point, i.e., $u_n = u(\xi_n)$. The Lagrange polynomial is defined as the interpolating polynomial through a set of nodal points, for example $\boldsymbol{\xi}$, that takes the value 1 at ξ_n and zero at all other points $\xi_{i\neq n}$:

$$\ell_n(r) = \prod_{\substack{i=1 \\ i\neq n}}^{N_P} \frac{r - \xi_i}{\xi_n - \xi_i} \tag{10}$$

The Lagrange polynomials serve as basis functions to construct the polynomial solution in each cell using

$$u(r) = \sum_{n=1}^{N_P} u_n \ell_n(r) \tag{11}$$

Since each u_n is simply a coefficient in equation (11), the derivative of this polynomial is just a linear combination of the derivatives of the Lagrange polynomials

$$\frac{d}{dr}u(r) = u_r(r) = \sum_{n=1}^{N_P} u_n \left[\frac{d}{dr}\ell_n(r)\right] \tag{12}$$

Here and in the remainder of this work, the subscript r is exclusively used to denote differentiation with respect to the local coordinate variable r of a polynomial function defined on the reference element. Note that this same idea can be used to compute the derivative for any polynomial function whose values are known at the solution points.

Using the nodal solutions, we can evaluate the value of the flux function at each solution point, i.e., $f_n^{\mathrm{D}} = f(u_n)$. The superscript 'D' is used to identify it as the *discontinuous flux function* because the resulting function only uses information inside a given cell and does not account for interaction with data from the neighboring cells. Since we know the value of the discontinuous flux at each solution point in a cell, we can compute its derivative using equation (12). However, if we employ the derivative of f^{D} to evaluate f_x in equation (9), we obtain erroneous solutions since such a derivative does not include the interaction of data across cell interfaces. Therefore, we seek to construct a so-called *continuous flux function*, denoted by $F(r)$, which accounts for this interaction between adjacent cells and approximates the discontinuous flux function in some sense, to which we can then calculate its derivative. The continuous flux function will be obtained by adding a correction to f^{D}, the discontinuous one.

To this end, at each (cell) interface, let u_{L} and u_{R} be the values taken on by the solution polynomials to its left and right, respectively. An upwind flux value common for the two adjacent cells, denoted by $f^* = f^*(u_{\mathrm{L}}, u_{\mathrm{R}})$, can then be defined via the wind direction. Here, for the case of the Euler or Navier-Stokes equations, we use Roe's flux[19] with an entropy fix.[20]

Let f^*_{LB} denote the upwind flux at the 'left boundary' of the cell and f^*_{RB} denote the upwind flux at the 'right boundary' of the cell. We can then construct the continuous flux function, $F(r)$, as a polynomial of degree $P+1$ by requiring that it takes on the upwind flux values at the two cell interfaces, and it approximates f^{D} via $P-1$ conditions. Instead of defining F, we define $F - f^{\mathrm{D}}$, which approximates the zero function. At the two interfaces, $r = \pm 1$, the function $F(r) - f^{\mathrm{D}}(r)$ takes on the following values, which are called the flux jumps:

$$F(-1) - f^{\mathrm{D}}(-1) = f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1) \qquad \text{and} \qquad F(1) - f^{\mathrm{D}}(1) = f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1) \tag{13}$$

Therefore, $F(r) - f^{\mathrm{D}}(r)$ has prescribed left and right interface values, is of degree $P+1$, and approximates zero.

We now separate the prescription of the jump at the left interface from that of the right. This separation plays a critical role in the new approach. Again, using the local coordinate, let g_{LB} be the *correction function* for the left interface of the reference element defined by

$$g_{\mathrm{LB}}(-1) = 1, \qquad g_{\mathrm{LB}}(1) = 0 \tag{14}$$

and g_{LB} is a polynomial of degree $P+1$ approximating the zero function in some sense. We always choose g_{RB} by reflection: $g_{\mathrm{RB}}(r) = g_{\mathrm{LB}}(-r)$. As a result,

$$g_{\mathrm{RB}}(-1) = 0, \qquad g_{\mathrm{RB}}(1) = 1 \tag{15}$$

All results within the present work use the Radau polynomials as the correction functions and this choice has been shown to recover a nodal discontinuous Galerkin (DG) method; further discussions regarding the choices in correction functions can be found in Ref. 6, 21–23.

To begin constructing the continuous flux function, we start by considering the left interface, $r = -1$. The polynomial

$$f^{\mathrm{D}}(r) + \left[f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)\right] g_{\mathrm{LB}}(r) \tag{16}$$

provides a correction at the left interface for $f^{\mathrm{D}}(r)$ by changing the flux value at this interface from $f^{\mathrm{D}}(-1)$ to f^*_{LB}, while leaving the value at the right interface unchanged, namely, $f^{\mathrm{D}}(1)$. Next, the polynomial

$$F(r) = f^{\mathrm{D}}(r) + \left[f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)\right] g_{\mathrm{LB}}(r) + \left[f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1)\right] g_{\mathrm{RB}}(r) \tag{17}$$

provides corrections to both interfaces; using equations (14) and (15), one can easily verify that $F(-1) = f^*_{\mathrm{LB}}$ and $F(1) = f^*_{\mathrm{RB}}$. Thus, the above $F(r)$ is of degree $P + 1$, takes on the upwind flux values at the two interfaces, and approximates $f^{\mathrm{D}}(r)$ in the same sense that g_{LB} and g_{RB} approximate the zero function.

The derivative of $F(r)$ at the solution point ξ_n is evaluated as

$$\left.\frac{\partial\, F(r)}{\partial r}\right|_{\xi_n} = F_r(\xi_n) = f^{\mathrm{D}}_r(\xi_n) + \left[f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)\right] g'_{\mathrm{LB}}(\xi_n) + \left[f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1)\right] g'_{\mathrm{RB}}(\xi_n) \tag{18}$$

Finally, F_r is transformed from reference space to physical space using the inverse mapping function, χ^{-1}, where it can then be employed to approximate f_x at each solution point within the physical grid cell. The solution to equation (9) is then advanced in time via an explicit Runge-Kutta method.

II.B.2. Spatial Discretization of Diffusion Terms

We now move our attention to the diffusion terms in equation (1). Consider the 1D diffusion equation

$$u_t = u_{xx} \tag{19}$$

which we choose to rewrite as a set of two coupled equations

$$u_t = \phi_x \tag{20a}$$

$$\phi = u_x \tag{20b}$$

To solve equation (20a) and thus equation (19), we must first solve equation (20b) such that ϕ is a polynomial of degree P, approximates u_x within each cell, and takes into account the interaction of data between adjacent cells. This can all be accomplished using the reconstruction idea from the previous section. Switching back to the reference element, equation (18) can be reformulated to create an analogous expression for the reconstructed first derivative, ϕ, as

$$\phi(\xi_n) = \phi_n = \phi^{\mathrm{D}}_n + \left[u^*_{\mathrm{LB}} - u(-1)\right] g'_{\mathrm{LB}}(\xi_n) + \left[u^*_{\mathrm{RB}} - u(1)\right] g'_{\mathrm{RB}}(\xi_n) \tag{21}$$

In equation (21), the discontinuous first derivative, $\phi^{\mathrm{D}}_n = \phi^{\mathrm{D}}(\xi_n) = u_r(\xi_n)$, is found using equation (12), and the common values at the interface, u^*, are obtained by simply averaging the interface values from the two cells on the interface

$$u^* = \frac{1}{2}\left(u_{\mathrm{L}} + u_{\mathrm{R}}\right) \tag{22}$$

Since ϕ_n is now known at all solution points, a discontinuous second derivative, ϕ^{D}_r, can be evaluated at each solution point using the derivative of the Lagrange polynomials similar to equation (12), i.e.,

$$\phi^{\mathrm{D}}_r(r) = \sum_{n=1}^{N_P} \phi_n \left[\frac{d}{dr}\ell_n(r)\right] \tag{23}$$

Using the reconstructed first derivatives found by equation (21) and the subsequent discontinuous second derivatives, we can apply the reconstruction procedure one final time to evaluate the right side of equation (20a). The resulting equation for the corrected first derivative of ϕ, and thus the corrected second derivative of u, is given by

$$\left.\frac{\partial\, \phi(r)}{\partial r}\right|_{\xi_n} = \phi^{\mathrm{D}}_r(\xi_n) + \left[\phi^*_{\mathrm{LB}} - \phi(-1)\right] g'_{\mathrm{LB}}(\xi_n) + \left[\phi^*_{\mathrm{RB}} - \phi(1)\right] g'_{\mathrm{RB}}(\xi_n) \tag{24}$$

All that remains is how to define the common derivatives, ϕ^*, at each interface. Similar to equation (22), we use a simple average for the common derivative,

$$\phi^* = \frac{1}{2}\left(\phi_{\mathrm{L}} + \phi_{\mathrm{R}}\right) \tag{25}$$

where ϕ_{L} and ϕ_{R} are respective left and right derivative states on the interface. The obvious choice for the left and right derivative states is $\phi_{\mathrm{L}} = \phi(1)$ for the left cell on the interface and $\phi_{\mathrm{R}} = \phi(-1)$ for the right cell; however, this choice results in a large stencil with a relatively small stability limit.[7] A compact stencil with a larger stability limit can be found through a one-sided correction. At each (cell) interface, let $\phi_{\mathrm{L}}^{\mathrm{D}} = u_r(1)$ and $\phi_{\mathrm{R}}^{\mathrm{D}} = u_r(-1)$ be the values taken on by equation (12) from the left and right cells on the interface, respectively. Application of the one-sided correction thus defines the left and right derivative states in the following way

$$\phi_{\mathrm{L}} = \phi_{\mathrm{L}}^{\mathrm{D}} + \left[u^* - u_{\mathrm{L}}\right] g'_{\mathrm{RB}}(1) \tag{26a}$$

$$\phi_{\mathrm{R}} = \phi_{\mathrm{R}}^{\mathrm{D}} + \left[u^* - u_{\mathrm{R}}\right] g'_{\mathrm{LB}}(-1) \tag{26b}$$

Using equations (22) and (26), and the identity $g'_{\mathrm{RB}}(1) = -g'_{\mathrm{LB}}(-1)$ resulting from the symmetry requirement on the correction functions, equation (25) can be simplified to

$$\phi^* = \frac{1}{2}\left[\phi_{\mathrm{L}}^{\mathrm{D}} + \phi_{\mathrm{R}}^{\mathrm{D}} + \left(u_{\mathrm{R}} - u_{\mathrm{L}}\right) g'_{\mathrm{RB}}(1)\right] \tag{27}$$

If the Radau polynomials are used for the correction functions in equations (21) to (27), the resulting method is equivalent to the BR2 method.[24] All results in the present work use this method to evaluate the diffusion terms of equation (1).

II.B.3. Temporal Discretization

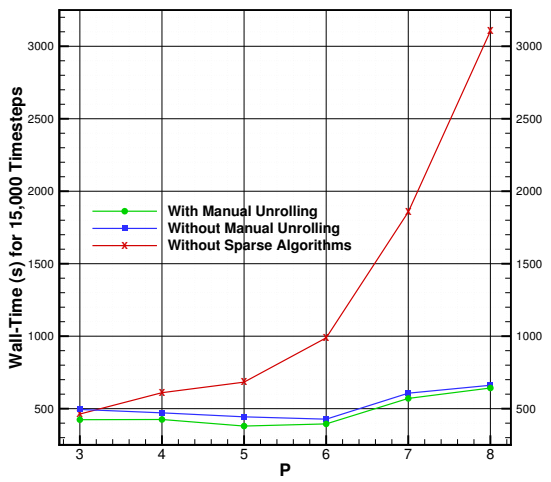
All of the methods currently implemented within GFR for advancing the solution in time are within the family of explicit Runge-Kutta (RK) schemes. The default scheme is the 3-stage strong stability preserving (SSP) RK method,[25] which is 3rd-order accurate in time. Additional time integration methods available within GFR are the low-storage, 5-stage/4th-order RK method of Carpenter and Kennedy[26] and the 5-stage/4th-order SSP RK method of Spiteri and Ruuth.[27]

All of the results presented in this paper were integrated in time using the 3-stage SSP RK method. The time step size is evaluated at each interface flux point using the maximum inviscid and viscous eigenvalues from the two adjoining cells, and the minimum overall time step is used as the global time step to maintain a time-accurate, unsteady solution.

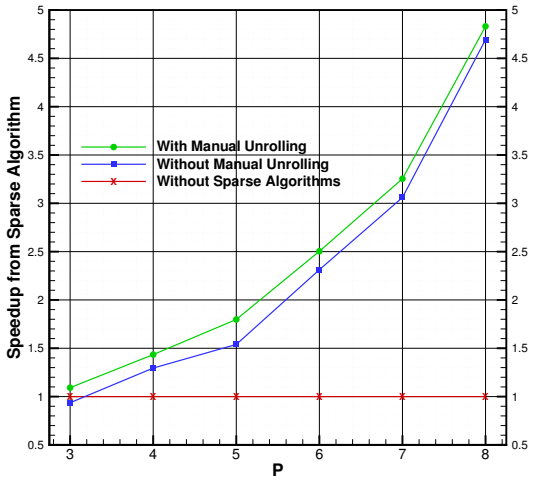
II.C. Data Structures and Code Optimization

The most efficient implementation of the FR method on structured grids utilizes the tensor product extension to higher dimensions. The tensor products can be programmed as a set of separate one-dimensional problems through an ordered, ijk-type data structure that is ideally suited for vectorization/optimization on modern computer processors. Unfortunately, such an implementation would severely handicap the applicability of a CFD code based on a method that is also known to work efficiently with unstructured geometries.

The FR formulation for unstructured geometries is largely matrix-oriented such that many of the steps within the method can be simplified to matrix operations, e.g., matrix-vector products and dot products. A generalized method for all geometry types can be created by reformulating the tensor products of structured geometries in the same matrix-oriented formulation for unstructured geometries. Unlike the mostly full matrices for unstructured geometries, reformulating the tensor products results in very sparse matrices, thus eliminating the efficiencies inherent with the use of structured geometries. For 2D simulations, the sparsity of these matrices can be ignored without drastically affecting the performance of the algorithm. This cannot be said for 3D simulations where the zeros already comprise roughly 94% of these matrices for just a $\mathcal{P}3$ solution. Combining this with a Courant-Friedrichs-Lewy (CFL) number that scales as $\mathcal{P}^{-2}$ for explicit methods,[15] running simulations above $\mathcal{P}5$ becomes unreasonably expensive unless we can take advantage of the sparsity of these matrices.



(a) Total wall-time for $\mathcal{P}3$–$\mathcal{P}8$ solutions



(b) Speedup relative to non-sparse algorithm

Figure 1: Comparison of the performance between different matrix algorithms found by running the Taylor-Green vortex problem for 15,000 time steps using approximately 128^3 DoF in each simulation.

Handling each geometry type separately from others would require the use of inefficient conditional statements at low levels within the solver, as well as multiple instances of duplicate code containing only small changes unique to each geometry. Therefore, one of our goals in developing GFR is to treat all grid geometries in a single, generic manner to enhance code clarity and readability. To facilitate this goal, a Fortran derived data type was created that stores a single matrix operator within an allocatable array for a specific combination of geometry type and solution order. For each matrix operator required by the FR method, we are able to create an array of this derived type that stores the matrices for each combination of geometry type and solution order valid for the current simulation.

With all of these matrix operators existing as an instance of this one derived data type, a generic mechanism for their use within the code was achieved through type-bound procedures, a new feature of Fortran that was introduced by the Fortran 2003 standard. By adding type-bound procedures with the pointer attribute to the definition of the derived type, the underlying algorithm controlling the matrix operations can be changed dynamically. Upon instantiation of each matrix object, the procedure pointers are associated with the Fortran intrinsic matrix functions (i.e., the dot_product and matmul functions). After all the matrices are initially created during the preprocessor phase of the simulation, the sparsity of each matrix is evaluated. Each one that is found to be sparse is converted to a compressed storage format and stored within the derived type, and the procedure pointers are then associated with the sparse matrix algorithms that utilize this compressed format. Implementing these matrices in this object-oriented programming style creates additional flexibility in selecting which algorithms are utilized every time the matrix operators are used.

Since these matrix operations comprise a large percentage of the computations within each time step, this abstraction provides opportunities to significantly reduce the overall computational cost. As an example, our initial implementation of the sparse dot product turned out to be slower than the intrinsic version for a $\mathcal{P}3$ simulation. Profiling of the sparse function highlighted significant overhead in evaluating the counter for the inner loop, because the compiler could not deduce that exactly four trips through the loop would occur every time the function was called. Therefore, the compiler was unable to properly optimize the sparse function by unrolling the inner loop, thus eliminating the need to evaluate the loop counter altogether. Using this information, we were able to quickly write new sparse functions with manually unrolled loops for many different solution orders, and associate these new functions with the type-bound procedure pointer.

Using the Taylor-Green vortex problem described in section III.B, we performed a series of tests to estimate the performance improvement found by the sparse algorithms. Each of the different matrix algorithms was used with $\mathcal{P}3$-$\mathcal{P}8$ solutions containing approximately 128^3 degrees of freedom (DoF), and each simulation was run for a total of 15,000 time steps. The total wall time for each simulation is shown in figure 1a and the speedup of the two sparse algorithms relative to the intrinsic functions is shown in figure 1b. The

most remarkable result from using the sparse algorithms is the computational cost per time step is relatively constant for all solution orders. In fact, the $\mathcal{P}5$ and $\mathcal{P}6$ solutions are actually less expensive per time step than the $\mathcal{P}3$ or $\mathcal{P}4$ solutions when comparing simulations with approximately the same DoF. Additionally, figure 1a clearly shows how expensive the solutions greater than $\mathcal{P}6$ become if the sparsity is ignored. It is also interesting to note that the intrinsic functions perform just as well as their sparse counterparts for the $\mathcal{P}3$ solution.

The matrix operators computed during the preprocessing phase are frequently used throughout each time step. Some specific examples of tasks involving these matrix operators include: computing derivatives through the derivative matrix, interpolating between sets of nodal points, use of the correction function, conversion between nodal and modal solution forms, and filtering. Roundoff errors occurring amid the creation of these operators are stored in the matrix elements; therefore, the errors are applied to the solution each and every time the operators are used. We take extra precaution to prevent this from happening by initially evaluating each operator using quadruple (128-bit) precision. Afterwards, each quad precision floating point value within each operator that is found to be smaller than double (64-bit) precision machine epsilon (2^{-52}) is assumed to be roundoff error and set to zero. All other values are simply converted from quad to double precision.

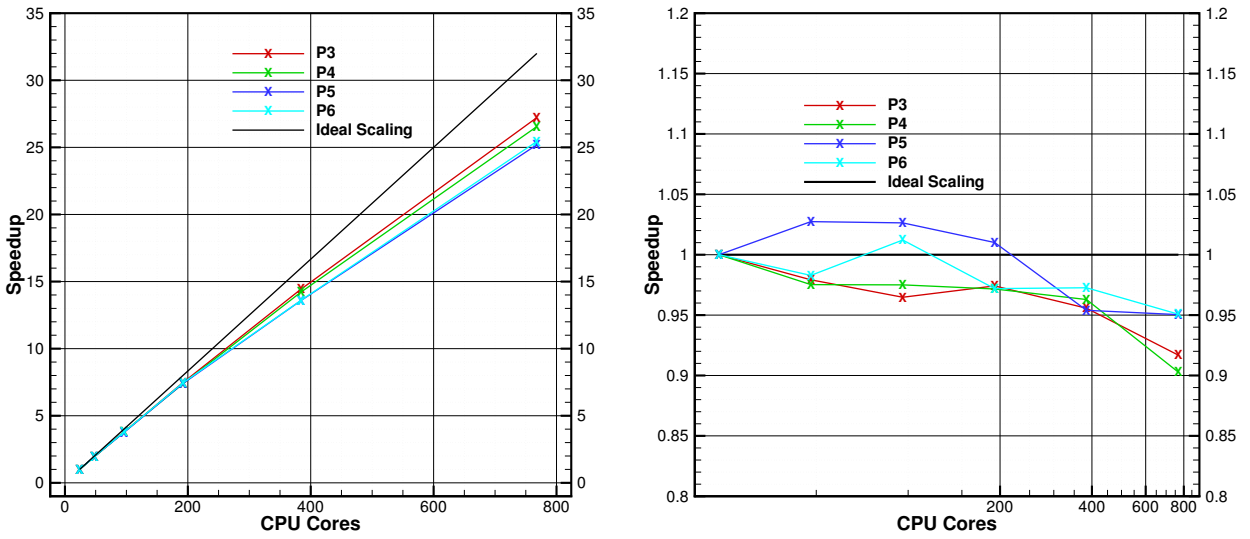
II.D. Parallel Processing

The parallel implementation is based on the single program, multiple data paradigm and utilizes MPI (Message Passing Interface)[28] for communication between a set of processes on a distributed-memory system. The application programming interface (API) provided by the METIS[29] library is used to partition the computational grid during the preprocessing phase of the simulation preventing the need to create any unnecessary intermediate grid files. Each MPI process corresponds to a single CPU core and the computational domain is partitioned such that each MPI process is assigned a unique grid partition. By default, the domain decomposition is accomplished through a k-way partitioning algorithm[30] with the goal of minimizing the total communication volume required by the final partitioned grid. If the default method fails to produce a quality partitioning, i.e., each process/partition is assigned at least one grid cell and all partitions are contiguous, one of the other partitioning algorithms within METIS is attempted. The second method uses the same k-way partitioning algorithm, but the goal is changed to minimizing the edge cut[31] which is just an approximation of the total communication required by the resulting partitioning. The final algorithm that is attempted uses a multilevel recursive bisection.[29]

In cell-centered finite-volume (FV) methods, each grid cell contains only a single DoF which makes the computational cost per grid cell a small fraction of the total computational cost. However, for nodal high-order methods the DoF per grid cell is $\mathcal{O}(\mathcal{P}^d)$, where d denotes the spatial dimension of the problem; therefore, the cost per grid cell is a much larger percentage of the total cost compared to FV methods. With communication significantly more expensive than computation, there is a limit for FV methods where the number of grid cells per partition becomes too small, and the performance starts to degrade rapidly as the communication costs begin to outweigh the computational costs. For nodal high-order methods, a partition can consist of just a few cells without severe communication bottlenecks, because the many DoF in each cell still require significant computational costs. The non-blocking communication functions in MPI are used to exchange information between processors in the background while computations local to each cell on a partition are evaluated concurrently to mask the communication costs. Exploiting these characteristics of high-order methods allows a problem to be partitioned efficiently across a very large number of processors with minimal communication costs.

Except for initially reading the global grid file, all large-scale file input/output (I/O) is done in parallel. There are two essential I/O tasks that inherently involve large amounts of data. The first task is the output of solution files used for visualizing the flow-field which is accomplished via the CGNS (CFD General Notation System) library.[32] The CGNS library provides an API to create such files in a standardized, compact, machine-independent binary format that is compatible with most CFD flow analysis and visualization software packages. The popular HDF5 (Hierarchical Data Format, Version 5)[33] data format is the default storage mechanism implemented within the CGNS library which allows for the utilization of the parallel I/O capabilities provided by the HDF5 libraries.

The second large-scale I/O task is the reading/writing of restart files. All restart file I/O within GFR is handled by the parallel I/O capabilities found in the MPI-2 extension of MPI.[34] By utilizing MPI user-defined datatypes, all processes can collectively write their own local data to the restart file independent of all other



(a) Strong scaling

(b) Weak scaling

Figure 2: The parallel scaling found by $\mathcal{P}3$–$\mathcal{P}6$ simulations of the Taylor-Green vortex problem for 2,000 time steps on a varying number of CPU cores. The left figure shows the strong scaling for approximately 128^3 DoF split evenly across the CPU cores. The right figure shows the weak scaling with each CPU core containing approximately 5,000 DoF.

processes. The MPI datatypes automatically reorder the local data for each process such that it matches the ordering of the global computational grid. Otherwise, the restart I/O would have to be performed either serially (wasting significant memory and creating a desynchronized process) or by writing one or multiple restart files that are dependent on the current partitioning.

The NASA Pleiades supercomputer was used to complete all of the simulations presented in this work. The Pleiades supercomputer is a distributed-memory cluster consisting of SGI compute nodes, each based on one of four different Intel CPU architectures, networked together via an InfiniBand interconnect and connected to a Lustre parallel file system. All the simulations were run on either the "Sandy Bridge"-based computing nodes that each contain two 8-core Intel Xeon E5-2670 processors or the "Haswell"-based computing nodes that each contain two 12-core Intel Xeon E5-2680v3 processors. The largest simulation attempted thus far used 4,800 CPU cores to successfully solve the Taylor-Green vortex problem on a grid containing 102^3 cells and 132,651,000 (510^3) DoF.

The most common method for testing parallel scalability, referred to as strong scaling, takes a problem of fixed size and measures the speedup gained by increasing the number of processors in order to decrease the amount of work assigned to each individual MPI process. A second test of parallel scalability, referred to as weak scaling, counters strong scaling by scaling the problem size with the number of processors such that the work assigned to each MPI process remains constant. Both of these methods were used to test the parallel scalability of GFR through a series of $\mathcal{P}3$–$\mathcal{P}6$ simulations of the Taylor-Green vortex problem, and each solution order was run for 2,000 time steps on a successive number of Haswell compute nodes (2^n, $n = 0, 5$). Each Haswell node contains two 12-core processors, therefore the baseline simulation using only a single compute node actually used 24 CPU cores; the remaining simulations used 48, 96, 192, 384, and 768 CPU cores.

The strong scaling results for GFR are shown in figure 2a where the DoF for all simulations on all processor counts was approximately 128^3 DoF. Ideally, an increase in the number of processors should result in a proportional reduction in the total wall time; this ideal speedup is shown as the black line in figure 2a. Through 192 CPU cores, the strong scaling is within reason for all degrees of the solution polynomial, with all showing a speedup within 92% of ideal at 192 cores. However, the scaling begins to digress with further processor increases, especially so for the $\mathcal{P}5$ and $\mathcal{P}6$ simulations. At 768 CPU cores, the $\mathcal{P}5$ case contains an average of only 12.1 cells per core and only 7.6 cells per core for the $\mathcal{P}6$ case. Because the total number of cells cannot be split evenly between the grid partitions for these two cases, each MPI process contained

either 12 or 13 cells for the $\mathcal{P}5$ case or either 7 or 8 cells for the $\mathcal{P}6$ case. These small one cell differences actually correspond to a significant load imbalance between processes, resulting in some waiting idly for others that have to do more work.

This inevitable limit of strong scaling is countered by weak scaling, and the results of the weak scaling tests with GFR are shown in figure 2b. The same successive number of compute nodes used for testing the strong scalability was also used to test the weak scalability, but here the simulations were scaled such that each CPU core contained approximately 5,000 DoF. The variations in the DoF in each simulation was accounted for by multiplying the overall wall time by the ratio 5,000/(actual DoF per CPU core). For all simulations using up to 16 computational nodes, the weak scaling is within 5% of the nominal scaling. However, the weak scalability begins to break down for all simulations using 32 nodes.

More than likely, both reductions in scalability at higher processor counts are caused by the increased complexity in the partition graph, thus creating significant contention for network bandwidth. At the moment, the domain decomposition and subsequent assignment of individual partitions to MPI processes is performed blindly such that the processor and partition localities are not reflective of one another. This method is more than adequate for lesser processor counts but it quickly breaks down as the number of processors increases; both are clearly reflected in figure 2.

III. Numerical Results

III.A. Isentropic Euler Vortex Problem

The isentropic Euler vortex problem is commonly used[9,13,16,35–46] to empirically quantify the order of accuracy of a numerical method because it is easy to implement and the exact solution is known at all times. This problem is ideal for high-order methods because their theoretical accuracy should allow them to propagate the vortex with minimal entropy production for an indefinite amount of time. Being able to sustain vortical flow structures without the numerical method contributing unwanted numerical dissipation is crucial for advanced turbulence methods like LES. All of these reasons are why this is becoming a popular test case for high-order methods that are designed to produce minimal numerical dissipation.

In Ref. 45, some commonly used variations of this problem were found to contain instabilities caused by poorly chosen boundary conditions and the size of the computational domain being too small. After discovering the underlying causes of these instabilities, a list was provided to offer general guidelines and recommendations regarding the use of the isentropic Euler vortex problem. One of the recommendations is to zero the mean flow velocity so the vortex remains fixed at its initial location and no longer propagates through the domain. By keeping the vortex stationary, all of the grid boundaries can be given characteristic boundary conditions since periodic boundary conditions are no longer required. In the absence of physical dissipation, periodic boundaries provided a means for numerical error to recirculate within the computational domain and accumulate until eventually becoming large enough to cause the system to become unstable. The purpose of this article is to provide an overview of the capabilities of GFR, so this section will only summarize the findings from Ref. 45 regarding the stationary vortex problem with characteristic boundary conditions.

The initial conditions for the vortex problem are found by superimposing perturbations in velocity and temperature onto a uniform mean flow. These perturbations are defined such that the entropy, $S = p/\rho^\gamma$, is constant throughout the entire grid domain, i.e., $\delta S = 0$. Since the vortex is stationary, the mean flow velocity, $(v_{x,\infty}, v_{y,\infty})$, is zero; additionally, the free-stream density, ρ_∞, and pressure, p_∞, are set to unity.

The perturbations in velocity and temperature are defined as

$$\begin{aligned} \delta v_x &= -\,y\,\Theta \\ \delta v_y &= +\,x\,\Theta \\ \delta T &= -\frac{(\gamma-1)}{2}\,\Theta^2 \end{aligned} \tag{28}$$

where Θ is a Gaussian function of the form

$$\Theta = \frac{5}{2\pi\sqrt{\gamma}}\,\mathrm{e}^{-\frac{1}{2}\left(1-x^2-y^2\right)} \tag{29}$$

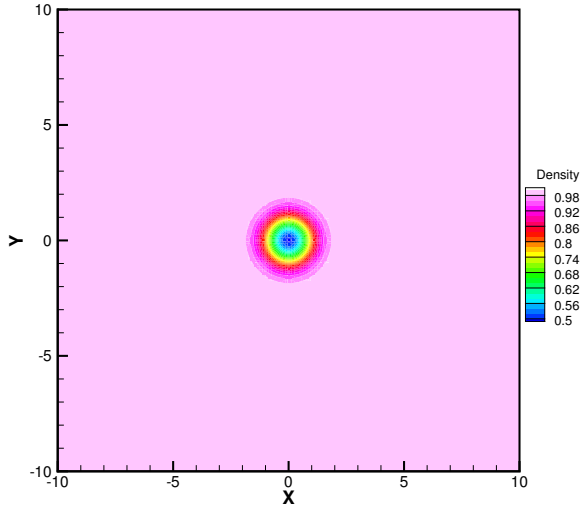


Figure 3: Contours of the initial density profile for the isentropic Euler vortex problem.

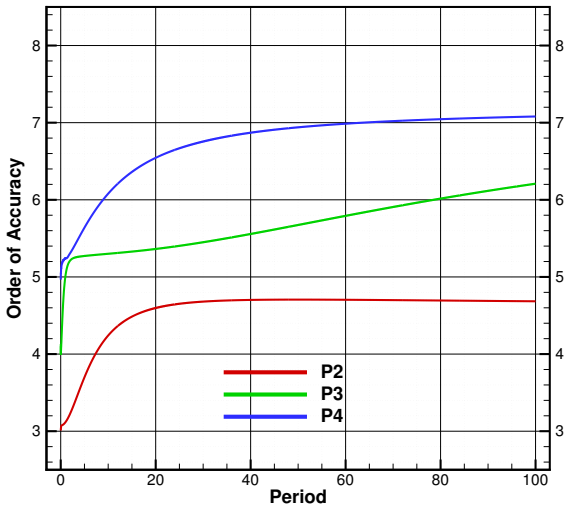


Figure 4: The development of the order of accuracy over time for the stationary isentropic Euler vortex using all characteristic boundaries.

These perturbations, along with the free-stream conditions and the isentropic relations between density, pressure, and temperature, are used to define the initial flow conditions of the primitive variables as

$$\begin{aligned} \tilde{\rho}_0 &= (1+\delta T)^{\frac{1}{\gamma-1}} \\ \tilde{v}_{x,0} &= \delta v_x \\ \tilde{v}_{y,0} &= \delta v_y \\ \tilde{p}_0 &= \tfrac{1}{\gamma}(1+\delta T)^{\frac{\gamma}{\gamma-1}} \end{aligned} \tag{30}$$

where the subscript 0 denotes a quantity given at $t = 0$. The tilde accents on the primitive variables indicate that each is a nondimensional quantity where ρ_∞, a_∞, and T_∞ have been used as the characteristic density, velocity, and temperature scales, respectively.

The last remaining item needed to complete the problem is the definition of the computational domain. All simulations of this problem were performed on a square domain with grid boundaries located at $(x, y) = (\pm 10, \pm 10)$. A series of grids with varying resolutions was created from the computational domain, with each grid consisting of regular, nonoverlapping quadrilateral cells of uniform size. Finally, all grid boundaries were set to characteristic boundary conditions to allow for information to flow in/out of the domain as needed. A contour plot showing the initial density profile resulting from this problem definition is shown in figure 3.

This problem is cleverly defined such that it can be run indefinitely and the exact analytical solution at any point in time is simply the initial conditions. Therefore, any deviation of the numerical solution from the initial conditions is an accurate measurement of the error generated by the numerical method, and a general sense of how the error evolves over time can be established. For the propagating vortex, time can additionally be measured in flow-through periods, which is the number of times the vortex propagates completely through the domain and returns back to its initial location. Even though a flow-through period for the stationary vortex is undefined, we will still use it to measure the elapsed time of our simulations. For the stationary case, one period is equivalent to the amount of time for a vortex propagating at $v_{x,\infty} = 1$ to complete one flow-through period.

The isentropic Euler vortex problem was used to complete an h-refinement study for $\mathcal{P}2$–$\mathcal{P}4$ solutions on grids containing 120^2, 160^2, and 200^2 cells. At each time interval that the solution error was evaluated, a linear fit of the density errors from each grid level was computed to see how the order of accuracy develops over time. The result of these linear fits using this set of simulations produces the order of accuracy versus time plot shown in figure 4. Initially, an accuracy of $\mathcal{P}+1$ is found for each polynomial order, and this matches

the results commonly found by DG-type methods for solving nonlinear equations. As time progresses, the accuracy increases for all polynomial orders reaching an approximate accuracy of $\mathcal{P}+3$ through 100 periods. This clearly verifies that the FR method achieves some degree of super-accuracy, that is accuracy greater than $\mathcal{P}+1$, for this problem.

III.B. Taylor-Green Vortex Problem

The Taylor-Green vortex (TGV) is a canonical problem in fluid dynamics developed to study vortex dynamics, turbulent transition, turbulent decay and the energy dissipation process. The TGV problem contains several key physical processes in turbulence in a simple construct and therefore is an excellent case for the evaluation of turbulent flow simulation methodologies. The problem consists of a cubic volume of fluid that contains a smooth initial distribution of vorticity, and periodic boundary conditions are applied to all boundary surfaces. As time advances the vortices roll-up, stretch and interact, eventually breaking down into turbulence. Because there is no external forcing, the small-scale turbulent motion will eventually dissipate all the energy in the fluid and it will come to rest.

III.B.1. Problem Definition

The definition of the TGV problem used here follows that specified by the AIAA First International Workshop on High-Order CFD Methods[39] held in 2012. The same problem was also a test case in the 2nd and 3rd iterations of workshop respectively held in 2013 and 2015, and is a test case for the upcoming 4th edition of the workshop in 2016. The solution found by a pseudo-spectral code[47] using 512^3 DoF is also included with the workshop test case, and it is used here as a reference solution for comparison with the solutions found by GFR.

The TGV problem models an incompressible flow at a Reynolds number of 1600 within a computational domain defined by a periodic cube with sides of length $2\pi L$. Within the domain, an initial distribution of velocity and its corresponding smooth vorticity distribution is specified by the following relations

$$\begin{aligned} v_x &= V_0 \sin\left(\frac{x}{L}\right)\cos\left(\frac{y}{L}\right)\cos\left(\frac{z}{L}\right) \\ v_y &= -V_0 \cos\left(\frac{x}{L}\right)\sin\left(\frac{y}{L}\right)\cos\left(\frac{z}{L}\right) \\ v_z &= 0 \\ p &= p_0 + \frac{\rho_0 V_0^2}{16}\left(\cos\left(\frac{2x}{L}\right)+\cos\left(\frac{2y}{L}\right)\right)\left(\cos\left(\frac{2z}{L}\right)+2\right) \end{aligned} \tag{31}$$

Because GFR solves the compressible Navier-Stokes equations, the reference Mach number is selected to be small enough as to provide an incompressible flow at the given Reynolds number, i.e., $M_0 = 0.1$. To complete the definition of the TGV problem, the reference length and temperature are respectively defined as $L = 1.0\,\mathrm{m}$ and $T_0 = 300\,\mathrm{K}$.

III.B.2. Data Processing

The primary method for evaluating the TGV solutions is examining the rate at which the fluid dissipates the kinetic energy in the domain. The integrated kinetic energy, E_k, within the domain is computed by

$$E_k = \frac{1}{\rho_0 \Omega}\int_\Omega \rho \frac{v_i v_i}{2}\,\mathrm{d}\Omega \tag{32}$$

where Ω is the volume of the computational domain. The kinetic energy dissipation rate (KEDR) can then be computed by differencing E_k in time,

$$\epsilon(E_k) = -\frac{dE_k}{dt} \tag{33}$$

This KEDR, ϵ, is written as a function of E_k to emphasize that it is computed directly from the kinetic energy in the domain.

A second "theoretical" kinetic energy dissipation rate can be computed from the integrated enstrophy, ζ, within the domain

$$\zeta = \frac{1}{\rho_0 \Omega} \int_\Omega \rho \frac{\omega_i \omega_i}{2} \, \mathrm{d}\Omega \tag{34}$$

where ω_i is the vorticity vector. It can be shown that for an incompressible flow, the enstrophy is directly related to the kinetic energy dissipation rate through a constant,

$$\epsilon(\zeta) = 2 \frac{\mu}{\rho_0} \zeta \tag{35}$$

The difference between the directly computed KEDR, $\epsilon(E_k)$, and the enstrophy based KEDR, $\epsilon(\zeta)$, will be examined.

Two additional quantities were computed to test the incompressible assumption. For a compressible flow where the bulk viscosity can be taken to be zero, the theoretical dissipation rate is the sum of two quantities; the contribution to the KEDR, obtained from the deviatoric strain rate tensor, S^d_{ij},

$$\epsilon_1 = 2 \frac{\mu}{\rho_0 \Omega} \int_\Omega S^d_{ij} S^d_{ij} \, \mathrm{d}\Omega \tag{36}$$

and the contribution to the KEDR, obtained from the pressure dilatation term.

$$\epsilon_3 = -\frac{1}{\rho_0 \Omega} \int_\Omega p \frac{\partial v_i}{\partial x_j} \, \mathrm{d}\Omega \tag{37}$$

For low Mach number flows, ϵ_3 should be negligible and the theoretical dissipation rates from equations (35) and (36) should be equivalent. Even though GFR solves the compressible Navier-Stokes equations, a comparison between these two theoretical dissipation rates was nearly identical for each simulation, meaning the incompressible assumption is valid here.

III.B.3. Grids

Two input parameters are required to run the TGV problem using GFR. The first parameter simply instructs the code to initialize the flow solution using a function based on the relations in equation (31). The second input parameter is an integer which allows the user to load a user-provided grid if given a negative value. Otherwise, the parameter specifies the desired DoF in each coordinate direction, and a grid is generated during the preprocessor phase of the simulation based on this value and the input value for the degree of the polynomial solution. For example, if the input file requests 256 DoF and a $\mathcal{P}4$ solution (i.e., 5^3 DoF per grid cell), the code would generate a grid containing 51^3 cells resulting in a total of 255^3 DoF.

An optional input parameter can be given to apply a random perturbation to all of the internal grid nodes. For each individual grid node, a new random value is evaluated for the perturbation in each coordinate direction, so the highest possible degree of irregularity can be imposed onto the grid. The value of this parameter is input as a percentage, relative to the uniform cell length, and controls the maximum amount of perturbation allowed in any direction. The perturbations were limited to only the interior nodes of the grid to prevent the creation of hanging nodes across linked periodic boundaries or deformation to the cubic shape of the computational domain.

In this work, both h- and p-refinement studies were performed. For $\mathcal{P}3$–$\mathcal{P}6$ solutions, simulations were performed on grids that provided approximately 64^3, 128^3, and 256^3 DoF for each of the solution orders. Additionally, simulations using $\mathcal{P}7$ and $\mathcal{P}8$ solutions were performed with approximately 256^3 DoF. Finally, a $\mathcal{P}4$ simulation was performed with approximately 512^3 DoF, and a second $\mathcal{P}4$ simulation with approximately 256^3 DoF was performed with a 20% random perturbation applied in all directions to each internal grid node. A summary of all these simulations is found in table 1 including details on the number of CPU cores used, the architecture of the CPU, and the average number of DoF and grid cells per CPU core.

III.B.4. Results

For the set of simulations with approximately 64^3 DoF, the $\mathcal{P}5$ and $\mathcal{P}6$ simulations blew up shortly after $t^* = 15.04$ and $t^* = 10.44$, respectively. This is clearly seen in figure 5a, and further attempts using successively smaller time steps failed to stabilize either of the simulations. A similar instability was found

Table 1: Summary of all the Taylor-Green vortex simulations

$\mathcal{P}$	DoF	Cells	CPU Cores	Architecture	DoF/Core	Cells/Core	Error in $\epsilon(\zeta)$ at peak KEDR
3	64^3	16^3	160	Sandy Bridge	1,638.4	25.6	51.58 %
4	65^3	13^3	192	Haswell	1,430.3	11.4	45.40 %
5	66^3	11^3	192	Haswell	1,497.4	6.9	35.67 %
6	63^3	9^3	120	Haswell	2,083.7	6.1	27.68 %
3	128^3	32^3	1,600	Sandy Bridge	1,310.7	20.5	24.16 %
4	130^3	26^3	1,536	Haswell	1,430.3	11.4	19.26 %
5	126^3	21^3	720	Haswell	2,778.3	12.9	14.97 %
6	126^3	18^3	480	Haswell	4,167.5	12.2	14.88 %
3	256^3	64^3	1,200	Haswell	13,981.0	218.5	5.96 %
4	255^3	51^3	1,200	Haswell	13,817.8	110.5	3.69 %
4*	255^3	51^3	2,400	Haswell	6,908.9	55.3	3.87 %
5	258^3	43^3	1,200	Haswell	14,311.3	66.3	3.04 %
6	259^3	37^3	1,200	Haswell	14,478.3	42.2	2.08 %
7	256^3	32^3	2,400	Haswell	6,990.5	13.7	2.01 %
8	252^3	28^3	2,400	Haswell	6,667.9	9.1	1.40 %
4	510^3	102^3	4,800	Haswell	27,635.6	221.1	0.75 %

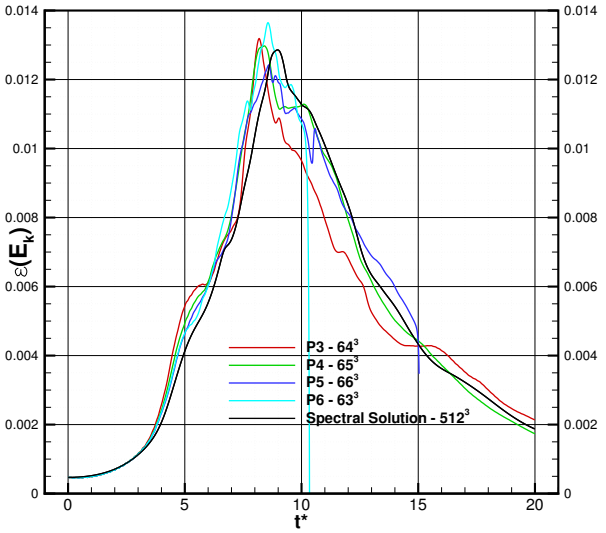
* Randomly perturbed grid

using a $\mathcal{P}7$ solution with 128^3 DoF, but the results of this simulation are not included since it only reached $t^* = 6.8$ before ultimately diverging. However, all of these instabilities were remedied by increasing the grid resolution through additional DoF as evident by the results in figure 7. This indicates that these instabilities could be driven by errors associated with polynomial aliasing, which are known to become more problematic as $\mathcal{P}$ increases for under-resolved simulations.[48,49] Additional studies applying filtering or over-integration to these simulations are required to verify this conjecture. Also note that in figure 7, the $\mathcal{P}7$ and $\mathcal{P}8$ simulations do not reach $t^* = 20$, but this is not the result of instabilities causing the solution to diverge. It is simply because the simulations reached the end of their allocated wall time on the Pleiades system. The same is true for the $\mathcal{P}4$ simulation using 510^3 DoF displayed in figure 8.

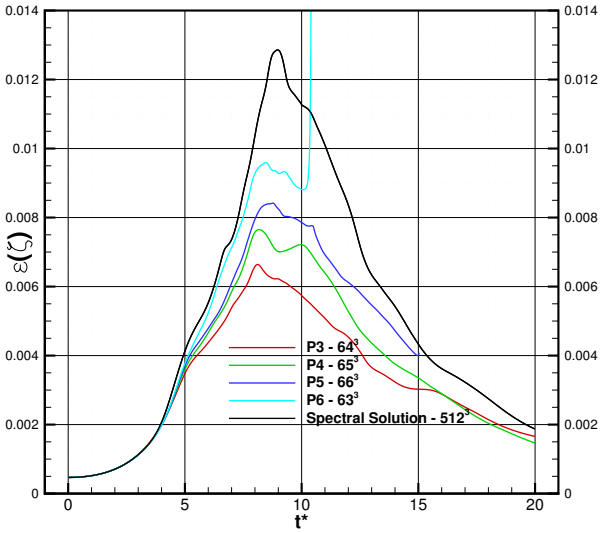
Figure 5, shows the KEDR for all the simulations containing approximately 64^3 DoF; specifically, figure 5a shows the KEDR directly computed by time-differencing the integrated kinetic energy using equations (32) and (33), and figure 5b shows the KEDR based on the integrated enstrophy which is computed by equations (34) and (35). Aside from the points where the $\mathcal{P}5$ and $\mathcal{P}6$ solutions begin to diverge, the overall trends of $\epsilon(E_k)$ follow the spectral solution for all cases in figure 5a. However, in figure 5b, the KEDR is greatly under-predicted by $\epsilon(\zeta)$, especially around the time in which the spectral solution reaches a peak rate of dissipation, i.e., $t^* = 8.97$. For the $\mathcal{P}3$–$\mathcal{P}6$ solutions, the peak value is respectively under-predicted by 52%, 45%, 36%, and 28%, yet this also illustrates the considerable improvement in accuracy gained by increasing $\mathcal{P}$.

The directly computed KEDR is representative of all the dissipation actually present in the simulation, whereas the enstrophy based KEDR is representative of the physical dissipation that theoretically exists from the resolved vortical structures within the solution. Therefore, the difference between these two values can be considered as an approximation to the numerical dissipation provided by the FR method. In figure 5b, the physical dissipation from the resolved vortical structures is significantly less than the reference spectral solution, yet in figure 5a, the total dissipation follows the trends of the same reference solution. This indicates that the numerical dissipation is providing a reasonable approximation of the physical dissipation found in the unresolved scales. With this reasoning, the FR method appears to be a viable candidate for accurate implicit large-eddy simulations (ILES).

This same general trend is found with increasing grid resolution, as seen in figure 6 for simulations

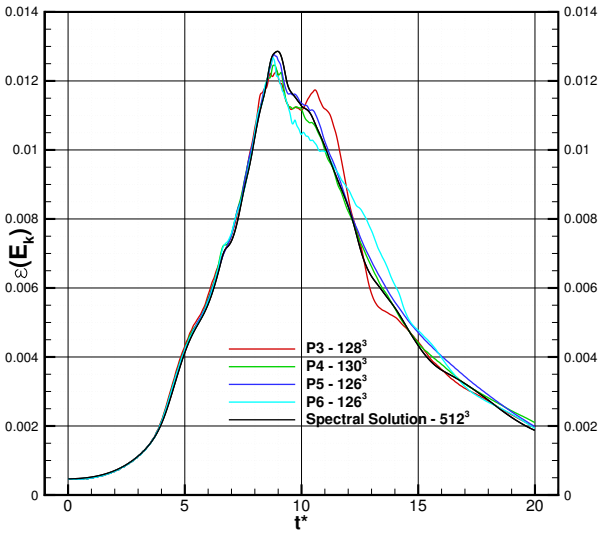


(a) Directly computed KEDR, $\epsilon(E_k)$

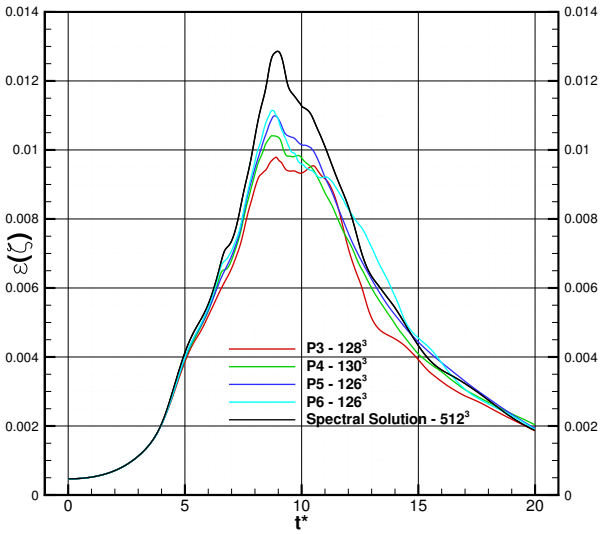


(b) Enstrophy based KEDR, $\epsilon(\zeta)$

Figure 5: Comparison of the KEDR results for $\mathcal{P}3$–$\mathcal{P}6$ simulations containing approximately 64^3 DoF.

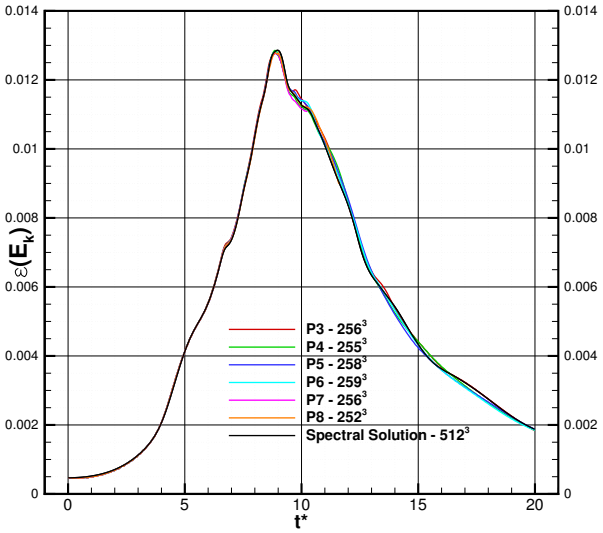


(a) Directly computed KEDR, $\epsilon(E_k)$

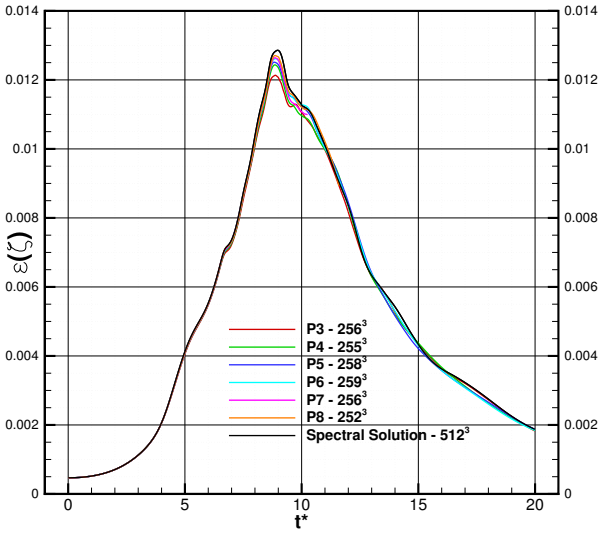


(b) Enstrophy based KEDR, $\epsilon(\zeta)$

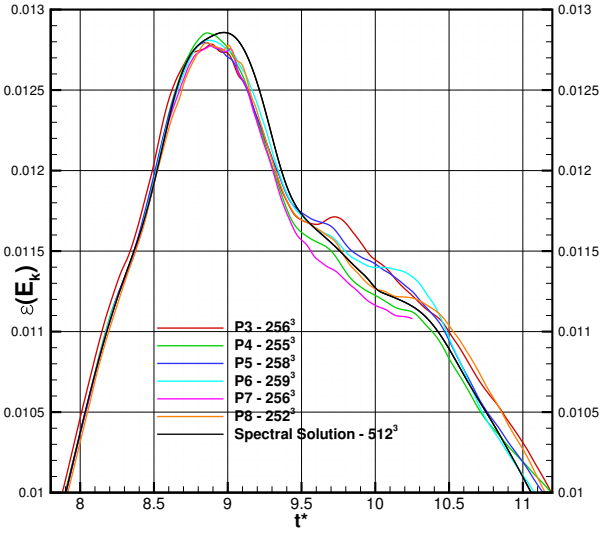
Figure 6: Comparison of the KEDR results for $\mathcal{P}3$–$\mathcal{P}6$ simulations containing approximately 128^3 DoF.



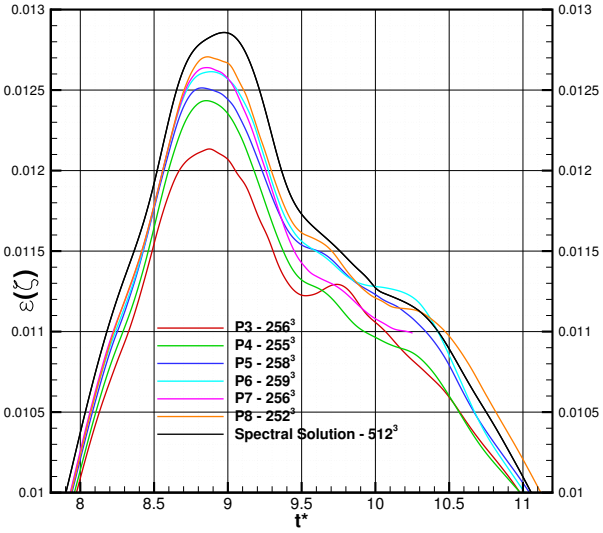
(a) Directly computed KEDR, $\epsilon(E_k)$



(b) Enstrophy based KEDR, $\epsilon(\zeta)$

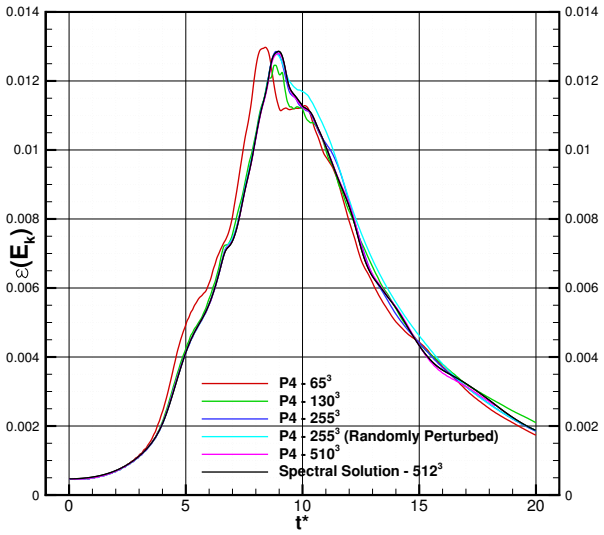


(c) Close up of $\epsilon(E_k)$

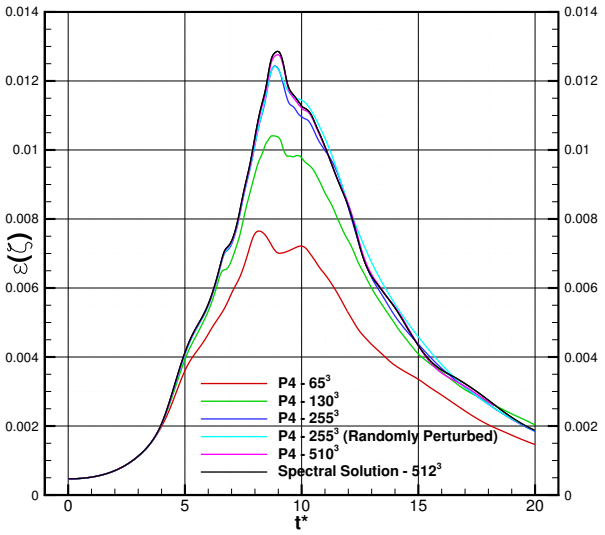


(d) Close up of $\epsilon(\zeta)$

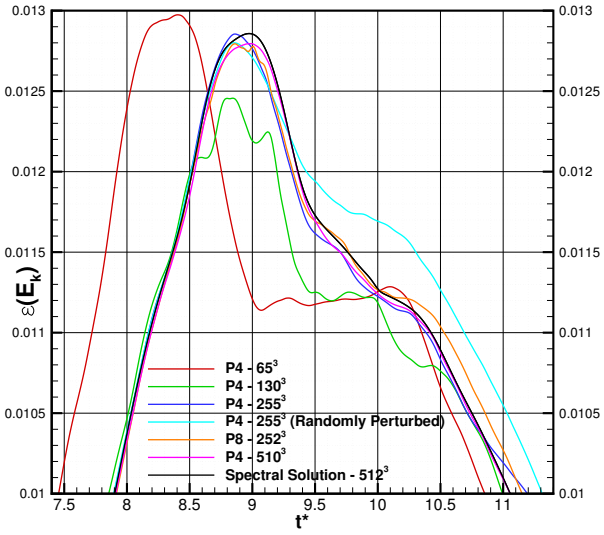
Figure 7: Comparison of the KEDR results for $\mathcal{P}3$–$\mathcal{P}8$ simulations containing approximately 256^3 DoF.



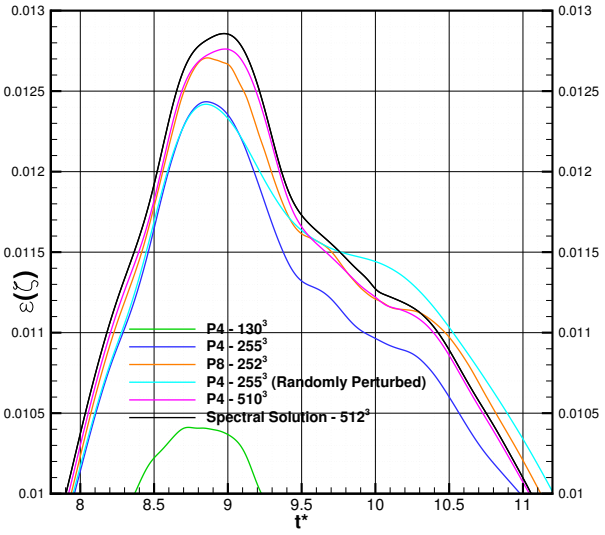
(a) Directly computed KEDR, $\epsilon(E_k)$



(b) Enstrophy based KEDR, $\epsilon(\zeta)$



(c) Close up of $\epsilon(E_k)$



(d) Close up of $\epsilon(\zeta)$

Figure 8: Comparison of the KEDR results for the $\mathcal{P}4$ solutions on all of the various grid levels. The $\mathcal{P}8$ solution with 252^3 DoF is also included in the close up views for additional comparisons.

containing approximately 128^3 DoF, and again in figure 7 for those with approximately 256^3 DoF. The last column of table 1 gives the error in $\epsilon(\zeta)$ for all of these simulations when the dissipation rate of the spectral solution is at its maximum. The error in $\epsilon(\zeta)$ decreases significantly with increasing grid resolution due to smaller and smaller vortical structures being resolved. This increase in the number of resolved vortical structures also corresponds to substantial improvements in $\epsilon(E_k)$ converging towards the spectral solution. In figure 6a, the simulations using approximately 128^3 DoF are much smoother and more accurate than those with 64^3 DoF, and in figures 7a and 7c, the simulations containing approximately 256^3 DoF are in very close agreement with the spectral solution.

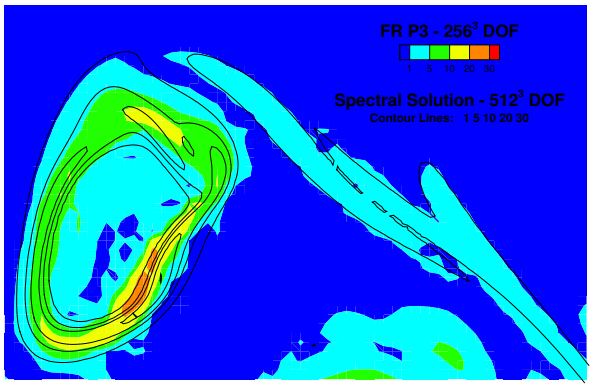
The results in figures 5 to 7 present p-refinement studies of the KEDR at three different grid resolutions. The last set of KEDR results, presented in figure 8, demonstrates the effect of h-refinement by varying the DoF used with a $\mathcal{P}4$ solution. The simulations containing 65^3, 130^3, and 255^3 DoF are the same simulations from the previous figures. Included with these are a simulation containing 510^3 DoF and a second simulation with 255^3 DoF using the randomly perturbed grid described in section III.B.3. Close up views of figures 8a and 8b, in the region with peak KEDR, are respectively found in figures 8c and 8d. The $\mathcal{P}8$ simulation with 252^3 DoF has also been included in figures 8c and 8d to compare the improvement in accuracy found by increasing $\mathcal{P}$ versus that found by increasing the grid resolution. Increasing the grid resolution to 510^3 DoF (orange curve) gives slightly better accuracy in the KEDR over increasing the solution order to $\mathcal{P}8$ (magenta curve), but both produce significant improvements in $\epsilon(\zeta)$ over the $\mathcal{P}4$ solution with 255^3 DoF (blue curve).

The next set of results for the Taylor-Green vortex problem demonstrate the accuracy of the actual vortical structures resolved by GFR. Contours of the vorticity norm at $t^* = 8$, located on the plane $x = -\pi L$ and in the region bounded by the lines $y = [0, \frac{\pi L}{2}]$ and $z = [\frac{\pi L}{2}, \pi L]$, are shown in figures 9 and 10 for each of the simulations with approximately 256^3 DoF and in figure 11 for the $\mathcal{P}4$ simulation with 510^3 DoF. Additionally, the contour lines found using a pseudo-spectral code with 512^3 DoF have been superimposed onto each of these figures as a reference solution for comparison. Note that the contour lines correspond exactly to the contour color levels.

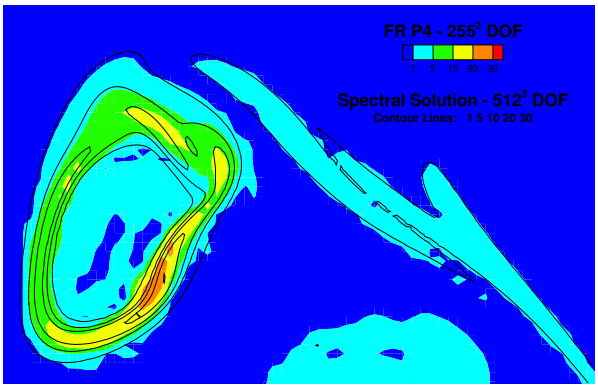
The solution from the spectral code resolves two primary vortical structures in the region of interest: a ring-like structure in the left half of the region and a linearly-shaped structure that extends from the bottom-right to just above the ring structure. In figure 9, both of these vortical structures from the spectral solution are resolved by GFR for all values of $\mathcal{P}$. However, for lower orders, an additional structure not present in the spectral solution is found located at the bottom of this region. As the solution accuracy increases up to $\mathcal{P}7$ and $\mathcal{P}8$, this new structure is dissipated. One final point of interest in figure 9 is how the alignments of the finer details within the ring-like vortical structure change with increasing $\mathcal{P}$. At lower orders, the interior edge of the ring structure is not well defined and the vorticity is spread across the interior of the ring. Additionally, the structure is more jagged and offset slightly in a counter-clockwise direction as compared to the spectral solution. As $\mathcal{P}$ increases, the details of the ring structure converge towards the spectral solution with the $\mathcal{P}8$ solution showing good agreement.

Figure 10 demonstrates the sensitivity of the FR method to irregularly spaced grids by comparing the two $\mathcal{P}4$ simulations with 255^3 DoF. The solution on the randomly perturbed grid in figure 10b is not significantly different from the solution on the uniform grid in figure 10a, but it does show increased diffusion throughout. The grid irregularity causes the interior edge of the ring structure to become even less defined due to additional spreading of the vorticity across the interior of the structure, and the edges of linear structure are smeared such that it deviates more from the spectral solution. The additional diffusion does provide some added benefit over the solution on the uniform grid, in that the structure not present in the spectral solution is dissipated and less prominent.

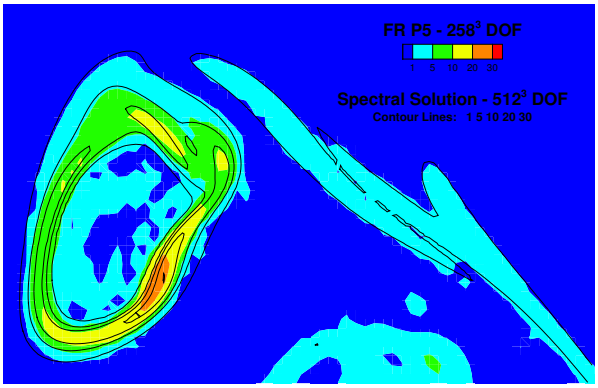
Finally, figure 11 shows that by increasing the grid resolution to 510^3 DoF for a $\mathcal{P}4$ solution, GFR is able to very nearly reproduce the spectral solution containing 512^3 DoF. The accuracy of this result provides some additional insight into the previous discussion using figures 8c and 8d to compare the accuracy improvement found by increasing grid resolution versus that found by increasing the solution order. Using the enstrophy based KEDR as the metric for accuracy, the simulation with additional grid resolution took longer than the simulation with an increased solution order, yet both gave similar improvements in accuracy. However, if the vorticity contours are used for the accuracy metric, comparing the improvement in accuracy going from figure 9b to 9f versus the improvement going from figure 9b to 11 shows that the increased grid resolution may be worth the extra computational cost. Combining all of the results for the TGV problem, the added value in both accuracy and flexibility provided by the two refinement methods, either separately or combined, is clearly visible.



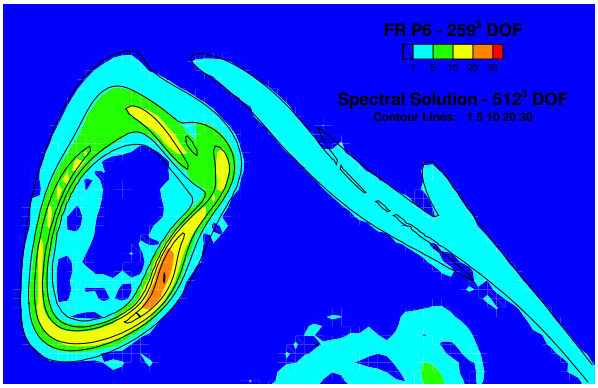
(a) $\mathcal{P}3$ with 256^3 DoF



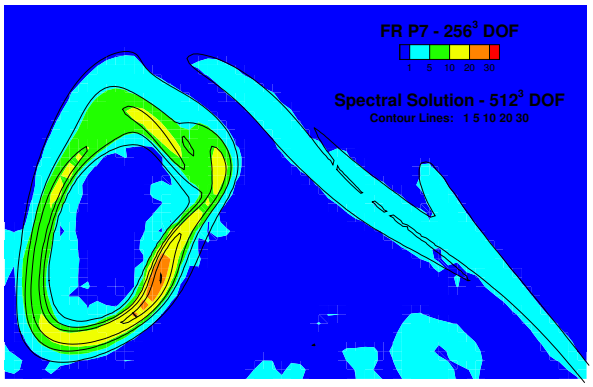
(b) $\mathcal{P}4$ with 255^3 DoF



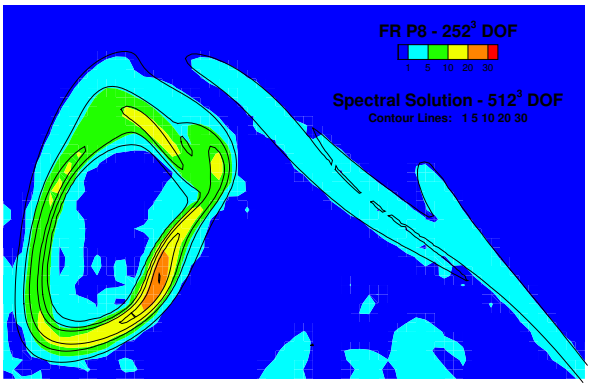
(c) $\mathcal{P}5$ with 258^3 DoF



(d) $\mathcal{P}6$ with 259^3 DoF

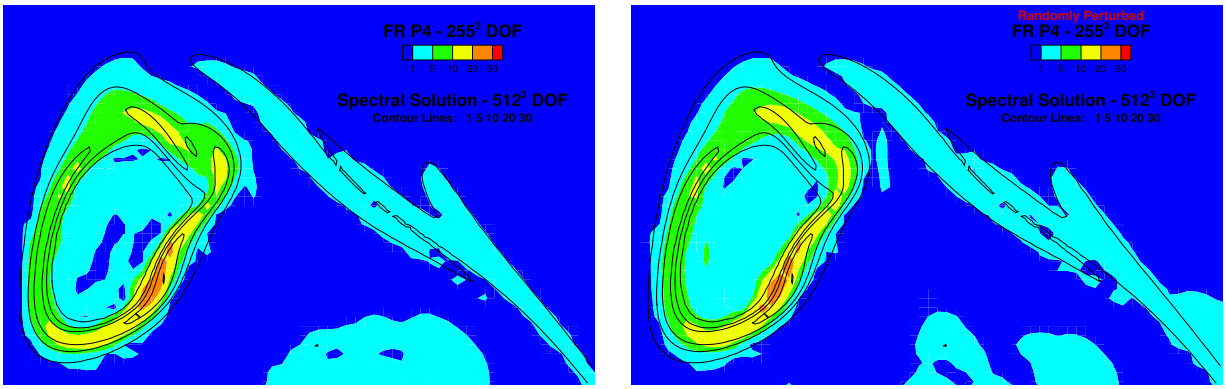


(e) $\mathcal{P}7$ with 256^3 DoF



(f) $\mathcal{P}8$ with 252^3 DoF

Figure 9: Comparison of $\mathcal{P}3$–$\mathcal{P}8$ solutions with approximately 256^3 DoF. The plots show contours of the vorticity norm at $t^* = 8$ on the plane $x = -\pi L$, in the region bounded by the lines $y = [0, \frac{\pi L}{2}]$ and $z = [\frac{\pi L}{2}, \pi L]$. The contour lines found using a pseudo-spectral code with 512^3 DoF have been superimposed as a reference solution for comparison.



(a) $\mathcal{P}4$ with 255^3 DoF (uniform Cartesian grid)

(b) $\mathcal{P}4$ with 255^3 DoF (randomly perturbed grid)

Figure 10: $\mathcal{P}4$ solutions with 255^3 DoF comparing the results between a uniform Cartesian grid and a randomly perturbed grid. The plots show contours of the vorticity norm at $t^* = 8$ on the plane $x = -\pi L$, in the region bounded by the lines $y = [0, \frac{\pi L}{2}]$ and $z = [\frac{\pi L}{2}, \pi L]$. The contour lines found using a pseudo-spectral code with 512^3 DoF have been superimposed as a reference solution for comparison.

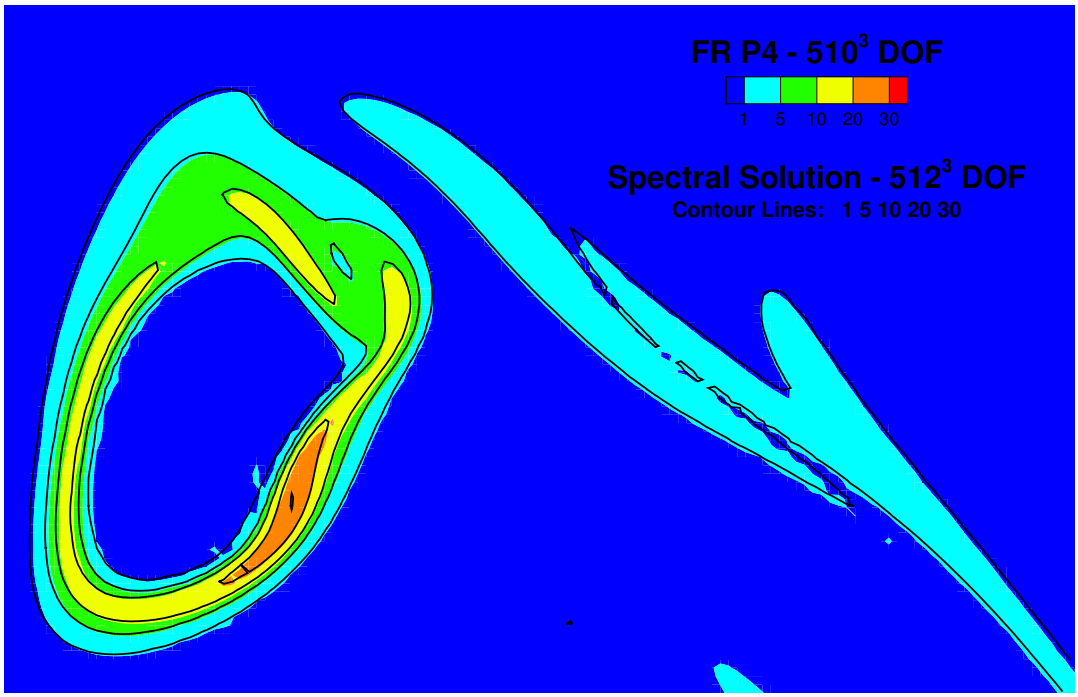


Figure 11: Contours from a $\mathcal{P}4$ simulation with 510^3 DoF showing the vorticity norm at $t^* = 8$ on the plane $x = -\pi L$, in the region bounded by the lines $y = [0, \frac{\pi L}{2}]$ and $z = [\frac{\pi L}{2}, \pi L]$. The contour lines found using a pseudo-spectral code with 512^3 DoF have been superimposed as a reference solution for comparison.

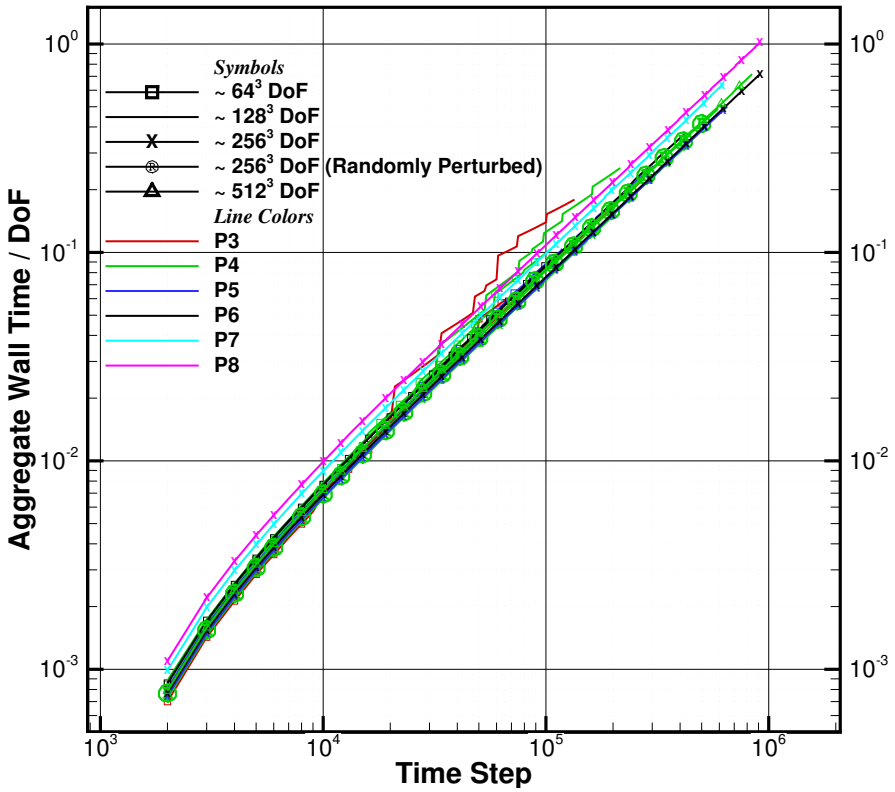


Figure 12: Comparison of the aggregate wall times per DoF versus time step for each of the Taylor-Green vortex simulations. This shows that for each time step, the computational cost per DoF is essentially independent of $\mathcal{P}$. Therefore, any additional computational cost incurred by increasing $\mathcal{P}$ must be caused by the smaller time step restriction increasing the number of time steps required to reach a given simulation time (t^*).

III.B.5. Performance

All but two of the Taylor-Green vortex simulations were run on Haswell compute nodes; the $\mathcal{P}3$ solutions containing 64^3 and 128^3 DoF were run on Sandy Bridge nodes. Because these two CPU architectures prevent a fair performance comparison between simulations, a wall-time normalization factor was created using the TauBench benchmark program.[50] Specifically, a series of five benchmarks was run on a single computational node of each type, making sure that: a) the same input parameters to TauBench were used for both node types, b) all CPU cores on the node were used for each benchmark, and c) the TauBench program was compiled for each node type using the same optimizations used to compile GFR. The averaged wall time for the five benchmarks on a given node architecture was then used to normalize the actual wall times for all simulations run on that same node architecture; the average of the benchmark wall times was 22.667 seconds for a Sandy Bridge node and 18.999 seconds for a Haswell node. Finally, since the number of CPU cores used for each simulation varied, the architecture-adjusted wall time was multiplied by the number of CPU cores to get an architecture-adjusted, aggregate wall time which provides a fair comparison of the computational cost between simulations.

For all the Taylor-Green vortex simulations, the computational cost per DoF is shown in figure 12 as a function of the number of completed time steps. There are two major deductions to be made from this figure. First, figure 12 simultaneously combines and reinforces the conclusions made from figures 1 and 2. Not counting the small benchmark cases from section II.C and section II.D, there are a total of 16 simulations of the Taylor-Green vortex problem presented in this work, and table 1 clearly shows the significant variations between all of the simulations in terms of grid size and the number of processors used. The fact that *all* 16 of these simulations are very closely aligned in figure 12 demonstrates a) the sparse matrix algorithms have essentially made the computational cost per time step independent of $\mathcal{P}$, and b) the current parallel

implementation scales reasonably well for all combinations of $\mathcal{P}$ and DoF used in this study. Second, if the cost per iteration is the same between two simulations with a similar number of DoF but different $\mathcal{P}$, the difference in wall time to reach a given value of t^* must be due to the difference in the number of time steps taken. Therefore, the stability restriction on the time step size for explicit methods remains one of the biggest obstacles preventing the use of FR, DG, and other high-order methods at very high orders, e.g., $\mathcal{P} \geq 6$.

One last comment needs to be made about the results in figure 12 regarding the jumps in the wall time present in a number of the simulations. Each of these jumps coincides with a CGNS solution file being written to disk at predetermined simulation times of the TGV problem; specifically, the jumps occur when t^* is 3, 5, 7, 8, 9, 11, and 15. At the time of this paper being prepared, the current stable release of CGNS, version 3.2.1, is the version implemented within GFR. This version of parallel CGNS, as well as all other previous versions, is known to experience major slowdowns when trying to open a file using a large number of processors. Version 3.3.0 is currently in its alpha release stage, and one of the major changes in this new update is significant performance improvements to the parallel CGNS functions.

IV. Summary and Conclusions

A new CFD code named GFR is currently being developed at NASA Glenn Research Center to ultimately provide efficient large-eddy simulation capabilities for complex aeropropulsion flows. A description of this code was presented, detailing the underlying numerical methods and some of its capabilities. This code uses the flux reconstruction method to solve the differential form of the Navier-Stokes equations, and various explicit Runge-Kutta schemes have been implemented for advancing the solution in time.

The FR method offers a simple and efficient method that is easy to implement and accurate to an arbitrary order on common grid cell geometries. A formulation of the FR method that unifies all grid geometries is achieved by reducing the method to a series of matrix operations. These matrix operators are very sparse for structured geometries, and type-bound procedure pointers, a new feature introduced by the Fortran 2003 standard, provide a path to significantly reduce computational costs by utilizing this sparsity. By implementing this strategy in GFR, the computational cost of using a $\mathcal{P}8$ solution is nearly 5 times less than ignoring the sparsity and using the intrinsic functions.

Parallelization of GFR is achieved through MPI, and concurrently running computations and non-blocking communication are used to mask the expensive costs associated with communicating data between processors. Both weak and strong scaling tests show good results with minimal tuning of the parallel implementation. Our results indicate the scalability begins to decrease (to approximately 80–90 % of ideal) for higher processor counts, but improvements are expected with adjustments to the partitioning algorithms and the method of distributing the partitions across the processors.

Using GFR, we were able to complete an h-refinement study of the isentropic Euler vortex problem and successfully verified the accuracy of the FR method for inviscid flows. Specifically, the results of the study found the expected $\mathcal{P} + 1$ order accuracy through one period and super-accuracy, i.e., greater than $\mathcal{P} + 1$, through 100 periods. The results from this case study demonstrate that vortical flow structures can be accurately predicted by the FR method for long amounts of time with minimal unwanted numerical dissipation introduced into the system.

The application of GFR to solve the Taylor-Green vortex problem demonstrated tremendous potential for its application to complex aerodynamic flows of engineering interest. The enstrophy based kinetic energy dissipation rate, $\epsilon(\zeta)$, is a theoretical measurement of the physical dissipation from the resolved vortical structures within the solution. This value was under-predicted in all simulations, especially for those containing more unresolved scales due to less grid resolution. However, the directly computed KEDR, $\epsilon(E_k)$, for each simulation followed the overall trend of the reference spectral solution with the highest grid resolutions showing very good agreement to this reference solution. The discrepancy between the theoretical and actual dissipation contained in the solution is representative of the numerical dissipation from the FR method. The accuracy of both $\epsilon(E_k)$ and $\epsilon(\zeta)$ consistently improved by increasing either or both grid resolution and $\mathcal{P}$, which suggests that the numerical dissipation from the FR method adequately approximates the physical dissipation contained within the unresolved scales. Altogether, these results indicate that the FR method appears to be a viable candidate for accurate ILES of aerodynamic flows for complex geometries.

References

[1]DeBonis, J., "Progress Towards Large-Eddy Simulations for Prediction of Realistic Nozzle Systems," *AIAA Journal of Propulsion and Power*, Vol. 23, No. 5, 2007, pp. 971–980.

[2]Forsythe, J., Squires, K., Wurtzler, K., and Spalart, P., "Detached-Eddy Simulation of Fighter Aircraft at High-Alpha," *Journal of Aircraft*, Vol. 41, No. 2, 2004, pp. 193–200.

[3]Mavripilis, D., Pelaez, J., and Kandil, O., "Large Eddy and Detached Eddy Simulations Using an Unstructured Multigrid Solver," *DNS/LES - Progress and Challenges. Proceedings of the Third AFOSR International Conference on DNS/LES*, Columbus, OH, 2001.

[4]Moreau, S., Christophe, J., and Roger, M., "LES of the Trailing-Edge Flow and Noise of a NACA0012 Airfoil Near Stall," *Proceedings of the Summer Program 2008, Center for Turbulence Research*, Stanford University/NASA Ames, 2008.

[5]Kang, S., Iaccarino, G., Ham, F., and Moin, P., "Prediction of Wall-Pressure Fluctuation in Turbulent Flows with an Immersed Boundary Method," *Journal of Computational Physics*, Vol. 228, No. 18, 2009, pp. 6753–6772.

[6]Huynh, H., "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods," AIAA Paper 2007-4079, Jun 2007.

[7]Huynh, H., "A Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin for Diffusion," AIAA Paper 2009-403, Jan 2009.

[8]Huynh, H., "High-Order Methods Including Discontinuous Galerkin by Reconstructions on Triangular Meshes," AIAA Paper 2011-44, Jan 2011.

[9]Vermeire, B. C., Cagnone, J.-S., and Nadarajah, S., "ILES Using the Correction Procedure via Reconstruction Scheme," AIAA Paper 2013-1001, Jan 2013.

[10]Vermeire, B. C., Nadarajah, S., and Tucker, P. G., "Canonical Test Cases for High-Order Unstructured Implicit Large Eddy Simulation," AIAA Paper 2014-0935, Jan 2014.

[11]Skarolek, V. and Miyaji, K., "Transitional Flow over a SD7003 Wing Using Flux Reconstruction Scheme," AIAA Paper 2014-0250, Jan 2014.

[12]Haga, T., Tsutsumi, S., Kawai, S., and Takaki, R., "Large-Eddy Simulation of a Supersonic Jet Using High-Order Flux Reconstruction Scheme," AIAA Paper 2015-0831, Jan 2015.

[13]Wang, Z. and Gao, H., "A Unifying Lifting Collocation Penalty Formulation Including the Discontinuous Galerkin, Spectral Volume/Difference Methods for Conservation Laws on Mixed Grids," *Journal of Computational Physics*, Vol. 228, No. 21, 2009, pp. 8161–8186.

[14]Castonguay, P., Williams, D. M., Vincent, P. E., Lopez, M., and Jameson, A., "On the Development of a High-Order, Multi-GPU Enabled, Compressible Viscous Flow Solver for Mixed Unstructured Grids," AIAA Paper 2011-3229, Jun 2011.

[15]Karniadakis, G. and Sherwin, S., *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford University Press, 2nd ed., 2013.

[16]Hesthaven, J. S. and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Vol. 54 of *Texts in Applied Mathematics*, Springer New York, 2008.

[17]Kopriva, D. A., "Metric Identities and the Discontinuous Spectral Element Method on Curvilinear Meshes," *Journal of Scientific Computing*, Vol. 26, No. 3, 2006, pp. 301–327.

[18]Kopriva, D. A., *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*, Springer Science & Business Media, 2009.

[19]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[20]Huynh, H. T., "Accurate Upwind Methods for the Euler Equations," *SIAM Journal on Numerical Analysis*, Vol. 32, No. 5, 1995, pp. 1565–1619.

[21]Vincent, P., Castonguay, P., and Jameson, A., "A New Class of High-Order Energy Stable Flux Reconstruction Schemes," *Journal of Scientific Computing*, Vol. 47, No. 1, 2011, pp. 50–72.

[22]Castonguay, P., Vincent, P., and Jameson, A., "A New Class of High-Order Energy Stable Flux Reconstruction Schemes for Triangular Elements," *Journal of Scientific Computing*, Vol. 51, No. 1, 2012, pp. 224–256.

[23]Vincent, P. E., Farrington, A. M., Witherden, F. D., and Jameson, A., "An Extended Range of Stable-Symmetric-Conservative Flux Reconstruction Correction Functions," *Computer Methods in Applied Mechanics and Engineering*, Vol. 296, 2015, pp. 248–272.

[24]Bassi, F. and Rebay, S., "A High Order Discontinuous Galerkin Method for Compressible Turbulent Flows," *Discontinuous Galerkin Methods: Theory, Computation, and Application*, edited by B. Cockburn, G. E. Karniadakis, and C.-W. Shu, Lecture Notes in Computational Science and Engineering, Springer, 2000, pp. 77–88.

[25]Gottlieb, S. and Shu, C.-W., "Total Variation Diminishing Runge-Kutta Schemes," *Mathematics of Computation*, Vol. 67, No. 221, 1998, pp. 73–85.

[26]Carpenter, M. H. and Kennedy, C. A., "Fourth-Order 2N-Storage Runge-Kutta Schemes," NASA TM-1994-109112, 1994.

[27]Spiteri, R. J. and Ruuth, S. J., "A New Class of Optimal High-Order Strong-Stability-Preserving Time Discretization Methods," *SIAM Journal on Numerical Analysis*, Vol. 40, No. 2, 2002, pp. 469–491.

[28]Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard (version 1.1)," Technical report, 1995, http://www.mpi-forum.org.

[29]Karypis, G. and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 1999, pp. 359–392.

[30]Karypis, G. and Kumar, V., "Multilevel k-way Partitioning Scheme for Irregular Graphs," *Journal of Parallel and Distributed Computing*, Vol. 48, No. 1, 1998, pp. 96–129.

[31]Hendrickson, B., "Graph Partitioning and Parallel Solvers: Has the Emperor No Clothes?" *In Proc. Irregular98*, Springer-Verlag, 1998, pp. 218–225.

[32]Rumsey, C. L., Wedan, B., Hauser, T., and Poinot, M., "Recent Updates to the CFD General Notation System (CGNS)," AIAA Paper 2012-1264, Jan 2012.

[33]The HDF Group, "Hierarchical Data Format, version 5," 1997–2016, http://www.hdfgroup.org/HDF5/.

[34]Message Passing Interface Forum, "MPI-2: Extensions to the Message-Passing Interface," Technical report, 1997, http://www.mpi-forum.org.

[35]Shu, C.-W., "Essentially Non-oscillatory and Weighted Essentially Non-oscillatory Schemes for Hyperbolic Conservation Laws," *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, edited by A. Quarteroni, Vol. 1697 of *Lecture Notes in Mathematics*, Springer Berlin Heidelberg, 1998, pp. 325–432.

[36]Wang, Z. J., Liu, Y., May, G., and Jameson, A., "Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations," *Journal of Scientific Computing*, Vol. 32, No. 1, 2007, pp. 45–71.

[37]Castonguay, P., Vincent, P., and Jameson, A., "Application of High-Order Energy Stable Flux Reconstruction Schemes to the Euler Equations," AIAA Paper 2011-686, Jan 2011.

[38]Vincent, P., Castonguay, P., and Jameson, A., "Insights from von Neumann Analysis of High-Order Flux Reconstruction Schemes," *Journal of Computational Physics*, Vol. 230, No. 22, 2011, pp. 8134–8154.

[39]Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 811–845.

[40]Gao, H. and Wang, Z., "A Conservative Correction Procedure via Reconstruction Formulation with the Chain-Rule Divergence Evaluation," *J. Comput. Physics*, Vol. 232, No. 1, 2013, pp. 7–13.

[41]Williams, D. M. and Jameson, A., "Nodal Points and the Nonlinear Stability of High-Order Methods for Unsteady Flow Problems on Tetrahedral Meshes," AIAA Paper 2013-2830, Jun 2013.

[42]Witherden, F. D., Farrington, A. M., and Vincent, P. E., "PyFR: An Open Source Framework for Solving Advection-Diffusion Type Problems on Streaming Architectures Using the Flux Reconstruction Approach," *Computer Physics Communications*, Vol. 185, No. 11, 2014, pp. 3028–3040.

[43]De Grazia, D., Mengaldo, G., Moxey, D., Vincent, P. E., and Sherwin, S. J., "Connections Between the Discontinuous Galerkin Method and High-Order Flux Reconstruction Schemes," *International Journal for Numerical Methods in Fluids*, Vol. 75, No. 12, 2014, pp. 860–877.

[44]Yu, M., Wang, Z., and Liu, Y., "On the Accuracy and Efficiency of Discontinuous Galerkin, Spectral Difference and Correction Procedure via Reconstruction Methods," *Journal of Computational Physics*, Vol. 259, 2014, pp. 70–95.

[45]Spiegel, S., Huynh, H. T., and DeBonis, J. R., "A Survey of the Isentropic Euler Vortex Problem Using High-Order Methods," AIAA Paper 2015-2444, Jun 2015.

[46]Spiegel, S., Huynh, H. T., and DeBonis, J. R., "De-Aliasing through Over-Integration Applied to the Flux Reconstruction and Discontinuous Galerkin Methods," AIAA Paper 2015-2744, Jun 2015.

[47]van Rees, W. M., Leonard, A., Pullin, D. I., and Koumoutsakos, P., "A Comparison of Vortex and Pseudo-Spectral Methods for the Simulation of Periodic Vortical Flows at High Reynolds Numbers," *Journal of Computational Physics*, Vol. 230, 2011, pp. 2794–2805.

[48]Kirby, R. M. and Karniadakis, G. E., "De-Aliasing on Non-Uniform Grids: Algorithms and Applications," *Journal of Computational Physics*, Vol. 191, No. 1, 2003, pp. 249–264.

[49]Gassner, G. J. and Beck, A. D., "On the Accuracy of High-Order Discretizations for Underresolved Turbulence Simulations," *Theoretical and Computational Fluid Dynamics*, Vol. 27, No. 3-4, 2013, pp. 221–237.

[50]DLR Germany, "TauBench - IPACS," http://www.ipacs-benchmark.org.