AIAA JOURNAL
Vol. 58, No. 1, January 2020

Impact of Number Representation for High-Order Implicit Large-Eddy Simulations

F. D. Witherden* and A. Jameson†
Texas A&M University, College Station, Texas 77843

DOI: 10.2514/1.J058434

High-order numerical methods for unstructured grids combine the superior accuracy of high-order spectral or finite difference methods with the geometric flexibility of low-order finite volume or finite element schemes. Over the past few years, they have shown promise in enabling implicit large-eddy simulations within the vicinity of complex geometrical configurations. However, the cost of such simulations remains prohibitive. In this paper, the impact of number representation on the efficiency and efficacy of these methods is considered. Theoretical performance models are derived and assessed. Specifically, four test cases are considered: 1) the viscous Taylor–Green vortex, 2) flow over a circular cylinder, 3) flow through a T106c low-pressure turbine cascade, and 4) flow around a NACA 0021 in deep stall. It is found that the use of single (in lieu of double) precision arithmetic does not have a significant impact on accuracy. The performance of the resulting simulations was found to improve between 1.4 and 2.9 times.

Nomenclature

$\mathfrak{C}_\alpha$	=	common solution at an interface
$\det \boldsymbol{A}$	=	matrix determinant
$\dim \boldsymbol{A}$	=	matrix dimensions
e	=	element type
$\mathfrak{F}_\alpha$	=	common normal flux at an interface
i, j, k	=	summation indices
$\ell_{e\rho}$	=	nodal basis polynomial ρ for element type e
$\boldsymbol{\mathcal{M}}_{en}$	=	transformed to physical mapping
N_D	=	number of spatial dimensions
N_V	=	number of field variables
n	=	element number
$\wp$	=	polynomial order
x, y, z	=	physical coordinates
$\tilde{x}, \tilde{y}, \tilde{z}$	=	transformed coordinates
α	=	field variable number
δ_{ij}	=	Kronecker delta
$\partial\boldsymbol{\Omega}_e$	=	boundary of $\hat{\boldsymbol{\Omega}}_e$
ρ, σ, ν	=	summation indices
$\boldsymbol{\Omega}$	=	solution domain
$\boldsymbol{\Omega}_e$	=	all elements in $\boldsymbol{\Omega}$ of type e
$\lvert\boldsymbol{\Omega}_e\rvert$	=	number of elements of type e
$\boldsymbol{\Omega}_{en}$	=	element n of type e in $\boldsymbol{\Omega}$
$\hat{\boldsymbol{\Omega}}_e$	=	standard element of type e

Superscripts

(f)	=	quantity at a flux point
$(f_\perp)$	=	normal quantity at a flux point
T	=	transpose
(u)	=	quantity at a solution point
$\sim$	=	quantity in transformed space
$\hat{}$	=	vector quantity of unit magnitude

I. Introduction

THERE is an increasing desire among industrial practitioners of computational fluid dynamics (CFD) to perform scale-resolving simulations (SRSs) of unsteady compressible flows within the vicinity of complex geometries. However, current generation industry-standard CFD software, predominantly based on first- or second-order-accurate Reynolds-averaged Navier–Stokes technology, is not well suited to the task. Henceforth, over the past decade, there has been significant interest in the application of high-order methods for mixed unstructured grids to such problems. Popular examples of high-order schemes for mixed unstructured grids include the discontinuous Galerkin (DG) method, first introduced by Reed and Hill [1], and the spectral difference (SD) methods originally proposed under the moniker "staggered-gird Chebyshev multidomain methods" by Kopriva and Kolias in 1996 [2] and later popularized by Sun et al. [3]. In 2007, Huynh [4] proposed the flux reconstruction (FR) approach: a unifying framework for high-order schemes on unstructured grids that incorporates both the nodal DG schemes of Ref. [5] and (at least for a linear flux function) any SD scheme. In addition to offering high-order accuracy on unstructured mixed grids, FR schemes are also compact in space; thus, when combined with explicit time marching, they offer a significant degree of element locality. This locality makes such schemes extremely good candidates for acceleration using either the vector units of modern CPUs or graphics processing units (GPUs) [6–9]. However, even with the synergistic combination of high-order methods and modern hardware platforms, SRSs remain expensive.

When performing a SRS, it is not typically necessary to resolve all of the turbulent scales as would be done in a direct numerical simulation (DNS). Instead, one may resolve only the large-scale structures and use a model to account for the dynamics of the small-scale structures, which are assumed to be isotropic and have a predominantly dissipative effect. This is accomplished with a subgrid scale model, and the resulting computation is termed a large-eddy simulation (LES). More recently, there has been a trend (at least within the high-order community) toward abnegating the subgrid scale model and instead relying on the numerical scheme to introduce the right amount of dissipation. Such a computation is known as an implicit LES (ILES) or, perhaps more appropriately, an under resolved DNS.

The majority of the scientific codes represent and manipulate numbers using floating-point arithmetic. Historically, floating-point arithmetic has existed in various vendor specific flavors. However, starting in the late 1980s the industry has converged toward the IEEE Standard for Floating-Point Arithmetic (IEEE 754) [10]. The standard defines both decimal (base 10) and binary (base two) floating-point formats at a range of precisions. In the consumer and high-performance computing markets, vendors typically implement the base-two binary32 and binary64 formats, which are commonly referred to as single and double precision, respectively. In the C programming language, these types are specified as `float` and `double`, respectively. The bit representation of these two formats is illustrated in Fig. 1.

Received 14 March 2019; revision received 11 August 2019; accepted for publication 12 August 2019; published online 17 September 2019.

*Assistant Professor, Department of Ocean Engineering, 3145 TAMU.
†Professor, Department of Ocean Engineering, 3145 TAMU.

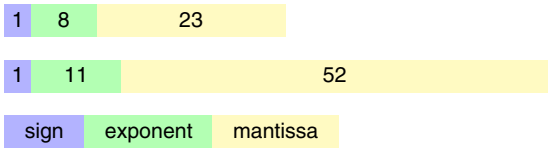

Fig. 1 Bit lengths for the binary32 and binary64 IEEE STD 754 floating-point numbers.

The extra bits in the exponent give double precision floats a range of $\sim 10^{\pm 308}$, as compared to $\sim 10^{\pm 38}$ for single precision. Similarly, the mantissa of a double precision number is accurate to ~ 16 significant figures as compared to just ~ 7 significant figures for a single precision number. As a consequence of this, double precision numbers are considered sufficient for much of the scientific community (albeit with some notable exceptions involving quad and arbitrary precision [11]). Hence, at the time of writing, double precision arithmetic is very much the norm. Nevertheless, single precision has been employed successfully in a variety of fields, including those of molecular dynamics [12–14] and the computational geosciences [15–17]. Within the context of CFD, Homann et al. [18] considered the impact of number precision for incompressible DNS using a pseudospectral method. Their conclusion was that number representation had no substantial impact on the turbulent statistics of interest. The goal of this paper is to assess whether single precision can also be reliably applied to compressible high-order implicit LES and, if so, how performance is affected.

The remainder of this paper is structured as follows. In Sec. II, we consider, from a theoretical standpoint, the performance impact of a wholesale conversion of a code from double to single precision. A brief overview of the FR approach for solving the Navier–Stokes equations on mixed unstructured grids in three dimensions is presented in Sec. III, with our implementation of the FR approach for modern hardware platforms, PyFR, being presented in Sec. IV. In Sec. V, we use PyFR to assess the impact of number representation for four test cases: the Taylor–Green vortex at $Re = 1600$, flow over a circular cylinder at $Re = 3900$, flow through a T106c low-pressure turbine cascade at $Re = 80{,}000$, and flow around a NACA 0021 airfoil at $Re = 270{,}000$ in deep stall. Finally, in Sec. VI, conclusions are drawn.

Throughout, we adopt a convention in which dummy indices on the right-hand side of an expression are summed. For example,

$$C_{ijk} = A_{ijl}B_{ilk} \equiv \sum_{l} A_{ijl}B_{ilk}$$

where the limits are implied from the surrounding context. All indices are assumed to be zero based.

II. Performance Considerations

There are a variety of mechanisms through which the use of single precision within an application, in lieu of double precision, can result in better performance. This improvement can, in turn, be translated into either a reduced overall time to solution or reduced resource requirements. In this section, we will explore these mechanisms as they relate to the capabilities of modern computing hardware.

Table 1 summarizes the characteristics of a variety of modern hardware platforms. Looking at the table, we see that, on Intel-based platforms, the ratio between the single precision and double precision peak floating point operation (FLOP)/s is exactly two. However, on GPU platforms from both NVIDIA and AMD, there is much more variation. The highest ratio is found on the NVIDIA Tesla M40, which has just 0.21 teraflop (TFLOP)/s of double precision compute but 7.00 TFLOP/s of single precision compute. The relative abundance of single precision compute on these platforms is one of the key motivators for this investigation. Assuming a wholesale conversion of a code to single precision, the factors that dictate the overall performance improvement are roughly as follows:

Table 1 Theoretical peak memory bandwidth and FLOP/s for a variety of modern hardware platforms[a]

		TFLOP/s		
Model	GB/s	Single	Double	Ratio
AMD Radeon R9 Nano	512	8.19	0.51	16
AMD FirePro W9100	320	5.24	2.62	2
Intel Xeon E5-2699 v4	77	1.55	0.77	2
Intel Xeon Phi 7290	400	6.91	3.46	2
NVIDIA Tesla V100	900	15.7	7.8	2
NVIDIA Tesla K40c	288	4.29	1.43	3
NVIDIA Tesla M40	288	7.00	0.21	32

[a]Boosting and throttling technologies may result in a deviation from the theoretical peaks, depending on the specifics of the workload.

A. Memory Requirements

A float is exactly half the size of a double; 4 B as compared to 8 B. Hence, by switching to single precision, there is the potential for a twofold reduction in the memory required by an application. This reduction, however, is predicated on all allocated memory being used for numerical data. For codes operating on unstructured grids, it is necessary to store additional data describing the connectivity of the grid. These data usually take the form of either pointers or integer arrays. With few exceptions, the size of these data is unaffected by the number representation. Hence, the reduction in memory follows from Amdahl's law [19] as

$$R = \frac{1}{(1-p) + p/2} \tag{1}$$

where R is the reduction in memory, and p is the proportion of application memory allocated to double precision variables.

B. Memory Bandwidth

On today's hardware, many popular numerical discretizations for unsteady problems are limited not by compute but by memory bandwidth [8]. In the general case, a transition to single precision may yield a performance improvement of up to two times; although, as with the memory requirements, it is important to also account for non-numerical data. For an unstructured code, this reduces the potential speedup somewhat.

An additional subtlety arises on GPU platforms that can, under a specific set of circumstances, improve the performance of bandwidth limited code by more than a factor of two. Unlike traditional CPU architectures, which define a fixed set of registers as part of the instruction set, GPUs are able to allocate their registers dynamically up to some defined architectural limit. These registers are of a fixed size: usually 32 bits. When the compiler runs out of available registers, it becomes necessary to spill values onto the stack that is backed by global memory. Register spillage is therefore undesirable because it consumes valuable memory bandwidth. A consequence of the registers being of a fixed size is that a kernel employing single precision will usually require fewer registers than the equivalent double precision kernel, and hence result in less spillage. The impact of this is most pronounced when the double precision variant of a kernel spills, whereas the single precision variant kernel does not. Here, the saved memory bandwidth is translated into a greater than expected performance improvement.

C. Memory Latency

When operating on unstructured grids, some degree of memory indirection is unavoidable. This indirection results in access to non-contiguous locations in memory. On modern hardware platforms, the penalty associated with a cache miss for reading a single precision number is equivalent to that for a double precision number. Hence, kernels limited by memory latency, as opposed to bandwidth, are unlikely to see any benefit from single precision.

D. Compute

The potential speedup from compute bound applications by switching to single precision varies greatly across hardware platforms. Looking at the peak TFLOP/s column in Table 1, we observe that the ratio of single to double precision computes varies from 2 to 32. The practicalities associated with achieving these speedups vary greatly between platforms. On CPUs, the instructions that operate on single precision vectors have similar throughputs and latencies to their double precision counterparts, except that each instruction now operates on twice as many elements. Accordingly, for kernels that are suitably vectorized, a factor-of-two speedup is eminently achievable.

The situation for GPUs is somewhat more complicated, however. Depending on the specifics of the hardware, it might only be possible to obtain peak single precision FLOP/s under very specific circumstances. For example, Lai and Seznec [20] estimated that a single precision matrix multiplication kernel on an NVIDIA GTX 680, which is a GPU with a theoretical peak of ∼3 TFLOP/s, is limited by architectural constraints to just 57.6% of this peak.

As in the case of memory bandwidth, there are nonlinearities associated with the switch from double to single precision that can result in larger than expected speedups. For example, on Intel and AMD CPUs, there are pairs of instructions for quickly computing single precision estimates of $\frac{1}{x}$ and $\frac{1}{\sqrt{x}}$, which can be polished using either one or two Newton–Raphson iterations to achieve a suitable level of accuracy. This results in code sequences that are more than two times faster than their double precision counterparts.

The improvements in compute performance associated with the use of single precision are, of course, restricted to the portions of a code that perform numerical computation. Taking r to be the ratio of single to double precision computes and applying Amdahl's law, we see that

$$S = \frac{1}{(1-p) + p/r} \tag{2}$$

where S is the maximum attainable speedup, and p is the proportion of time spent performing numerical computations.

E. Disk Input/Output

Given that the available I/O throughput on parallel file systems has not kept pace with the increasing compute capability of modern hardware, it is not uncommon for disk I/O to constitute a meaningful fraction of the total wall-clock time. For applications for which the file dumps contain purely volumetric data, the improvement from switching to single precision is roughly a factor of two in both disk I/O and storage/archival requirements.

F. Power Draw

Single precision floating-point operations require fewer joules than double precision floating-point operations. As an example of this, we consider the power drawn by an NVIDIA K40c while performing a large matrix–matrix multiplication. When running single precision, the reported power draw is ∼193 W at ∼3.1 TFLOP/s; for double precision, the reported power draw is ∼199 W at ∼1.2 TFLOP/s. Thus, while the use of single precision will not necessarily reduce the peak power draw, it can be expected to reduce the energy bill by a factor that is proportional to the speedup.

III. Flux Reconstruction

For a detailed overview of the multidimensional flux reconstruction approach, the reader is referred to the work of Witherden et al. [8]. In this section, we shall describe the approach within the context of the three dimensional Navier–Stokes equations. In conservative form, these can be expressed as

$$\frac{\partial u_\alpha}{\partial t} + \nabla \cdot \boldsymbol{f}_\alpha = 0 \tag{3}$$

where α is the field variable index, $u = u(\boldsymbol{x}, t) = \{\rho, \rho v_x, \rho v_y, \rho v_z, E\}$ is the solution, ρ is the mass density of the fluid, $\boldsymbol{v} = (v_x, v_y, v_z)^T$ is the fluid velocity vector, and E is the total energy per unit volume. For a perfect gas, the pressure p and total energy can be related by the ideal gas law

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \|\boldsymbol{v}\|^2 \tag{4}$$

where $\gamma = c_p/c_v$. The flux can be written as $\boldsymbol{f}_\alpha = \boldsymbol{f}_\alpha(u, \nabla u) = \boldsymbol{f}^{(\text{inv})} - \boldsymbol{f}^{(\text{vis})}$, where the inviscid terms are given by

$$\boldsymbol{f}^{(\text{inv})} = \begin{pmatrix} \rho v_x & \rho v_y & \rho v_z \\ \rho v_x^2 + p & \rho v_y v_x & \rho v_z v_x \\ \rho v_x v_y & \rho v_y^2 + p & \rho v_z v_y \\ \rho v_x v_z & \rho v_y v_z & \rho v_z^2 + p \\ v_x(E+p) & v_y(E+p) & v_z(E+p) \end{pmatrix} \tag{5}$$

and the viscous terms are given according to

$$\boldsymbol{f}^{(\text{vis})} = \begin{pmatrix} 0 & 0 & 0 \\ \mathcal{T}_{xx} & \mathcal{T}_{yx} & \mathcal{T}_{zx} \\ \mathcal{T}_{xy} & \mathcal{T}_{yy} & \mathcal{T}_{zy} \\ \mathcal{T}_{xz} & \mathcal{T}_{yz} & \mathcal{T}_{zz} \\ v_i\mathcal{T}_{ix} + \Delta\partial_x T & v_i\mathcal{T}_{iy} + \Delta\partial_y T & v_i\mathcal{T}_{iz} + \Delta\partial_z T \end{pmatrix} \tag{6}$$

In the preceding, we have defined $\Delta = \mu c_p/Pr$, where μ is the dynamic viscosity and Pr is the Prandtl number. The components of the stress-energy tensor are given by

$$\mathcal{T}_{ij} = \mu(\partial_i v_j + \partial_j v_i) - \frac{2}{3}\mu\delta_{ij}\nabla \cdot \boldsymbol{v} \tag{7}$$

where δ_{ij} is the Kronecker delta. Using the ideal gas law, the temperature can be expressed as

$$T = \frac{1}{c_v}\frac{1}{\gamma - 1}\frac{p}{\rho} \tag{8}$$

with partial derivatives thereof being given according to the quotient rule.

To solve this system using flux reconstruction, we start by rewriting Eq. (3) as a first-order system

$$\frac{\partial u_\alpha}{\partial t} + \nabla \cdot \boldsymbol{f}_\alpha(u, \boldsymbol{q}) = 0 \tag{9a}$$

$$\boldsymbol{q}_\alpha - \nabla u_\alpha = 0 \tag{9b}$$

where $\boldsymbol{q}$ is an auxiliary variable.

Take $\mathcal{E}$ to be the set of supported element types in $\mathbb{R}^3$. Consider using these various element types to construct a conformal body-fitted mesh of a domain $\boldsymbol{\Omega}$. Inside of each element $\boldsymbol{\Omega}_{en}$, where the index e corresponds to the element type and n to the type-local element number, we require that

$$\frac{\partial u_{en\alpha}}{\partial t} + \nabla \cdot \boldsymbol{f}_{en\alpha} = 0 \tag{10a}$$

$$\boldsymbol{q}_{en\alpha} - \nabla u_{en\alpha} = 0 \tag{10b}$$

It is convenient, for reasons of both mathematical simplicity and computational efficiency, to work in a transformed space. We accomplish this by introducing, for each element type, a standard element $\hat{\boldsymbol{\Omega}}_e$ that exists in a transformed space: $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{z})^T$. Next, we assume the existence of a mapping function for each element such that

$$x_i = \mathcal{M}_{eni}(\tilde{\boldsymbol{x}}), \qquad \boldsymbol{x} = \boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}}),$$
$$\tilde{x}_i = \mathcal{M}^{-1}_{eni}(\boldsymbol{x}), \qquad \tilde{\boldsymbol{x}} = \boldsymbol{\mathcal{M}}^{-1}_{en}(\boldsymbol{x})$$

along with the relevant Jacobian matrices

$$\boldsymbol{J}_{en} = J_{enij} = \frac{\partial \mathcal{M}_{eni}}{\partial \tilde{x}_j}, \qquad J_{en} = \det \boldsymbol{J}_{en},$$
$$\boldsymbol{J}^{-1}_{en} = J^{-1}_{enij} = \frac{\partial \mathcal{M}^{-1}_{eni}}{\partial x_j}, \qquad J^{-1}_{en} = \det \boldsymbol{J}^{-1}_{en} = \frac{1}{J_{en}}$$

These definitions provide us with a means of transforming quantities to and from standard element space. Taking the transformed solution, flux, and gradients inside each element to be

$$\tilde{u}_{en\alpha} = \tilde{u}_{en\alpha}(\tilde{\boldsymbol{x}}, t) = J_{en}(\tilde{\boldsymbol{x}}) u_{en\alpha}(\boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}}), t) \tag{11a}$$

$$\tilde{\boldsymbol{f}}_{en\alpha} = \tilde{\boldsymbol{f}}_{en\alpha}(\tilde{\boldsymbol{x}}, t) = J_{en}(\tilde{\boldsymbol{x}}) \boldsymbol{J}^{-1}_{en}(\boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}})) \boldsymbol{f}_{en\alpha}(\boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}}), t) \tag{11b}$$

$$\tilde{\boldsymbol{q}}_{en\alpha} = \tilde{\boldsymbol{q}}_{en\alpha}(\tilde{\boldsymbol{x}}, t) = \boldsymbol{J}^{T}_{en}(\tilde{\boldsymbol{x}}) \boldsymbol{q}_{en\alpha}(\boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}}), t) \tag{11c}$$

and letting $\tilde{\nabla} = \partial / \partial \tilde{x}_i$, it can be readily verified that

$$\frac{\partial u_{en\alpha}}{\partial t} + J^{-1}_{en} \tilde{\nabla} \cdot \tilde{\boldsymbol{f}}_{en\alpha} = 0 \tag{12a}$$

$$\tilde{\boldsymbol{q}}_{en\alpha} - \tilde{\nabla} u_{en\alpha} = 0 \tag{12b}$$

as required.We next proceed to associate a set of solution points with each standard element. For each type $e \in \mathcal{E}$, take $\{\tilde{\boldsymbol{x}}^{(u)}_{ep}\}$ to be the chosen set of points where $0 \le \rho < N^{(u)}_e(\wp)$, and $\wp$ is the polynomial order. These points can then be used to construct a nodal basis set $\{\ell^{(u)}_{ep}(\tilde{\boldsymbol{x}})\}$ with the property that $\ell^{(u)}_{ep}(\tilde{\boldsymbol{x}}^{(u)}_{e\sigma}) = \delta_{\rho\sigma}$. To obtain such a set, we first take $\{\psi_{e\sigma}(\tilde{\boldsymbol{x}})\}$ to be an orthonormal basis that spans a selected order $\wp$ polynomial space defined inside $\hat{\boldsymbol{\Omega}}_e$. Next, we compute the elements of the generalized Vandermonde matrix $\mathcal{V}_{ep\sigma} = \psi_{ep}(\tilde{\boldsymbol{x}}^{(u)}_{e\sigma})$. With these, a nodal basis set can be constructed as $\ell^{(u)}_{ep}(\tilde{\boldsymbol{x}}) = \mathcal{V}^{-1}_{ep\sigma} \psi_{e\sigma}(\tilde{\boldsymbol{x}})$. Along with the solution points inside of each element, we also define a set of flux points on $\partial\hat{\boldsymbol{\Omega}}_e$. We denote the flux points for a particular element type as $\{\tilde{\boldsymbol{x}}^{(f)}_{ep}\}$, where $0 \le \rho < N^{(f)}_e(\wp)$. Let the set of corresponding normalized outward-pointing normal vectors be given by $\{\hat{\tilde{\boldsymbol{n}}}^{(f)}_{ep}\}$. It is critical that each flux point pair along an interface share the same coordinates in physical space. For a pair of flux points $ep n$ and $e'\rho' n'$ at a nonperiodic interface, this can be formalized as $\boldsymbol{\mathcal{M}}_{en}(\tilde{\boldsymbol{x}}^{(f)}_{ep}) = \boldsymbol{\mathcal{M}}_{e'n'}(\tilde{\boldsymbol{x}}^{(f)}_{e'\rho'})$.

The first step in the FR approach is to go from the discontinuous solution at the solution points to the discontinuous solution at the flux points

$$u^{(f)}_{e\sigma n\alpha} = u^{(u)}_{epn\alpha} \ell^{(u)}_{ep}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma}) \tag{13}$$

where $u^{(u)}_{epn\alpha}$ is an approximate solution of field variable α inside of the nth element of type e at solution point $\tilde{\boldsymbol{x}}^{(u)}_{ep}$. This can then be used to compute a common solution

$$\mathfrak{C}_\alpha u^{(f)}_{epn\alpha} = \mathfrak{C}_\alpha \underbrace{u^{(f)}_{e\sigma n\alpha}} = \left(\frac{1}{2} - \beta\right) u^{(f)}_{epn\alpha} + \left(\frac{1}{2} + \beta\right) \underbrace{u^{(f)}_{e\sigma n\alpha}} \tag{14}$$

where β controls the degree of up-/downwinding as part of a local discontinuous Galerkin (LDG) approach [21]. Here, we have taken $\widetilde{ep n}$ to be the element type, the flux point number, and the element number of the adjoining point at the interface. Because grids are unstructured, the relationship between $ep n$ and $\widetilde{ep n}$ is indirect. This necessitates the use of a lookup table.

Furthermore, associated with each flux point is a vector correction function $\boldsymbol{g}^{(f)}_{ep}(\tilde{\boldsymbol{x}})$ constrained such that

$$\hat{\tilde{\boldsymbol{n}}}^{(f)}_{e\sigma} \cdot \boldsymbol{g}^{(f)}_{ep}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma}) = \delta_{\rho\sigma} \tag{15}$$

with a divergence that sits in the same polynomial space as the solution. Using these fields, we can express the solution to Eq. (12b) as

$$\tilde{\boldsymbol{q}}^{(u)}_{e\sigma n\alpha} = [\hat{\tilde{\boldsymbol{n}}}^{(f)}_{ep} \cdot \tilde{\nabla} \cdot \boldsymbol{g}^{(f)}_{ep}(\tilde{\boldsymbol{x}}) \{\mathfrak{C}_\alpha u^{(f)}_{epn\alpha} - u^{(f)}_{epn\alpha}\} + u^{(u)}_{evn\alpha} \tilde{\nabla} \ell^{(u)}_{ev}(\tilde{\boldsymbol{x}})]_{\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{x}}^{(u)}_{e\sigma}} \tag{16}$$

where the term inside the curly brackets is the "jump" at the interface, and the final term is an order $\wp - 1$ approximation of the gradient obtained by differentiating the discontinuous solution polynomial. We can now compute physical gradients as

$$\boldsymbol{q}^{(u)}_{e\sigma n\alpha} = \boldsymbol{J}^{-T(u)}_{e\sigma n} \tilde{\boldsymbol{q}}^{(u)}_{e\sigma n\alpha} \tag{17}$$

$$\boldsymbol{q}^{(f)}_{e\sigma n\alpha} = \ell^{(u)}_{ep}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma}) \boldsymbol{q}^{(u)}_{epn\alpha} \tag{18}$$

where $\boldsymbol{J}^{-T(u)}_{e\sigma n} = \boldsymbol{J}^{-T}_{en}(\tilde{\boldsymbol{x}}^{(u)}_{e\sigma})$. Having solved the auxiliary equation, we are now able to evaluate the transformed flux

$$\tilde{\boldsymbol{f}}^{(u)}_{epn\alpha} = J^{(u)}_{epn} \boldsymbol{J}^{-1(u)}_{epn} \boldsymbol{f}_\alpha(u^{(u)}_{epn}, \boldsymbol{q}^{(u)}_{epn}) \tag{19}$$

where $J^{(u)}_{epn} = \det \boldsymbol{J}_{en}(\tilde{\boldsymbol{x}}^{(u)}_{ep})$. This can be seen to be a collocation projection of the flux. With this, it is possible to compute the normal transformed flux at each of the flux points

$$\tilde{f}^{(f_\perp)}_{e\sigma n\alpha} = \ell^{(u)}_{ep}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma}) \hat{\tilde{\boldsymbol{n}}}^{(f)}_{e\sigma} \cdot \tilde{\boldsymbol{f}}^{(u)}_{epn\alpha} \tag{20}$$

Considering the physical normals at the flux points, we see that $\boldsymbol{n}^{(f)}_{e\sigma n} = \boldsymbol{J}^{-T(f)}_{e\sigma n} \hat{\tilde{\boldsymbol{n}}}^{(f)}_{e\sigma}$ where $\boldsymbol{J}^{-T(f)}_{e\sigma n} = \boldsymbol{J}^{-T}_{en}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma})$, which is the outward-facing normal vector in physical space. The normal vector can also be expressed as $n^{(f)}_{e\sigma n} \hat{\boldsymbol{n}}^{(f)}_{e\sigma n}$, where $n^{(f)}_{e\sigma n}$ is the magnitude. As the interfaces between two elements conform, we must have $\hat{\boldsymbol{n}}^{(f)}_{e\sigma n} = -\underbrace{\hat{\boldsymbol{n}}^{(f)}_{e\sigma n}}$. With these definitions, we are now in a position to specify an expression for the common normal flux at a flux point pair as

$$\mathfrak{F}_\alpha f^{(f_\perp)}_{e\sigma n\alpha} = -\mathfrak{F}_\alpha \underbrace{f^{(f_\perp)}_{e\sigma n\alpha}} = \mathfrak{F}^{(\text{inv})}_\alpha - \mathfrak{F}^{(\text{vis})}_\alpha \tag{21}$$

where $\mathfrak{F}^{(\text{inv})}_\alpha$ is obtained by performing a Riemann solve on the inviscid portion of the flux. Further details on approximate Riemann solvers can be found in the work of Toro [22]. The viscous portion of the common normal flux is specified as

$$\mathfrak{F}^{(\text{vis})}_\alpha = \hat{\boldsymbol{n}}^{(f)}_{e\sigma n} \cdot \left\{\left(\frac{1}{2} + \beta\right) \boldsymbol{f}^{(\text{vis})}_{e\sigma n\alpha} + \left(\frac{1}{2} - \beta\right) \boldsymbol{f}^{(\text{vis})}_{\underbrace{e\sigma n\alpha}}\right\} + \tau(u^{(f)}_{e\sigma n\alpha} - u^{(f)}_{\underbrace{e\sigma n\alpha}}) \tag{22}$$

with τ being a penalty parameter. The common normal fluxes in Eq. (21) can now be taken into transformed space via

$$\mathfrak{F}_\alpha \tilde{f}^{(f_\perp)}_{e\sigma n\alpha} = J^{(f)}_{e\sigma n} n^{(f)}_{e\sigma n} \mathfrak{F}_\alpha f^{(f_\perp)}_{e\sigma n\alpha} \tag{23}$$

$$\mathfrak{F}_\alpha \tilde{f}^{(f_\perp)}_{\underbrace{e\sigma n\alpha}} = J^{(f)}_{\underbrace{e\sigma n}} n^{(f)}_{\underbrace{e\sigma n}} \mathfrak{F}_\alpha f^{(f_\perp)}_{\underbrace{e\sigma n\alpha}} \tag{24}$$

where $J^{(f)}_{e\sigma n} = \det \boldsymbol{J}_{en}(\tilde{\boldsymbol{x}}^{(f)}_{e\sigma})$.

It is now possible to compute an approximation for the divergence of the continuous flux. The procedure is directly analogous to the one used to calculate the transformed gradient in Eq. (16):

$$(\tilde{\nabla}\cdot\tilde{\boldsymbol{f}})^{(u)}_{e\rho n\alpha} = [\tilde{\nabla}\cdot\boldsymbol{g}^{(f)}_{e\sigma}(\tilde{\boldsymbol{x}})\{\tilde{\mathfrak{F}}_{\alpha}\tilde{f}^{(f_\perp)}_{e\sigma n\alpha} - \tilde{f}^{(f_\perp)}_{e\sigma n\alpha}\} + \tilde{\boldsymbol{f}}^{(u)}_{ev n\alpha}\cdot\tilde{\nabla}\ell^{(u)}_{ev}(\tilde{\boldsymbol{x}})]_{\tilde{\boldsymbol{x}}=\tilde{\boldsymbol{x}}^{(u)}_{e\rho}} \tag{25}$$

which can then be used to obtain a semidiscretized form of the governing system

$$\frac{\partial u^{(u)}_{e\rho n\alpha}}{\partial t} = -J^{-1(u)}_{e\rho n}(\tilde{\nabla}\cdot\tilde{\boldsymbol{f}})^{(u)}_{e\rho n\alpha} \tag{26}$$

where $J^{-1(u)}_{e\rho n} = \det \boldsymbol{J}^{-1}_{en}(\tilde{\boldsymbol{x}}^{(u)}_{e\rho}) = 1/J^{(u)}_{e\rho n}$.

This semidiscretized form is simply a system of ordinary differential equations in t and can be marched forward in time through one of a variety of schemes.

As presented, the aforementioned scheme employs a simple collocation projection to map the discontinuous flux into the polynomial space of the solution. For the Euler and Navier–Stokes equations, this flux is nonlinear, and hence this procedure will result in an aliasing. So, long as the simulation is sufficiently well resolved, this is not typically an issue. However, when simulating high-Reynolds-number flows on coarse grids, it is possible for aliasing-driven instabilities to develop and destabilize the simulation. One of the more reliable means for mitigating this is to perform antialiasing via an approximate L^2 projection, as described in Refs. [5,23,24]. This approach attempts to approximate the L^2 optimal projection from the function space of the flux into the polynomial space of the solution. Performing this projection requires the numerical evaluation of integrals inside both the volume of an element and on the various faces. These integrals are typically approximated using high-strength multidimensional quadrature rules (such as those described in Refs. [25–27]), with the effectiveness of the approach being contingent on the accuracy of the quadrature approximations. This can necessitate the use of a large number of quadrature points relative to the number of solution/flux points, and thus greatly increase the cost of a time step.

IV. PyFR

PyFR [8,28] is a Python framework for solving the compressible Navier–Stokes equations on mixed unstructured grids using the FR approach. The key functionality of PyFR v1.5.0 is summarized in Table 2. Kernels within PyFR are specified in a hardware agnostic domain-specific language based off of the Mako templating engine. At runtime, so-called backendswithin PyFR translate these kernels into the native language of the current platform. These kernels are then compiled, linked, and dynamically loaded. This architecture ensures that all computationally intensive operations are performed using native code, with the Python layer simply being responsible for invoking these kernels and marshalling data transfers.

The design of PyFR makes it particularly well suited to the task of evaluating the impact of number representation on implicit large-eddy simulations. First, PyFR has been designed from the ground up to support both single and double precision modes of operation. Second, PyFR is portable across a range of hardware platforms, hence allowing the performance of single precision operation to be compared and contrasted across multiple pieces of hardware. Third, the data structures employed by PyFR encourage vectorization and give rise to coalesced memory access patterns. This is accomplished through use of the structure-of-arrays layout, which places field variables of the same type adjacent to each other in memory. Finally, PyFR is capable of operating across a range of performance regimes that span the spectrum from bandwidth to compute bound. Specifically, PyFR casts the majority of operations within an FR step as matrix–matrix multiplications of the form

Table 2 Key functionality of PyFR v1.5.0

Parameter	Value
Dimensions	Two-dimensional, three-dimensional
Elements	Triangles, quadrilaterals, hexahedra, tetrahedra, prisms, pyramids
Spatial orders	Arbitrary
Time steppers	Step-size controlled Runge–Kutta
Precisions	Single, double
Backends	C/OpenMP, CUDA, OpenCL, pyMIC
Communication	MPI
File format	Parallel HDF5
Governing systems	Euler, compressible Navier–Stokes

$$\boldsymbol{C}\leftarrow\boldsymbol{A}\boldsymbol{B} + \beta\boldsymbol{C} \tag{27}$$

where $\boldsymbol{A}$ is a constant operator matrix, $\boldsymbol{B}$ and $\boldsymbol{C}$ are state matrices, and $\beta = 0$ or $\beta = 1$ depending on the context of the operation. These operations (especially in three dimensions) are generally compute bound. However, when tensor product elements (quadrilaterals and hexahedra) are employed with a compatible combination of solution and flux points, the operator matrix $\boldsymbol{A}$ can exhibit sparsity.

Unfortunately, the degree of sparsity is not sufficient for standard sparse basic linear algebra subprograms (BLAS) packages such as NVIDIA cuSPARSE and Intel MKL. PyFR addresses this issue through the use of the codeveloped GiMMiK code generation package of Wozniak et al. [29]. Given a constant operator matrix $\boldsymbol{A}$, GiMMiK will (at runtime) generate a bespoke kernel for performing this specific matrix multiplication. As GiMMiK embeds the elements of $\boldsymbol{A}$ directly into the source of the kernel, it is able to trivially eliminate any zero multiplications. The effect of this is that, for these simulations, PyFR transitions into being bound by memory bandwidth. Although GiMMiK is primarily geared toward generating sparse matrix multiplication routines, it has also shown promise for operator matrices that are small and dense [29].

A detailed analysis of the performance of PyFR on NVIDIA GPUs can be found in Refs. [8,9]. The impact of the GiMMiK matrix multiplication library, including sustained FLOP and bandwidth numbers, can be found in Ref. [29]. Strong and weak scaling studies for a variety of test cases can be found in Refs. [8,28,30].

When incorporating support for multiple precisions into PyFR, great care was taken to ensure that all calculations are performed at the requested precision and that all constant data, including the initial condition and the geometric terms, are stored at this precision. When running with single precision, the domain-specific language also suffixes all floating-point constants by `.f` so as to prevent unwanted promotions to double precision.

V. Numerical Experiments

All simulations herein were performed using a prerelease version of PyFR v1.5.0 with GiMMiK v2.0.0.

A. Taylor–Green Vortex

1. Background

Simulation of the Taylor–Green vortex breakdown using the compressible Navier–Stokes equations has been undertaken for the comparison of high-order numerical schemes. The case has been a constituent of the first three high-order workshops and is specified inside of a fully periodic box $\boldsymbol{\Omega} = [-\pi L, \pi L]^3$, where L is the characteristic length, at a Reynolds number of $Re = 1600$ with an effectively incompressible Mach number of $M = 0.1$ [31]. To assess the impact of number representation on this problem, four sets of simulations using structured hexahedral grids were undertaken. Details regarding these simulations can be found in Table 3. The initial conditions are given as

$$\rho(\boldsymbol{x}, t=0) = \frac{p}{RT_0} \tag{28}$$

$$p(\boldsymbol{x}, t=0) = p_0 + \frac{\rho_0 v_0^2}{16}\left(\cos\frac{2x}{L} + \cos\frac{2y}{L}\right)\left(\cos\frac{2z}{L} + 2\right) \tag{29}$$

$$\boldsymbol{v}(\boldsymbol{x}, t=0) = (\hat{\boldsymbol{x}} - \hat{\boldsymbol{y}})v_0 \sin\frac{x}{L}\cos\frac{y}{L}\cos\frac{z}{L} \tag{30}$$

Table 3 Solver and mesh configurations for the Taylor–Green vortex experiment[a]

			Memory, GB	
Order	N_E	$\sum N_u$	Single	Double
$\wp = 2$	86^3	258^3	6.4	12.2
$\wp = 3$	64^3	256^3	5.4	10.3
$\wp = 4$	52^3	260^3	5.1	9.8
$\wp = 5$	43^3	258^3	4.6	9.0

[a]The total number of elements and solution points in the simulations are indicated by N_E and $\sum N_u$, respectively.

where ρ_0 is the unperturbed density, p_0 is the unperturbed pressure, RT_0 is the product of the specific gas constant with a constant initial temperature, $v_0 = M\sqrt{\gamma RT_0}$ is the unperturbed velocity, and $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are unit vectors in the x and y directions, respectively. In the proposed numerical experiment, we let $L = 1, \rho_0 = 1$, and $RT_0 = 1$; from the ideal gas law, we take $\gamma = 1.4$. Hence, $v_0 \simeq 0.118$, with the dynamic viscosity being given by

$$\mu = \frac{\rho_0 v_0 L}{Re} \simeq 7.40 \times 10^{-5} \tag{31}$$

The case is run until $t = 20t_c$, where time is normalized according to $t_c = L/v_0 \simeq 8.45$. Although the Taylor–Green vortex does not have an analytical solution, there is spectral DNS data available due to van Rees et al. [32]. Two salient metrics are the temporal evolution of the domain-integrated kinetic energy and enstrophy, which are defined as

$$\hat{E}_k(t) = \frac{1}{|\boldsymbol{\Omega}|}\frac{1}{\rho_0 v_0^2}\int_{\boldsymbol{\Omega}} \frac{1}{2}\rho(\boldsymbol{x},t)\|\boldsymbol{v}(\boldsymbol{x},t)\|^2 \,\mathrm{d}^3\boldsymbol{x} \tag{32}$$

$$\hat{\mathcal{E}}(t) = \frac{1}{|\boldsymbol{\Omega}|}\frac{1}{\rho_0 v_0^2}\int_{\boldsymbol{\Omega}} \frac{1}{2}\rho(\boldsymbol{x},t)\|\boldsymbol{\omega}(\boldsymbol{x},t)\|^2 \,\mathrm{d}^3\boldsymbol{x} \tag{33}$$

where $|\boldsymbol{\Omega}| = (2\pi)^3$ is the volume of the domain, and $\boldsymbol{\omega} = \nabla \times \boldsymbol{v}$ is the vorticity. We note here that, as the flow is effectively incompressible (hence, $\nabla \cdot \boldsymbol{v} = 0$), the enstrophy should be related to the decay rate of the kinetic energy via

$$\hat{\mathcal{E}} = -2\frac{\mu}{\rho_0}\partial_t \hat{E}_k \tag{34}$$

Denoting the spectral DNS quantities with the superscript $*$, we can introduce the following two error metrics as

$$\sigma = \max_t |\partial_t \hat{E}_k - (\partial_t \hat{E}_k)^\star| / \min_t (\partial_t \hat{E}_k)^\star \tag{35}$$

$$\varsigma = \max_t |\hat{\mathcal{E}} - \hat{\mathcal{E}}^\star| / \max_t \hat{\mathcal{E}}^\star \tag{36}$$

Because both $\hat{E}_k$ and $\hat{\mathcal{E}}$ are integral quantities over the entire domain, they can be readily evaluated using Gaussian quadrature. The time derivative of the kinetic energy $\partial_t \hat{E}_k$ can be approximated by fitting a cubic spline through $\hat{E}_k$ and then evaluating its derivative.

For all of the simulations, a DG scheme was used for the spatial discretization, a Rusanov Riemann solver [33] was used to calculate the inviscid fluxes at element interfaces, and the explicit low-storage RK45[2R+] Runge–Kutta scheme of Kennedy et al. [34] was used to advance the solution in time. The local temporal error was managed by using a proportional-integral (PI) type step-size controller with absolute and relative error tolerances set to 10^{-5}. Solution points and flux points were placed at a tensor product construction of Gauss–Legendre quadrature points. The upwind and penalty parameters for the LDG scheme were set to $\beta = 1/2$ and $\tau = 1/10$, respectively. The Prandtl number was taken to be $Pr = 0.72$.

2. *Results*

Volume renderings of the vorticity within the domain can be seen in Fig. 2. Plots of the $-\partial_t \hat{E}_k$ and $\hat{\mathcal{E}}$ as a function of time can be seen in Figs. 3 and 4, respectively. From the figures, it is clear that there is no perceptible difference between the single and double precision simulations for either quantity at any of the four orders of accuracy. Figure 5 shows the maximum percentage errors in the kinetic energy decay rate σ and the enstrophy ς when compared with the spectral DNS data of van Rees et al. [32]. These errors are, again, virtually identical for the single and double precision calculations. Hence, these results suggest that, for the direct simulation of transitional flows, single precision is sufficient.

3. *Performance*

The performance of the simulations was assessed using a pair of NVIDIA K40c GPUs running CUDA 7.5. Simulations were run with PyFR for 200RK45[2R+] time steps with a small fixed Δt. For this particular test case, we note that the PyFR operator matrices exhibit a substantial degree of sparsity. Therefore, on NVIDIA GPUs, it is possible to run PyFR with either dense cuBLAS or GiMMiK. In the former case, we expect PyFR to be compute bound; whereas in the latter case, PyFR is likely to be limited by available memory bandwidth. The results for both of these configurations can be seen in Tables 4 and 5, respectively.

Looking at the tables. the first thing we note is the dramatic decrease in the total number of floating-point operations required per RK45[2R+] stage when GiMMiK is employed. Indeed, with GiMMiK, the number of operations is almost constant with respect to the polynomial order. This implies that, on a degree-of-freedom basis, high-order methods are no more expensive than their low-order counterparts. Second, we observe that the GiMMiK simulations are substantially faster than those using dense BLAS. However, we also note that the performance for the cuBLAS simulations never sustains above ~50% of peak FLOP/s. Switching to single precision with cuBLAS gives rise to speedups between 1.41 and 1.83 that, given the FLOP bound nature of the problem, is less than one might expect given the threefold increase in peak FLOP/s. This suggests that the SGEMM routines in cuBLAS are not well tuned for the types of matrix multiplications encountered in FR.

When GiMMiK is enabled, the simulations go from being compute bound to bandwidth bound, as can be seen from the low sustained GFLOP/s numbers. For $\wp = 2, 3, 4$, the speedup is roughly in line with what one would expect for a bandwidth limited code. At $\wp = 5$, the performance of the double precision simulation begins to regress, whereas the single precision simulation does not, hence the

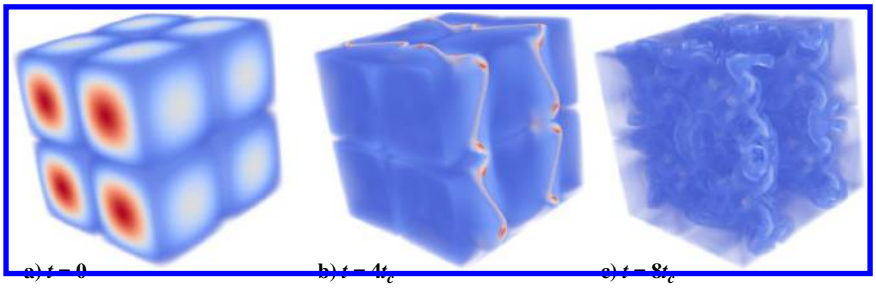

Fig. 2 Volume renders of vorticity for the double precision $\wp = 5$ Taylor–Green simulation.

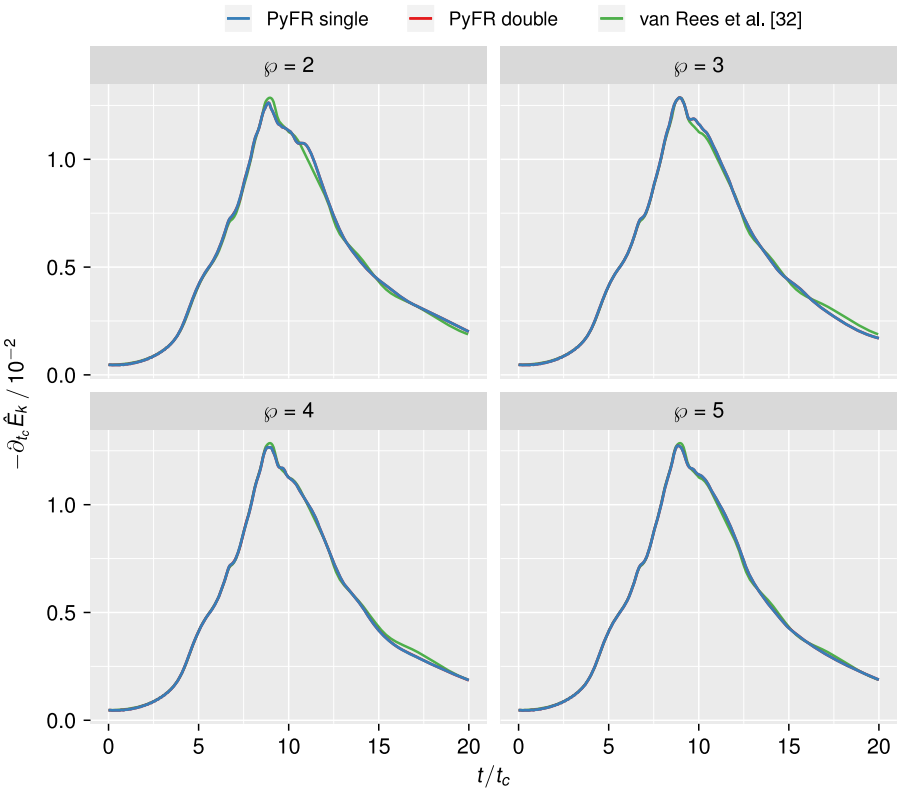

Fig. 3 Rate of kinetic energy decay for the Taylor–Green test simulation compared against the spectral DNS data of van Rees et al. [32].

larger than expected speedup. As the order is increased, the number of nonzeros in the operator matrices increases. It is known [29] that GiMMiK performance is heavily impacted by architectural register limits. Dropping down to single precision thus results in a substantial reduction in the number of required registers, and hence avoids unwanted spillage to global memory.

B. Cylinder at $Re = 3900$

1. *Background*

Flow over a circular cylinder has been the focus of various experimental and numerical studies [9,35–39]. Characteristics of the flow are known to be highly dependent on the Reynolds number Re, which is defined as

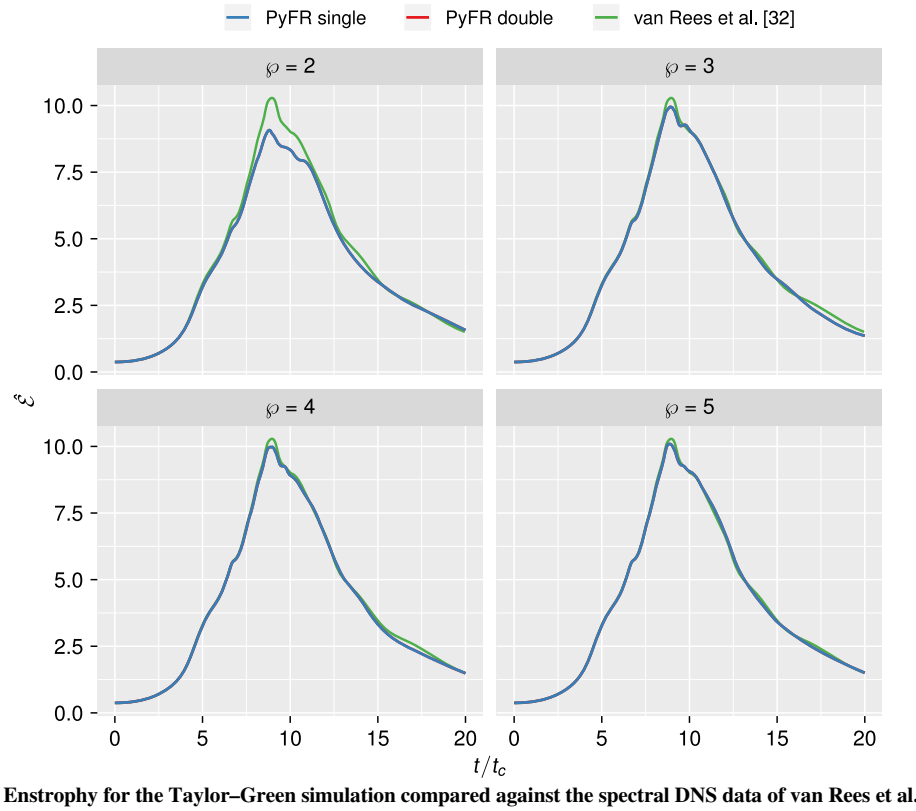

Fig. 4 Enstrophy for the Taylor–Green simulation compared against the spectral DNS data of van Rees et al. [32].

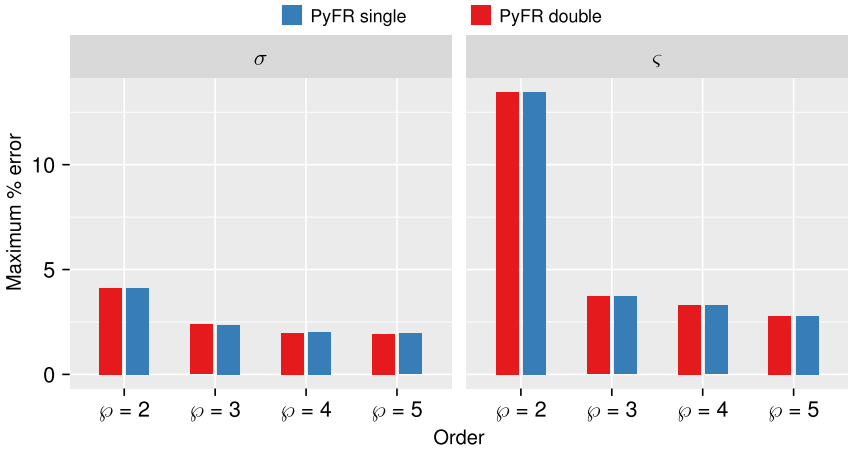

Fig. 5 Relative error in the kinetic energy decay rate and the enstrophy compared against the spectral DNS data of van Rees et al. [32].

$$Re = \frac{\rho_\infty v_\infty D}{\mu} \tag{37}$$

where ρ_∞ is the freestream density, v_∞ is the freestream fluid speed, D is the cylinder diameter, and μ is the dynamic viscosity. In the present study, flow over a circular cylinder at $Re = 3900$ with an effectively incompressible Mach number of $M = 0.2$ is considered. This case sits in the shear-layer transition regime identified by Williamson [40], and it contains several complex flow features, including separated shear layers, turbulent transition, and a fully turbulent wake. Recent studies by Lehmkuhl et al. [41] and Witherden et al. [9] have shown that, at this Reynolds number, the flow field oscillates at a low frequency between two shedding modes: a high-energy mode, and a low-energy mode. To decisively observe both modes, it is necessary to average over 1000 or so convective times.

In this section, a computational domain with dimensions of $[-9D, 25D]$, $[-9D, 9D]$, and $[0, \pi D]$ in the streamwise, crosswise, and spanwise directions, respectively, is used. The cylinder is centered at $(0, 0, 0)$. The spanwise extent was chosen based on the results of Norberg [35], who found no significant influence on statistical data when the spanwise dimension was doubled from πD to $2\pi D$. Indeed, a span of πD has been used in the majority of previous numerical studies [36–38], including the recent DNS study of Lehmkuhl et al. [41]. The streamwise and crosswise dimensions are comparable to the experimental and numerical values used by Parnaudeau et al. [39]. The overall domain dimensions are also comparable to those used for DNS studies by Lehmkuhl et al. [41]. The domain is periodic in the spanwise direction, with a no-slip isothermal wall boundary condition applied at the surface of the cylinder and a Riemann-invariant boundary condition, as detailed in Ref. [42], applied at the farfield in order to enforce the freestream.

To nondimensionalize the system, we take the diameter of the cylinder to be $D = 1, \rho_\infty = 1$, and $p_\infty = 1$. Hence, $v_\infty = M\sqrt{\gamma} \simeq 0.234$, with time being normalized according to $t_c = D/v_\infty \simeq 4.32$. The dynamic viscosity is given by

$$\mu = \frac{\rho_\infty v_\infty D}{Re} \simeq 6.07 \times 10^{-5} \tag{38}$$

Following Witherden et al. [9], the domain was meshed using prismatic elements in the near-wall boundary-layer region and tetrahedral elements in the wake and farfield. The elements on the boundary were quadratically curved and designed to fully resolve the near-wall boundary-layer region at $\wp = 4$. Specifically, the maximum skin-friction coefficient was estimated a priorias $C_f \approx 0.075$ based on the LES results of Breuer [37]. The height of the first element was then specified such that, when $\wp = 4$, the first solution point from the wall sits at $y+ \approx 1$, where nondimensional wall units are calculated in the usual fashion as $y+ = u_\tau y/\nu$ with $u_\tau = \sqrt{C_f/2}v_\infty$. The resulting mesh has 116 elements in the circumferential direction and 20 elements in the spanwise direction, with these numbers being chosen to help reduce face aspect ratios at the edges of the prismatic layer, which facilitates transition to the fully unstructured tetrahedral elements in the farfield. In total, the mesh contained 79,344 prismatic elements and 227,298 tetrahedral elements. A cutaway of the mesh can be seen in Fig. 6.

For the simulations, a DG scheme was used for the spatial discretization, a Rusanov Riemann solver was used to calculate the inviscid fluxes at element interfaces, and the explicit low-storage RK45[2R+] Runge Kutta scheme of Kennedy et al. [34] was used to advance the solution in time. A local temporal error was managed by using a PI-type step-size controller with absolute and relative error tolerances set to 10^{-5}. Inside of the tetrahedral elements (and following the guidance of Witherden et al. [43]), the quadrature points of Shunn and Ham [26] were used as the solution points. Based on the analysis of Witherden and Vincent [44], the flux points on the triangular faces were chosen to be the quadrature points of Williams et al. [27]. Flux points on the quadrilateral faces were taken to be a tensor product construction of the Gauss–Legendre quadrature points. Inside of the prismatic elements, a tensor product of the triangular flux points with a line of Gauss–Legendre points was employed as the solution points. The upwind and penalty parameters for the LDG scheme were set to $\beta = 1/2$ and $\tau = 1/10$, respectively. The Prandtl number was taken to be $Pr = 0.72$.

2. *Results*

Both simulations were started at $\wp = 1$ and run to $t = 80$. They were then restarted with $\wp = 2$ and run until $t = 400$; at which point, the order was increased to $\wp = 3$ and the simulations advanced until

Table 4 Performance for the Taylor–Green vortex with PyFR on two NVIDIA K40c GPUs using dense cuBLAS[a]

		$t_w/\sum N_u/10^{-9}$ s		GFLOP/s		
Order	GFLOP, stage	Single	Double	Single	Double	Speedup
$\wp = 2$	1.12×10^2	8.7	12.2	748.4	531.0	1.41
$\wp = 3$	2.01×10^2	7.7	11.6	1525.9	1012.2	1.51
$\wp = 4$	3.51×10^2	9.3	15.7	2191.0	1298.4	1.69
$\wp = 5$	5.27×10^2	11.1	20.3	2756.6	1508.6	1.83

[a]The wall-clock time per solution point per RK stage is denoted by $t_w/\sum N_u$.

Table 5 Performance for the Taylor–Green vortex with PyFR on two NVIDIA K40c GPUs using GiMMiK[a]

		$t_w/\sum N_u/10^{-9}$ s		GFLOP/s		
Order	GFLOP, stage	Single	Double	Single	Double	Speedup
$\wp = 2$	1.84×10^1	4.8	8.9	222.1	120.5	1.84
$\wp = 3$	1.82×10^1	4.2	7.9	252.3	134.6	1.88
$\wp = 4$	1.92×10^1	4.4	8.6	255.9	129.7	1.97
$\wp = 5$	1.96×10^1	4.5	13.1	250.8	87.0	2.88

[a]The wall-clock time per solution point per RK stage is denoted by $t_w/\sum N_u$.

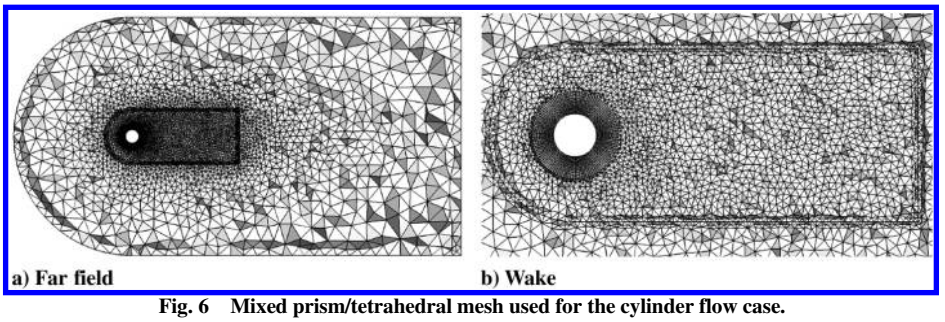

Fig. 6 Mixed prism/tetrahedral mesh used for the cylinder flow case.

$t = 520$. Finally, the order was increased to $\wp = 4$ and the simulations run until $t = 1480$. Time averages were collected starting at $t = 680$ until $t = 1480$, with a window size of $\Delta t_{\text{avg}} = 800$, with averages being accumulated every 30 time steps. This window covers approximately $185t_c$. Given this period, we do not expect to be able to clearly pick out the two shedding modes with either the double or single precision simulations. However, it has been demonstrated [8] that PyFR running with double precision in this configuration is capable of obtaining excellent agreement with the DNS results of Lehmkuhl et al. [41] and the experimental results of Parnaudeau et al. [39].

Isosurfaces of density colored by velocity magnitude can be seen in Fig. 7. Figure 8 shows the time- and span-averaged pressure coefficient $C_p = (p - p_\infty)/(1/2)\rho_\infty v_\infty^2$ on the surface of the cylinder. The streamwise and cross-stream velocity profiles at three points of $x/D = 1.06$, $x/D = 1.54$, and $x/D = 2.02$ can be seen in Figs. 9 and 10, respectively. The two lines associated with the numerical results of Lehmkuhl et al. [41] correspond to the two distinct shedding modes associated with the problem.

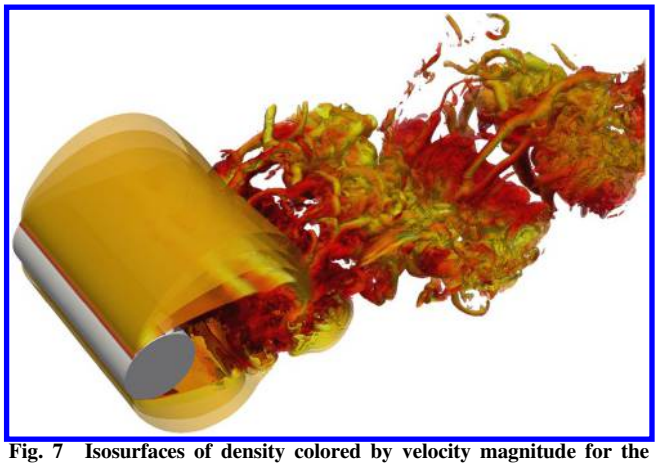

Fig. 7 Isosurfaces of density colored by velocity magnitude for the double precision circular cylinder test case at $\wp = 4$.

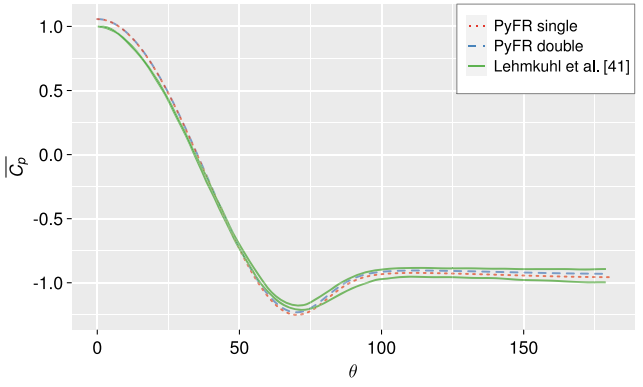

Fig. 8 Time- and span-averaged pressure coefficient on the surface of the cylinder compared with the numerical results of Lehmkuhl et al. [41].

From the figures, we see that both the single and double precision simulations appear to be at an intermediate state between the two shedding modes. Differences in the time averages between the single and double precision simulations are not entirely unexpected. As the flow develops, instabilities build up along the z axis that result in the simulations transitioning to fully turbulent flow. Because the scale of these instabilities is connected to the chosen number precision, the time at which the flow transitions to turbulence is different for the single and double precision simulations. In turn, this can result in the initial shedding mode being different for the two simulations. This combined with the averaging window of $\sim 185t_c$ can thus give rise to differences in the time-averaged statistics.

3. *Performance*

The performance of the simulation was assessed using a single NVIDIA K40c GPU using CUDA 7.5. Simulations were run with PyFR at $\wp = 4$ for 100 RK45[2R+] time steps with a small fixed Δt. GiMMiK was employed for all simulations. In the case of the prismatic elements, which have a partial tensor product structure, GiMMiK is able successfully exploit the sparsity in the resulting operator matrices. For the tetrahedral elements, for which the operator matrices are predominantly dense but of awkward dimensions, GiMMiK is able to generate specialized kernels tailored to these specific dimensions.

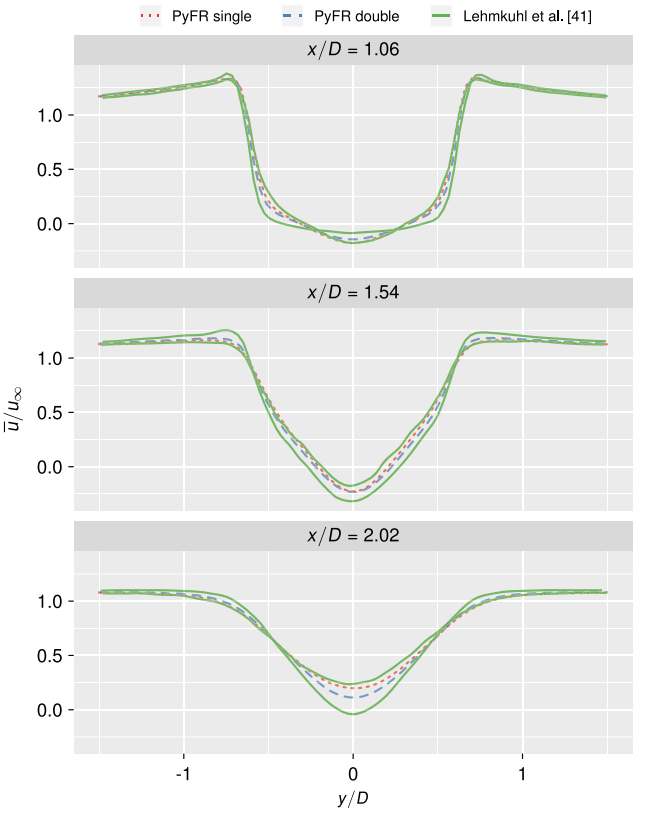

Fig. 9 Time- and span-averaged streamwise velocity profiles for the cylinder compared with the numerical results of Lehmkuhl et al. [41].

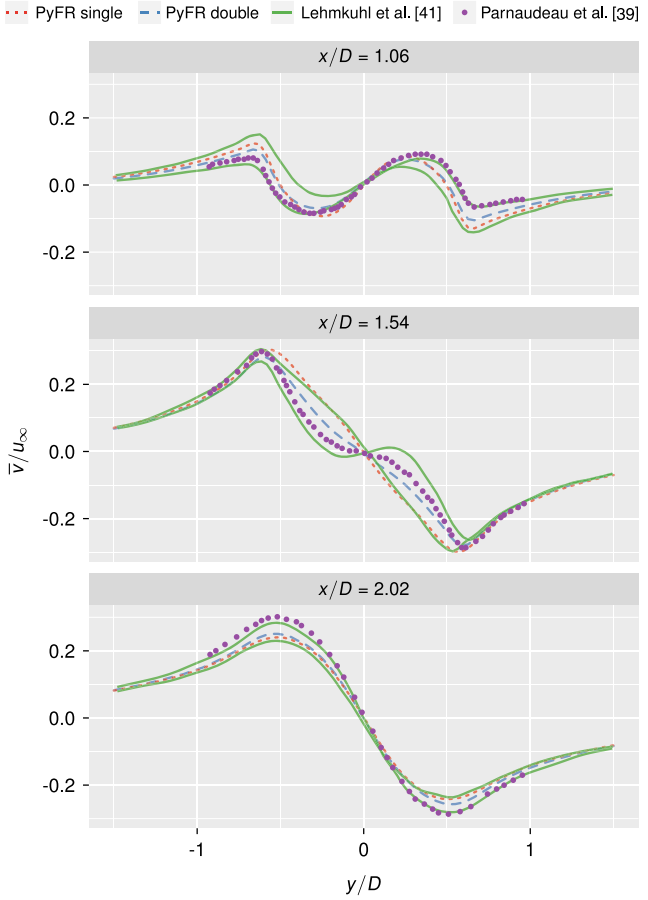

Fig. 10 Time- and span-averaged cross-stream velocity profiles for the cylinder compared with the numerical results of Lehmkuhl et al. [41] and the experimental results of Parnaudeau et al. [39].

In the double precision case, the average wall-time per-solution point per Runge–Kutta (RK) stage is 28.6 ns. Switching to single precision reduces this time down to 17.7 ns. This corresponds to a speedup of 1.62. When comparing these numbers to those of the Taylor–Green vortex (Tables 4 and 5), it is important to remember that the Taylor–Green simulations were performed on a pair of K40c GPUs. After accounting for this factor, we see performance is slightly improved relative to the $\wp = 4$ Taylor–Green simulations with cuBLAS but is slower than those with GiMMiK.

The overall speedup factor of 1.62 is also somewhat less than the factor of 1.97 that we observed for the GiMMiK-enabled Taylor–Green simulations. Compared with hexahedra, both prisms and tetrahedra have a larger ratio of flux points to solution points. This increase translates directly into in additional memory indirection. As discussed in Sec. II, memory indirection can result in kernels becoming bound by memory latency as opposed to bandwidth. In this scenario, the potential for performance improvements is limited.

C. T106c Low-Pressure Turbine Cascade

1. *Background*

The T106 low-pressure turbine cascade [45] is a popular test case for evaluating CFD solvers. The T106c cascade is defined by imposing a pitch-to-cord ratio of $s/c = 0.95$ and is supported by a wide body of numerical and experimental data [46–49]. From the specification of the T106c cascade, the chord is $c = 0.09301$ m and the pitchwise inlet flow angle is 32.7 deg. In this study, we will consider the case at a Reynolds number of $Re = 80{,}000$, with an outlet Mach number of $M = 0.65$. These conditions are of interest because it will allow the impact of number representation to be quantified for a fully compressible simulation at a meaningful Reynolds number. The working fluid for the case is taken to be air with a total temperature at the inlet of $T_i = 298.15$ K. The ratio of specific heats is $\gamma = 1.4$ with the specific gas constant being given by $R \simeq 287.1\ \mathrm{J \cdot kg^{-1} \cdot K^{-1}}$.

Additional characteristics of the flow can be determined using the isentropic flow equations. The exit temperature can be determined as

$$T_e = \frac{T_i}{1 + (\gamma + 1/2)M^2} \simeq 274.92\ \mathrm{K} \tag{39}$$

with the exit velocity being given by

$$v_e = M\sqrt{\gamma R T_e} \simeq 216.07\ \mathrm{m \cdot s^{-1}} \tag{40}$$

Using Sutherland's law, the viscosity at the exit is given by

$$\mu_e = \mu_0 \left(\frac{T_e}{T_0}\right)^{3/2} \frac{T_0 + S}{T_e + S} \tag{41}$$

where $\mu_0 \simeq 1.716 \times 10^{-5}$ Pa s is the viscosity for air at the reference temperature, $T_0 \simeq 273.15$ K, and $S \simeq 110.4$ K is Sutherland's temperature. Hence, $\mu_e \simeq 1.7248 \times 10^{-5}$ Pa s, with the density at the outlet being given by

$$\rho_e = \frac{Re\mu_e}{v_e c} \simeq 0.0687\ \mathrm{kg/m^{-3}} \tag{42}$$

Finally, the pressure at the outlet and total pressure can be determined as

$$p_e = \rho_e R T_e \simeq 5419.3\ \mathrm{Pa} \tag{43}$$

$$p_i = p_e \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\gamma/\gamma - 1} \simeq 7198.5\ \mathrm{Pa} \tag{44}$$

which serve to fully specify the problem.

The linear T106c cascade used in the experiments of Michálek et al. [46] consisted of six blades: each with a span $h = 2.4c \simeq 0.22$ m. For the case under consideration, the inlet turbulence intensity was approximately 0.9%. To simplify the problem setup somewhat, we will simulate just a single blade without any inlet turbulence and a reduced span of $0.2c \simeq 0.0186$ m. The domain is periodic in the crosswise and spanwise directions. On the surface of the blade, an adiabatic wall condition is applied. A fixed total pressure condition is enforced on the inlet of the domain. This condition also enforces the desired pitchwise angle for the incoming flow. At the exit, a Riemann-invariant boundary condition is applied. We note here that, because the Riemann-invariant conditions are designed to minimize reflections at the boundary, they do not guarantee a strong enforcement of the exit pressure p_e. Hence, some tuning is required around the pressure enforced by the boundary condition in order to obtain the desired (averaged) exit pressure.

To mesh the domain, unstructured hexahedral elements were employed. The elements on the boundary of the blade were quadratically curved and designed to fully resolve the near-wall boundary-layer region at $\wp = 2$. Specifically, the height of the first element was specified such that, when $\wp = 2$, the first solution point from the wall sits at $y+ \approx 1$ with

$$y+ = \frac{u_\tau \rho_e y}{\mu_e} = \frac{\sqrt{C_f} v_e \rho_e y}{\sqrt{2}\mu_e} \approx \frac{(2\log_{10} Re - 0.65)^{-1.15} v_e \rho_e y}{\sqrt{2}\mu_e} \tag{45}$$

where, in the third step, we have used the Schlichting skin-friction formula to estimate C_f based off of the Reynolds number. The resulting mesh has a total of eight layers in the spanwise direction. In total, the mesh contains 59,936 elements. A cutaway of the mesh can be seen in Fig. 11.

For the simulations, a DG scheme was used for the spatial discretization, a Rusanov Riemann solver was used to calculate the inviscid fluxes at element interfaces, and the explicit low-storage RK45[2R+] Runge Kutta scheme of Kennedy et al. [34] was used to advance the

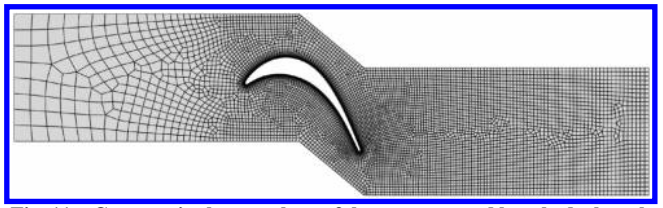

Fig. 11 Cutaway in the x–y plane of the unstructured hexahedral mesh used for the T106c test case.

solution in time. The local temporal error was managed by using a PI-type step-size controller with absolute and relative error tolerances set to 10^{-5}. Solution points and flux points were placed at a tensor product construction of Gauss–Legendre quadrature points. The upwind and penalty parameters for the LDG scheme were set to $\beta = 1/2$ and $\tau = 1/10$, respectively. The Prandtl number was taken to be $Pr = 0.71$.

2. *Results*

The simulations were run at $\wp = 1$ with a viscosity of $\mu = 3.4 \times 10^{-5}$ until $t = 2.5 \times 10^{-2}$. At this point, the simulations were restarted with the correct viscosity of 1.7248×10^{-5} at order $\wp = 2$ and advanced to 3.5×10^{-2}. Time averaging of the pressure field was enabled at $t = 3.25 \times 10^{-2}$, with the average being accumulated every 50 time steps. This corresponds to approximately two passes over the chord.

A plot of isosurfaces of Q-criteria colored by velocity magnitude for the fully developed flow are shown in Fig. 12. To compare with experimental data, we consider the isentropic Mach number, which is defined as

$$M^{\text{isentropic}} = \sqrt{\frac{2}{\gamma - 1}\left(\left(\frac{p_e}{p}\right)^{\Gamma} - 1\right)} \tag{46}$$

where $\Gamma = (\gamma - 1)/\gamma$. A plot of the chordwise distribution of this can be seen in Fig. 13. Looking at the figure, the first thing we note is that the single and double precision results are virtually identical. We also see that there is a reasonable degree of agreement between PyFR and the experimental data of Michálek et al. [46]. However, there are some discrepancies around peak suction at $x/c \sim 0.6$. We note, however, that such overprediction has also been observed by both the studies of Hillewaert et al. [48] and Garai et al. [49]. As our objective here is to assess the impact of number representation, we do not give any further consideration to this deviation.

Fig. 12 Isosurfaces of Q-criterion colored by velocity magnitude for the T106c test case at single precision with $\wp = 2$.

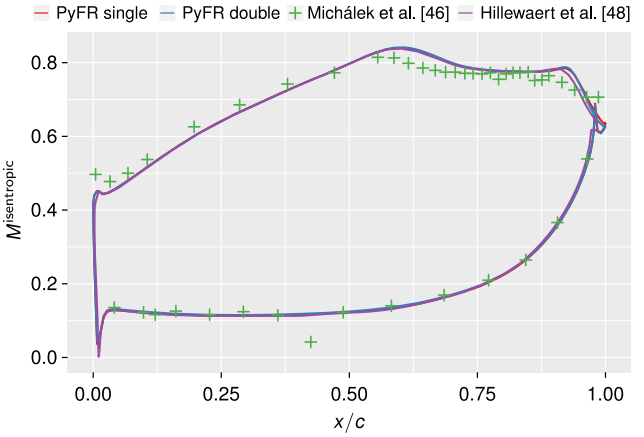

Fig. 13 Chordwise distribution of time- and span-averaged isentropic Mach number for T106c test case compared against experimental data of Michálek et al. [46] (extracted from Pacciani et al. [47]) and computational study of Hillewaert et al. [48].

3. *Performance*

The performance of PyFR in terms of the wall-clock time per stage does not depend on the specifics of the mesh. As such, the performance for the T106c test case at $\wp = 2$ is similar to the Taylor–Green vortex simulation at this order. Although there are some minor differences, due to the enforcement of boundary conditions, the impact on runtime is negligible. The reader is henceforth referred to the discussion in Sec. V.A.

D. NACA 0021 Airfoil in Deep Stall

1. *Background*

Flow around a NACA 0021 airfoil in deep stall at a Reynolds number of $Re = 270{,}000$ and Mach $M = 0.1$ is an extremely challenging test case for scale-resolving approaches. In the present study, the airfoil is set at a 60 deg angle of attack to the oncoming freestream flow. Substantial experimental [50] and computational data exist [51,52] for this configuration. The first ILES simulation of this case was performed by Park et al. [23] in 2017 using the PyFR solver.

For this test case, a computational domain with an extent of $30c$ in the streamwise and crosswise directions is used, where c corresponds to the chord of the airfoil. The span is taken as $4c$. Riemann-invariant boundary conditions were applied in the far field, a no-slip adiabatic wall boundary condition was applied on the airfoil surface, and a periodic condition was imposed in the spanwise direction.

To mesh the domain, unstructured hexahedral elements were employed. The elements on the boundary of the airfoil were quadratically curved and designed to fully resolve the boundary layer such that, at $\wp = 4$, the first solution point sits as $y+ \approx 1$. The resulting grid contained a total of 206,528 elements. A cutaway of the grid can be seen in Fig. 14.

Compared with previous simulations, the high Reynolds number and coarse grid necessitate the use of a stabilization strategy in order to damp and/or counter the aliasing-driven instabilities that would otherwise develop [23]. The simulation can therefore be regarded as being on the limit in terms of resolution, and thus is an extremely good candidate for highlighting any substantial differences between single and double precision. Furthermore, the presence of low-frequency dynamics requires that the system be simulated for a relatively large number of convective times.

Simulations were run using a DG scheme for the spatial discretization. A Rusanov Riemann solver was used to calculate the inviscid fluxes at element interfaces, and the explicit low-storage RK45[2R+] Runge–Kutta scheme of Kennedy et al. [34] was used to advance the solution in time. The local temporal error was managed by using a PI-type step-size controller with absolute and relative error tolerances set to 10^{-6}. Solution points and flux points were placed at a tensor product construction of Gauss–Legendre quadrature points. To stabilize the simulation, both the interior and interface fluxes were

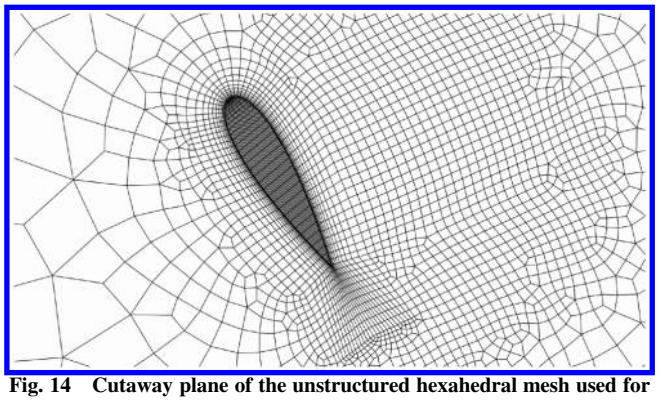

Fig. 14 Cutaway plane of the unstructured hexahedral mesh used for the NACA 0012 test case.

antialiased via an approximate L^2 projection using 11th-order Gauss–Legendre quadrature rules. The upwind and penalty parameters for the LDG scheme were set to $\beta = 1/2$ and $\tau = 1/10$, respectively. The Prandtl number was taken to be $Pr = 0.71$.

2. *Results*

The simulations were run at $\wp = 1$ without any antialiasing until $t = 100$. At this point, the simulations were restarted at $\wp = 4$ with antialiasing and run for a further 300 time units. This corresponds to approximately 300 passes over the chord. A comparison of the time-averaged lift and drag coefficients can be found in Table 6, and a plot showing the power spectrum density (PSD) of the lift coefficient C_L against the Strouhal number St can be seen in Fig. 15.

Looking at the table, we observe that the single precision simulation predicts lift and drag values that are ∼2.4% and ∼2.5% greater than the double precision simulation, respectively. We further observe that the double precision simulation values are very close to the experimental values of Swalwell [50]. As a fourth reference point, we also consider the double precision simulation of Park et al. [23], which was also conducted using PyFR using a very similar configuration to that described here. These results can be seen to be somewhat closer to our single precision results, especially with regard to the drag. Because the instantaneous dynamics of the problem are highly chaotic, some variation in the averages is expected. Moving on to the PSD of the lift coefficient, we find that both the single and double precision simulations are able to accurately predict the two peak shedding modes observed in the experiment. The fact that the single precision simulation is capable of predicting these peaks is promising and suggests that it is a suitable choice for this class of problems.

3. *Performance*

On account of the quadrature-based antialiasing procedure, this simulation falls into the FLOP bound regime. Comparing the performance of the two simulations, a speedup factor of 1.87 is observed. This factor is comparable to the $\wp = 5$ result in Table 4. Although the NACA 0021 simulation is performed at $\wp = 4$, and thus would typically expect to observe a speedup in line with the $\wp = 4$ result from Table 4 (1.69), the presence of antialiasing increases the sizes of the various operator matrices such that they are now similar to and, in some cases, even exceed those encountered in a $\wp = 5$ simulation.

Table 6 Time-averaged lift C_L and drag C_D coefficients for the NACA 0021 airfoil test case compared against the reference simulations of Park et al. [23] and the experimental data of Swalwell [50]

Simulation	Span	C_L	C_D
PyFR (single)	$4c$	0.957	1.594
PyFR (double)	$4c$	0.935	1.555
PyFR (Park et al. [23])	$4c$	0.953	1.579
Experimental (Swalwell [50])	$7.2c$	0.932	1.545

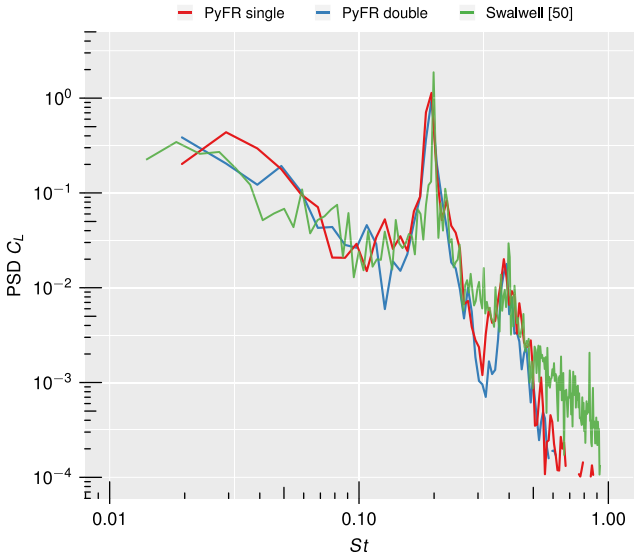

Fig. 15 PSD of C_L for two PyFR simulations compared against experimental data of Ref. [50]. Although PyFR data were obtained from time- and surfaced-averaged lift coefficient, experimental data were obtained from a sectional lift coefficient at a fixed spanwise location.

VI. Conclusions

In this paper, the impact of number precision was assessed for high-order implicit large-eddy simulations. A detailed theoretical analysis of the performance considerations associated with a switch from double to single precision has been presented. The impact of number representation on four unsteady test cases was shown: the Taylor–Green vortex, flow over a cylinder, flow over a T106c low-pressure turbine cascade, and flow over a NACA 0021 airfoil in deep stall. For all test cases, the use of single precision was found to have little impact on the overall accuracy of the simulations. This suggests that, for many real-world test cases, single precision arithmetic is indeed sufficient. For bandwidth bound simulations, speedup factors between 1.4 and 2.9 were observed, hence resulting in a greatly reduced time to solution.

Acknowledgments

The authors would like to thank the U.S. Air Force Office of Scientific Research for their support via grant FA9550-14-1-0186 under the direction of Jean-Luc Cambier; and Margot Gerritsen for access to the XStream graphics processing unit computing cluster, which is supported by the National Science Foundation Major Research Instrumentation program (ACI-1429830). The authors would also like to thank NVIDIA for hardware donations.

References

[1] Reed, W. H., and Hill, T. R., "Triangular Mesh Methods for the Neutron Transport Equation," Los Alamos Scientific Lab. TR LA-UR-73-479, Los Alamos, NM, 1973.

[2] Kopriva, D. A., and Kolias, J. H., "A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows," *Journal of Computational Physics*, Vol. 125, No. 1, 1996, pp. 244–261. doi:10.1006/jcph.1996.0091

[3] Sun, Y., Wang, Z. J., and Liu, Y., "High-Order Multidomain Spectral Difference Method for the Navier–Stokes Equations on Unstructured Hexahedral Grids," *Communications in Computational Physics*, Vol. 2, No. 2, 2007, pp. 310–333.

[4] Huynh, H. T., "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods," AIAA Paper 2007-4079, 2007. doi:10.2514/6.2007-4079

[5] Hesthaven, J. S., and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Vol. 54, Springer, New York, 2008. doi:10.1007/978-0-387-72067-8

[6] Klöckner, A., Warburton, T., Bridge, J., and Hesthaven, J. S., "Nodal Discontinuous Galerkin Methods on Graphics Processors," *Journal of Computational Physics*, Vol. 228, No. 21, 2009, pp. 7863–7882. doi:10.1016/j.jcp.2009.06.041

[7] 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, 2011. doi:10.2514/6.2011-3229

[8] 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. doi:10.1016/j.cpc.2014.07.011

[9] Witherden, F. D., Vermeire, B. C., and Vincent, P. E., "Heterogeneous Computing on Mixed Unstructured Grids with PyFR," *Computers and Fluids*, Vol. 120, Oct. 2015, pp. 173–186. doi:10.1016/j.compfluid.2015.07.016

[10] "IEEE Standard for Floating-Point Arithmetic," *IEEE STD 754-2008*, IEEE Publ., Piscataway, NJ, Aug. 2008, pp. 1–70. doi:10.1109/IEEESTD.2008.4610935

[11] Bailey, D. H., "High-Precision Floating-Point Arithmetic in Scientific Computation," *Computing in Science and Engineering*, Vol. 7, No. 3, 2005, pp. 54–61. doi:10.1109/MCSE.2005.52

[12] Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., and Lindahl, E., "GROMACS: High Performance Molecular Simulations Through Multi-Level Parallelism from Laptops to Supercomputers," *SoftwareX*, Vols. 1–2, Sept. 2015, pp. 19–25. doi:10.1016/j.softx.2015.06.001

[13] Board, J. A., Causey, J. W., Leathrum, J. F., Windemuth, A., and Schulten, K., "Accelerated Molecular Dynamics Simulation with the Parallel Fast Multipole Algorithm," *Chemical Physics Letters*, Vol. 198, Nos. 1–2, 1992, pp. 89–94. doi:10.1016/0009-2614(92)90053-P

[14] Hess, B., Kutzner, C., Van Der Spoel, D., and Lindahl, E., "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation," *Journal of Chemical Theory and Computation*, Vol. 4, No. 3, 2008, pp. 435–447. doi:10.1021/ct700301q

[15] Heinecke, A., Breuer, A., Rettenberger, S., Bader, M., Gabriel, A.-A., Pelties, C., Bode, A., Barth, W., Liao, X.-K., Vaidyanathan, K., et al., "Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Publ., Piscataway, NJ, 2014, pp. 3–0. doi:10.1109/SC.2014.6

[16] Gasperini, P., and Vannucci, G., "FPSPACK: A Package of FORTRAN Subroutines to Manage Earthquake Focal Mechanism Data," *Computers and Geosciences*, Vol. 29, No. 7, 2003, pp. 893–901. doi:10.1016/S0098-3004(03)00096-7

[17] Komatitsch, D., Michéa, D., and Erlebacher, G., "Porting a High-Order Finite-Element Earthquake Modeling Application to NVIDIA Graphics Cards Using CUDA," *Journal of Parallel and Distributed Computing*, Vol. 69, No. 5, 2009, pp. 451–460. doi:10.1016/j.jpdc.2009.01.006

[18] Homann, H., Dreher, J., and Grauer, R., "Impact of the Floating-Point Precision and Interpolation Scheme on the Results of DNS of Turbulence by Pseudo-Spectral Codes," *Computer Physics Communications*, Vol. 177, No. 7, 2007, pp. 560–565. doi:10.1016/j.cpc.2007.05.019

[19] Amdahl, G. M., "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ACM Press, New York, 1967, pp. 483–485. doi:10.1145/1465482.1465560

[20] Lai, J., and Seznec, A., "Performance Upper Bound Analysis and Optimization of SGEMM on Fermi and Kepler GPUs," *2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, IEEE Publ., Piscataway, NJ, 2013, pp. 1–10. doi:10.1109/CGO.2013.6494986

[21] Cockburn, B., and Shu, C.-W., "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems," *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463. doi:10.1137/S0036142997316712

[22] Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, Springer, New York, 2009. doi:10.1007/b79761

[23] Park, J., Witherden, F., and Vincent, P., "High-Order Implicit Large-Eddy Simulations of Flow over a NACA0021 Aerofoil," *AIAA Journal*, Vol. 55, No. 7, June 2017, pp. 2186–2197. doi:10.2514/1.J055304

[24] Witherden, F. D., "On the Development and Implementation of High-Order Flux Reconstruction Schemes for Computational Fluid Dynamics," Ph.D. Thesis, Imperial College London, 2015.

[25] Witherden, F., and Vincent, P., "On the Identification of Symmetric Quadrature Rules for Finite Element Methods," *Computers and Mathematics with Applications*, Vol. 69, No. 10, 2015, pp. 1232–1241. doi:10.1016/j.camwa.2015.03.017

[26] Shunn, L., and Ham, F., "Symmetric Quadrature Rules for Tetrahedra Based on a Cubic Close-Packed Lattice Arrangement," *Journal of Computational and Applied Mathematics*, Vol. 236, No. 17, 2012, pp. 4348–4364. doi:10.1016/j.cam.2012.03.032

[27] Williams, D. M., Shunn, L., and Jameson, A., "Symmetric Quadrature Rules for Simplexes Based on Sphere Close Packed Lattice Arrangements," *Journal of Computational and Applied Mathematics*, Vol. 266, Aug. 2014, pp. 18–38. doi:10.1016/j.cam.2014.01.007

[28] Vincent, P. E., Witherden, F. D., Farrington, A. M., Ntemos, G., Vermeire, B. C., Park, J. S., and Iyer, A. S., "PyFR: Next-Generation High-Order Computational Fluid Dynamics on Many-Core Hardware," AIAA Paper 2015-3050, 2015. doi:10.2514/6.2015-3050

[29] Wozniak, B. D., Witherden, F. D., Russell, F. P., Vincent, P. E., and Kelly, P. H., "GiMMiK—Generating Bespoke Matrix Multiplication Kernels for Accelerators: Application to High-Order Computational Fluid Dynamics," *Computer Physics Communications*, Vol. 202, May 2016, pp. 12–22. doi:10.1016/j.cpc.2015.12.012

[30] Vincent, P., Witherden, F., Vermeire, B., Park, J. S., and Iyer, A., "Towards Green Aviation with Python at Petascale," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Publ., Piscataway, NJ, 2016, p. 1. doi:10.1109/SC.2016.1

[31] Wang, Z., et al., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 811–845. doi:10.1002/fld.v72.8

[32] van Rees, W. M., Leonard, A., Pullin, D., 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, No. 8, 2011, pp. 2794–2805. doi:10.1016/j.jcp.2010.11.031

[33] Rusanov, V. V., "Calculation of Interaction of Non-Steady Shock Waves with Obstacles," *Journal of Computational Mathematics and Physics*, No. 1, 1961, pp. 267–279.

[34] Kennedy, C. A., Carpenter, M. H., and Lewis, R. M., "Low-Storage, Explicit Runge–Kutta Schemes for the Compressible Navier–Stokes Equations," *Applied Numerical Mathematics*, Vol. 35, No. 3, 2000, pp. 177–219. doi:10.1016/S0168-9274(99)00141-5

[35] Norberg, C., "LDV Measurements in the Near Wake of a Circular Cylinder," *International Journal for Numerical Methods in Fluids*, Vol. 28, No. 9, 1998, pp. 1281–1302. doi:10.1002/(ISSN)1097-0363

[36] Ma, X., Karamanos, G. S., and Karniadakis, G. E., "Dynamics and Low-Dimensionality of a Turbulent Near Wake," *Journal of Fluid Mechanics*, Vol. 410, May 2000, pp. 29–65 doi:10.1017/S0022112099007934

[37] Breuer, M., "Large Eddy Simulation of the Subcritical Flow Past a Circular Cylinder," *International Journal for Numerical Methods in Fluids*, Vol. 28, No. 9, 1998, pp. 1281–1302. doi:10.1002/(ISSN)1097-0363

[38] Kravchenko, A. G., and Moin, P., "Numerical Studies of Flow over a Circular Cylinder at $Re_D = 3900$," *Physics of Fluids*, Vol. 12, No. 2, 2000, pp. 403–417. doi:10.1063/1.870318

[39] Parnaudeau, P., Carlier, J., Heitz, D., and Lamballais, E., "Experimental and Numerical Studies of the Flow over a Circular Cylinder at Reynolds Number 3900," *Physics of Fluids*, Vol. 20, No. 8, 2008, Paper 085101. doi:10.1063/1.2957018

[40] Williamson, C. H. K., "Vortex Dynamics in the Cylinder Wake," *Annual Review of Fluid Mechanics*, Vol. 28, No. 1, 1996, pp. 477–539. doi:10.1146/annurev.fl.28.010196.002401

[41] Lehmkuhl, O., Rodriguez, I., Borrell, R., and Oliva, A., "Low-Frequency Unsteadiness in the Vortex Formation Region of a Circular

Cylinder," *Physics of Fluids*, Vol. 25, No. 8, 2013, Paper 085109. doi:10.1063/1.4818641

[42] Jameson, A., and Baker, T., "Solution of the Euler Equations for Complex Configurations," AIAA Paper 1983-1929, 1983. doi:10.2514/6.1983-1929

[43] Witherden, F., Park, J., and Vincent, P., "An Analysis of Solution Point Coordinates for Flux Reconstruction Schemes on Tetrahedral Elements," *Journal of Scientific Computing*, Vol. 69, No. 2, Nov. 2016, pp. 905–920. doi:10.1007/s10915-016-0204-y

[44] Witherden, F. D., and Vincent, P. E., "An Analysis of Solution Point Coordinates for Flux Reconstruction Schemes on Triangular Elements," *Journal of Scientific Computing*, Vol. 61, No. 2, 2014, pp. 398–423. doi:10.1007/s10915-014-9832-2

[45] Wood, J., Strasizar, T., and Hathaway, M., "Test Case E/CA-6, Subsonic Turbine Cascade T106," *Test Cases for Computation of Internal Flows*, AGARD AR-275, Neuilly-Sur-Seine, France, July 1990.

[46] Michálek, J., Monaldi, M., and Arts, T., "Aerodynamic Performance of a Very High Lift Low Pressure Turbine Airfoil (T106C) at Low Reynolds and High Mach Number with Effect of Free Stream Turbulence Intensity," *Journal of Turbomachinery*, Vol. 134, No. 6, 2012, Paper 061009. doi:10.1115/1.4006291

[47] Pacciani, R., Marconcini, M., Arnone, A., and Bertini, F., "An Assessment of the Laminar Kinetic Energy Concept for the Prediction of High-Lift, Low-Reynolds Number Cascade Flows," *Journal of Power and Energy*, Vol. 225, No. 7, 2011, pp. 995–1003.

[48] Hillewaert, K., de Wiart, C. C., Verheylewegen, G., and Arts, T., "Assessment of a High-Order Discontinuous Galerkin Method for the Direct Numerical Simulation of Transition at Low-Reynolds Number in the T106C High-Lift Low Pressure Turbine Cascade," *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition, American Society of Mechanical Engineers*, ASME Paper V02BT39A034, New York, 2014. doi:10.1115/GT2014-26739

[49] Garai, A., Diosady, L. T., Murman, S. M., and Madavan, N., "DNS of Flow in a Low-Pressure Turbine Cascade with Elevated Inflow Turbulence Using a Discontinuous-Galerkin Spectral-Element Method," *Proceedings of ASME Turbo Expo 2016*, ASME Paper GT2016-56700, New York, 2016.

[50] Swalwell, K. E., "The Effect of Turbulence on Stall of Horizontal Axis Wind Turbines," Ph.D. Thesis, Monash Univ., Clayton, VIC, Australia, 2005.

[51] Haase, W., Braza, M., and Revell, A., *DESider–A European Effort on Hybrid RANS-LES Modelling: Results of the European-Union Funded Project, 2004-2007*, Vol. 103, Springer Science and Business Media, New York, 2009. doi:10.1007/978-3-540-92773-0

[52] Garbaruk, A., Leicher, S., Mockett, C., Spalart, P., Strelets, M., and Thiele, F., "Evaluation of Time Sample and Span Size Effects in DES of Nominally 2D Airfoils Beyond Stall," *Progress in Hybrid RANS-LES Modelling*, edited by S. H. Peng, P. Doerffer, and W. Haase, Vol. 111, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, Berlin, 2010, pp. 87–99. doi:10.1007/978-3-642-14168-3_7

P. G. Tucker
Associate Editor