Array 17 (2023) 100268

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

Evaluation of next-generation high-order compressible fluid dynamic solver on cloud computing for complex industrial flows

R. Al Jahdali [a,*], S. Kortas [c], M. Shaikh [c], L. Dalcin [a], M. Parsani [a,b]

[a] King Abdullah University of Science and Technology (KAUST), Computer Electrical and Mathematical Science and Engineering Division (CEMSE), Extreme Computing Research Center (ECRC), 23955-6900 Thuwal, Saudi Arabia
[b] King Abdullah University of Science and Technology (KAUST), Physical Sciences and Engineering Division (PSE), 23955-6900 Thuwal, Saudi Arabia
[c] KAUST Supercomputing Core Laboratory, 23955-6900 Thuwal, Saudi Arabia

ARTICLE INFO

Keywords:
Cloud computing
Amazon Web Services Elastic Compute Cloud
Fluid mechanics
Compressible Navier–Stokes equations
Fully-discrete entropy stable algorithms

ABSTRACT

Industrially relevant computational fluid dynamics simulations frequently require vast computational resources that are only available to governments, wealthy corporations, and wealthy institutions. Thus, in many contexts and realities, high-performance computing grids and cloud resources on demand should be evaluated as viable alternatives to conventional computing clusters. In this work, we present the analysis of the time-to-solution and cost of an entropy stable collocated discontinuous Galerkin (SSDC) compressible computational fluid dynamics framework on Ibex, the on-premises cluster at KAUST, and the Amazon Web Services Elastic Compute Cloud for complex compressible flows. SSDC is a prototype of the next generation computational fluid dynamics frameworks developed following the road map established by the NASA CFD vision 2030. We simulate complex flow problems using high-order accurate fully-discrete entropy stable algorithms. In terms of time-to-solution, the Amazon Elastic Compute Cloud delivers the best performance, with the Graviton2 processors based on the Arm architecture being the fastest. However, the results also indicate that the Ibex nodes based on the AMD Rome architecture deliver good performance, close to those observed for the Amazon Elastic Compute Cloud. Furthermore, we observed that computations performed on the Ibex on-premises cluster are currently less expensive than those performed in the cloud. Our findings could be used to develop guidelines for selecting high-performance computing cloud resources to simulate realistic fluid flow problems.

1. Introduction

The process of predicting fluid flow (such as gases and liquids), mass transfer, chemical reactions, and other associated phenomena with a computer during the design or production process is known as computational fluid dynamics (CFD). State-of-the-art CFD is critical in modeling many physical phenomena, including biomedicine [1], plasma science [2], climate [3,4], weather [5,6], and aerodynamics [7,8], and for elucidating and analyzing the underlying mechanisms of these complex phenomena. In the aeronautics and aerospace fields, aggressive use of CFD has resulted in significant reductions in wind tunnel time as well as a reduction in the number of experimental rig tests. Thus, through the use of CFD, industries, governments and national laboratories have been able to produce products faster and at reduced costs [8], saving hundreds of millions of dollars. In addition to reducing testing requirements, CFD has the additional potential of providing superior understanding and insight into the critical physical phenomena limiting component performance, thereby opening up new frontiers in a variety of fields.

Among the critical goals, NASA's vision roadmap [9] indicates the computational procedure and algorithms to perform realistic aerodynamic simulations by 2030 with the next generation of computing architectures. The path to the final aerodynamic simulation goal includes a scheduled demonstration of various critical technologies and production scalable entropy-stable solvers are one of them [9]. A prototype of these new solvers, named SSDC, where "Entropy" is represented in the acronym by its thermodynamic symbol, "S", has been developed and deployed for industrial collaborations [10]. SSDC combines a state-of-the-art, provably stable, adaptive order solver, with software engineering that exploits hybrid shared-distributed memory capabilities.

High-performance computing (HPC) has had a significant influence on CFD. The demand for HPC systems increases with the increasing

* Corresponding author.
E-mail addresses: Rasha.Aljahdali@kaust.edu.sa (R. Al Jahdali), samuel.kortas@kaust.edu.sa (S. Kortas), mohsin.shaikh@kaust.edu.sa (M. Shaikh), dalcinl@gmail.com (L. Dalcin), matteo.parsani@kaust.edu.sa (M. Parsani).

https://doi.org/10.1016/j.array.2022.100268
Received 25 October 2022; Received in revised form 1 December 2022; Accepted 5 December 2022
Available online 9 December 2022

complexity of the simulation, which is a primary element in fluid flow simulations. Owning high-performance computer facilities helps to process simulations faster and allows one to perform large-scale simulations. Supercomputer centers and other HPC computing capabilities are being used to perform massive simulations in academia, national laboratories, and industry. Nevertheless, the cost of the computing cluster may be exorbitant to purchase and maintain, and therefore, it may not be able to keep up with today's computational demands. Hence, on-demand grids and cloud resources should be viewed as alternatives to traditional computing clusters in various situations and circumstances. A major advantage of cloud resources is the opportunity to use cutting-edge hardware without the requirement for financial investment or IT maintenance costs. In addition, instead of an annual license, the software can be purchased on a pay-per-use basis. Because the industry is interested in using on-demand computing services for engineering simulations, benchmarking the performance of next-generation of compressible CFD solver prototypes, such as SSDC, on cloud resources is essential.

Currently, the industry is experimenting with a variety of ways for incorporating cloud computing into CFD workloads. Thus, it is pivotal to evaluate the performance of parallel systems to determine whether they are capable of running complex industrial CFD problems with upcoming solvers. On one side, important studies in this direction for well established classic solvers based on finite volume, finite differences, and finite elements (up to second order accuracy) have been reported, for instance, by Peña-Monferrer et al. [11], Ashton et al. [12] and Turner et al. [13] for complex industry problems. In particular, in early 2020, the results of the performance of the open-source OpenFOAM software on the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) service for racing vehicles employing a hybrid RANS-LES model were presented by Ashton and co-authors [12]. The authors demonstrated that AWS might provide an HPC environment that would enable wider usage of high-fidelity CFD methods by permitting higher core counts and reducing turnaround time. In early 2021, in Peña-Monferrer et al. [11] presented a hybrid cloud solution for efficient simulation and analysis of drop dispersions by breaking down and study the CFD data analysis pipeline into small microservice-like processes. Turner and co-authors [13] gave an exhaustive analysis of how well GPU and CPU architectures work and how much they cost for a complete aircraft RANS simulation using the CFD code zCFD. On the other side, studies on the performance of adaptive non-linearly stable (high-order) compressible algorithms on the HPC cloud computing is lacking. Those spatial and temporal discretizations, in the wake of the final NASA 2030 CFD vision report [9], received an impetus from leading U.S. national laboratories, (*e.g.*, NASA and Sandia National Laboratories), top-tier academic institutions in Europe, U.S., and the Middle East. In particular, at the end of 2021, the first fully discrete entropy stable solver of any order on unstructured for compressible CFD, named SSDC, was presented in [10]. The SSDC's scalable algorithms have been proven to run fast and effectively on hardware ranging from laptops to supercomputers with more than 180,000 processors, such as the Shaheen XC40 supercomputer. However, no studies on the performance of these new solvers on the HPC cloud computing is available in literature.

Here, we provide a comprehensive cost analysis of the SSDC solver for simulating flow problems on AWS EC2. The purpose is to assess the viability of the HPC cluster on the Amazon cloud for solving complex CFD industry flow problems and identify a set of AWS EC2 instances that deliver the shortest time and the lowest possible price. We use four test cases [10,14,15]: the turbulent flow over two spheres in tandem at a Reynolds number and a Mach number of $Re = 3.9\times10^3$ and $Ma = 0.1$, respectively; the flow past a delta wing with the experimental sting at $Re = 10^6$ and $Ma = 0.07$; the NASA juncture flow experiment at $Re = 2.4\times10^6$ and $Ma = 0.189$; and the flow past the Imperial front wing at $Re = 2.2.\times10^5$ and $Ma = 0.036$.

Numerous commercial solvers are currently deployed and utilized on the cloud to simulate fluid flow problems based on the Reynolds-averaged Navier–Stokes (RANS) approach. However, there is no doubt that large eddy simulation (LES) has the potential to deliver results that are more accurate and reliable and will be one of the main approaches to solving challenging fluid flows problems. This study reports the first step towards high-performance CFD simulations on the AWS cloud cluster for industrial workloads based on LES approaches with an *hp*-adaptive compressible solver. The main contributions of this work are as follows. First, we deploy and test a novel prototype of the next generation of compressible CFD solver "SSDC" on the HPC cluster on the cloud. SSDC is a prototype of the framework defined in the NASA CFD 2030 vision, and it has novel capabilities in terms of adaptivity in space and time. Second, we manage and operate the AWS cloud cluster by optimizing the PETSc library and SSDC framework for each of the architectures used in the study. The installation details of the optimized PETSc library can be found in https://github.com/ecrc/petsc_installation_aws_cloud_2022. The former step is done in conjunction with the Slurm scheduler to simplify the management of jobs. Since we are simulating the LES model, a large number of AWS EC2 instances is needed. Third, we compare the on-premises cluster and the most recent architectures on the AWS cloud in terms of performance and cost.

The paper is organized as follows. In Section 2, we present the key ideas and elements of the spatial and temporal algorithms implemented in SSDC, whereas in Section 3, we give a brief overview of the software implementation and infrastructure. Section 4 briefly describes the computing environments of the on-premise Ibex cluster and Amazon EC2 cloud computing environment. Section 5 presents the performance results of the test case studies to analyze the performance of the SSDC solver on Ibex and Amazon EC2 clusters. Finally, the conclusion and future work are drawn in Section 6.

2. The compressible Navier–Stokes equations

In this section, we give an overview of the discretization of the compressible Navier–Stokes equations. A detailed presentation of the key elements of the spatial discretization is presented for the advection–diffusion equation in multiple dimensions in Appendix A.

In the framework of the method of line approach, we first present the spatial discretization and subsequently describe the temporal integration approach. The compressible Navier–Stokes equations in Cartesian coordinates read

$$\begin{aligned}
&\frac{\partial \boldsymbol{q}}{\partial t} + \sum_{m=1}^{3} \frac{\partial \boldsymbol{\mathcal{F}}^{I}_{x_m}}{\partial x_m} = \sum_{m=1}^{3} \frac{\partial \boldsymbol{\mathcal{F}}^{V}_{x_m}}{\partial x_m}, \quad \forall \left(x_1, x_2, x_3\right) \in \Omega, \quad t \geq 0,\\
&\boldsymbol{q}\left(x_1, x_2, x_3, t\right) = \boldsymbol{g}^{(B)}\left(x_1, x_2, x_3, t\right), \quad \forall \left(x_1, x_2, x_3\right) \in \Gamma, \quad t \geq 0,\\
&\boldsymbol{q}\left(x_1, x_2, x_3, 0\right) = \boldsymbol{g}^{(0)}\left(x_1, x_2, x_3\right), \quad \forall \left(x_1, x_2, x_3\right) \in \Omega,
\end{aligned} \tag{1}$$

where the vectors $\boldsymbol{q}$, $\boldsymbol{\mathcal{F}}^{I}_{x_m}$ and $\boldsymbol{\mathcal{F}}^{V}_{x_m}$ denote the conserved variables, the inviscid fluxes, and the viscous fluxes, respectively. The boundary data, $\boldsymbol{g}^{(B)}$, and the initial condition, $\boldsymbol{g}^{(0)}$, are assumed to be in $L^2(\Omega)$, with the further assumption that $\boldsymbol{g}^{(B)}$ will be set to coincide with linear, well-posed boundary conditions, prescribed in such a way that either entropy conservation or entropy stability is achieved. System (1) is closed with the assumption of the ideal gas model. The conserved variable vector can be written as

$$\boldsymbol{q} = \left[\rho, \rho\mathcal{U}_1, \rho\mathcal{U}_2, \rho\mathcal{U}_3, \rho\mathcal{E}\right]^{\mathrm{T}},$$

where ρ denotes the density, $\boldsymbol{\mathcal{U}} = \left[\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3\right]^{\mathrm{T}}$ is the velocity vector, and $\mathcal{E}$ is the specific total energy. The inviscid fluxes are given as

$$\begin{aligned}
\boldsymbol{\mathcal{F}}^{I}_{x_m} = \big[&\rho\mathcal{U}_m, \rho\mathcal{U}_m\mathcal{U}_1 + \delta_{m,1}\mathcal{P}, \rho\mathcal{U}_m\mathcal{U}_2 + \delta_{m,2}\mathcal{P},\\
&\rho\mathcal{U}_m\mathcal{U}_3 + \delta_{m,3}\mathcal{P}, \rho\mathcal{U}_m\mathcal{H}\big]^{\mathrm{T}},
\end{aligned}$$

where $\mathcal{P}$ is the pressure, $\mathcal{H}$ is the specific total enthalpy and $\delta_{i,j}$ is the Kronecker delta. The required constituent relations are

$$\mathcal{H} = c_p\mathcal{T} + \frac{1}{2}\boldsymbol{\mathcal{U}}^{\mathrm{T}}\boldsymbol{\mathcal{U}}, \quad \mathcal{P} = \rho R\mathcal{T}, \quad R = \frac{R_u}{M_w},$$

where $\mathcal{T}$ is the temperature, R_u is the universal gas constant, M_w is the molecular weight of the gas, and c_p is the specific heat capacity at constant pressure. Finally, the specific thermodynamic entropy is given as

$$s = \frac{R}{\gamma - 1}\log\left(\frac{\mathcal{T}}{\mathcal{T}_\infty}\right) - R\log\left(\frac{\rho}{\rho_\infty}\right), \quad \gamma = \frac{c_p}{c_p - R},$$

where $\mathcal{T}_\infty$ and ρ_∞ are the reference temperature and density. The viscous fluxes $\boldsymbol{\mathcal{F}}^V_{x_m}$ are given by

$$\boldsymbol{\mathcal{F}}^V_{x_m} = \left[0, \tau_{1,m}, \tau_{2,m}, \tau_{3,m}, \sum_{i=1}^{3}\tau_{i,m}\mathcal{U}_i - \kappa\frac{\partial\mathcal{T}}{\partial x_m}\right]^{\mathrm{T}}, \tag{2}$$

while the viscous stresses are defined as

$$\tau_{i,j} = \mu\left(\frac{\partial\mathcal{U}_i}{\partial x_j} + \frac{\partial\mathcal{U}_j}{\partial x_i} - \delta_{i,j}\frac{2}{3}\sum_{n=1}^{3}\frac{\partial\mathcal{U}_n}{\partial x_n}\right), \tag{3}$$

where $\mu(\mathcal{T})$ is the dynamic viscosity and $\kappa(\mathcal{T})$ is the thermal conductivity.

The compressible Navier–Stokes equations given in (1) have a convex extension, that when integrated over the physical domain, Ω, depends only on the boundary data and negative semi-definite dissipation terms. This convex extension depends on an entropy function, S, that is constructed from the thermodynamic entropy as

$$S = -\rho s,$$

and provides a mechanism for proving stability in the L^2 norm. The entropy variables $\boldsymbol{\mathcal{W}}$ are an alternative variable set related to the conservative variables via a one-to-one mapping. They are defined in terms of the entropy function S by the relation $\boldsymbol{\mathcal{W}}^{\mathrm{T}} = \partial S/\partial\boldsymbol{\mathcal{Q}}$ and they are extensively used in the entropy stability proofs of the algorithms used herein; see, for instance, [16–19] and the references therein. In addition, they simultaneously symmetrize the inviscid and the viscous flux Jacobians in all three spatial directions. Following the analysis described in [16,20], we multiply Eqs. (1) by the (local) entropy variables $\boldsymbol{\mathcal{W}}$ and arrive at the integral form of the (scalar) entropy equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega S\,\mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t}\eta \le \sum_{m=1}^{3}\int_\Gamma\left(\boldsymbol{\mathcal{W}}^{\mathrm{T}}\boldsymbol{\mathcal{F}}^V_{x_-}\mathcal{F}_x\right)n_x\,\mathrm{d}\Gamma - DT, \tag{4}$$

where n_x is the mth component of the outward facing unit normal to Γ and

$$DT = \sum_{m,j=1}^{3}\int_\Omega\left(\frac{\partial\boldsymbol{\mathcal{W}}}{\partial x_m}\right)^{\mathrm{T}}\mathsf{C}_{m,j}\frac{\partial\boldsymbol{\mathcal{W}}}{\partial x_j}\,\mathrm{d}\Omega. \tag{5}$$

More details about the continuous entropy analysis can be found [21–23].

2.1. Spatial discretization of the compressible Navier–Stokes equations

The details of the algorithm for conforming and nonconforming interfaces can be found in [10,16,19,24–26]. Herein, we summarized the main steps of the algorithm for conforming interfaces.

To approximate the compressible Navier–Stokes equations, the domain Ω is divided into K nonoverlapping elements. Then, following the procedure outlined for the spatial discretization of convection–diffusion Eqs. (A.1), on the κth element, the generic entropy stable discretization of (1) reads

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{q}_\kappa}{\mathrm{d}t} + \sum_{m=1}^{3}2\mathsf{D}^{I,\kappa}_{x_m}\circ\mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa,\boldsymbol{q}_\kappa\right)\mathbf{1}_\kappa = \sum_{m,j=1}^{3}\mathsf{D}^{V_1,\kappa}_{x_m}\left[\mathsf{C}_{m,j}\right]\boldsymbol{\theta}_j \\ + \mathbf{SAT}^I + \mathbf{SAT}^V + \mathbf{diss}^I + \mathbf{diss}^{V_1},\end{aligned} \tag{6}$$

where $\boldsymbol{q}_\kappa$ is the numerical solution vector at the mesh nodes, while $\mathbf{diss}^I$ and $\mathbf{diss}^V$ denote the interface dissipation contributions for the inviscid and viscous parts of the equations, respectively.

The continuous entropy stability analysis is mimicked by approximating the derivatives of the inviscid fluxes as follows [27], *i.e.*,

$$\frac{\partial f^I_{x_m}}{\partial x_m} \approx 2\mathsf{D}^{I,\kappa}_{x_m}\circ\mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa,\boldsymbol{q}_\kappa\right)\mathbf{1}_\kappa, \tag{7}$$

where $\mathsf{D}^{I,\kappa}_{x_m}$ is a differentiation matrix for the x_m direction, $\circ$ denotes the Hadamard product, $\mathsf{F}_{x_m}\left(\boldsymbol{q}_\kappa,\boldsymbol{q}_\kappa\right)$ is a two-point flux function matrix, and $\mathbf{1}_\kappa$ is a vector of ones.

The differentiation matrix $\mathsf{D}^{I,\kappa}_{x_m}$ is constructed as

$$\mathsf{D}^{I,\kappa}_{x_m} \equiv \frac{1}{2}\mathsf{J}^{-1}_\kappa\sum_{l=1}^{3}\left(\mathsf{D}_{\xi_l}\left[\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right]_\kappa + \left[\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right]_\kappa\mathsf{D}_{\xi_l}\right), \tag{8}$$

where J_κ denotes the determinant of the discrete Jacobian, D_{ξ_l} is an SBP operator approximating $\frac{\partial}{\partial\xi}$, while $\left[\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right]_\kappa$ indicates the discrete metric terms which must satisfy a discrete version of the geometric conservation law (GCL) constraints [19,25,28]. In addition, $\mathsf{D}^{V_1,\kappa}_{x_m}$ is the differentiation matrix, constructed as

$$\mathsf{D}^{V_1,\kappa}_{x_m} \equiv \mathsf{J}^{-1}_\kappa\sum_{l=1}^{3}\mathsf{D}_{\xi_l}\left[\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right]_\kappa. \tag{9}$$

Furthermore, the derivative of the entropy variables, w, is approximated as

$$\boldsymbol{\theta}_a = \mathsf{D}^{V_2,\kappa}_{x_a}\boldsymbol{w}_\kappa + \mathbf{SAT}^{V_2} \approx \frac{\partial w}{\partial x_a}. \tag{10}$$

The differentiation matrix $\mathsf{D}^{V_2,\kappa}_{x_j}$ in (10) is defined as

$$\mathsf{D}^{V_2,\kappa}_{x_j} \equiv \mathsf{J}^{-1}_\kappa\sum_{a=1}^{3}\left[\mathcal{J}\frac{\partial\xi_a}{\partial x_j}\right]_\kappa\mathsf{D}_{\xi_a}. \tag{11}$$

The metric terms $\left[\mathcal{J}\frac{\partial\xi_l}{\partial x_m}\right]_\kappa$ and $\left[\mathcal{J}\frac{\partial\xi_a}{\partial x_j}\right]_\kappa$ can be computed in different ways. In the SSDC solver, these metrics terms are computed based upon the optimization procedure of Crean et al. [28]. Algorithmic details for conforming interfaces are given in [25]; for p- and hp-nonconforming interfaces, the interested reader is referred to [19,24].

The entropy function of the semidiscretization (6) mimics closely Eq. (4). In fact, following closely the entropy stability analysis, the SBP operators and their equivalent telescoping forms yield [16,19,22,24–26]

$$\frac{d}{dt}\mathbf{1}^{\mathrm{T}}\hat{\mathsf{P}}\mathbf{S} = \frac{d}{dt}\eta = \mathbf{BT} - \mathbf{DT} + \boldsymbol{\Upsilon}, \tag{12}$$

which is the semi-discrete analog of (4). Here **BT** is the discrete boundary term (i.e., the discrete version of the first integral term on the right-hand side of (4)), **DT** is the discrete dissipation term (i.e., the discrete version of the second term on the right-hand side of (4)) and $\boldsymbol{\Upsilon}$ enforces interface coupling and boundary conditions [16,20]. For completeness, we note that the matrix $\hat{\mathsf{P}}$ may be thought of as the mass matrix in the context of the discontinuous Galerkin finite element method.

In our framework, the boundary conditions necessary to close system (1) preserve the entropy stability of the interior operators described above. Precisely, solid inviscid and viscous wall boundary conditions are imposed as described in [20,29] whereas, for the far-field, we use the approach described in [30]. The implementation of boundary conditions for a general framework (not necessarily entropy stable) can be found in [31].

2.2. Temporal discretization of the compressible Navier–Stokes equations

A general (explicit or implicit) s-stage Runge–Kutta (RK) scheme can be concisely encapsulated using its Butcher tableau [32], which is composed of a matrix A of dimensions $s \times s$ and two vectors b and c of length s. The basic idea of the relaxation procedure is to enforce conservation, dissipation, or other solution properties with respect to

a convex functional by scaling the weights b_i of the RK method by a real-value parameter $\tilde{\gamma}$. Hence, the time step from $u^n \approx u(t_n)$ is given by

$$\begin{aligned} y_i &= u^n + \Delta t \sum_{j=1}^{s} a_{ij}\, f(t_n + c_j \Delta t, y_j), \\ u_{\tilde{\gamma}}^{n+1} &= u^n + \tilde{\gamma}_n \Delta t \sum_{i=1}^{s} b_i f(t_n + c_i \Delta t, y_i), \end{aligned} \tag{13}$$

where the stage values of the RK method are denoted by y_i. The parameter $\tilde{\gamma}_n$ is computed using the global relaxation procedure, *i.e.*, $\tilde{\gamma}_n$ is a root of a global nonlinear algebraic equation for η [33]. We recover the step of the classic RK method if $\tilde{\gamma}_n = 1$.

In the SSDC solver, global relaxation RK schemes are used to integrate in time systems of ODEs (6) such that Eq. (12) is fulfilled. For performance and robustness reasons, we select the secant method as the root finding algorithm for computing the relaxation parameter $\tilde{\gamma}_n$ [14,34].

3. Software implementation

The SSDC solver used in this work is being developed in the Advanced Algorithms and Numerical Simulations Laboratory (AANSLab), which is part of the Extreme Computing Research Center (ECRC) at King Abdullah University of Science and Technology (KAUST). The SSDC framework is built on top of the highly-scalable Portable and Extensible Toolkit for Scientific computing (PETSc) [35], its mesh topology abstraction (DMPlex) [36], and its scalable differential–algebraic equation solver components [37]. The spatial discretization features *hp*-adaptive capabilities on unstructured quadrilateral/hexahedral meshes. Support for nonconforming meshes relies on the p4est software library [38,39] and its bridge to PETSc's DMPlex [40]. Leveraging the capabilities of the PETSc library allows support for different mesh formats including fluent, Exodus II, CGNS and GMSH. Triangle/tetrahedral meshes are converted on the fly into quadrilateral/hexahedral elements; uniform and non-uniform mesh refinements algorithms are also available.

4. HPC cluster's architectures

Amazon EC2 offers a variety of instance types that are suitable for a wide range of applications. It is possible to mix and match CPU, memory, storage, and networking resources by using different instance types. Depending on the workload, we can choose from a variety of instance sizes available for each type of instance. AWS Cloud provides highly flexible computing platforms that are suited for performing HPC applications in terms of resource availability and configurability. CFD usually necessitates a large amount of computer resources, and its software architecture makes it suited for parallel processing; this requires the use of an HPC or cloud computing infrastructure. In order to deal with parallel computation needs, an adequate resource management system is also necessary. AWS ParallelCluster is one of the possible solutions to the aforementioned requirements. Here, we will briefly describe the compute environments of Amazon EC2 cloud and on-premise resource Ibex cluster.

4.1. Ibex cluster

Ibex is a heterogeneous cluster composed of nodes with various CPU architectures and an assortment of GPUs hosted at KAUST. Ibex is made up of more than 400 nodes that are constantly monitored by the systems team. This heterogeneous cluster has a mix of CPU architectures, including Intel Xeon Platinum 8000 series (Skylake-SP) 1st generation, Intel Xeon Platinum 8276M (Cascade Lake-SP) 1st generation, AMD EPYC 7642 (AMD Rome), and various GPUs nodes with P100s, V100s, GTX1080Ti, and RTX2080Ti, and with variable amounts of RAM ranging from 360 GB to 700 GB. The network connectivity on Ibex depends on InfiniBand HDR Director Switch. It has the following features and benefits: Mellanox HDR Infiniband is capable of 200 Gbps, it uses in compute and storage nodes with HDR-100 at 100 Gbps speed. Table 1 summarizes the main features of the Ibex's architectures used in this work. We highlight that Ibex has a daily occupancy of about 75%, with Intel architectures in high demand.

Table 1
Details of the Ibex's architectures.

Model	# CPUs	Memory (GiB)	Network bandwidth (Gbps)
Intel Xeon Platinum 8000	40	350	Up to 100
Intel Xeon Platinum 8276M	40	350	Up to 100
AMD EPYC 7642	128	375	Up to 100

Table 2
Amazon EC2 instances feature details.

Model	# vCPUs	Memory (GiB)	Network bandwidth (Gbps)
c5d.4xlarge	16	32	Up to 10
c5d.9xlarge	36	72	10
c5a.4xlarge	16	32	Up to 10
c5a.8xlarge	32	64	10
c6 g.4xlarge	16	32	Up to 10
c6 g.8xlarge	32	64	12

4.2. Amazon ParallelCluster

AWS ParallelCluster is an open source cluster management tool that is supported by AWS to deploy and manage HPC clusters in the AWS cloud. It enables the operation of HPC clusters in the AWS Cloud Environment and provides a wide range of configuration options. AWS ParallelCluster offers a variety of batch schedulers for managing nodes, resources, and parallel workloads. Users can use them to run large-scale scientific and engineering tasks. AWS cloud has many data centers throughout the world. In this work, all the experiments were conducted on Amazon's cloud data center in US East region in Northern Virginia. Our assessments cover the possible instance types which are similar to those available on KAUST's Ibex cluster described in Section 4.1 with in addition one Arm architecture as described at the end of this section. Compute Optimized instances are suited for compute intensive applications that benefit from powerful CPUs. In this work, we have built AWS ParallelCluster and we included the following compute-optimized instances: C5d, C5a, and C6 g instances. This family's instances are ideal for HPC. C5d will run on the Intel Xeon Scalable Processor 2nd generation or the Intel Xeon Platinum 8000 series (Skylake-SP) 1st generation with a sustained turbo frequency of up to 3.4 GHz and a single core turbo frequency of up to 3.5 GHz using Intel Turbo Boost Technology at launch. C5a instances provide the best x86 price/performance for a wide range of compute-intensive tasks, using 2nd generation 3.3 GHz AMD EPYC 7002 series processors built on a 7 nm technology node for greater efficiency. C5a instances can deliver up to 20 Gbps of network capacity as well as 9.5 Gbps of dedicated bandwidth to Amazon. EC2 C6 g instances are powered by AWS Graviton2 processors based on the Arm architecture. Table 2 shows the instances feature details.

5. Performance of the SSDC solver on Ibex and AWS EC2 clusters

The purpose of this study is to analyze the performance of the SSDC solver on Ibex and Amazon EC2 clusters for complex flow problems and provide the current cost per core hour. The default number of CPU cores for an EC2 instance depends on its type. We only allowed the instance to use the available physical CPU cores. This means we disabled hyperthreading. In the following sections, we report the arithmetic average of the wall-clock time (WCT) in seconds of three independent runs and the associated estimated cost in United States Dollars (USD).

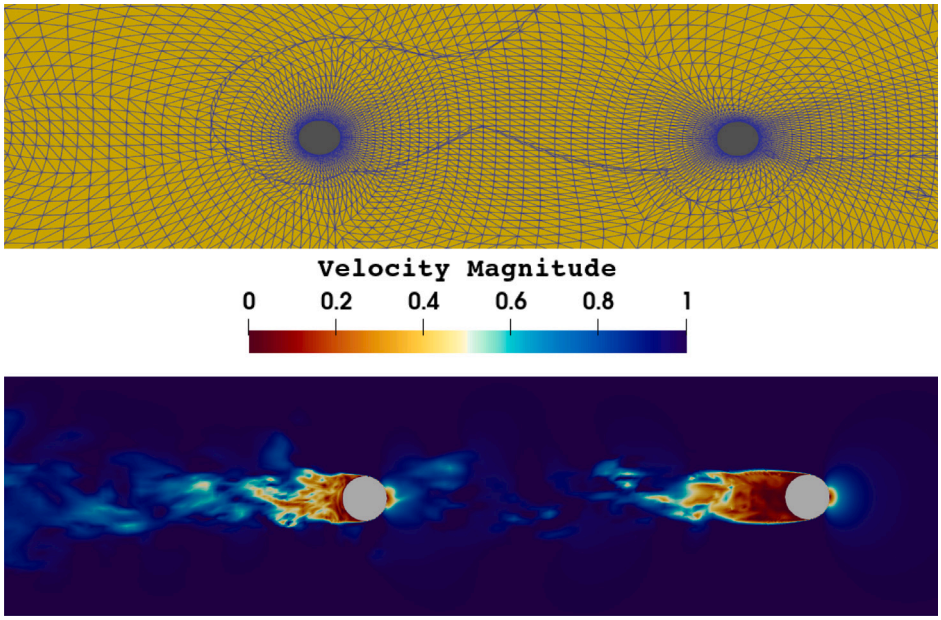
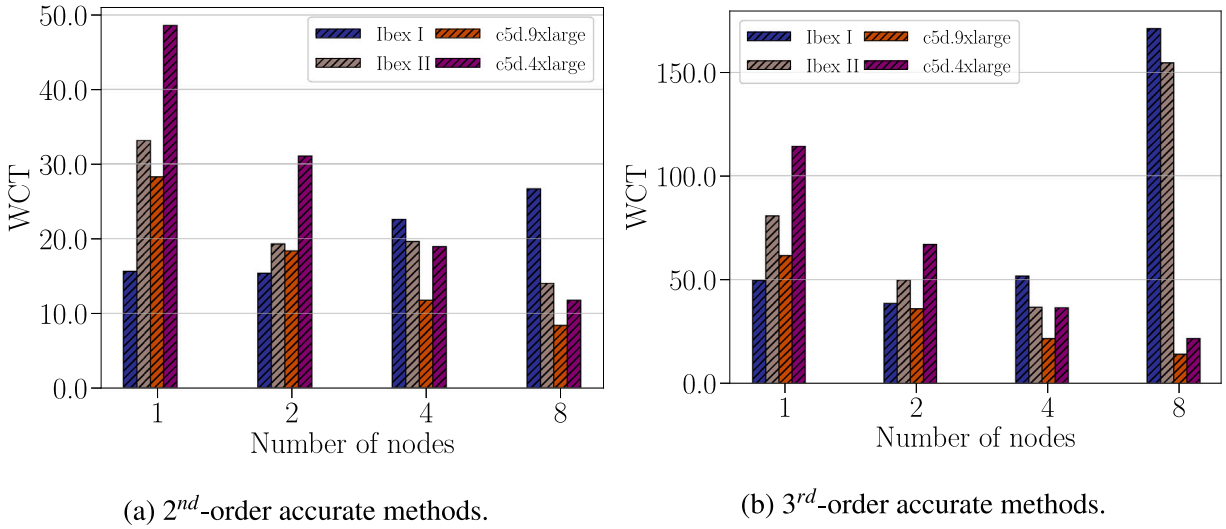
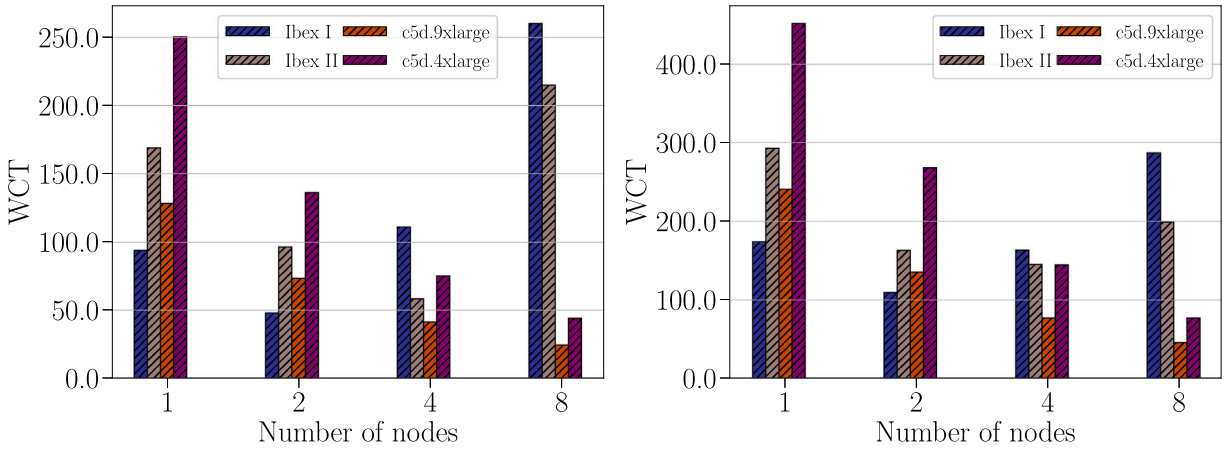


Fig. 1. Upper panel: Illustration of the mesh. Lower panel: Contour plots of the velocity magnitude for the flow past two identical spheres in tandem at time $t = 100$.



(a) 2^{nd}-order accurate methods.

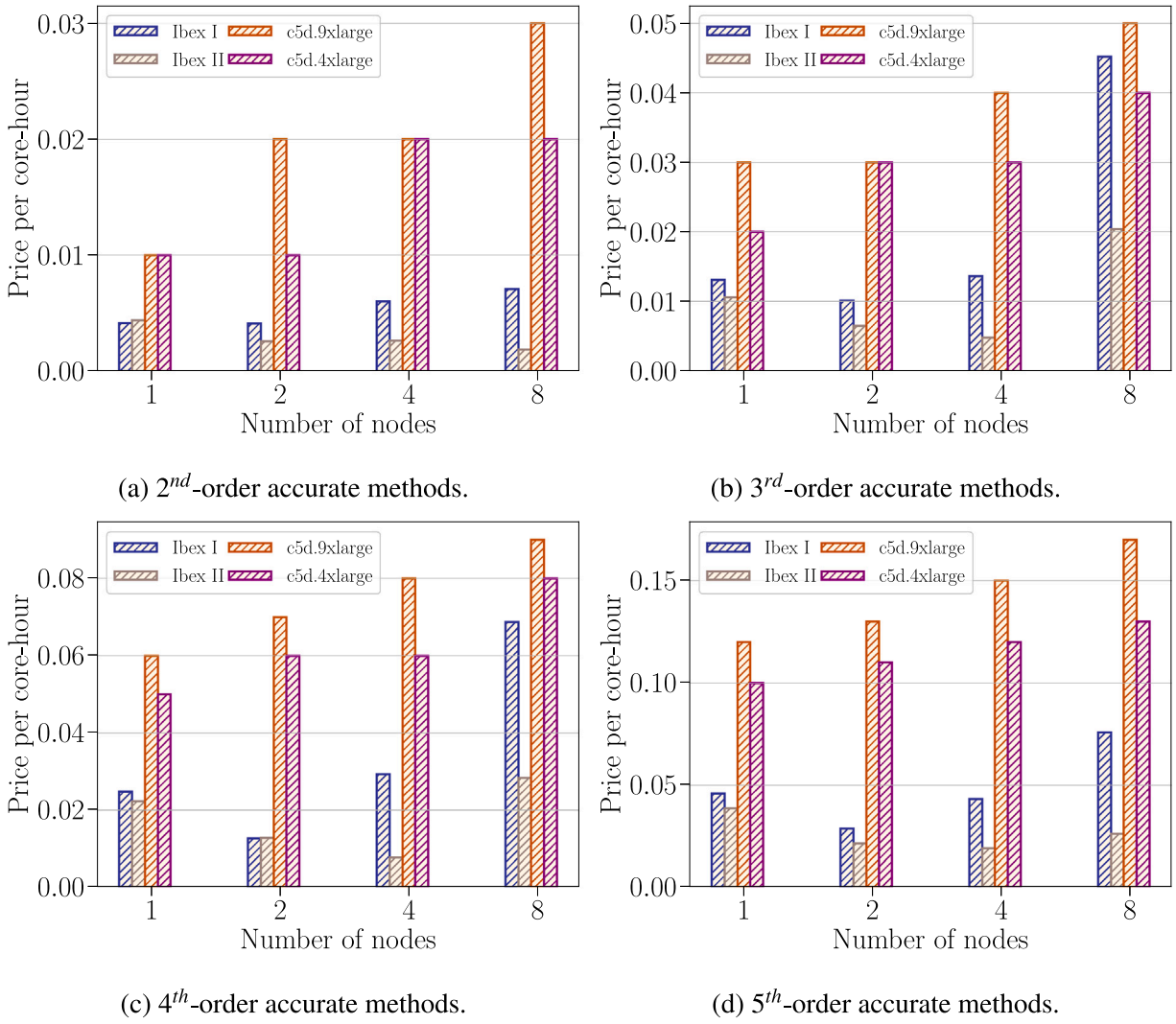
(b) 3^{rd}-order accurate methods.



(c) 4^{th}-order accurate methods.

(d) 5^{th}-order accurate methods.

Fig. 2. Flow past two spheres in tandem: Wall-clock time in seconds for each simulation against the number of nodes for Intel CPU architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) 2^{nd}-order accurate methods.

(b) 3^{rd}-order accurate methods.

(c) 4^{th}-order accurate methods.

(d) 5^{th}-order accurate methods.

Fig. 3. Flow past two spheres in tandem: Cost performance in USD of Intel CPU architectures.

5.1. Flow past two spheres in tandem

We simulate the flow past two equally sized spheres with a diameter D, held fixed in a rectangular domain located at a separation distance of 10D [41]. The Reynolds and Mach numbers are set to $Re_D = \mathcal{U}_\infty \mathrm{D}/\nu = 10^4$ and $Ma_\infty = 0.1$, respectively. The Prandtl number is set to $Pr = 0.7$. The quite upstream flow conditions of the first sphere are used to define the similarity parameters, e.g., $\mathcal{U}_\infty$ is the free-stream velocity.

Regardless of the geometry's simplicity, capturing the flow in this regime is relatively difficult. The relevance of such flows around several bodies, specifically around two spheres, is considered significant in many practical applications, as it allows a better understanding of the effect of the wake behind a leading bluff body on the flow around a trailing one, for instance. A non-exhaustive list of important applications ranging from industrial fluidized beds to bio-reactors, to the combustion of aerosols, could be liquid–gas two-phase flows [42], suppression of icing on the solid surface [43], and oil droplets [44]. The complexity of the flow fields is shown in the lower panel of Fig. 1 by plotting the velocity magnitude at time $t = 100$.

We perform the numerical simulations using one of the grids `TandemSpheresHexMesh2Pm` provided by Steve Karman of Pointwise for the HiOCFD5 [41]. An illustration of the grid structure is shown in the upper panel of Fig. 1. In this study, we use this mesh in combination with a solution polynomial order of $p = 10$ (leading to a formally eleventh-order accurate scheme), yielding a total number of degrees of freedom (DOFs) of $\approx 2.616 \times 10^7$.

Figs. 2 and 4 show the WCT in seconds for the Ibex cluster and the EC2 instances using various CPU architectures. On the on-premises Ibex cluster, we simulated the flow problems using 40 (dark blue — Ibex I) and 20 (gray — Ibex II) physical cores. Moreover, each processor is responsible for one MPI thread. The wall-clock time for an on-premises cluster based on Intel's Cascade Lake architecture grows proportionally with the node count, as shown in Fig. 2. This pattern holds in the 40 and 20 physical core setups. On-premise cluster performance degrades for several reasons, including the problem's size and the fact that the nodes are non-exclusive, with a daily occupancy of the Ibex cluster of approximately 75%. Thus, the more computing nodes, (1) the greater the likelihood that computing resources will be shared with other users, and (2) the more partitioned the job, the more likely it is that network bandwidth will be shared. We highlight that this behavior can also be observed in the case of exclusive access to the nodes. In fact, for a "sufficiently" large number of CPU cores count and hence, sufficiently smaller local problems, communication between partitions cannot be hidden behind computations. Thus, the solver's performance degrades and departs from the ideal behavior.

For the EC2 c5d instances and all orders of accuracy, the simulations are speeded up as the number of CPUs increases. Moving from one to

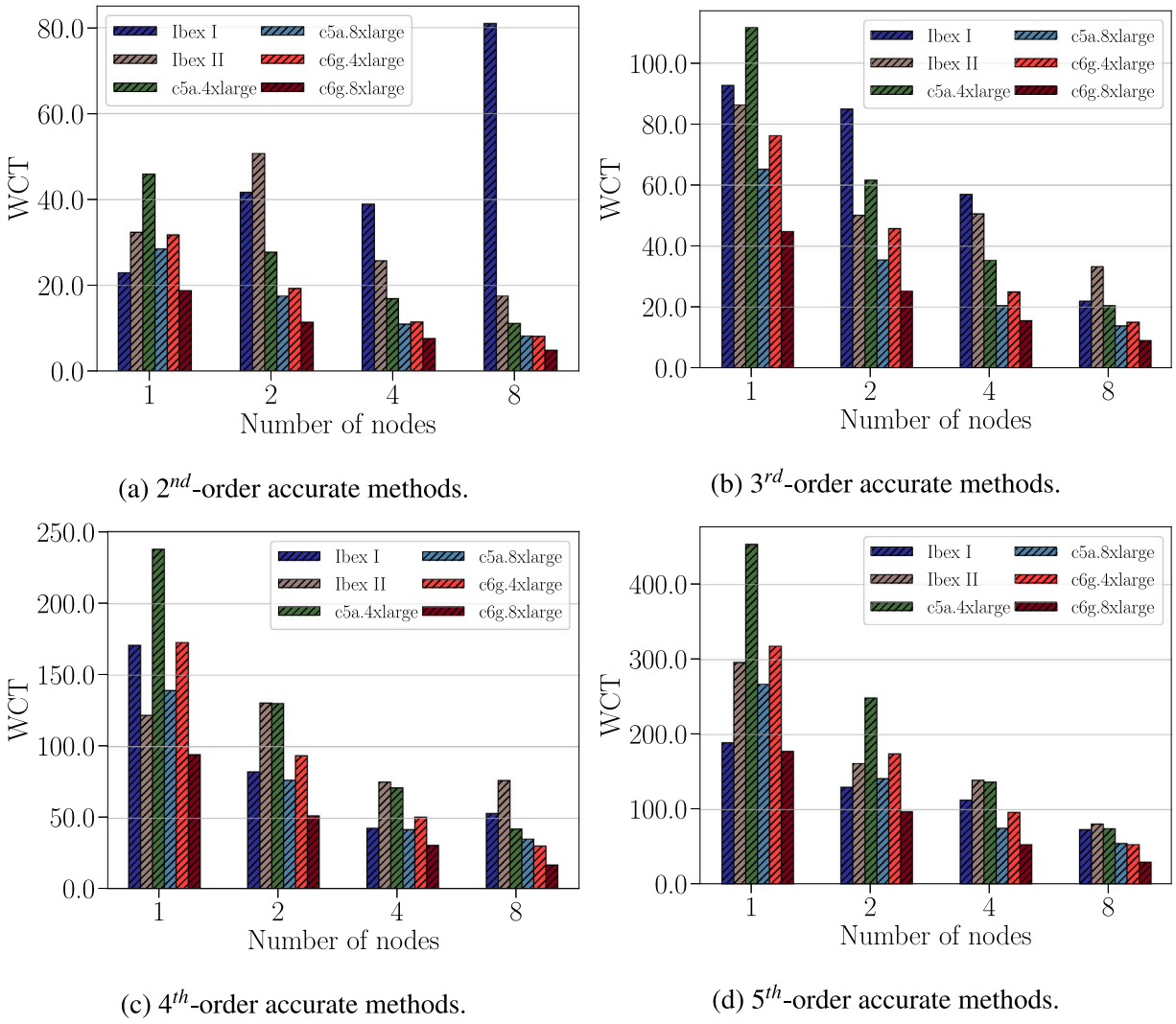


Fig. 4. Flow past two spheres in tandem: Wall-clock time in seconds for each simulation against the number of nodes for AMD & Arm CPU architectures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

eight computing nodes, in particular, results in a speed-up factor of about 5.6 for both cases (8 being the perfect scaling factor). Overall, the c5d.9xlarge EC2 instance delivers the shortest time-to-solution. Notably, although the on-premises cluster's wall-clock time increases as the number of compute nodes increases, up to two Ibex nodes running 40 and 20 MPI threads are still faster than EC2 instances. The executions on the on-premises cluster with 20 MPI threads are marginally faster or comparable to the executions on the c5d.4xlarge instance with four computing nodes. Therefore, it seems that utilizing half of a node's physically available cores has a favorable effect on wall-clock time. In fact, only half of the cores utilize the shared intra-node network, and more crucially, the workload per core permits better communications to be hidden behind computations.

As observed from Fig. 2, the results are favorable for simulations conducted on 8 EC2 nodes. In particular, the c5d.9xlarge EC2 instance delivers the results in the least amount of time. However, in terms of cost, Fig. 3 leads to a different conclusion: For any number of nodes and order of accuracy tested, it is cheaper to run on the Ibex cluster. Specifically, the simulations run with 20 physical cores cost at least two times less than those performed on AWS EC2 instances. Except for the third-order accurate solver on 8 nodes, running in an on-premises cluster is always advantageous, even when using the maximum number of available physical cores (*i.e.*, 40).

For the on-premises cluster with the AMD Rome architecture, the performance results are shown in Fig. 4. We observe that for the second-order accurate solver, the wall-clock time oscillates and does increase drastically when 8 nodes, and 40 cores per node are used. This is again the combined effect of the problem size and the network connectivity shared with other users — a common scenario for a parallel cluster. On the contrary, for the third-, fourth-, and fifth-order accurate solvers, we observe that by increasing the number of nodes, the wall-clock time decreases substantially. Increasing the number of nodes of EC2 AMD EPYC (*i.e.*, c5a instances) also leads to a reduction in time-to-solution. While the WCT decreases by around 35%-to-40% when moving from the c5a.4xlarge to the c5a.8xlarge. Thus, more on-chip cores are beneficial for this problem size. Fig. 4 also exhibits the performance of the Arm-based AWS Graviton2 processors (*i.e.*, c6 g instances). As we can observe, the c6 g instances perform well and deliver the least wall-clock time. Additionally, the time to solution is roughly halved when the nodes are doubled—indicating that a nearly perfect scaling is achieved. Furthermore, using four Arm nodes, the time-to-solution is nearly identical to that required by eight AMD EPYC nodes. Nevertheless, as shown in Fig. 4, the WCT on the Arm-based architecture c6 g.8xlarge and the Ibex cluster with eight nodes using 40 MPI processes per node are almost identical. Thus, the AMD

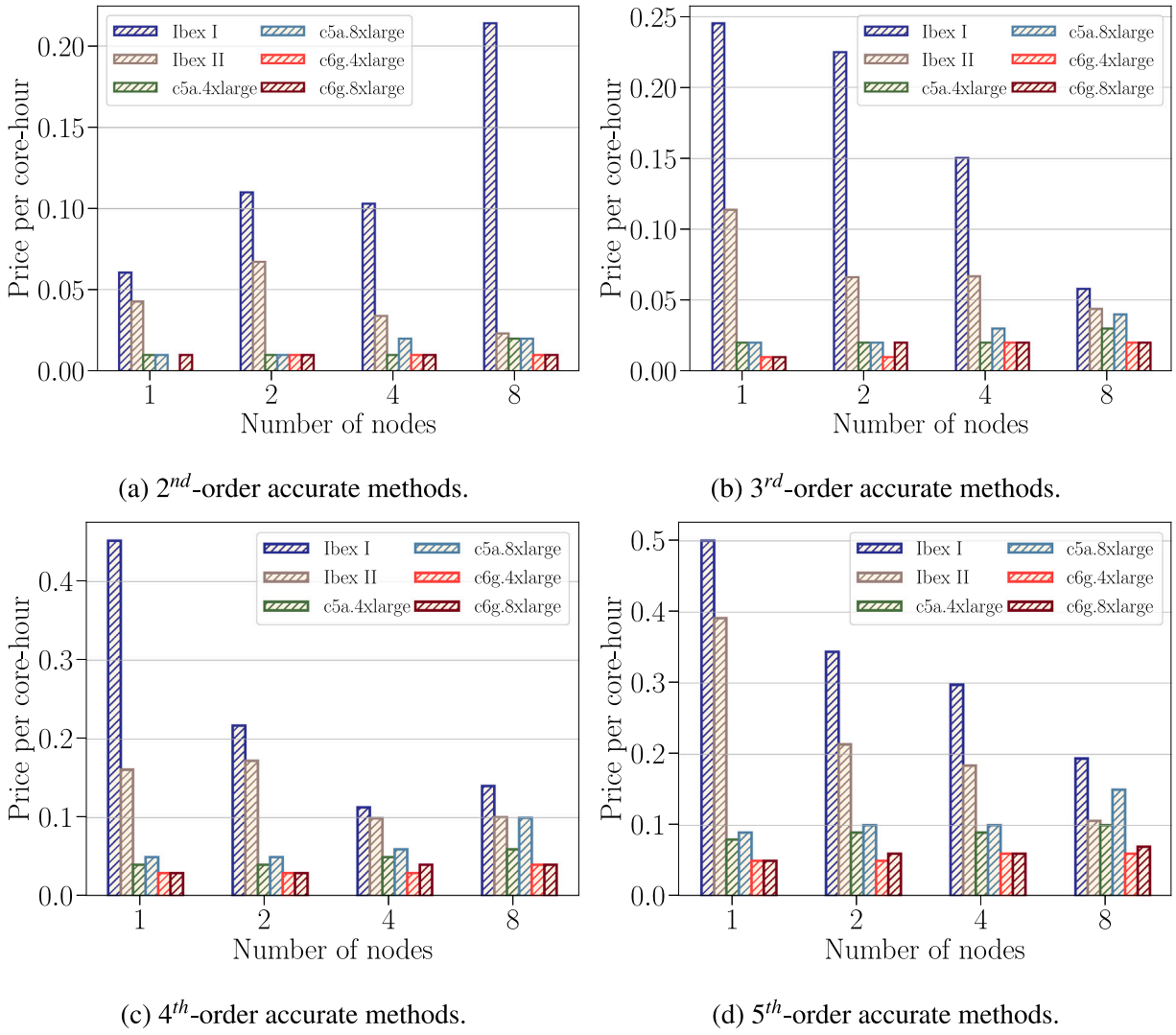


Fig. 5. Flow past two spheres in tandem: Cost performance in USD of AMD & Arm CPU architectures.

Rome nodes of the on-premises cluster can deliver similar performance compared to the c6 g.8xlarge instance.

In Figs. 5, we show the cost of simulating the test case with AMD and Arm architectures. Compared to the AWS ParallelCluster, the Ibex cluster offers significantly more affordable computations. While between all AWS EC2 instances, the Arm nodes are the least expensive for c6 g.4xlarge and c6 g.8xlarge. Finally, it is worth noting that the cost and performance of the EC2 c6 g instances are similar to those of the AWS EC2 c5d instances for all orders of accuracy.

5.2. Flow around delta wing

We study the flow around a 65° swept delta wing. We use the geometry reported by Hummel and Redeker [45] for the Second International Vortex Flow Experiment. In this work, we use the medium radius leading edge configuration, $r_{LE}/\bar{c} = 0.0015$, where $\bar{c} = 0.653$ m. The delta wing has a mean aerodynamic chord of $\ell = 0.667$ m, a root chord length of $c_r = 1.47\ell$, and a wing span of $b = 1.37\ell$. Furthermore, the central region is flat, and it has no twist or camber. A Cartesian coordinate system is positioned at the delta wing's apex with the x_1 coordinate pointing downstream, the x_2 coordinate pointing in a spanwise direction, and the x_3 coordinate perpendicular to the flat plate. We consider the sting as part of the setup up to the position $x_1/c_r = 1.758$.

The grids consist of $\approx 9.209 \times 10^4$ hexahedral cells. As shown in 6(a), the grid is divided into three blocks, where each block is assigned a different solution polynomial degree, p. Given the degree of the solution and the number of cells in each block, the number of DOFs is $\approx 1.435 \times 10^7$. The simulations are carried out for an angle of attack of AoA = 13°, a Mach number $Ma = 0.07$ and a Reynolds number of $Re = 10^6$, based on the mean aerodynamic chord.

The performance of the simulation for both on-premises cluster Ibex using Intel Cascade Lake architecture and AWS ParallelCluster using EC2 c5d instances are present in Fig. 7. The performance of the on-premise cluster is degraded as the number of cores increases. This is because of the interplays of two factors: the high occupancy and non-exclusivity of the Intel Cascade nodes on the Ibex cluster, and the considerable increase of the fraction of the communication time on the overall computational time for smaller and smaller local problems (communication cannot be hidden behind computations). When the size problem becomes "sufficiently" large, we observe that an increment of the number of nodes corresponds to a decrease in time-to-solution, as shown in Fig. 7(d). However, the performance degrades again for eight nodes, where we observe a growth of the time-to-solution. In contrast, for the AWS EC2 c5d instances, the simulation is speed-up as the number of nodes increases for all orders of accuracy.

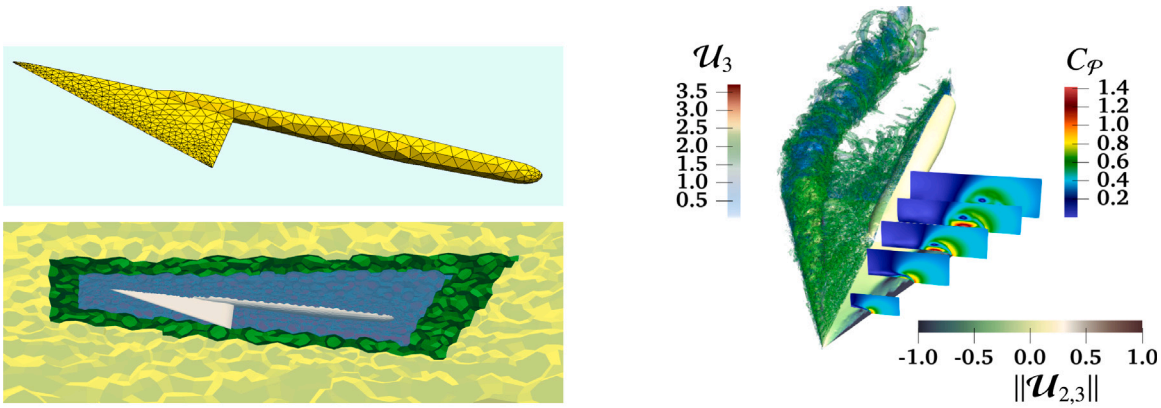


Fig. 6. (a) Geometry (top) and solution polynomial degree distribution (bottom) for the 65° swept delta wing test case; $p = 2$ in the far-field region (yellow), $p = 5$ in the region surrounding the delta wing and its support (blue), and $p = 3$ elsewhere (green). (b) Average flow field past the 65° swept delta wing: the Q-criterion colored by the normalized velocity magnitude (left) and mean axial velocity (right) at $x_1/c_r = 0.2, 0.4, 0.6, 0.8$, and 0.95; the wing surface is colored using the time-averaged pressure coefficient. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

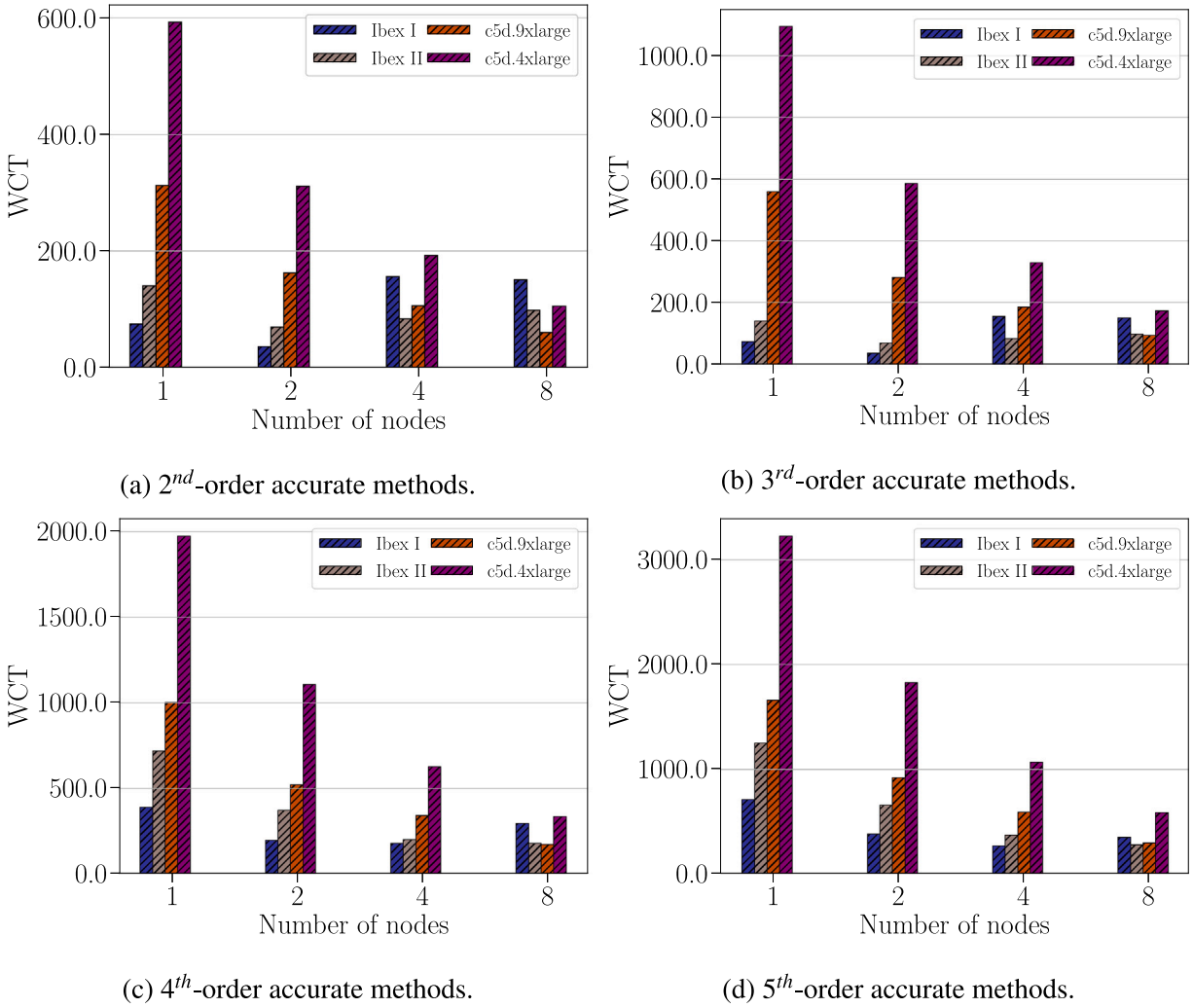


Fig. 7. Flow around a delta wing: Wall-clock time in seconds for each simulation against the number of nodes for Intel CPU architectures.

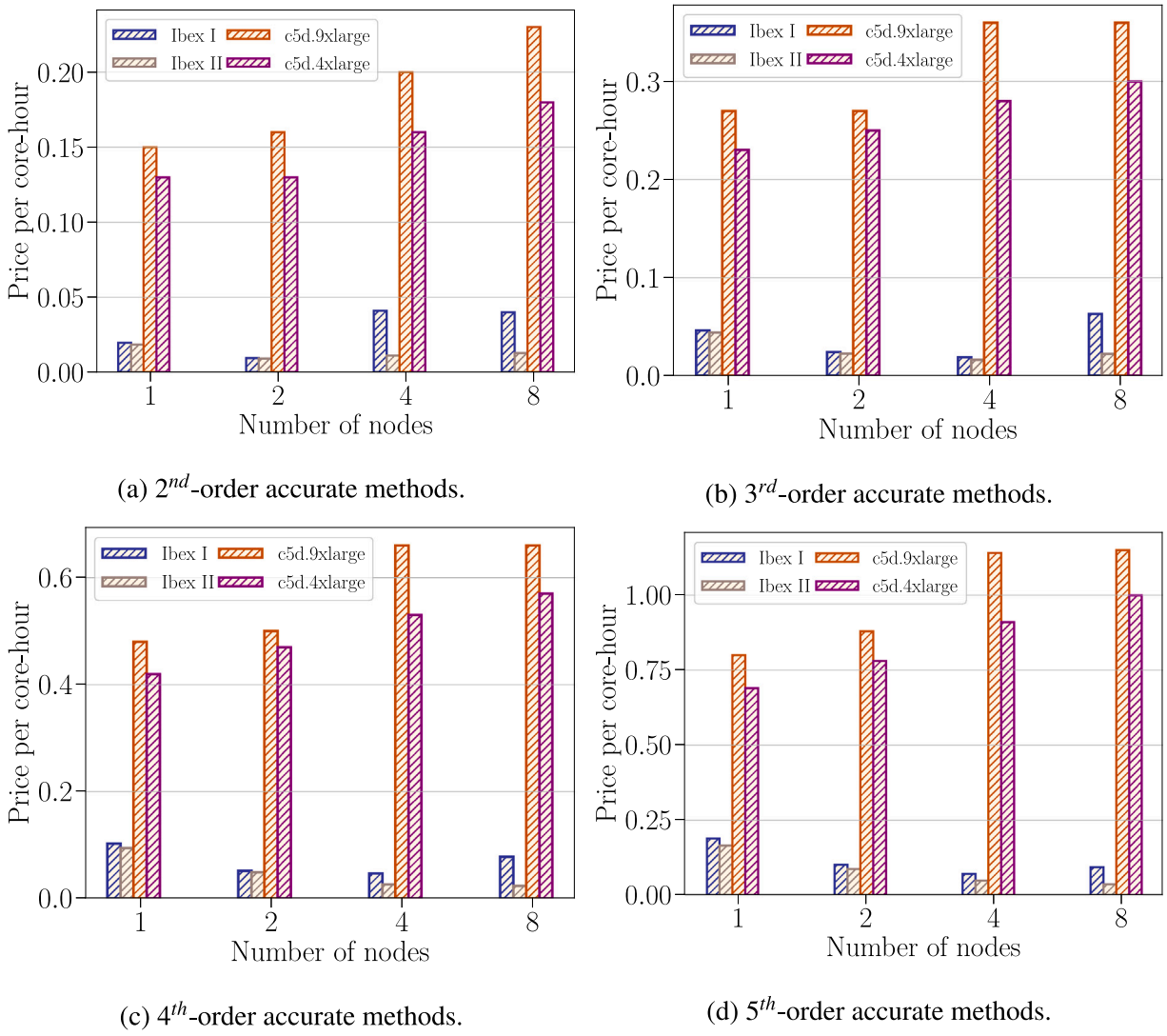


Fig. 8. Flow around a delta wing: Cost performance in USD of Intel CPU architectures.

In particular, a speed-up factor of about 1.8 is delivered for both AWS EC2 c5d instances.

Fig. 8 shows the cost analysis of running on Ibex and the c5d instances. The runs performed with the on-premises cluster using 40 and 20 MPI threads show better cost-efficiency than the AWS EC2 c5d instances. In fact, Ibex provides up to 70% better pricing performance than c5d instances, regardless of solver accuracy and the number of nodes.

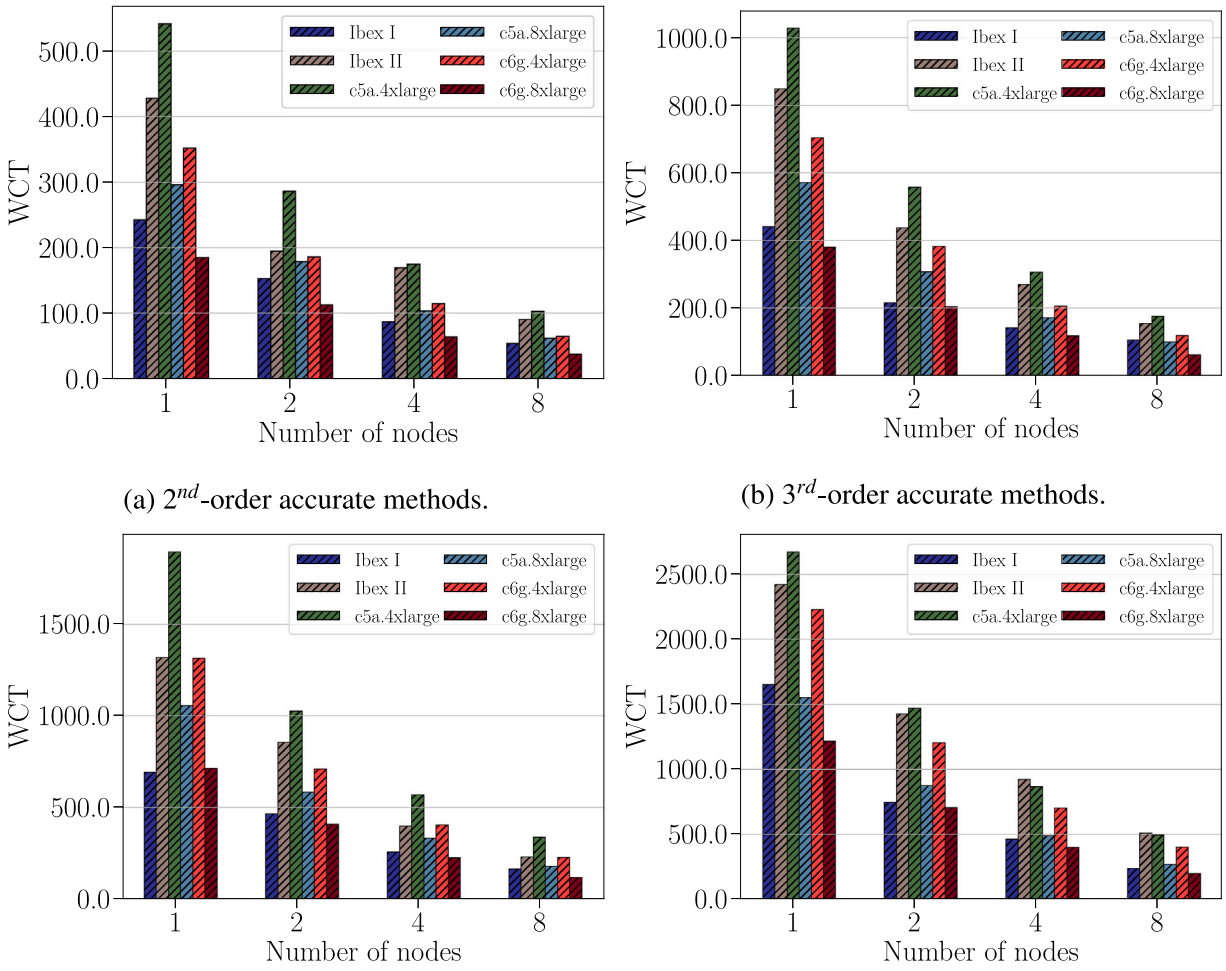
As done for the flow past the two spheres in tandem, we also study the performance of the solver and the cost of each simulation for the AMD and Arm architectures. Fig. 9 shows the wall-clock time in seconds for the on-premise cluster with the AMD Rome architecture, the EC2 AMD c5a instances, and the EC2 Arm c6 g instances. The performance results of the Ibex and AWS clusters is mostly matching the results obtained for the previous test case, *i.e.,* the wall-clock time decreases substantially by increasing the number of nodes. In particular, by doubling the number of nodes we observe a speed-up factor of about 1.7 for all the runs. As for the previous application, moving from the c5a.4xlarge to the c5a.8xlarge instances lead to a reduction in the wall-clock time by around 35%-to-40%. Furthermore, the smallest wall-clock time is obtained using the EC2 Arm c6 g instances, for all orders of accuracy. In addition, almost perfect scaling is observed. Doubling the nodes by switching from c6 g.4xlarge to c6 g.8xlarge leads to halves the solution time.

A detailed cost analysis is shown in Fig. 10. Among the EC2 instances, the compute-optimized c6 g instances perform better than the c5a instances for all the number of nodes and order of accuracy. However, on the on-premises cluster, using the largest number of physical cores possible is always the most convenient solution.

In the next section, we further assess the performance of the solver on-premises cluster Ibex and the AWS ParallelCluster by considering two more complex industrially-relevant flow problems. Precisely, we will simulate the NASA junction flow experiment and the flow past a Formula (1) front wing. The accurate simulation of these industrial problems via large eddy simulation (LES) is representative of the type of simulations that exascale will allow performing in a 24-hour turnaround, a typical requirement for industry standards (see, for instance, [46]). In our context, we use these test cases to explore the influence of the problem size on the performance of both clusters.

5.3. NASA juncture flow experiment

In this section, we simulate the NASA juncture configuration. This test case has a wing based on the DLR-F6 geometry and is equipped



(a) 2^{nd}-order accurate methods.

(b) 3^{rd}-order accurate methods.

(c) 4^{th}-order accurate methods.

(d) 5^{th}-order accurate methods.

Fig. 9. Flow around a delta wing: Wall-clock time in seconds for each simulation against the number of nodes for AMD & Arm CPU architectures.

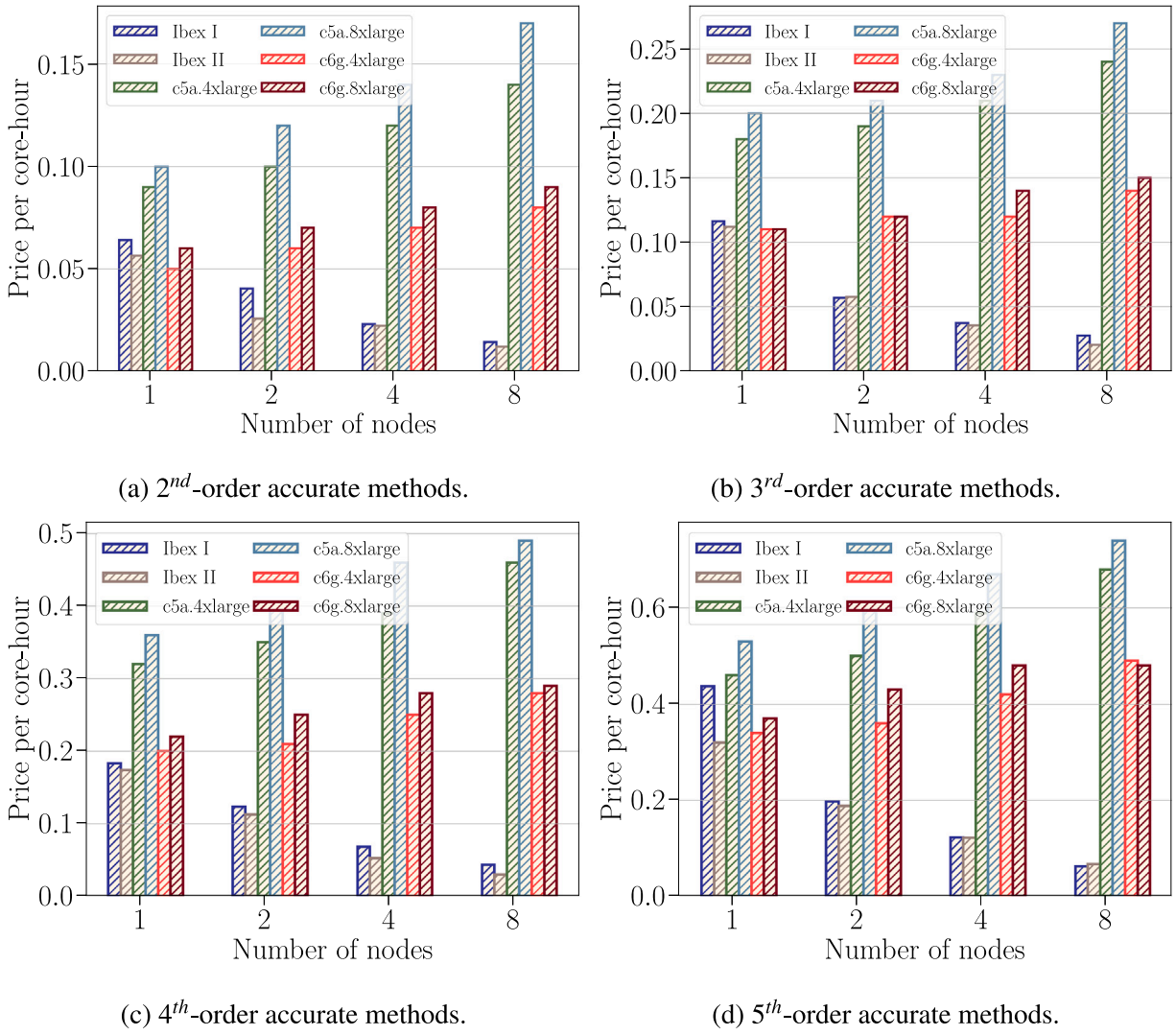
with a leading edge horn to reduce the effect of the horseshoe vortex around the wing-fuselage juncture [47]. The Reynolds and Mach numbers based on the freestream conditions are 2.4×10^6 and 0.189, respectively. The angle of attack is $-2.5°$. We perform simulations by neglecting the sting and the mast. An overview of the mesh is shown in Fig. 11(a). The grid is broken up into three blocks, each of which represents a different degree of approximation (p) for the solution field. Specifically, we utilize $p = 1$ in the far-field region (dark green), $p = 3$ in the region surrounding the model (dark orange), and $p = 2$ in the remaining part (dark yellow). The total number of hexahedral elements is $\approx 6.762 \times 10^5$, and the number of DOFs is $\approx 4.091 \times 10^7$. In Fig. 11(b), we plot the Q-criterion using isocontours colored by normalizing the instantaneous velocity, *i.e.*, $\mathcal{U}_1/|\mathcal{U}_\infty|$.

In Fig. 12, we show the WTC for different numbers of nodes for the Ibex cluster and the AWS EC2 instances. For all the type of nodes except the Intel Cascade Lake, doubling the number of nodes yields an efficiency of about 95%. For the Intel Cascade Lake architecture with 40 MPI threads, moving from one to four nodes leads to a rapid decrease in efficiency. Eventually, for eight nodes, the time to solution increases. The fastest time-to-solution is delivered by the Cascade Lake with 20 MPI threads per node and the c6g8xlarge AWS EC2 instance. In terms of cost, the on-premise cluster simulations done on Intel architectures are much cheaper than the AWS EC2 cd5 instances, as illustrated in Fig. 13. For the AMD-Ibex nodes and the c5a and c6 g instances, the on-premises cluster still delivers the smallest cost.

5.4. Flow past a Formula (1) front wing

Here, we consider the flow past a Formula (1) front wing [48]. We refer to this test case as the Imperial Front Wing, based on the front wing and endplate design of the McLaren 17D race car [49]. We denote by h the distance between the ground and the lowest part of the front wing endplate and by c the chord length of the main element. The position of the wing in the tunnel is further characterized by a pitch angle of 1.094°. Here we use $h/c = 0.36$ which can be considered as a relatively low front ride height, with high ground effect and hence higher loads on the wing. The corresponding Reynolds number is $Re = 2.2 \times 10^5$, based on the main element chord c of 250 mm and a free stream velocity U of 25 m/s. The Mach number is set to $Ma = 0.036$. This corresponds to a practically incompressible flow.

The computational domain is divided into 3.4×10^6 hexahedral elements with a maximum aspect ratio of approximately 250. The solution polynomial degree is set to $p = 2$ — a formally third-order accurate scheme. Thus, the total number of DOFs is approximately 9.18×10^7. The grid is constructed using the commercial software Pointwise V18.3 released in September 2019; solid boundaries are described using a quadratic mesh. The panel of Fig. 14(a) illustrates an overview of the front wing geometry and the mesh, where the contour plot of the time-averaged pressure coefficient on the surface of the front wing is shown in Fig. 14(b).



(a) 2^{nd}-order accurate methods.

(b) 3^{rd}-order accurate methods.

(c) 4^{th}-order accurate methods.

(d) 5^{th}-order accurate methods.

Fig. 10. Flow around a delta wing: Cost performance in USD of AMD & Arm CPU architectures.

The variation of the wall-clock time against the number of nodes for different types of CPU architecture is shown in Fig. 15. Overall, we observe that the simulations speed up as the number of CPUs increases for all architectures on the on-premises cluster and the AWS EC2 instances. These results confirm that the problem size influenced the performance of the on-premises cluster Ibex presented in the previous sections. Fig. 15(a) shows the wall-clock time used by the Intel Cascade Lake architecture on the Ibex cluster and the AWS EC2 c5d instances. The simulation on 8 nodes of the on-premises cluster with 40 MPI threads produced favorable results. For all four computations, doubling the number of nodes reduces the time-to-solution by a factor of approximately 1.8. However, for 8 nodes Intel Cascade Lake architecture and 40 MPI threads, this factor reduces to 1.7. In Fig. 15(b), we report the performance of the Ibex cluster using AMD processes and AWS ParallelCluster using c5a and c6 g instances. As we can observe, the c6 g.8xlarge AWS EC2 instance delivers the results in the least amount of time with a parallel efficiency of approximately 70%. By comparing Figs. 15(a) and 15(b), we observe that the c6 g.8xlarge AWS EC2 instance achieve the best performance among all the CPU architectures.

Although parallel performance is important, the cost of the simulations is a significant concern. As illustrated in Fig. 16, the cost of computations on the on-premises cluster is much lower than that of the AWS ParallelCluster. Across all the AWS EC2 instances, the c6 g.8xlarge Arm architecture is the cheapest one.

6. Conclusion

In this work, we evaluate the performance of a prototype of next generation high-order entropy stable solvers for compressible flows on unstructured grids on the Amazon Web Services Elastic Compute Cloud and the on-premise resource Ibex cluster hosted at KAUST. The study aims to establish the possibility of using Amazon's cloud-based high-performance computing service to address complex computational fluid dynamics industry flow problems and propose a set of Elastic Cloud Computing instances that provide the fastest time to the solution and offer more affordable computations. In terms of time-to-solution, the Amazon Web Services Elastic Compute Cloud delivers the best performance, with the Graviton2 processors based on the Arm architecture being the fastest. However, the results also indicate that the nodes based on the AMD Rome architecture of Ibex deliver very good performance, close to those observed for the Amazon Cloud service. In addition, we found that the simulations on the Ibex cluster are currently less expensive than those performed on the cloud for all orders of accuracy. Future work will demonstrate the performance at higher core counts and include post-processing elements of the computational fluid dynamics process.

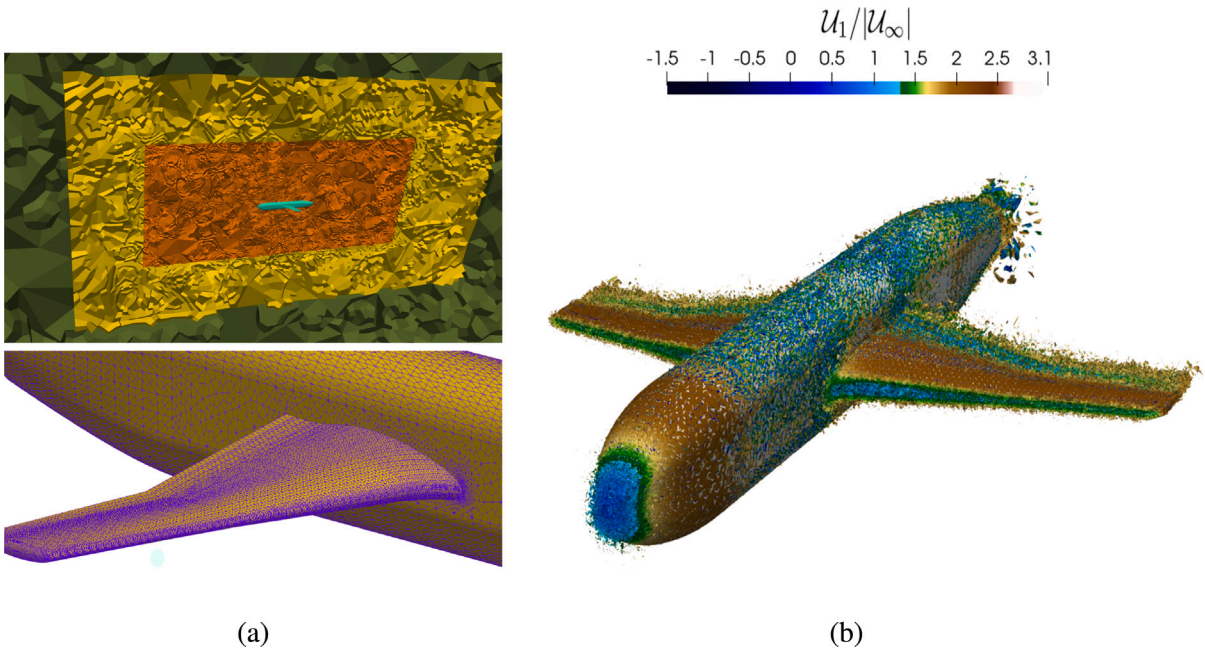
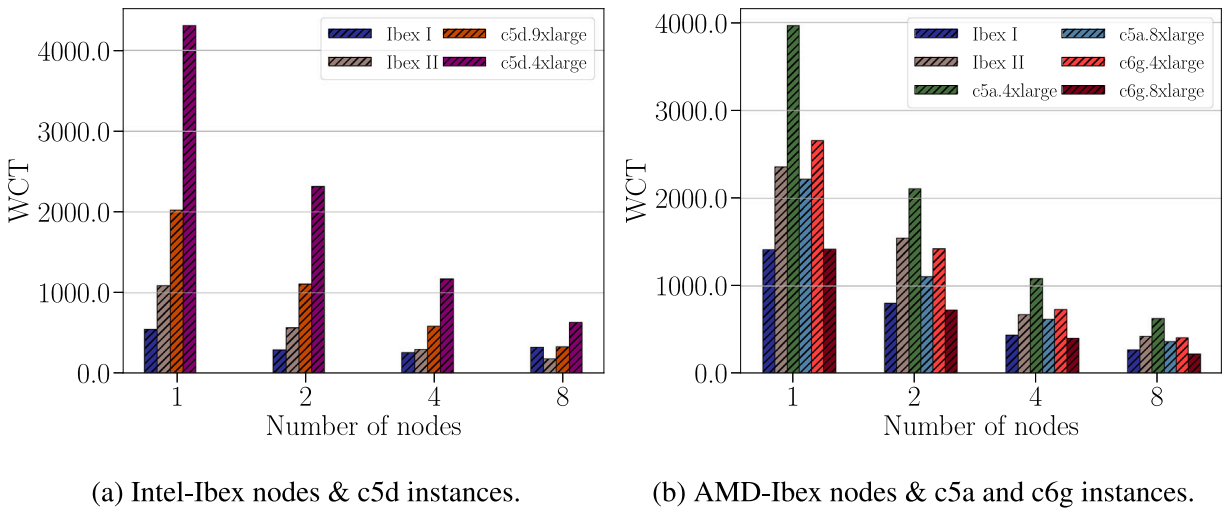


Fig. 11. NASA juncture flow experiment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Intel-Ibex nodes & c5d instances. (b) AMD-Ibex nodes & c5a and c6g instances.

Fig. 12. NASA juncture flow experiment: Wall-clock time in seconds for each simulation against the number of nodes.

CRediT authorship contribution statement

R. Al Jahdali: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **S. Kortas:** Resources, Software. **M. Shaikh:** Resources, Software. **L. Dalcin:** Resources, Software, Review & editing. **M. Parsani:** Conceptualization, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

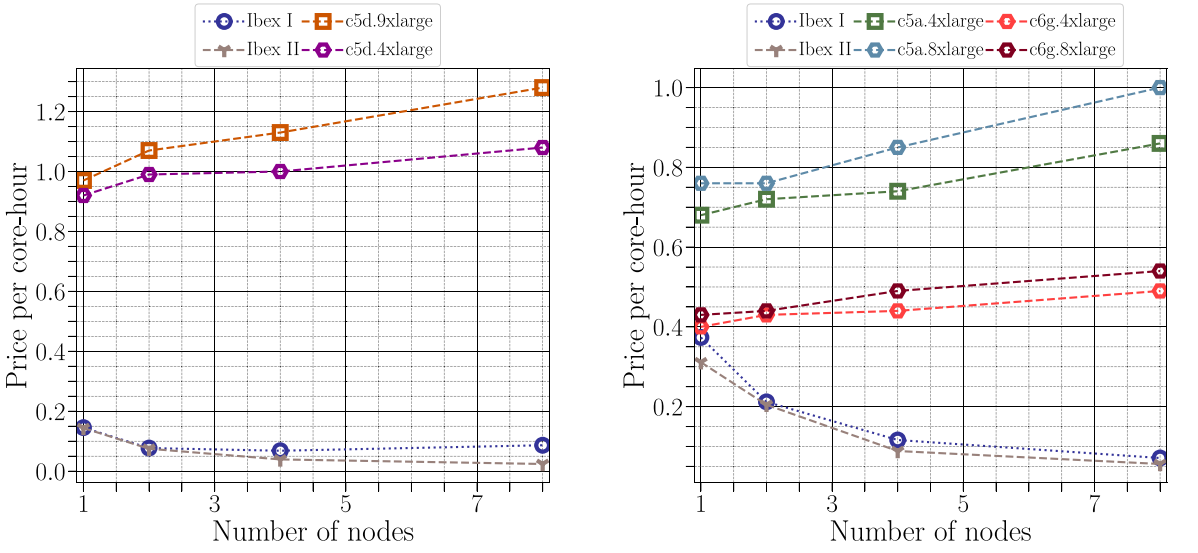
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

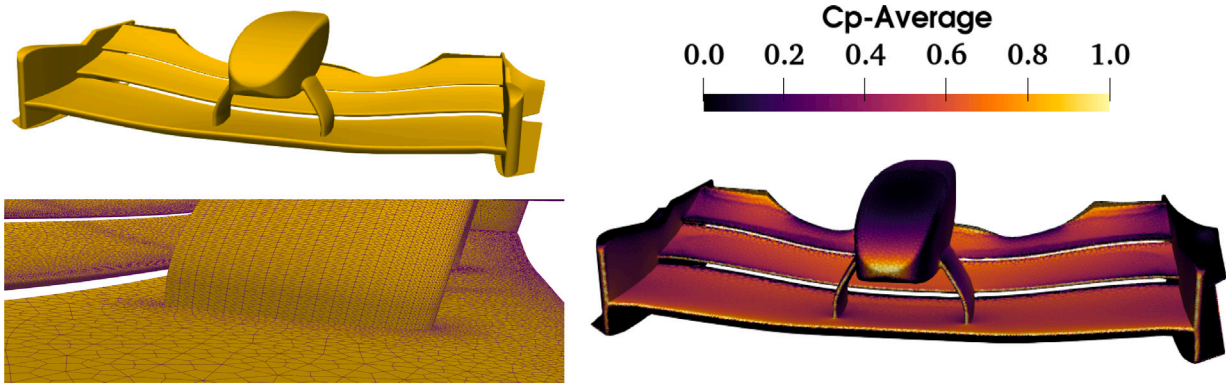
Acknowledgments

The work described in this paper was supported by King Abdullah University of Science and Technology, Saudi Arabia through the award OSR-2019-CCF-3666. The authors are also thankful for the computing resources of the Supercomputing Laboratory and the Extreme Computing Research Center at King Abdullah University of Science and Technology.



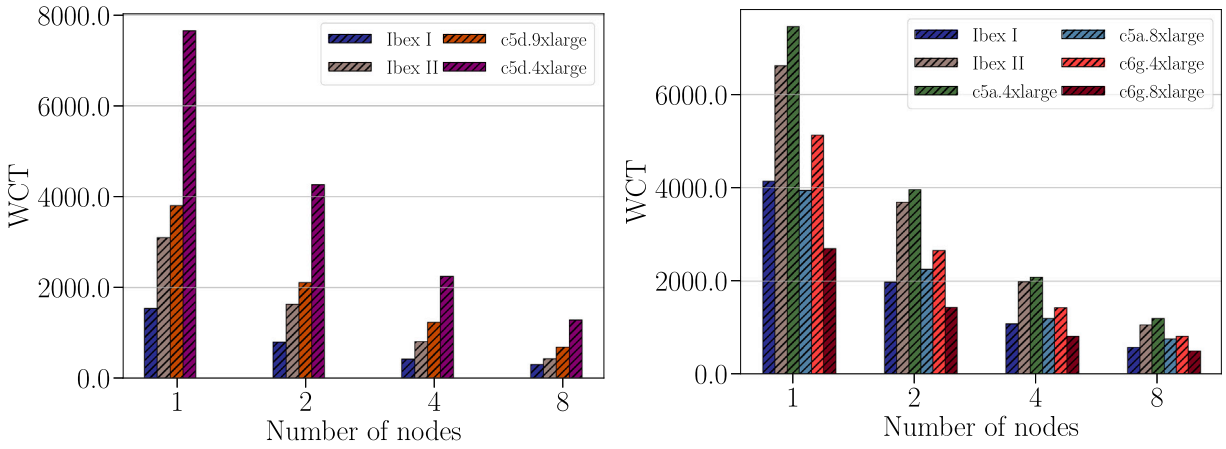
(a) Intel-Ibex nodes & c5d instances. (b) AMD-Ibex nodes & c5a and c6g instances.

Fig. 13. NASA juncture flow experiment: Cost performance in USD of Ibex, and AWS EC2 instances.



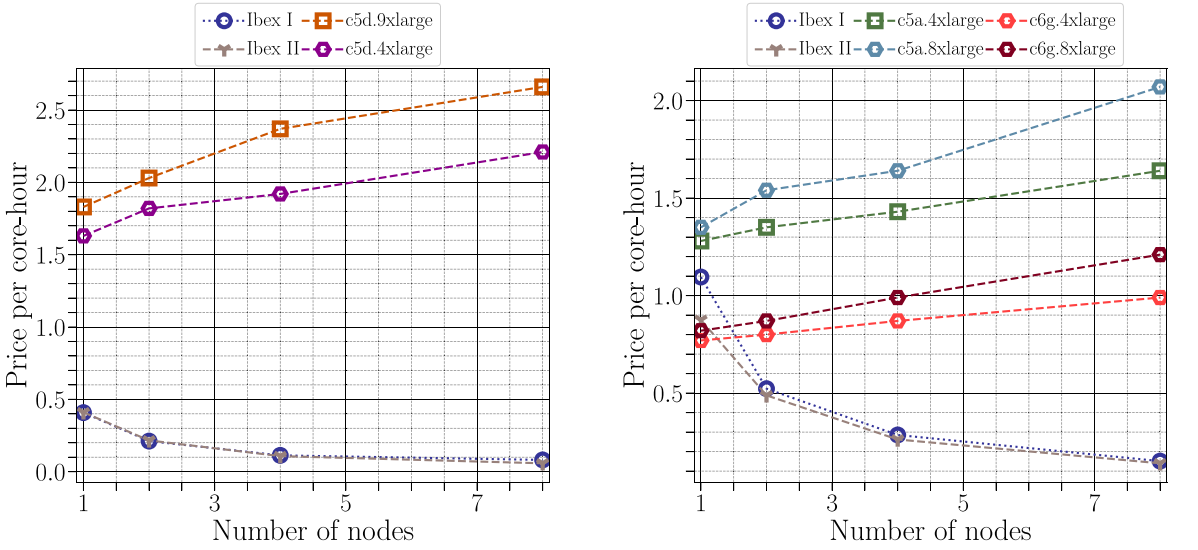
(a) (b)

Fig. 14. Formula (1) front wing (Imperial Front Wing).



(a) Intel-Ibex nodes & c5d instances. (b) AMD-Ibex nodes & c5a and c6g instances.

Fig. 15. Formula (1) front wing: Wall-clock time in seconds for each simulation against the number of nodes.



(a) Intel-Ibex nodes & c5d instances.

(b) AMD-Ibex nodes & c5a and c6g instances.

Fig. 16. Formula (1) front wing: Cost performance in USD of Ibex, and AWS EC2 instances.

Appendix A. An overview of the semidiscrete entropy stable spatial discretization

This Appendix gives some of the details of the spatial discretization implemented in the SSDC framework. As a model problem, we use the advection-diffusion equation in multiple dimensions.

A.1. Mapping

In general, to solve partial differential equations numerically, we partition the physical domain $\Omega \subset \mathbb{R}^3$ into K non-overlapping elements, where K is a whole number greater than zero. Then, each element in physical space is transformed using a local and invertible curvilinear coordinate transformation that is compatible at shared interfaces, meaning that the push-forward element-wise mappings are continuous across physical element interfaces. To achieve that, one maps from the reference coordinates $(\xi_1, \xi_2, \xi_3) \in [-1, 1]^3$ to the physical element (see Fig. A.17) by the push-forward transformation $(x_1, x_2, x_3) = X(\xi_1, \xi_2, \xi_3)$, which, in the presence of curved elements, is usually a high-order degree polynomial. This procedure requires no explicit knowledge nor construction of the pull-back mappings in unstructured mesh schemes.

A.2. Summation-by-parts operators

The physical (spatial) domain $\Omega \subset \mathbb{R}^3$ with boundary $\partial\Omega$ is discretized using tensor-product elements. In this work, the derivatives in each element are discretized using one-dimensional SBP operators [50, 51] which are given for completeness in Definition 1.

Definition 1. A matrix operator, $\mathsf{D}_\xi \in \mathbb{R}^{N\times N}$, is an SBP operator of degree p approximating the derivative $\frac{\partial}{\partial\xi}$ on the domain $\xi \in [\alpha, \beta]$ with nodal distribution $\boldsymbol{\xi}$ having N_l nodes, if

1. $\mathsf{D}_\xi \boldsymbol{\xi}^j = j\boldsymbol{\xi}^{j-1}$, $j = 0, 1, \ldots, p$;
2. $\mathsf{D}_\xi \equiv \left(\mathsf{P}_\xi\right)^{-1} \mathsf{Q}_\xi$, where the norm matrix, P_ξ, is symmetric positive definite;

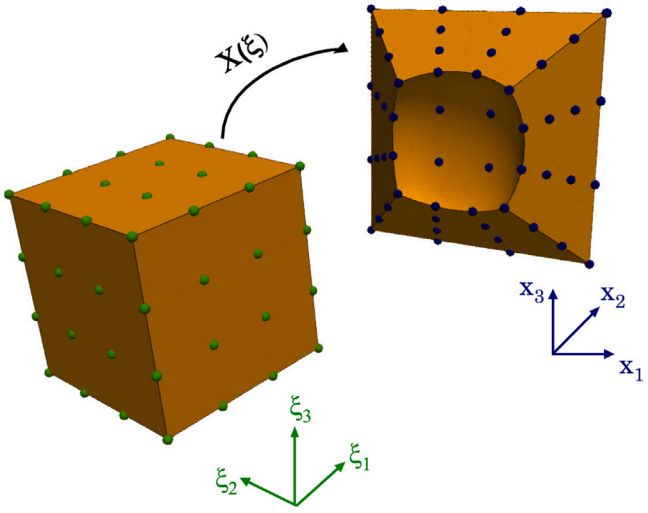


Fig. A.17. Example of mapping procedure of the reference element (left) to a physical element (right).

3. $\mathsf{Q}_\xi \equiv \left(\mathsf{S}_\xi + \frac{1}{2}\mathsf{E}_\xi\right)$, $\mathsf{S}_\xi = -\left(\mathsf{S}_\xi\right)^{\mathrm{T}}$, $\mathsf{E}_\xi = \left(\mathsf{E}_\xi\right)^{\mathrm{T}}$, $\mathsf{E}_\xi = \mathrm{diag}\,(-1, 0, \ldots, 0, 1) = \boldsymbol{e}_N \boldsymbol{e}_N{}^{\mathrm{T}} - \boldsymbol{e}_1 \boldsymbol{e}_1{}^{\mathrm{T}}$, $\boldsymbol{e}_1 \equiv [1, 0, \ldots, 0]^{\mathrm{T}}$, and $\boldsymbol{e}_N \equiv [0, 0, \ldots, 1]^{\mathrm{T}}$.

An SBP operator of degree p is hence an operator that differentiates exactly monomials up to degree p.

In this work, we use a collocated discontinuous Galerkin approach with diagonal norm SBP operators are constructed on the Legendre–Gauss–Lobatto (LGL) nodes [10]. The SBP operators used in this work are explicitly constructed in [22]. These operators are extended to multiple dimensions by using tensor products ($\otimes$).

A.3. Semidiscretization of the linear advection-diffusion equation

The linear convection–diffusion equation in Cartesian physical coordinates reads

$$
\begin{aligned}
&\frac{\partial \boldsymbol{q}}{\partial t}+\sum_{m=1}^{3} \frac{\partial\left(a_m \boldsymbol{q}\right)}{\partial x_m}=\sum_{m=1}^{3} \frac{\partial^2\left(b_m \boldsymbol{q}\right)}{\partial x_m^2}, \\
&\forall\left(x_1, x_2, x_3\right) \in \Omega, \quad t \geq 0, \\
&\boldsymbol{q}\left(x_1, x_2, x_3, t\right)=\boldsymbol{g}^{(B)}\left(x_1, x_2, x_3, t\right), \\
&\forall\left(x_1, x_2, x_3\right) \in \partial \Omega, \quad t \geq 0, \\
&\boldsymbol{q}\left(x_1, x_2, x_3, 0\right)=\boldsymbol{g}^{(0)}\left(x_1, x_2, x_3, 0\right), \\
&\forall\left(x_1, x_2, x_3\right) \in \Omega,
\end{aligned}
\tag{A.1}
$$

where $(a_m \boldsymbol{q})$ are the inviscid fluxes, a_m are the (constant) components of the convection speed, $\frac{\partial(b_m q)}{\partial x_m}$ are the viscous fluxes, and b_m are the (constant and positive) diffusion coefficients. If the coefficients b_m are set to zero, we will have a purely hyperbolic system. The boundary data, $\boldsymbol{g}^{(B)}$, and the initial condition, $\boldsymbol{g}^{(0)}$, are assumed to be in $L^2(\Omega)$, with the further assumption that $\boldsymbol{g}^{(B)}$ is prescribed so that linear stability (energy stability) is achieved. Here, derivatives are approximated with SBP differentiation operators defined in computational space. Therefore, we use the Jacobian of the push-forward mapping and the chain rule to transform Eq. (A.1) from physical to computational space as

$$
J_\kappa \frac{\partial \boldsymbol{q}}{\partial t}+\sum_{l,m=1}^{3} J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial\left(a_m \boldsymbol{q}\right)}{\partial \xi_l}=\sum_{l,a,m=1}^{3} J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial}{\partial \xi_l}\left(\frac{\partial \xi_a}{\partial x_m} \frac{\partial\left(b_m \boldsymbol{q}\right)}{\partial \xi_a}\right), \tag{A.2}
$$

where J_κ is the determinant of the metric Jacobian. Moving the metric terms $J_\kappa \frac{\partial \xi_l}{\partial x_m}$ inside the derivative, and using the product rule, leads to

$$
\begin{aligned}
&J_\kappa \frac{\partial \boldsymbol{q}}{\partial t}+\sum_{l,m=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} a_m \boldsymbol{q}\right)-\sum_{l,m=1}^{3} a_m \boldsymbol{q} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m}\right)= \\
&\sum_{l,a,m=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial \xi_a}{\partial x_m} \frac{\partial\left(b_m \boldsymbol{q}\right)}{\partial \xi_a}\right)-\sum_{l,a,m=1}^{3} \frac{\partial \xi_a}{\partial x_m} \frac{\partial\left(b_m \boldsymbol{q}\right)}{\partial \xi_a} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m}\right).
\end{aligned}
\tag{A.3}
$$

The last terms on the left- and right-hand sides of (A.3) are zero via the geometric conservation law (GCL) relations

$$
\sum_{l=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m}\right)=0, \quad m=1,2,3, \tag{A.4}
$$

yielding the strong conservation form of the convection–diffusion equation in computational space

$$
J_\kappa \frac{\partial \boldsymbol{q}}{\partial t}+\sum_{l,m=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} a_m \boldsymbol{q}\right)=\sum_{l,a,m=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial \xi_a}{\partial x_m} \frac{\partial\left(b_m \boldsymbol{q}\right)}{\partial \xi_a}\right). \tag{A.5}
$$

Now, consider the following differentiation matrices for the three-dimensional case:

$$
\mathsf{D}_{\xi_1} \equiv \mathsf{D}_\xi \otimes \mathsf{I}_{N_2} \otimes \mathsf{I}_{N_3}, \ \mathsf{D}_{\xi_2} \equiv \mathsf{I}_{N_1} \otimes \mathsf{D}_\xi \otimes \mathsf{I}_{N_3}, \ \mathsf{D}_{\xi_3} \equiv \mathsf{I}_{N_1} \otimes \mathsf{I}_{N_2} \otimes \mathsf{D}_\xi,
$$

where I_{N_l} is an $N_l \times N_l$ identity matrix and N_l represents the number of LGL points per direction in a given element. The diagonal matrix containing the metric Jacobian is defined as

$$
\mathsf{J}_\kappa \equiv \operatorname{diag}\left(J_\kappa(\boldsymbol{\xi}^{(1)}), \ldots, J_\kappa(\boldsymbol{\xi}^{(N_\kappa)})\right),
$$

while the diagonal matrix of the metric terms, $\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa$, has to be chosen to be a discretization of

$$
\operatorname{diag}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m}(\boldsymbol{\xi}^{(1)}), \ldots, J_\kappa \frac{\partial \xi_l}{\partial x_m}(\boldsymbol{\xi}^{(N_\kappa)})\right),
$$

where $N_\kappa \equiv N_1 N_2 N_3$ is the total number of LGL nodes in the κth element. To construct a stable scheme, we canonically split the inviscid terms into one half of the inviscid terms in (A.2) and one half of the inviscid terms in (A.3) (see [22]), while the viscous terms are handled in strict conservation form. At the continuous level, this process leads to

$$
\begin{aligned}
&J_\kappa \frac{\partial \boldsymbol{q}}{\partial t}+\frac{1}{2} \sum_{l,m=1}^{3}\left\{\frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} a_m \boldsymbol{q}\right)+J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial}{\partial \xi_l}\left(a_m \boldsymbol{q}\right)\right\} \\
&-\frac{1}{2} \sum_{l,m=1}^{3}\left\{a_m \boldsymbol{q} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m}\right)\right\}=\sum_{l,a,m=1}^{3} \frac{\partial}{\partial \xi_l}\left(J_\kappa \frac{\partial \xi_l}{\partial x_m} \frac{\partial \xi_a}{\partial x_m} \frac{\partial\left(b_m \boldsymbol{q}\right)}{\partial \xi_a}\right),
\end{aligned}
\tag{A.6}
$$

where the last set of terms on the left-hand side are zero by the GCL conditions (A.4). Then, a stable semi-discrete form can be constructed similarly to the split form (A.6) by discretizing the inviscid portion of (A.2) and (A.5) using D_{ξ_l}, J_κ, and $\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa$, and by averaging the results. The viscous terms are obtained from the discretization of the viscous portion of (A.5). This process leads to

$$
\begin{aligned}
&\mathsf{J}_\kappa \frac{\mathrm{d} \boldsymbol{q}_\kappa}{\mathrm{d} t}+\frac{1}{2} \sum_{l,m=1}^{3} a_m\left\{\mathsf{D}_{\xi_l}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa+\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa \mathsf{D}_{\xi_l}\right\} \boldsymbol{q}_\kappa \\
&-\frac{1}{2} \sum_{l,m=1}^{3}\left\{a_m \operatorname{diag}\left(\boldsymbol{q}_\kappa\right) \mathsf{D}_{\xi_l}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa \mathbf{1}_\kappa\right\}= \\
&\sum_{l,m,a=1}^{3} b_m \mathsf{D}_{\xi_l} \mathsf{J}_\kappa^{-1}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa\left[\mathcal{J} \frac{\partial \xi_a}{\partial x_m}\right]_\kappa \mathsf{D}_{\xi_a} \boldsymbol{q}_\kappa+\mathbf{SAT}_\kappa,
\end{aligned}
\tag{A.7}
$$

where $\mathbf{1}_\kappa$ is a vector of ones of size N_κ. The semi-discrete form (A.7) highlights a set of discrete GCL conditions (the analog of the continuous GCL conditions (A.4))

$$
\sum_{l=1}^{3} \mathsf{D}_{\xi_l}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa \mathbf{1}_\kappa=\mathbf{0}, \quad m=1,2,3. \tag{A.8}
$$

The satisfaction of conditions (A.8) will result in the telescoping, provably stable, semi-discrete form

$$
\begin{aligned}
&\mathsf{J}_\kappa \frac{\mathrm{d} \boldsymbol{q}_\kappa}{\mathrm{d} t}+\frac{1}{2} \sum_{l,m=1}^{3} a_m\left\{\mathsf{D}_{\xi_l}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa+\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa \mathsf{D}_{\xi_l}\right\} \boldsymbol{q}_\kappa= \\
&\sum_{l,m,a=1}^{3} b_m \mathsf{D}_{\xi_l} \mathsf{J}_\kappa{}^{-1}\left[\mathcal{J} \frac{\partial \xi_l}{\partial x_m}\right]_\kappa\left[\mathcal{J} \frac{\partial \xi_a}{\partial x_m}\right]_\kappa \mathsf{D}_{\xi_a} \boldsymbol{q}_\kappa+\mathbf{SAT}_\kappa.
\end{aligned}
\tag{A.9}
$$

At one of the ith interfaces of the κth element, the $\mathbf{SAT}_{\kappa,i}$ in the normal direction is defined as

$$
\mathbf{SAT}_{\kappa,i}=-\frac{1}{2} \mathsf{P}^{-1} \widehat{\mathcal{E}}\left(a_n \boldsymbol{q}_\kappa-\beta D \boldsymbol{q}_\kappa-\mathbf{1}_i g\right), \tag{A.10}
$$

where P is the norm-matrix of the SBP operator, and $\widehat{\mathcal{E}}$ is a matrix that extracts from the solution vector only the solution values associated to the LGL points that lie on the ith interface. The scalar a_n is the advection velocity projected in the normal direction, while the scalar β is defined as $\beta = 2|a|/\mathrm{Pe}$. The acronym Pe stands for the Pèclet number. The symbol $\mathbf{1}_i$ represents a vector of ones of size $(p+1)^{dim-1}$. For an interior interface, g contains the information of the adjoining solution element, whereas g is constructed using boundary data for a boundary interface.

References

[1] Chung B, Cebral JR. CFD for evaluation and treatment planning of aneurysms: review of proposed clinical uses and their challenges. Ann Biomed Eng 2015;43(1):122–38.

[2] Tang WM, Chan VS. Advances and challenges in computational plasma science. Plasma Phys. Controlled Fusion 2005;47(2):R1–34.

[3] Schneider T, Teixeira J, Bretherton CS, Brient F, Pressel KG, Schär C, Siebesma AP. Climate goals and computing the future of clouds. Nature Clim Change 2017;7(1):3–5.

[4] Neumann P, Düben P, Adamidis P, Bauer P, Brück M, Kornblueh L, Klocke D, Stevens B, Wedi N, Biercamp J. Assessing the scales in numerical weather and climate predictions: will exascale be the rescue? Phil Trans R Soc A 2019;377(2142):20180148.

[5] Bauer P, Thorpe A, Brunet G. The quiet revolution of numerical weather prediction. Nature 2015;525(7567):47–55.

[6] Richardson LF. Weather prediction by numerical process. Cambridge University Press; 2007.
[7] Anderson JD. Basic philosophy of CFD. In: Computational fluid dynamics. Springer; 2009, p. 3–14.
[8] Spalart PR, Venkatakrishnan V. On the role and challenges of CFD in the aerospace industry. Aeronaut J 2016;120(1223):209–32.
[9] Slotnick J, Khodadoust A, Alonso J, Darmofal D, Gropp W, Lurie E, Mavriplis D. CFD vision 2030 study: A path to revolutionary computational aerosciences. NASA-CR-2014-218178, 2014.
[10] Parsani M, Boukharfane R, Nolasco IR, Del Rey Fernández DC, Zampini S, Hadri B, Dalcin L. High-order accurate entropy-stable discontinuous collocated Galerkin methods with the summation-by-parts property for compressible CFD frameworks: Scalable SSDC algorithms and flow solver. J Comput Phys 2020;424:109844.
[11] Peña-Monferrer C, Manson-Sawko R, Elisseev V. HPC-cloud native framework for concurrent simulation, analysis and visualization of CFD workflows. Future Gener Comput Syst 2021;123:14–23.
[12] Ashton N, Sachs S, Foti L, Eberhardt S. Towards high-fidelity CFD on the cloud for the automotive and motorsport sectors. In: WCX SAE world congress experience. SAE International; 2020.
[13] Turner M, Appa J, Ashton N. Performance of CPU and GPU HPC architectures for off-design aircraft simulations. In: AIAA scitech 2021 forum. 2021, p. 1–7.
[14] Al Jahdali R, Dalcin L, Parsani M. On the performance of relaxation and adaptive explicit runge–kutta schemes for adaptive high-order compressible flow simulations. J Comput Phys 2022;In press.
[15] Ranocha H, Dalcin L, Parsani M, Ketcheson DI. Optimized Runge-Kutta methods with automatic step size control for compressible computational fluid dynamics. Commun Appl Math Comput 2021;1–38.
[16] Carpenter MH, Fisher TC, Nielsen EJ, Frankel S. Entropy stable spectral collocation schemes for the Navier–Stokes equations: Discontinuous interfaces. SIAM J Sci Comput 2014;36(5):835–67.
[17] Parsani M, Carpenter MH, Fisher T, Nielsen E. Entropy stable staggered grid discontinuous spectral collocation methods of any order for the compressible Navier–Stokes equations. SIAM J Sci Comput 2016;38(5):3129–62.
[18] Friedrich L, Winters AR, Del Rey Fernández DC, Gassner GJ, Parsani M, Carpenter MH. An entropy stable h/p non-conforming discontinuous Galerkin method with the summation-by-parts property. J Sci Comput 2018;77(2):689–725.
[19] Fernández DCDel Rey, Carpenter MH, Dalcin L, Zampini S, Parsani M. Entropy stable h/p-nonconforming discretization with the summation-by-parts property for the compressible Euler and Navier–Stokes equations. SN Partial Differ Equ Appl 2020;1(2):1–54.
[20] Parsani M, Carpenter MH, Nielsen EJ. Entropy stable wall boundary conditions for the three-dimensional compressible Navier–Stokes equations. J Comput Phys 2015;292:88–113.
[21] Dafermos CM. Hyperbolic conservation laws in continuum physics. Berlin: Springer-Verlag; 2010.
[22] Carpenter MH, Parsani M, Fisher TC, Nielsen EJ. Entropy stable staggered grid spectral collocation for the Burgers' and compressible Navier-Stokes equations. NASA TM-2015-218990, 2015.
[23] Svärd M. A convergent numerical scheme for the compressible Navier–Stokes equations. SIAM J Numer Anal 2016;54(3):1484–506.
[24] Fernández DCDel Rey, Carpenter MH, Dalcin L, Fredrich L, Winters AR, Gassner GJ, Parsani M. Entropy-stable p-nonconforming discretizations with the summation-by-parts property for the compressible Navier–Stokes equations. Comput & Fluids 2020;210:104631.
[25] Nolasco IR, Dalcin L, Fernández DCDel Rey, Zampini S, Parsani M. Optimized geometrical metrics satisfying free-stream preservation. Comput & Fluids 2020;207:104555.
[26] Parsani M, Carpenter MH, Nielsen EJ. Entropy stable discontinuous interfaces coupling for the three-dimensional compressible Navier–Stokes equations. J Comput Phys 2015;290:132–8.
[27] Del Rey Fernández DC, Carpenter MH, Dalcin L, Zampini S, Parsani M. Entropy stable h/p non-conforming discretization with the summation-by-parts property for the compressible Euler and Navier–Stokes equations. SN Partial Differ Equ Appl 2020;1(2):1–54.
[28] Crean J, Hicken JE, Del Rey Fernández DC, Zingg DZ, Carpenter MH. Entropy-stable summation-by-parts discretization of the Euler equations on general curved elements. J Comput Phys 2018;356:410–38.
[29] Dalcin L, Rojas D, Zampini S, Del Rey Fernández DC, Carpenter MH, Parsani M. Conservative and entropy stable solid wall boundary conditions for the compressible Navier–Stokes equations: Adiabatic wall and heat entropy transfer. J Comput Phys 2019;397:108775.
[30] Svärd Magnus, Özcan Hatice. Entropy-stable schemes for the Euler equations with far-field and wall boundary conditions. J Sci Comput 2014;58(1):61–89.
[31] Mengaldo Gianmarco, De Grazia Daniele, Witherden Freddie, Farrington Antony, Vincent Peter, Sherwin Spencer, Peiro Joaquim. A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics. In: 7th AIAA theoretical fluid mechanics conference. 2014, p. 2923.
[32] Butcher JC. Numerical methods for ordinary differential equations. Chichester: John Wiley & Sons Ltd; 2016.
[33] Ranocha H, Sayyari M, Dalcin L, Parsani M, Ketcheson DI. Relaxation Runge-Kutta methods: Fully-discrete explicit entropy-stable schemes for the compressible Euler and Navier–Stokes equations. SIAM J Sci Comput 2020;42(2):A612–38.
[34] Rogowski M, Dalcin L, Parsani M, Keyes DE. Performance analysis of relaxation runge–kutta methods. Int J High Perform Comput Appl 2022;10943420221085947.
[35] Balay S, Abhyankar S, Adams MF, Benson S, Brown J, Brune P, Buschelman K, Constantinescu E, Dalcin L, Dener A, Eijkhout V, Gropp WD, Hapla V, Isaac T, Jolivet P, Karpeev D, Kaushik D, Knepley MG, Kong F, Kruger S, May DA, McInnes L Curfman, Mills R Tran, Mitchell L, Munson T, Roman JE, Rupp K, Sanan P, Sarich J, Smith BF, Zampini S, Zhang H, Zhang H, Zhang J. PETSc/TAO users manual. Technical report ANL-21/39 - revision 3.16, Argonne National Laboratory; 2021.
[36] Knepley MG, Karpeev DA. Mesh algorithms for PDE with Sieve I: Mesh distribution. Sci Program 2009;17(3):215–30.
[37] Abhyankar S, Brown J, Constantinescu EM, Ghosh D, Smith BF, Zhang H. PETSc/TS: A modern scalable ODE/DAE solver library. Technical report, 2018.
[38] Burstedde C, Wilcox LC, Ghattas O. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM J Sci Comput 2011;33(3):1103–33.
[39] Isaac T, Burstedde C, Wilcox LC, Ghattas O. Recursive algorithms for distributed forests of octrees. SIAM J Sci Comput 2015;37(5):C497–531.
[40] Isaac T, Knepley MG. Support for non-conformal meshes in PETSc's DMPlex interface. 2015, arXiv preprint arXiv:1508.02470.
[41] Cenaero. HiOCFD5, 5th international workshop on high-order CFD methods. 2018, URL https://how5.cenaero.be.
[42] Kimball E, Whitaker T, Kevrekidis YG, Benziger JB. Drops, slugs, and flooding in polymer electrolyte membrane fuel cells. AIChE J 2008;54(5):1313–32.
[43] Cebeci T, Kafyeke F. Aircraft icing. Annu Rev Fluid Mech 2003;35(1):11–21.
[44] Madani S, Amirfazli A. Oil drop shedding from solid substrates by a shearing liquid. Colloids Surf A 2014;441:796–806.
[45] Hummel D, Redeker G. A new vortex flow experiment for computer code validation. In: RTO/AVT symposium on vortex flow and high angle of attack aerodynamics, meeting proc. RTO-MP-069. 2003, p. 8–31.
[46] Mengaldo Gianmarco, Moxey David, Turner Michael, Moura Rodrigo Costa, Jassim Ayad, Taylor Mark, Peiro Joaquim, Sherwin Spencer. Industry-relevant implicit large-eddy simulation of a high-performance road car via spectral/hp element methods. SIAM Rev 2021;63(4):723–55.
[47] Rumsey CL, Morrison JH. Goals and status of the NASA juncture flow experiment. NATO, STO-MP-AVT-246, 2016.
[48] Pegrum JM. Experimental study of the vortex system generated by a formula 1 front wing (Ph.D. thesis), Imperial College London; 2007.
[49] Buscariolo FF, Hoessler J, Moxey D, Jassim A, Gouder K, Basley J, Murai Y, Assi GRS, Sherwin SJ. Spectral/hp element simulation of flow past a formula one front wing: validation against experiments. 2019.
[50] Svärd M, Nordström J. Review of summation-by-parts schemes for initial boundary–value problems. J Comput Phys 2014;268:17–38.
[51] Del Rey Fernández DC, Hicken JE, Zingg DW. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. Comput & Fluids 2014;95:171–96.