Computer Physics Communications 279 (2022) 108453

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

UCNS3D: An open-source high-order finite-volume unstructured CFD solver ☆,☆☆

Antonis F. Antoniadis [a], Dimitris Drikakis [b], Pericles S. Farmakis [c,d,e], Lin Fu [f,g], Ioannis Kokkinakis [b], Xesús Nogueira [h], Paulo A.S.F. Silva [a], Martin Skote [a], Vladimir Titarev [i], Panagiotis Tsoutsanis [a,*]

[a] Centre for Computational Engineering Sciences, Cranfield University, Cranfield MK43 0AL, United Kingdom
[b] University of Nicosia, Nicosia CY-2417, Cyprus
[c] Laboratory for Laser Energetics, University of Rochester, NY, 14623, USA
[d] Flash Center for Computational Science, Department of Physics and Astronomy, University of Rochester, NY, 14627, USA
[e] Department of Mechanical Engineering, University of Rochester, NY, 14623, USA
[f] Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[g] Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[h] Universidade da Coruña, Group of Numerical Methods in Engineering-GMNI, Center for Technological Innovation in Construction and Civil Engineering- CITEEC, Civil Engineering School, Campus de Elviña, A Coruña 15071, Spain
[i] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow 119333, Russia,

ARTICLE INFO

Article history:
Received 23 February 2022
Received in revised form 8 June 2022
Accepted 13 June 2022
Available online 17 June 2022

Dataset link: https://doi.org/10.17862/cranfield.rd.16447212.v1

Dataset link: https://doi.org/10.17862/cranfield.rd.11836164.v1

Dataset link: https://doi.org/10.17862/cranfield.rd.8983772.v1

Dataset link: https://doi.org/10.17862/cranfield.rd.19146182.v1

Keywords:
CFD
High-order
Finite-volume
Parallel
HPC
Open-source

ABSTRACT

UCNS3D is an open-source computational solver for compressible flows on unstructured meshes. State-of-the-art high-order methods and their associated benefits can now be implemented for industrial-scale CFD problems due to the flexibility and highly-automated generation offered by unstructured meshes. We present the governing equations of the physical models employed in UCNS3D, and the numerical framework developed for their solution. The code has been designed so that extended to other systems of equations and numerical models is straightforward. The employed methods are validated towards a series of stringent well-established test problems against experimental or analytical solutions, where the full capabilities of UCNS3D in terms of applications spectrum, robustness, efficiency, and accuracy are demonstrated.

Program summary

Program title: UCNS3D (Unstructured Compressible Flow Solver)
CPC Library link to program files: https://doi.org/10.17632/222zh873kh.1
Developer's repository link: https://github.com/ucns3d-team/UCNS3D
Licensing provisions: GNU General Public License 3
Programming language: Fortran2008
Nature of problem: UCNS3D is intended for the simulation of compressible flows in 2D and 3D unstructured meshes, by employing high-resolution, high-order methods capable of providing physically meaningful results in a computational efficient manner. The solver is designed for a broad range of problems encountered in engineering applications such as transitional, fully turbulent, and multicomponent flows with several fidelity level modelling options available.
Solution method: The present software includes multiple physical models, numerical methods, and modelling techniques such as iLES, RANS, DES for unstructured meshes. The software has been

☆ The review of this paper was arranged by Prof. N.S. Scott.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail addresses: a.f.antoniadis@cranfield.ac.uk (A.F. Antoniadis), drikakis.d@unic.ac.cy (D. Drikakis), pfar@lle.rochester.edu (P.S. Farmakis), linfu@ust.hk (L. Fu), yanisint@gmail.com (I. Kokkinakis), xesus.nogueira@udc.es (X. Nogueira), paulo.a.silva@cranfield.ac.uk (P.A.S.F. Silva), m.skote@cranfield.ac.uk (M. Skote), vladimir.titarev@frccsc.ru (V. Titarev), panagiotis.tsoutsanis@cranfield.ac.uk (P. Tsoutsanis).

https://doi.org/10.1016/j.cpc.2022.108453
0010-4655/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

developed such that the inclusion of additional physical models and numerical methods can be easily accommodated.

© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Compressible flows are found in an overwhelming range of applications across several disciplines. These include turbulent flows around aircraft, automotive vehicles, space launch vehicles, windturbines; multiphysics flows in high-energy-density physics like supernova explosions and inertial confinement fusion, liquid jet atomization in scramjet engines, shockwave lithotripsy. Improving our understanding of these flows is key to several of the most important challenges facing today's society, such as the drive to netzero, understanding the origins of the cosmos, and curing diseases. The active research and development landscape in CFD, has led to breakthroughs in method developments, high effectiveness in simulation workflows and computational performance, lower uncertainty in simulated predictions and with a richer understanding of increasingly complex multi-disciplinary physics.

The numerical simulation of compressible flows, requires an arsenal of numerical methods, physics models, and efficient computational paradigms due to the simultaneous presence of a plethora of physical phenomena with competing requirements, e.g., shock waves, turbulence, cavitation, and hydrodynamic instabilities. Phenomena with strong gradients such as shock waves require non-oscillatory numerical methods to sharply capture, while delicate smooth structures such as vortices require highly accurate schemes to resolve. Another challenge is that the resolution required to study and understand such phenomena for cases of practical interest, where the Reynolds number is high, often necessitates the use of large scale high-performance computing facilities (HPC). Therefore, it is of paramount importance for the CFD software implementation to possess a favourable parallel computational efficiency. Moreover, most flows of practical interest are usually encountered in settings where the geometry of interest could be highly complicated, and several geometrical design iterations might be required, leading to unstructured meshes having established themselves as the modern passkey for this challenging requirement.

One of the strategies that has been widely adopted for addressing all of the aforementioned challenges across the finite-volume (FV) [1–17], finite-element (FE) [18–26], spectral Finite Volume (SFV) [27–31], and combinations of them [32–37] is to develop and apply high-resolution and high-order numerical methods. High-resolution, high-order methods reduce unphysical oscillations across discontinuities, providing enhanced accuracy in flow predictions in smooth regions, while being more computational efficient than low-order methods.

UCNS3D is an open-source high-order compressible CFD solver based on unstructured meshes. The robustness, flexibility, computational efficiency, and accuracy are the key attributes by which the entire computational framework of UCNS3D including methods, models, and algorithms, is based on. The arsenal of high-resolution numerical schemes comprises of: Central schemes, Monotone Upstream centred scheme for Conservation Laws (MUSCL) schemes, Weighted Essentially non-oscillatory (WENO) schemes, Central WENO (CWENO) schemes, CWENOZ schemes, Multidimensional Optimal Order Detection (MOOD) schemes ranging from 1st to 7th order of spatial accuracy, and a wide range of temporal discretisation methods for steady and transient flow problems. The multicomponent flows are solved using the diffuse-interface paradigm, and turbulent flow simulation modes include RANS (Reynolds-Averaged Navier-Stokes), DES (Detached Eddy Simulation), DDES (Delayed Detached Eddy Simulation), iLES (implicit Large Eddy Simulation), and DNS (Direct Numerical Simulation). To ensure that the benefits of high-order methods are realised for practical applications, the parallel implementation strategically employs both MPI and OpenMP interfaces suited to harness the computing power available in a computationally efficient manner, as will be further discussed in this paper. UCNS3D offers the unique flexibility of combining high-order FV methods with several levels of fidelity for simulating compressible flows. This is clearly demonstrated in the following example of simulating the turbulent flow past an aircraft. Depending on the computational budget available, carrying out transient iLES might not be always feasible. Therefore, the user can instead resort to the RANS equations while still taking advantage of deploying high-order methods to improve the accuracy of the simulation results.

UCNS3D is not the only compressible CFD code that can support unstructured meshes, since other established solvers such as OpenFoam [38], SU2 [39], PyFR [21], solve compressible flow problems on unstructured meshes as well. However, UCNS3D is a CFD solver within which the user can deploy very high-order FV schemes for several governing equations including turbulent flows, flows with rotating geometries, as well as multicomponent flows. While, similarly to other established solvers, it can also serve as a platform for developers and researchers to further develop and expand the existing physical models and algorithms embedded into the solver's structure.

The main value proposition of UCNS3D is to aid researchers, academics and CAT (Computer Aided Technology) analysts, all who require high-fidelity predictions from a CFD simulation, by (i) omitting the need to generate additional meshes, and (ii) by providing an intuitive coding framework to extend the physics domains further without the inconvenience of having to implement core numerics, algorithms and data bookkeeping components of the code.

There are two types of users UCNS3D is well suited for. Firstly, those who perform flow analysis to enhance their physics understanding and apply these insights to improve the design of discrete manufactured products and systems, e.g., civil aviation, automotive and motorsports, renewable energy. And then those in research groups who realise research and development using UCNS3D through collaborations by extending its physics domains and improving the core computational framework capabilities [1,40–42]. UCNS3D is a computational platform that enables researchers to collaborate, co-develop, and share ideas to push further the boundaries of science.

The remainder of this paper is structured as follows. Section 2 describes the governing equations used in UCNS3D. Section 3 is devoted to presenting the complete numerical framework developed in UCNS3D. Section 4 outlines the features, structure and description of input/output files, and package installation and implementation and how they are linked with the previously presented governing equations and numerical framework. Section 5 follows with several representative test problems in 2D and 3D where the high-order of accuracy, and robust non-oscillatory properties of the framework are highlighted. The cases considered include the 2D vortex evolution, the 2D interaction of a shock wave with entropy wave, the 3D viscous iLES of the Taylor-Green vortex, RANS of full aircraft at cruise conditions, the iLES of the SD7003 airfoil, the RANS simulation of the Caradone & Tung rotor, and finally, the interaction between a helium bubble and a shockwave. A represen-

tative example of the parallel performance of UCNS3D is detailed in Section 6, while the conclusions are provided in Section 7.

2. Governing equations

In this section, some of the most important systems implemented in UCNS3D are described.

2.1. Linear advection equation

The linear advection equation considered is as follows:

$$\frac{\partial U}{\partial t} + \nabla \cdot (\mathbf{v}U) = 0, \tag{1}$$

where U is the conserved scalar, and $\mathbf{v}$ is the wave speed vector. This equation is the starting point for the development, validation and verification for any method developed in the software.

2.2. Navier-Stokes equations

The compressible Navier-Stokes equations are considered, written in conservative form as:

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t} + \nabla \cdot (\vec{\mathbf{F}}_c(\mathbf{U}) - \vec{\mathbf{F}}_v(\mathbf{U}, \nabla\mathbf{U})) = \mathbf{S}(\mathbf{U}, \nabla\mathbf{U}), \tag{2}$$

where $\mathbf{U}$ is the vector of the conserved variables, $\mathbf{S}$ is the source term vector and unless otherwise specified it is going to be assumed to be zero, and $\vec{\mathbf{F}}_c$ and $\vec{\mathbf{F}}_v$ are the inviscid and viscous flux vectors, respectively, as

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}, \vec{\mathbf{F}}_c = \begin{bmatrix} \rho u_n \\ \rho u u_n + n_x p \\ \rho v u_n + n_y p \\ \rho w u_n + n_z p \\ u_n(E+p) \end{bmatrix}, \quad \vec{\mathbf{F}}_v = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\Theta_x + n_y\Theta_y + n_z\Theta_z \end{bmatrix}, \tag{3}$$

where ρ is the density; u, v, w are the velocity components in x, y and z Cartesian coordinates, respectively, and u_n is the velocity normal to the bounded surface area, defined by $u_n = n_x u + n_y v + n_z w$. Ideal gas is assumed where the total energy per unit mass is calculated by $E = p/(\gamma-1) + (1/2)\rho(u^2+v^2+w^2)$, where p is the pressure, $\gamma = 1.4$ is the ratio of specific heats for air at normal atmospheric conditions; The laminar viscosity is related to the temperature through the Sutherland's law as

$$\frac{\mu_l}{\mu_0} = \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S}, \tag{4}$$

S is the Sutherland temperature and the subscript 0 denotes a reference state for the corresponding variables. The work of viscous stresses and heat conduction, Θ, is given by:

$$\begin{aligned} \Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \frac{\mu_l}{Pr}\frac{\gamma}{(\gamma-1)}\frac{\partial T}{\partial x}, \\ \Theta_y &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + \frac{\mu_l}{Pr}\frac{\gamma}{(\gamma-1)}\frac{\partial T}{\partial y}, \\ \Theta_z &= u\tau_{zz} + v\tau_{zy} + w\tau_{zz} + \frac{\mu_l}{Pr}\frac{\gamma}{(\gamma-1)}\frac{\partial T}{\partial z}. \end{aligned} \tag{5}$$

The viscous stress tensor τ_{ij} is defined by is

$$\tau_{ij} = (\mu_l)\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j} + \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}_i} - \frac{2}{3}\frac{\partial \mathbf{u}_k}{\partial \mathbf{x}_k}\delta_{ij}\right), \tag{6}$$

where δ_{ij} is the Kronecker delta and the subscripts i, j, k refer to the Cartesian coordinate components $\mathbf{x} = (x, y, z)$. Unless otherwise stated, the reference values are taken at atmospheric conditions (sea level): dynamic viscosity $\mu_0 = 1.7894 \times 10^{-5}$ kg/(ms); reference temperatures $T_0 = 288.16$ K; $S = 110.4$ K; and Prandtl number $Pr = 0.72$. The inviscid Euler equations are simply obtained by setting the viscous fluxes $\vec{\mathbf{F}}_v = 0$.

2.3. Reynolds-Averaged Navier-Stokes equations

For the RANS equations, the Spalart-Allmaras (SA) turbulence model [43] is coupled to Eq. (2), and the equation for this model is given by:

$$\begin{aligned} \frac{\partial \tilde{\nu}}{\partial t} + \nabla \cdot \left(\mathbf{v}\tilde{\nu}\right) =& C_{b1}\tilde{S}\tilde{\nu} + \frac{1}{\sigma}\left(\nabla \cdot ((\nu_l + \tilde{\nu})\nabla\tilde{\nu}) + C_{b2}\nabla\tilde{\nu}^2\right) \\ &- C_{w1} f_w \left(\frac{\tilde{\nu}}{d}\right)^2, \end{aligned} \tag{7}$$

where $\mathbf{v}$ being the Cartesian velocity vector, ν_l being the kinematic laminar viscosity and $\tilde{\nu}$ being the SA turbulent viscosity working variable. The turbulence parameter $\tilde{\nu}$ is related to eddy viscosity μ_t by:

$$\mu_t = \rho\tilde{\nu} f_{v1}, \quad \text{where} \quad f_{v1} = \frac{(\rho\tilde{\nu}/\mu_l)^3}{(\rho\tilde{\nu}/\mu_l)^3 + C_{v1}^3}, \tag{8}$$

and

$$\tilde{S} = \Omega + f_{v2}\frac{\tilde{\nu}}{\kappa^2 d^2}, \quad f_{v2} = 1 - \frac{(\rho\tilde{\nu}/\mu_l)}{1 + (\rho\tilde{\nu}/\mu_l) f_{v1}}, \tag{9}$$

Ω being the vorticity magnitude, and d being the wall distance. For the complete definitions of all the parameters, the readers are referred to the original work of Spalart-Allmaras [43]. Finally, with the inclusion of SA turbulence model, Eq. (5) is revised, and the term $\frac{\mu_l}{Pr}$ is modified to include the turbulence model contribution $\left(\frac{\mu_l}{Pr} + \frac{\mu_t}{Pr_t}\right)$, where μ_t and Pr_t being the turbulent dynamic viscosity and Prandtl number respectively. Similarly, Eq. (6) is revised, and the term (μ_l) is replaced by $(\mu_l + \mu_t)$. The SA turbulence model is also deployed for DES [44] and DDES [45] within UCNS3D, where the wall distance d is replaced by the equivalent expressions for DES and DDES accordingly [45].

2.4. Multiple reference framework

To avoid any relative motions between parts of the spatial domain, the frame of reference is decomposed into two parts: rotational and stationary. This procedure is known as Multiple Reference Frame (MRF) and works by switching the observer view of the problem. By defining an axisymmetric subdomain with a rotating reference frame attached to the blade at a particular position, an unsteady problem can be reformulated as steady. Then, the governing equations in each subdomain are computed differently. A useful mathematical approach to deal with more than one frame of reference is using the absolute velocity formulation, as it does not require special reference transformation at the interface between the subdomains. In this sense, the governing equations for each subdomain are computed with respect to its reference frame, but the velocities are stored in the absolute frame. The inviscid flux, $\vec{\mathbf{F}}_c$, and the source, $\mathbf{S}$, on Eq. (2) are modified as

$$\vec{\mathbf{F}}_c = \begin{bmatrix} \rho u_r \\ \rho u_r u + n_x p \\ \rho u_r v + n_y p \\ \rho u_r w + n_z p \\ E u_r + p u \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0 \\ \rho\left(\omega_2 w - \omega_3 v\right) \\ \rho\left(-\omega_1 w + \omega_3 u\right) \\ \rho\left(\omega_1 v - \omega_2 u\right) \\ 0 \end{bmatrix}, \tag{10}$$

where u_r is the relative velocity ($\vec{\mathbf{u}}_\mathbf{r} = \vec{\mathbf{u}} - \vec{\omega} \times \vec{\mathbf{r}}$), the steady angular velocity is $\vec{\omega} = (\omega_1, \omega_2, \omega_3)$ and $\vec{\mathbf{r}}$ is the radius vector pointing from the specified rotation centre. Additionally, the wall boundary condition is also modified to $\vec{\mathbf{u}} = \vec{\omega} \times \vec{\mathbf{r}}$.

2.5. 5-equation multicomponent model

The quasi-conservative five-equation model of Allaire et al. [46] is also considered for inviscid compressible multicomponent flows that has been widely-used in conjunction with high-resolution methods [47]. For two fluids, this involves two continuity equations, a momentum equation per dimension, an energy equation, and the non-conservative advection equation of the volume fraction of one of the two fluids as given below

$$\frac{\partial \mathbf{U}(\mathbf{x}, t)}{\partial t} + \nabla \cdot \vec{\mathbf{F}}_c(\mathbf{U}) = \mathbf{S}(\mathbf{U}, \nabla \mathbf{U}), \tag{11}$$

where $\mathbf{U}$ is the vector of the conserved variables, $\vec{\mathbf{F}}_c$ is the inviscid flux vector and $\mathbf{S}$ is the source term vector defined as

$$\mathbf{U} = \begin{bmatrix} a_1\rho_1 \\ a_2\rho_2 \\ \rho u \\ \rho v \\ \rho w \\ E \\ a_1 \end{bmatrix}, \vec{\mathbf{F}}_c = \begin{bmatrix} a_1\rho_1 u_n \\ a_2\rho_2 u_n \\ \rho u u_n + n_x p \\ \rho v u_n + n_y p \\ \rho w u_n + n_z p \\ u_n(E + p) \\ a_1 u_n \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ a_1 \nabla \cdot \mathbf{u} \end{bmatrix}, \tag{12}$$

where u_n is the velocity normal to the bounded surface area, defined by $u_n = n_x u + n_y v + n_z w$, ρ is the density, p is the pressure, E is the total energy and a is the volume fraction. The stiffened gas equation of state (EOS) is employed for closing the five-equation model. It has been primarily selected due to its application for flow problems involving gases, liquids and solids. The stiffened gas EOS characterises each fluid pressure as:

$$p_i = (\gamma_i - 1)\rho_i \epsilon_i - \gamma_i \pi_{\infty,i}, \tag{13}$$

where $\pi_{\infty,i} \geq 0$ is a reference pressure, and will be set to $\pi_\infty = 0$ for gases. The averaged density ρ and $\rho\epsilon$ are given by the following equations as

$$\rho = \sum_i a_i \rho_i, \tag{14}$$

$$\rho\epsilon = \sum_i a_i \rho_i \epsilon_i, \tag{15}$$

where ϵ is the internal energy, with $\rho\epsilon = E - \frac{1}{2}\rho(u^2 + v^2 + w^2)$. The EOS of the mixture reads

$$\xi = \frac{1}{\gamma - 1} = \sum_i \frac{a_i}{\gamma_i - 1}, \tag{16}$$

$$\frac{\pi_\infty \gamma}{\gamma - 1} = \sum_i a_i \frac{\pi_{\infty,i}\gamma_i}{\gamma_i - 1}, \tag{17}$$

$$p = (\gamma - 1)\rho\epsilon - \gamma\pi_\infty. \tag{18}$$

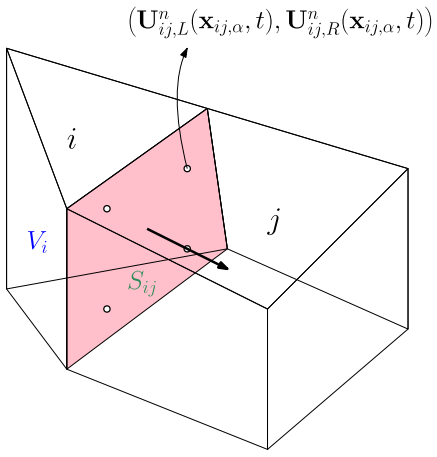


Fig. 1. Drawing illustrating the interface between the considered cell i and its neighbour j highlighting the normal vector and the quadrature points at the interface.

3. Numerical framework

Consider the unsteady non-linear hyperbolic system of conservation laws on a 3D domain Ω, written in its conservative form:

$$\frac{\partial \mathbf{U}(\mathbf{x}, t)}{\partial t} + \nabla \cdot (\vec{\mathbf{F}}_c(\mathbf{U}) - \vec{\mathbf{F}}_v(\mathbf{U}, \nabla \mathbf{U})) = \mathbf{S}(\mathbf{U}, \nabla \mathbf{U}), \tag{19}$$

where $\mathbf{U}$ is the vector of the conserved variables, $\vec{\mathbf{F}}_c$ and $\vec{\mathbf{F}}_v$ are the inviscid and viscous flux vectors respectively, and $\mathbf{S}$ is the source term vector. The physical domain Ω in consists of any combination of conforming tetrahedral, hexahedral, prism or pyramids in 3D, and quadrilateral or triangular in 2D. All the elements are indexed by a unique mono-index i. Integrating Eq. (19) over the mesh element i using a high-order explicit finite-volume formulation the following equation is obtained:

$$\begin{aligned} \frac{d\mathbf{U}_i}{dt} = &- \frac{1}{|V_i|} \sum_{l=1}^{N_f} \sum_{\alpha=1}^{N_{qp}} \vec{\mathbf{F}}_{c_l}\left(\mathbf{U}^n_{l,L}(\mathbf{x}_{l,\alpha}, t), \mathbf{U}^n_{l,R}(\mathbf{x}_{l,\alpha}, t)\right) \omega_\alpha |S_l| \\ &+ \frac{1}{|V_i|} \sum_{l=1}^{N_f} \sum_{\alpha=1}^{N_{qp}} \vec{\mathbf{F}}_{v_l}(\mathbf{U}^n_{l,L}(\mathbf{x}_{l,\alpha}, t), \mathbf{U}^n_{l,R}(\mathbf{x}_{l,\alpha}, t), \\ &\quad \nabla \mathbf{U}^n_{l,L}(\mathbf{x}_{l,\alpha}, t), \nabla \mathbf{U}^n_{l,R}(\mathbf{x}_{l,\alpha}, t)) \omega_\alpha |S_l| \\ &+ \mathbf{S}_i, \end{aligned} \tag{20}$$

where $\mathbf{U}_i$ are the volume averaged conserved variables

$$\mathbf{U}_i = \frac{1}{|V_i|} \int_{V_i} \mathbf{U}(x, y, z)\, dV, \tag{21}$$

and $\vec{\mathbf{F}}_{c_l}$ and $\vec{\mathbf{F}}_{v_l}$ are the numerical flux function in the direction normal to the cell interface between two cells as seen in Fig. 1, N_f is the number of faces per element, N_{qp} is the number of quadrature points used for approximating the surface integrals, $|S_l|$ is the surface area of the corresponding face, and $\mathbf{U}^n_{l,L}(\mathbf{x}_{l,\alpha}, t)$ and $\mathbf{U}^n_{l,R}(\mathbf{x}_{l,\alpha}, t)$ are the high-order approximations of the solutions for the considered cell and its neighbour respectively and $\nabla \mathbf{U}^n_{l,L}(\mathbf{x}_{l,\alpha}, t)$ and $\nabla \mathbf{U}^n_{l,R}(\mathbf{x}_{l,\alpha}, t)$ are the high-order approximations of the gradients for the considered cell and its neighbour respectively; while α corresponds to different Gaussian integration points $\mathbf{x}_\alpha$ and weights ω_α over each face. The volume, surface and line integrals are numerically approximated by suitable quadrature rules, see [48] for details on numerical approximations of multiple integrals.

3.1. Spatial discretisation

The reconstruction process adopted in UCNS3D follows the approaches of Tsoutsanis et al. [49,50], Titarev et al. [51] that have been previously applied to smooth and discontinuous flow problems [1–4,40,52–61] and only the key components are going to be described herein and the reader is referred to [49–51]. For a cell i a high-order polynomial $p_i(x, y, z)$ of order r can be built that provides $r+1$ order of accuracy, by requiring it to have the same average as a general quantity $\mathbf{U}_i$. This can be formulated as:

$$\mathbf{U}_i = \frac{1}{|V_i|}\int_{V_i} p_i(x, y, z)\, dV. \tag{22}$$

In order to reduce scaling effects, usually encountered when dealing with unstructured meshes of various elements shapes and sizes, a transformation from physical space (x, y, z) to a reference space (ξ, η, ζ) is performed as introduced by Dumbser et al. [6,49]. Each non-triangular or tetrahedral element is decomposed into triangular or tetrahedral elements, and one of the decomposed elements is used as a reference element for transforming to the new system of coordinates, as suggested by Tsoutsanis et al. [50]. The decomposition strategy is discussed in [50]. One important property of the transformation is that the spatial average of the conserved variable $\mathbf{U}_i$ does not change during transformation,

$$\mathbf{U}_i = \frac{1}{|V_i|}\int_{V_i} \mathbf{U}(x, y, z)\, dV \equiv \frac{1}{|V_i'|}\int_{V_i'} \mathbf{U}(\xi, \eta, \zeta)\, d\xi d\eta d\zeta, \tag{23}$$

where V_i' is the volume of the considered cell i in the reference co-ordinate system. The reconstruction is performed by building a central stencil $\mathcal{S}^1$ by recursively adding neighbouring elements, consisting of $M+1$ cells including the considered cell i. Several stencil selection algorithms have been developed in, UCNS3D but the most frequently used one is the stencil-based compact algorithm (SBC) introduced in, [62] where the reader is referred to for more details. For improved robustness, we employ $M = 2K$ cells in the stencil as reported in several previous studies [6,17,49,62–66], where K is the total number of polynomial coefficients given by:

$$K(r, d) = \frac{1}{d!}\prod_{l=1}^{d}(r+l), \tag{24}$$

where $d \in [2, 3]$ is the number of space dimensions. The central stencil $\mathcal{S}^1$ is given by

$$\mathcal{S}_i^c = \bigcup_{m=0}^{M_c} V_m, \tag{25}$$

where the index m refers to the local numbering of the elements in the stencil with the element with index 0 being the considered cell i, and the index c referring to the stencil number (in case of multiple stencils) where for the central stencil $c = 1$. The entire stencil of the considered cell i is transformed in reference space $\mathcal{S}_i'^c$, where the r^{th} order reconstruction polynomial is an expansion over local polynomial basis functions $\phi_k(\xi, \eta, \zeta)$ given by:

$$p(\xi, \eta, \zeta) = \sum_{k=0}^{K} a_k \phi_k(\xi, \eta, \zeta) = \mathbf{U}_0 + \sum_{k=1}^{K} a_k \phi_k(\xi, \eta, \zeta), \tag{26}$$

where $\mathbf{U}_0$ corresponds to the vector of conserved variables at the considered cell i, and a_k are the degrees of freedom of the polynomial. The degrees of freedom a_k for the polynomial for each cell m are obtained by satisfying the condition that the cell average of the reconstruction polynomial $p(\xi, \eta, \zeta)$ must be equal to the cell average of the solution $\mathbf{U}_m$:

$$\begin{aligned}\int_{V_m'} p(\xi, \eta, \zeta)\, d\xi d\eta d\zeta &= |V_m'|\mathbf{U}_0 + \sum_{k=1}^{K}\int_{V_m'} a_k \phi_k\, d\xi d\eta d\zeta \\ &= |V_m'|\mathbf{U}_m, \quad m = 1, \ldots, M.\end{aligned} \tag{27}$$

It needs to be stressed that since for hexahedral, quadrilateral, prisms and pyramid cells the transformation to reference space into a unit element cell cannot be guaranteed, the basis functions ψ_k also need to satisfy equation (22). The basis functions employed ψ_k for all the elements in the stencil are defined as follows:

$$\phi_k(\xi, \eta, \zeta) \equiv \psi_k(\xi, \eta, \zeta) - \frac{1}{|V_0'|}\int_{V_0'} \psi_k\, d\xi d\eta d\zeta \quad k = 1, 2, \ldots, K, \tag{28}$$

and in the present study ψ_k are Legendre polynomials basis functions. Denoting the integrals of the basis function k over the cell m in the stencil, and the vector of right-hand side by A_{mk} and b respectively as given by

$$A_{mk} = \int_{V_m'} \phi_k\, d\xi d\eta d\zeta, \quad b_m = |V_m'|(\mathbf{U}_m - \mathbf{U}_0), \tag{29}$$

the equations for degrees of freedom a_k can be rewritten in a matrix form as:

$$\sum_{k=1}^{K} A_{mk} a_k = b_m, \quad m = 1, 2, \ldots M. \tag{30}$$

The resulting linear system is solved by a QR decomposition based on Householder transformation [67] while using a Moore-Penrose pseudo-inverse of A_{mk} which is only computed once at the beginning of the simulation to for improving computational speed as detailed in [62]. Up to 7th-order accurate least-square polynomial reconstructions are available in UCNS3D for 2D and 3D problems.

3.1.1. Central

The central scheme, implemented in UCNS3D, refers to the linear scheme utilising only one central stencil, as described previously. Its standalone use however is limited due to the non-existing non-oscillatory properties, and it can only be used in conjunction with the MOOD algorithm, or in flow problems with low-Mach number.

3.1.2. MUSCL

UCNS3D implements the MUSCL scheme where a high-order variation of the solution within every cell is approximated by the corresponding polynomial, whose degrees of freedom a_k are computed during the least-squares reconstruction process. The scheme can be written as:

$$\mathbf{U}_{l,\alpha} = \mathbf{U}_i + \theta_i \cdot \sum_{k=1}^{K} a_k \phi_k(\xi_a, \eta_a, \zeta_a), \tag{31}$$

where $\mathbf{U}_{l,\alpha}$ is the extrapolated reconstructed solution at a face l, and at a quadrature point α, $\mathbf{U}_i$ is the value for the conserved variable of the considered element i, and (ξ_a, η_a, ζ_a) are the coordinates of the quadrature point at the l face. All polynomials, for all the faces and for all the quadrature points, are then limited by the limiter θ_i which is valid for the cell i to prevent any spurious oscillations. In the MUSCL framework, the following steps are taken:

1. Find the minimum and maximum values of the conserved variable from the neighbouring elements and the considered cell i
2. Compute the unlimited least square reconstructed solution $\mathbf{U}_{l,\alpha}$
3. Compute the maximum allowable value for $\theta_{l,\alpha}$, which corresponds to the slope limiter value at side l and quadrature point α for cell i
4. Select the minimum value of the slope limiter from all the faces, all the quadrature points α such that $\theta_i = \min(\theta_{l,a})$ where l is the index of the face of the considered cell.

It is the third step that differentiates the limiting function employed, and UCNS3D deploys the widely used limiters of Barth and Jespersen [68], and of Venkatakrishnan [69]. Both of these slope limiters are restricted to 2nd-order of spatial accuracy. However, for up to 4th-order accurate MUSCL schemes, the Michalak and Ollivier-Gooch [70] limiter (MOG) as well as its further extended bounds versions MOGE, MOGV developed in [61] are also available. The MOGE and MOGV variants redefine the elements included for obtaining the minimum and maximum values of step 2, by including all the stencil elements and all the vertex neighbours of the considered cell, respectively. The reader is referred to [61] for a detailed description of the limiting functions used.

3.1.3. WENO

The WENO scheme used in UCNS3D, employs a non-linear combination of various reconstruction polynomials from the central stencil and additional directional stencils, and each polynomial is weighted according to the smoothness of its solution, and it is based on the approaches of [6,49,50,65]. The polynomials are given as:

$$p_i(\xi,\eta,\zeta)^{\text{weno}} = \sum_{s=1}^{s_t} \omega_s p_s(\xi,\eta,\zeta), \tag{32}$$

where s_t is the total number of stencils. Substituting back to Eq. (26) for $p_s(\xi,\eta,\zeta)$, we obtain the following expression:

$$p_s(\xi,\eta,\zeta) = \sum_{k=0}^{K} a_k^{(s)} \phi_k(\xi,\eta,\zeta). \tag{33}$$

Using the condition that the sum of all weights is unity, yields:

$$\begin{aligned} p_i(\xi,\eta,\zeta)^{\text{weno}} &= \mathbf{U}_0 + \sum_{k=1}^{K} \left(\sum_{s=0}^{s_t} \omega_s a_k^s \right) \phi_k(\xi,\eta,\zeta) \\ &\equiv \mathbf{U}_0 + \sum_{k=1}^{K} \tilde{a}_k \phi_k(\xi,\eta,\zeta), \end{aligned} \tag{34}$$

where $\tilde{a}_k$ are the reconstructed degrees of freedom; and the non-linear weight ω_m is defined as:

$$\omega_s = \frac{\tilde{\omega}_s}{\sum_{s=1}^{s_t} \tilde{\omega}_s} \quad \text{where} \quad \tilde{\omega}_s = \frac{\lambda_s}{(\epsilon + \mathcal{SI}_s)^b}. \tag{35}$$

The smoothness indicator $\mathcal{SI}_m$ is given by:

$$\mathcal{SI}_s = \sum_{1 \le |\beta| \le r} \int_{V_0'} \left(\mathcal{D}^\beta p_s(\xi,\eta,\zeta) \right)^2 (d\xi, d\eta, d\zeta), \tag{36}$$

where β is a multi-index, r is the polynomial's order, λ_m is the linear weight. The central stencil is assigned a large linear weight λ_1 while the remaining directional stencils are assigned a value of $\lambda_s = 1$ and a value to prevent division by zero of $\epsilon = 10^{-6}$ is used, $b = 4$ and $\mathcal{D}$ is the derivative operator. The smoothness indicator is a quadratic function of the degrees of freedom (a_k^s) and Eq. (36) can be rewritten as:

$$\mathcal{SI}_s = \sum_{k=1}^{K} a_k^s \left(\sum_{q=1}^{K} \mathcal{OI}_{kq} a_q^s \right), \tag{37}$$

where the oscillation indication matrix $\mathcal{OI}_{kq}$ is given by:

$$\mathcal{OI}_{kq} = \sum_{1 \le |\beta| \le r} \int_{V_0'} \left(\mathcal{D}^\beta \phi_k(\xi,\eta,\zeta) \right) \left(\mathcal{D}^\beta \phi_q(\xi,\eta,\zeta) \right) (d\xi, d\eta, d\zeta), \tag{38}$$

and can be precomputed and stored at the beginning of the simulation. The WENO reconstruction can be carried out with respect to the characteristic variables, and the reader is referred to [50,51] and references therein regarding the implementation. For the directional stencils, several algorithms have been developed [62] and are available in UCNS3D, with the default being Type 3, due to its favourable balance in terms of robustness and computational cost.

3.1.4. CWENO

The CWENO scheme developed by Tsoutsanis and Dumbser [1] improves both the computational efficiency and robustness of the original WENO schemes. The key characteristic of the CWENO scheme is the combination of an optimal (high-order) polynomial p_{opt} using the central stencil with lower-order polynomials employing the directional stencils. At the presence of smooth data the optimal polynomial is recovered and the desired-order of accuracy is obtained, whereas at the presence of discontinuous data at least one of the lower-order polynomials (from the directional stencils) could contain smooth data, therefore reducing the oscillations in the computed solution. All the polynomials involved are subject to the same requirements as previously set of matching the cell averages of the solution. The computational savings compared to the WENO schemes arise from the reduced size of the directional stencils, and the fact that the directional stencils are contained in the central stencil, as it can be seen in Fig. 2. The definition of an optimal polynomial given by:

$$p_{opt}(\xi,\eta,\zeta) = \sum_{s=1}^{s_t} \lambda_s p_s(\xi,\eta,\zeta), \tag{39}$$

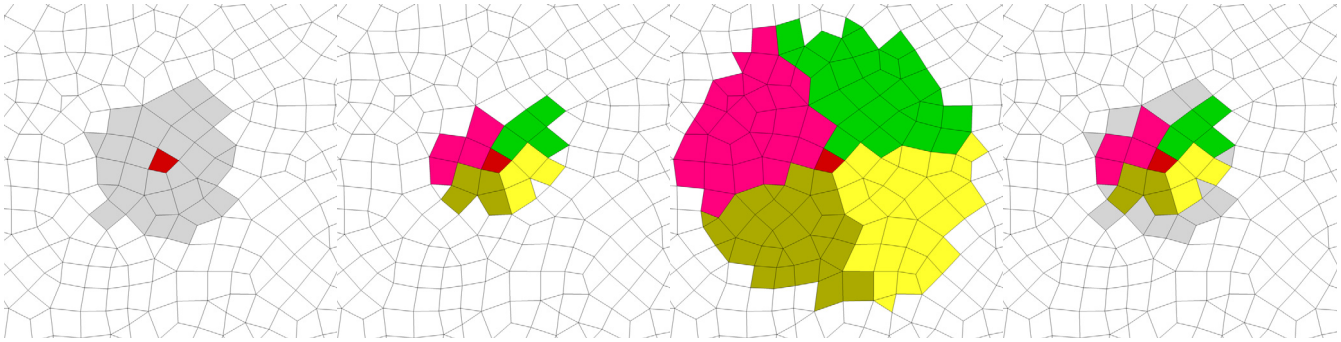
where s is the stencil index, with $s = 1$ being the central, $s = (2, 3, .., s_t)$ being the directional, s_t being the total number of stencils, and λ_s being the linear coefficient for each stencil, whose sum is equal to 1. The p_1 polynomial is not computed directly, but computed by subtracting the lower-order polynomials from the optimum polynomial as follows:

$$p_1(\xi,\eta,\zeta) = \frac{1}{\lambda_1} \left(p_{opt}(\xi,\eta,\zeta) - \sum_{s=2}^{s_t} \lambda_s p_s(\xi,\eta,\zeta) \right). \tag{40}$$

The CWENO reconstruction polynomial is a non-linear combination of all the polynomials as follows:

$$p(\xi,\eta,\zeta)^{\text{cweno}} = \sum_{s=1}^{s_t} \omega_s p_s(\xi,\eta,\zeta), \tag{41}$$

where ω_s correspond to the non-linear weights assigned to each polynomial, and in regions with smooth data $\omega_s \approx \lambda_s$, hence obtaining the high-order approximation from the central stencil, and



(a) Central Stencil (b) CWENO/CWENOZ directional stencils (c) WENO directional stencils (d) CWENO/CWENOZ stencils

Fig. 2. Examples of central and directional stencils for $\mathcal{P}=4$ for the WENO, and CWENO/CWENOZ schemes. The considered cell is illustrated in red colour, the central stencil elements by grey colour, and each of the directional stencils illustrated by different colours. It can be noticed that the CWENO/Z schemes employ significantly smaller directional stencils compared to WENO schemes. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

in regions of discontinuous solutions the reconstructed solution will be mostly influenced from the lower-order polynomials of the directional stencils, where $\tilde{a}_k$ are the reconstructed degrees of freedom; and the non-linear weight ω_s is defined as:

$$\omega_s = \frac{\tilde{\omega}_s}{\sum_{s=1}^{s_t} \tilde{\omega}_s} \quad \text{where} \quad \tilde{\omega}_s = \frac{\lambda_m}{(\epsilon + \mathcal{SI}_s)^b}. \tag{42}$$

Similarly, to the WENO scheme $\epsilon = 10^{-6}$ is used and $b=4$. For the present study we employ $r=1$ for the directional polynomials resulting in 2nd-order of accuracy, and any arbitrary order of accuracy for the polynomial associated with the central stencil. The smoothness indicators used for CWENO scheme are the same as in the WENO scheme defined previously. The linear weights are assigned by firstly assigning the non-normalised linear weight for the central stencil λ_1' an arbitrary value, and then normalising this as follows:

$$\lambda_1 = 1 - \frac{1}{\lambda_1'}, \tag{43}$$

with the linear weights associated with lower-order polynomials being assigned the same linear weights as follows:

$$\lambda_s = \frac{1-\lambda_1}{s_t - 1}, \tag{44}$$

where s_t is the total number of stencils.

3.1.5. CWENOZ

The CWENOZ scheme developed by Tsoutsanis and Dumbser [1] follows in principle the CWENO scheme, the key differentiation being the approximation of the non-linear weights. The definition of the optimal polynomial remains the same as before, and the CWENOZ reconstruction polynomial is given as a non-linear combination of all the polynomials in the following manner:

$$p(\xi,\eta,\zeta)^{\text{cwenoz}} = \sum_{s=1}^{s_t} \omega_s p_s(\xi,\eta,\zeta), \tag{45}$$

where ω_s correspond to the non-linear weights assigned to each polynomial. The WENOZ strategy of combining unequal degree polynomials was introduced by Borges et al. and Castro et al. [71,72] and has been adapted for unequal polynomials, reconstruction stencils and arbitrary elements as has recently been reported by [9,73]. The non-linear weights are now defined as:

$$\omega_s = \frac{\tilde{\omega}_s}{\sum_{s=1}^{s_t} \tilde{\omega}_s} \quad \text{where} \quad \tilde{\omega}_s = \lambda_s \left(1 + \frac{\tau}{\epsilon + \mathcal{SI}_s}\right). \tag{46}$$

With τ being the universal oscillation indicator and taken as the absolute difference between the smoothness indicators as follows:

$$\tau = \left(\frac{\sum_{s=2}^{s_t} |\mathcal{SI}_s - \mathcal{SI}_1|}{s_t - 1} \right)^b. \tag{47}$$

Similarly to the WENO scheme $\epsilon = 10^{-6}$ is used and $b=4$. We employ $r=1$ for the directional polynomials resulting in 2nd-order of accuracy, and any arbitrary order of accuracy for the polynomial associated with the central stencil, while the linear weight's assignment procedure is the same with the CWENO.

3.2. Low-Mach number treatment

In low-Mach number regions, it is known that Godunov type schemes exhibit an artificially large velocity jump at the cell interfaces, as it was demonstrated by Thornber et al. [74]. The proposed solution was to modify the extrapolated reconstructed values at the cell interfaces in order to take into account the correct flow physics of low speed flows. Alternative low-Mach number treatments include the semi-implicit pressure based approaches [75–81] that have been successfully applied for all Mach and low-Mach number flow problems. It was later on demonstrated by Simmonds et al. [60] that for unstructured meshes the original low-Mach number treatment did not provide favourable and robust improvements, since only the normal velocity jumps at the cell interfaces required modifications and not the tangential ones. Therefore, the `UCNS3D` employs the low-Mach number treatment of Simmonds et al. [60] where only the normal components $\hat{n}$ of the velocity at the interfaces are modified, and not their tangential $n_{\parallel}$.

$$\begin{aligned} u_L^{\hat{n}*} &= \frac{(1+z)u_L^{\hat{n}} + (1-z)u_R^{\hat{n}}}{2}, \\ u_R^{\hat{n}*} &= \frac{(1+z)u_R^{\hat{n}} + (1-z)u_L^{\hat{n}}}{2}, \\ z &= min(1, max(M_L, M_R)) \end{aligned} \tag{48}$$

with M_L, M_R corresponding to the Mach number based on the left and right-states respectively, and $u_L^{\hat{n}*}$, $u_R^{\hat{n}*}$ denotes the modified extrapolated normal component of velocities for the left- and

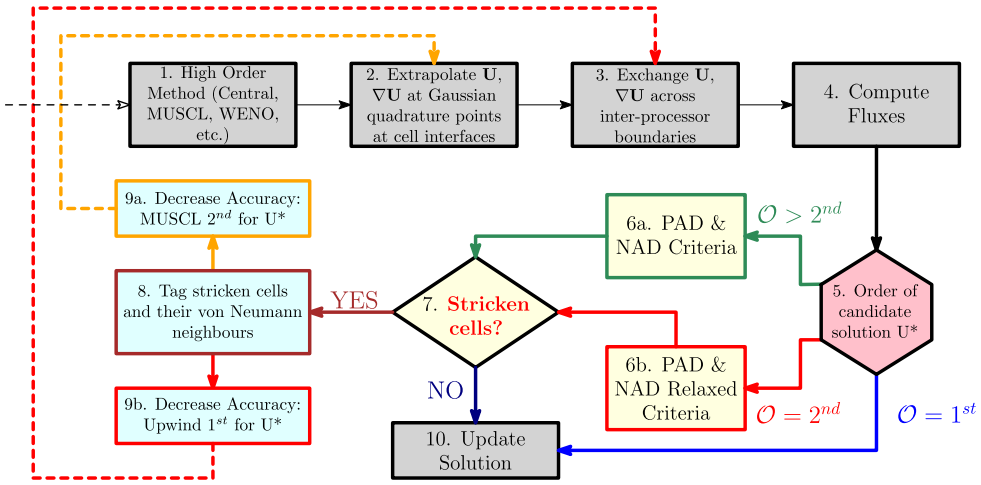


Fig. 3. Flow-chart of the MOOD-augmented UCNS3D code. In its current configuration, the high-order numerical scheme selected by the user (Central, CWENOZ, MUSCL, etc.) "piggybacks" the *a posteriori* MOOD algorithm. The switch of accuracy criteria of the PAD/NAD sensors for the parachuting and the bulletproof scheme is also shown as steps 5, 6a, and 6b. The steps with dark orange outlines refer to the transition from a higher order of accuracy to the 2^{nd} order MUSCL scheme, the first *auxiliary*, while those outlined with red refer to *parachuting* the solution to the *bulletproof* first order accurate Godunov's flux. A solution decremented to first order is exempted of the PAD/NAD due to its conservativity property.

right-states respectively, and $u_L^{\hat{n}}$, $u_R^{\hat{n}}$ denote the reconstructed extrapolated values for the velocity in the direction normal to the face/edge for the left- and right-states respectively. The local Mach number for the left and right states M_L, M_R is calculated based on the velocity magnitude of all the velocity components independent of the normal direction. It has to be noted that the low-Mach number treatment is mostly benefiting 2nd-order and 3rd-order schemes, since the velocity jumps are reduced when employing very high-order schemes. More importantly the hexahedral and quadrilateral elements are the ones that experience the most notable improvements, whereas the triangular, tetrahedral, prisms, and pyramids are rather less sensitive to the treatment due to their increased degrees of freedom (orientation), something that has also been reported by Rieper [82].

3.3. MOOD

The Multidimensional Optimal Order Detection (MOOD) algorithm, introduced by Clain et al. [17], has been established as a unique *a posteriori* method to enhance the non-oscillatory properties of any numerical method. A further improvement to the original method, developed by Farmakis et al. [40], is the method of choice implemented in UCNS3D, and has been tested successfully for very intensive computational problems with sharp discontinuities, density and velocity gradients, f.e. high-speed 2D and 3D converging flows perturbed by the Rayleigh-Taylor and Richtmyer-Meshkov instabilities. The key tenet of the MOOD algorithm is that for all the cells in the computational domain, a candidate solution should be both physically and numerically admissible. If these conditions are not satisfied in a given cell, the order of accuracy of the scheme is reduced locally until the candidate solution is admissible. Specifically, following the solution of the fluxes, each candidate solution calculated using any of the high-order methods implemented in UCNS3D (such as Central, CWENO, MUSCL, etc.) will be passed through two detectors:

1. *Physical Admissible Detector* (PAD): This detector checks that the solution is physical, that is, all points must have positive density and positive pressure at all times. In essence, this detector will identify any point exhibiting NaN values.
2. *Numerical Admissible Detector* (NAD) [5]: This is a more flexible version of the Discrete Maximum Principle (DMP) [17]. The detector checks that the solution is monotonic, and that no new extrema are created. It compares the candidate solution with the solution obtained in the previous Runge-Kutta step.

If the candidate solution does not satisfy the PAD & NAD criteria, then the code *parachutes* into a 2nd-order MUSCL scheme, and a relaxed version of the PAD & NAD criteria is applied again. This transition with more relaxed criteria has been strategically opted to maintain the solution of a cell both admissible and of the highest possible order. In case the new candidate solution is also not admissible, then a *bulletproof* scheme is used. For UCNS3D, this is Godunov's flux. Being a first-order Upwind scheme, Godunov's method is known to satisfy the PAD & NAD criteria by definition. At this point, it should be noted that MOOD's *a posteriori* nature gives rise to one important complication. If a solution is not admissible in a *single* cell, this entails that the solutions for its neighbouring cells will have to be recomputed as well; these solutions might be contaminated by the stricken cell's solution through the flux operation. If not addressed, contaminated solutions might lead to a cascade of non-admissible solutions in subsequent iterations. It is for this reason that when a cell is marked by application of the PAD & NAD criteria, its neighbouring cells are also identified accordingly. This enables the creation of the necessary stencil, as anticipated by the next scheme up the MOOD cascade. The entire operation of the MOOD implementation of Farmakis et al. in UCNS3D can be seen in the flow chart in Fig. 3. The collection of cells $\mathcal{V}_i$ represents the set of first neighbours of the cell i in consideration. In its present form, the NAD criterion assesses the conservative variables' vector, as suggested by [64]. The candidate solution $\mathbf{U}_i^*$ for a cell i during any Runge-Kutta stage should be within a certain range provided by the $\mathcal{V}_i$ region defined previously. The superscript $(^n)$ indicates here the previous Runge-Kutta step, not the previous time step:

$$\min_{y\in\mathcal{V}_i}(\mathbf{U}^n(y)) - \delta \leq \mathbf{U}_i^* \leq \min_{y\in\mathcal{V}_i}(\mathbf{U}^n(y)) + \delta\,. \tag{49}$$

The original DMP-relaxed margins, as introduced by [83], can be seen in step 6a. of the flow chart. Denoted by $(^\circ)$, they read:

$$\delta^\circ = \max_{y\in\mathcal{V}_i}\left(10^{-4}, 10^{-3}\cdot\left[\max_{y\in\mathcal{V}_i}(\mathbf{U}^n(y)) - \min_{y\in\mathcal{V}_i}(\mathbf{U}^n(y))\right]\right), \tag{50}$$

while the "tempered" criteria, employed in step 6b. of the flow chart and denoted by $\binom{R}{}$, ensure a prudent, as well as smooth transition along the cascade, from second to first order of accuracy. They are selected as:

$$\delta^R = \max_{y\in\mathcal{V}_i}\left(10^{-4}, 10^{-1}\cdot\left[\max_{y\in\mathcal{V}_i}(\mathbf{U}^n(y)) - \min_{y\in\mathcal{V}_i}(\mathbf{U}^n(y))\right]\right). \quad (51)$$

The relaxed criteria, developed for the UCNS3D, reduce significantly the computational overhead of the MOOD method, without reducing the efficacy of the framework to ensure the non-oscillatory properties solution. This has to do also with the favourable non-oscillatory properties of the limiters employed for the 2nd-order MUSCL scheme, for instance the Barth and Jespersen limiter [68]. For more details regarding the implementation and testing of the algorithm, the readers are referred to the work of Farmakis et al. [40].

3.4. Rotating frame domain decomposition

The frame of reference is defined at the solver level with no need of mesh-based interface. This approach employs a shape geometry formulation on the solver algorithm to identify in which reference frame the element is located. The rotational zone is defined as a cylinder defined by two points coordinates and radius. These input parameters as well as the rotational velocity are defined on the MRF.DAT file.

The velocity components are computed in absolute formulation within the appropriate flux correction according to the element frame of reference. The rotational velocity is computed at each Gaussian quadrature points taking into account its relative distance to the rotating axis. Two additional computation tasks are performed on the calculation process to change the frame from inertial frame to rotational: inclusion of extra source term and flux correction. For a more detailed description of the implementation and validation of this technique, the reader must refer to [4,84].

3.5. Fluxes approximation

For the inviscid fluxes several well-established and verified approximate Riemann solvers are employed such as the approximate HLLC (Harten-Lax-van Leer-Contact) Riemann solver of Toro et al. [85,86], the Roe Riemann solver [87], and the Rusanov Riemann solver [88]. For the viscous fluxes, the gradients are computed using the previously described least-square linear reconstruction with additional constraints for the boundary conditions as detailed in [50,89]. Only in the presence of bad-quality elements, a Green Gauss gradient approximation may be used, as described in the work of Tsoutsanis et al. [50]. For the viscous stress tensor and the heat flux vector in the Navier-Stokes equations, the gradients of the discontinuous states for the approximation of the viscous fluxes are averaged by including the penalty terms similar to previous approaches [90–92] in the following manner:

$$\nabla\mathbf{U} = \frac{1}{2}\left(\nabla\mathbf{U}_L + \nabla\mathbf{U}_R\right) + \frac{\alpha}{L_{int}}\left(\mathbf{U}_R - \mathbf{U}_L\right)\vec{n}, \quad (52)$$

where L_{int} is the distance between the cell centres of adjacent cells, and $\alpha = 4/3$ similarly to previous approaches [90,91].

3.6. Temporal discretisation

The temporal discretisation in UCNS3D follows the method of lines paradigm, whereas a separate discretisation of space and time can be deployed, since it offers favourable flexibility.

3.6.1. Steady simulations

For steady state simulations, Eq. (20) can be rewritten in the following semi-discrete form

$$\frac{d\mathbf{U}_i}{dt} = \mathbf{R}_i, \quad (53)$$

where $\mathbf{R}_i$ is the right-hand side residual, which should converge to the machine zero. The two options available include an explicit first-order forward Euler time stepping [93], and an implicit first-order backward Euler time where Eq. (53) takes the following form for the latter:

$$\frac{d\mathbf{U}_i^n}{dt} = \frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{dt} = \mathbf{R}_i^{n+1}. \quad (54)$$

Linearising in time, Eq. (54) gives

$$\frac{d\mathbf{U}_i^n}{dt} = \mathbf{R}_i^n + \frac{\partial\mathbf{R}_i^n}{\partial\mathbf{U}}, \quad (55)$$

where $\mathbf{R}_i$ should be equal to zero, hence (55) becomes

$$\left(\frac{\mathbf{I}}{dt} - \frac{\partial\mathbf{R}_i^n}{\partial\mathbf{U}}\right)d\mathbf{U}_i = \mathbf{R}_i^n. \quad (56)$$

The solution at each element i is updated via $\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + d\mathbf{U}_i$, where $\mathbf{I}$ stands for the identity matrix. The term $\frac{\partial\mathbf{R}_i^n}{\partial\mathbf{U}}$ stands for the Jacobian matrix and contains the linearisation of the inviscid, viscous flux vectors as well as the source terms. A first-order approximation of the numerical fluxes is employed for the approximation of the Jacobians since due to its simplicity, compact stencil support, and memory requirements. The simple flux function chosen is based on the Rusanov flux, given as

$$\begin{aligned}\mathbf{R}_i\left(\mathbf{U}_i, \mathbf{U}_j, n_{ij}\right) =& \frac{1}{2}\left(\mathbf{F}_{\mathbf{c},\mathbf{v}}^{\mathbf{n_{ij}}}(\mathbf{U}_i, \nabla\mathbf{U}_i) + \mathbf{F}_{\mathbf{c},\mathbf{v}}^{\mathbf{n_{ij}}}(\mathbf{U}_j, \nabla\mathbf{U}_j)\right) \\ &- \frac{1}{2}|\lambda_c^{ij} + \lambda_v^{ij}|\left(\mathbf{U}_j - \mathbf{U}_i\right),\end{aligned} \quad (57)$$

with the maximum convective and viscous eigenvalue written respectively

$$\begin{aligned}\lambda_c^{ij} &= |\mathbf{V}_{ij}\cdot\mathbf{n}_{ij}| + a_{ij}, \\ \lambda_v^{ij} &= \frac{\mu_{ij}}{\rho_{ij}|x_j - x_i|},\end{aligned} \quad (58)$$

where $\mathbf{n}_{ij}$ is the normal vector to the element interface, $\mathbf{V}_{ij}$ is the velocity vector and a_{ij} is the speed of sound. The viscous eigenvalue is approximated by the viscous spectral radius $x_j - x_i$. Within UCNS3D the linear system of Eq. (56) with the Rusanov flux can be solved by a matrix-free lower-upper symmetric Gauss-Seidel (LU-SGS). The original LU-SGS method was proposed for the structured meshes [94,95]. We use its extension to unstructured meshes [96,97], with our implementation similar to [98,99], an LU-SGS implementation with full diagonal and off-diagonal elements as described in [53], and a simple block Jacobi method. Finally, the boundary conditions are also treated implicitly according to Batten et al. [100]. Both the explicit and implicit time-stepping schemes are augmented by local time-stepping for convergence acceleration.

3.6.2. Unsteady simulations

For transient simulations UCNS3D employ explicit Strong Stability Preserving (SSP) Runge-Kutta methods ranging from 1st- to 4th-order of accuracy in time, including the implementations of Gottlieb and Shu [93], and Spiteri and Ruuth [101] where the readers are referred for further details. For flows problems that we

want to not be limited by the severe CFL limit of explicit schemes, the well established and widely used dual-time stepping strategy [102] is employed where the transient problem is solved as a series of steady-state problems, and therefore the previously established techniques can be used to accelerate the convergence to steady state. Rewriting Eq. (20) in the following semi-discrete form

$$\frac{d\mathbf{U}_i}{d\tau} = \mathbf{R}_i^*(\mathbf{U}^*)/V_i^{n+1}, \tag{59}$$

with the unsteady residual defined as:

$$\begin{aligned}\mathbf{R}_i^*(\mathbf{U}^*) =& \mathbf{R}_i(\mathbf{U}^*) + \frac{3}{2\Delta t}(V_i^{n+1}\mathbf{U}_i^*) - \frac{2}{\Delta t}(V_i^n\mathbf{U}_i^n) \\ &+ \frac{1}{2\Delta t}(V_i^{n-1}\mathbf{U}_i^{n-1}).\end{aligned} \tag{60}$$

Where τ is the pseudo-time, Δt is the physical time, and $\mathbf{R}_i(\mathbf{U}^*)$ is the residual. When the desired convergence to the pseudo-steady state problem is achieved, then $\mathbf{U}_i^{n+1} = \mathbf{U}_i^*$.

3.7. Source terms

The source terms are approximated within each control volume, with Gaussian quadrature rules suitable for the polynomial order selected for the spatial discretisation. Special treatment is applied only in the source term of the 5 equation multicomponent Euler Eq. (11). In that case the approach of Johnsen and Colonius [103] is adopted and the source term is numerically approximated as surface integral, rather than a volume one, while using the same velocity estimate as the one used for the evaluation of the fluxes as shown below:

$$\int_{V_i} a_1 \nabla \cdot \mathbf{u}\, dV \approx \int_{V_i} a_1\, dV \cdot \int_{\partial V_i} (u_n)^{Riem.}\, dS. \tag{61}$$

4. Overview and features

In this section, we provide an overview of UCNS3D, including the package installation, the features, the code structure, description on input and output files. We summarise the implementation details of the software and how several aspects of the governing equations and numerical framework previously introduced in Sections 2 and 3 are represented in the code, so that other scientists can leverage the high-order framework for their intended applications and for further developments.

4.1. Package installation

UCNS3D is available at http://www.ucns3d.com. It is written in Fortran 2008 with MPI [104] and OpenMP [105] embedded functionality. UCNS3D utilises BLAS libraries [106] for several floating-point operations such as Matrix-Vector and Matrix-Matrix multiplications. UCNS3D also employs the ParMetis and Metis libraries [107] for partitioning the computational domain, and the TecIO library [108] for writing binary files using the Tecplot interface. The UCNS3D package includes several directories and their description is provided in Table 1. The users should consult the README file at https://github.com/ucns3d-team/UCNS3D for instructions on how to compile and run UCNS3D, and the following set of features refer to version UCNS3D 1.3 version. The current operating systems supported are linux distributions such as Ubuntu, Suse, Redhat, CentoS and MacOS as well as in Windows 10 using WSL and WSL2. The user should ensure that the correct locations of the selected compilers and libraries are defined prior to the compilation of UCNS3D and that the appropriate options are selected in the Makefile file. The users are referred to https://github.com/ucns3d-team/UCNS3D for instructions on how to compile and run the solver.

Table 1
Description of the files included in UCNS3D.

Name	Description
CODE/	Source code files
CODE/Makefile	selection of compiler and libraries to be used
CODE/Makefile.common	Makefile input, does not need to be modified
SCRIPTS/	Example scripts for several HPC systems
ARCHER_LIB/	Example makefile for ARCHER2
MacOS/	Instruction and makefile for MacOS
TRANSLATORS/	Translators from STAR-CD, UGRID to native format
LICENSE	The GNU public license file
README	Readme file with instructions on how to compile, install and run UCNS3D
PARAMETERS_README	Detailed description of every option of the parameter file UCNS3D.DAT
README_TESTS	Description of the tests provided
README_UCNS3D	Description of how to develop solution profiles and boundaries in UCNS3D

4.2. Features

UCNS3D implements both two- and three-dimensional unstructured mesh framework, that consists of triangular, quadrilateral, prism, hexahedral, pyramid, and tetrahedral elements. It employs a wide range of spatial numerical schemes of order ranging from 1st to 7th-order. These schemes include MUSCL, WENO, CWENO, CWENOZ, and MOOD. Temporal scheme order ranges from 1st to 4th-order, in either explicit or implicit time-stepping methods, including SSP Runge-Kutta methods and dual-time stepping method. Gradient calculation algorithms include the least-square and Green-Gauss methods, as well as several well-established approximate Riemann solvers such as HLL, HLLC, Roe, etc. Spatial integrals are approximated via multidimensional Gaussian quadrature rules of compatible order with the chosen spatial order. The key details of the methods employed are outlined in Section 3. UCNS3D can solve several compressible flow problems governed by the following equations: Euler, Navier-Stokes, Reynolds-Averaged Navier-Stokes, 5-equation multi-component flows, as detailed in Section 2. Various boundary conditions have been implemented such as periodic, no-slip wall, inflow, outflow and pressure far-field. New features that are under consideration or currently under implementation are discussed in Section 7.

UCNS3D is a fully parallel solver employing both MPI and OpenMP interfaces, and it has been developed with practicality in mind, hence operations such as mesh partitioning are executed during runtime, while the MPI-IO library is utilised to write/read binary checkpoint files and take advantage of scalable IO operations supported by HPC infrastructure. UCNS3D has been successfully ported and tested in several HPC facilities including ARCHER, ARCHER2, HLRS-Hawk, HLRS-Hazelhen, SuperMUC, CSD3, and MARCONI [41,42,109,110]. And the performance on a representative benchmark is outlined in Section 6. UCNS3D is provided with a representative collection of 2D and 3D tests including the mesh files and parameter files as a starting point for users to familiarise and validate any further development done. The tests are provided in the https://doi.org/10.5281/zenodo.3375432 repository.

4.3. Code structure

The only requirement for the mesh files for UCNS3D is that the mesh files should include the boundary conditions. Therefore, regardless of the mesh file format provided the user should ensure that the boundary conditions are specified during the grid-generation process. UCNS3D solves the governing equations with

Table 2
Input files for UCNS3D.

Name	Description
ucns3d_p	UCNS3D executable
GRID.bnd	Grid boundary file
GRID.cel	Grid connectivity file
GRID.vrt	Grid vertex coordinates file
UCNS3D.DAT	Parameter file for controlling the operation of UCNS3D (including equations, discretisations, I/O operations, turbulence modelling etc)
MRF.DAT	Parameter file for the Multiple Reference Framework of UCNS3D
MOOD.DAT	Parameter file controlling the operation for the MOOD algorithm of UCNS3D
MULTI.DAT	Parameter file specifying all the components for the 5-equation multicomponent model of UCNS3D

Table 3
Output files for UCNS3D.

Name	Description
Errors.dat	Errors in $\mathcal{L}_\infty$ and $\mathcal{L}_2$ error norms, and computational times
Statistics.dat	Computational times statistics, for scalability benchmarks
residual.dat	file containing the residuals of the conserved variables iterations for steady state flow problems
history.txt	History of the operations performed during the simulation
FORCE.dat	History of the Lift and Drag forces
RESTART.dat	Checkpoint file with instantaneous (or converged) solution
RESTART_AV.dat	Checkpoint file with time-averaged solution
OUT_*.	instantaneous (or converged) solution of the entire domain (binary or ASCII, Tecplot or vtk output)
SURF_*	instantaneous (or converged) solution for the wall bounded surfaces/edges of the domain (binary or ASCII, Tecplot or vtk output)
OUT_AVER_*	time-averaged solution of the entire domain (binary or ASCII, Tecplot or vtk output)
SURF_AVER_*	time-averaged solution for the wall bounded surfaces/edges of the domain (binary or ASCII, Tecplot or vtk output)
PROBE_*	time-history of solution in terms of primitive variables for the numbered probe position (ASCII)

the prescribed boundary conditions, with the numerical method selected, until either the maximum number of iterations, the maximum wall-clock time, or the final time is reached. UCNS3D exports history, statistics, and solution files in either ASCII or binary format that can be viewed in visualisation software packages such as Tecplot [111], Paraview [112], Visit [113] etc.

The code is structured in individual Fortran modules, each one of them is associated with one particular operation such as boundary conditions, Riemann solvers, mesh partitioning, time discretisation, parallel communications, solver parameters, input/output operations, etc. The naming convention for all the module files follows the operation that each module executes.

4.4. Description of input/output files

UCNS3D requires the executable ucns3d_p in the same directory with the files associated with the grid that can be provided in either of the popular formats of fluent *.msh, in UGRID format *.ugrid, or STAR-CD format STAR files. The file that controls the operation of UCNS3D is the parameter file UCNS3D.DAT, where all the options are outlined. If other parameter files are present then the relative mode of UCNS3D will be enabled. For instance, if the MRF.DAT file is present in the same directory then the multiple-reference framework will be engaged. Depending on the parameters and options selected, several files will be generated including binary checkpoint (restart) file, solution files for visualisation, force file, statistics for parallel benchmarking, etc. The list of input and output files is provided in Table 2 and Table 3 respectively.

4.5. Implementation

UCNS3D has been implemented by utilising several modules, where each of the modules can contain several functions and subroutines. The modules can be categorised with respect to the nature of the operations contained in them. For example there are certain modules associated with the geometrical operations while others are associated with flow operations, as shown in Fig. 4. Therefore the modules of UCNS3D are described in the relevant sections that follow.

4.5.1. Geometry

This category of modules, is related to the several geometrical operations required including computing cell-centres, finding neighbours, building stencils, etc.

1. ***grid_p.f90***: contains several operations associated with the unstructured mesh such as the connectivity including subroutines dedicated to finding the direct side neighbours, the construction of the central and directional stencils as shown in Fig. 2 and several stencil selection algorithms [62], finding the shape of every element etc.
2. ***grid_t.f90***: consists of the operations related to computing several geometrical characteristics of the unstructured mesh such as the volume, edges, cell-centres, Gaussian quadrature points. The normal vectors at the surfaces/edges as shown in Fig. 1 and their corresponding rotation matrix and their inverse are also computed for projecting the solution along the direction of the normal at the cell-face and project back, and finally a subroutine that finds the cells associated with each probe position defined by the user is included.
3. ***bc_p.f90***: the subroutines that determine the boundary conditions for every bounded cell, and also determining the corresponding periodic cells make up this module.

4.5.2. Polynomials

This category of modules, is related to the operations such as computing basis functions for the reconstruction polynomials and the derivatives of them.

1. ***basis_p.f90***: contains the functions for computing the basis ψ_k as defined in Eq. (28) for any given point for any cell, according to the order and type of polynomial for 2D and 3D.
2. ***der_r.f90***: consists of functions that compute the derivatives $\mathcal{D}^\beta \phi_k(\xi, \eta, \zeta)$ for the generic and Legendre polynomials, that are required in several places such as the approximation of the smoothness indicators in Eq. (38), for the approximation of the viscous stresses from the least-square polynomial Eq. (6) etc.

4.5.3. Reconstruction

This category of modules, is related to multiple operations required for obtaining reconstructed solutions. They can be further subcategorised to Initial, Schemes and Matrix as follows:

Initial

The initial category, refers to operations that can be performed prior to advancing the solution, since in the case where the mesh remains fixed in time, the simulation speed can be reduced by limiting the number of operations that involve matrices that do not change with time. Such examples include the least-square matrices etc.

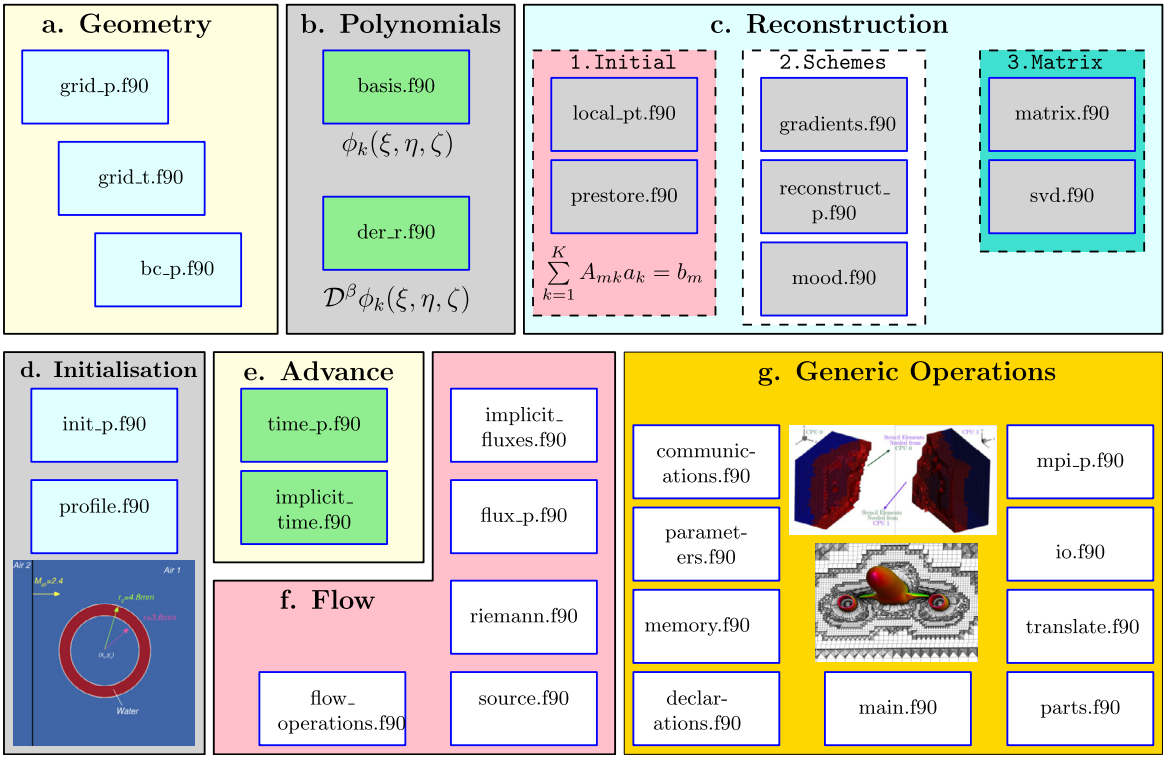


Fig. 4. Drawing of the UCNS3D modules arranged according to the nature of their contained operations: a) Geometry, b) Polynomials, c) Reconstruction d) Initialisation e) Flow g) Generic Operations.

1. ***local_pt.f90***: contains all the subroutines related to the transformation of the cells and their associated stencils from physical space (x, y, z) to a reference space (ξ, η, ζ) as defines in [49], and a subroutine for deciding which gradients approximation methods to employ for different cells based on quality characteristics of the cells.
2. ***prestore.f90***: mainly consists of subroutines for prestoring the Moore-Penrose pseudo-inverse of matrix A_{mk} reconstruction matrix as shown in Eq. (30) for each stencil of every cell, as well as the oscillation indication matrix $\mathcal{OI}_{kq}$ for the WENO, CWENO, CWENOZ variants as defined in Eq. (38).

Schemes

The schemes category, refers to operations that are dependent upon the spatial-discretisation scheme chosen, and it involves procedures to compute the gradient of the flow variables, and perform the reconstruction.

1. ***gradients.f90***: contains several subroutines for computing the gradients for the conserved variables required for the convective fluxes, and the velocity and temperature gradients required for the diffusive fluxes. Several subroutines are available for computing the gradients using a least-square or Green-Gauss method.
2. ***reconstruct_p.f90***: all the subroutines dedicated to the reconstruction process, including the central, MUSCL, WENO/ CWENO/ CWENOZ variants as defined in subsections 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5 respectively are contained in this module.
3. ***mood.f90***:this module contains the PAD and NAD criteria for the MOOD operator of Eq. (51) and Eq. (50), as well as a subroutine that detects which neighbouring cells need to be fixed as well.

Matrix

The matrix category, refers to operations that are related to the solution of the least-square matrix problem for obtaining the unknown coefficients (degrees of freedom) of the polynomials.

1. ***svd.f90***: contains the subroutines for the SVD decomposition solution procedure for the least square reconstruction process of the system shown in Eq. (30).
2. ***matrix.f90***: contains the subroutines for the QR decomposition using a Householder transformation for the least square reconstruction process shown in Eq. (30).

4.5.4. Initialisation

This category of modules, is related to all the operations for initialising the flow field, either from user-defined profiles, or from check-point (RESTART) files.

1. ***init_p.f90***: consists of the operations for the initialisation of the flow-field either from prescribed initial solution profiles, or from a previous simulation and the associated check pointing file.
2. ***profile.f90***: contains subroutines and functions dedicated for specifying the initial condition of the flow-field. There are several previously defined initial condition profiles that correspond to well established test problems such as the Taylor-Green vortex, the 2D and 3D explosion, the double Mach reflection and others. The users can use these subroutines for introducing new initial condition profiles of their choice.

4.5.5. Advance

This category of modules, is related to the time advancement of the solution, and the several methods available to perform this operation.

1. ***time_p.f90***: contains the subroutines associated with activities related to the time-advancement of the solution. These activities include computing the time-step size (with global or local time-stepping technique), explicit SSP Runge-Kutta time discretisation schemes for transient flow problems, implicit and explicit time-stepping schemes for steady state-problems, and implicit and explicit time-stepping schemes using the dual-time stepping paradigm.
2. ***implicit_time.f90***: all the subroutines responsible for solving the linear system of Eq. (56) for the implicit-time stepping method, using either a block-Jacobi or an LU-SGS method [98,99] are contained in this module.

4.5.6. Flow

This category of modules, is related to all the flow-operations including computing viscosity, using approximate Riemann solvers, computing the inviscid and viscous fluxes, and source terms.

1. ***flow_operations.f90***: consists of functions and subroutines for flow-related operations such as applying the boundary conditions, computing the viscosity based on Sutherland's law as seen in Eq. (4), transforming from primitive to conservative variables and vice versa, transforming from conservative to characteristic variables and vice versa, applying the low-Mach number correction as defined in Eq. (48), customising the inflow/outflow boundary conditions, computing the instantaneous and time-averaged shear stresses, computing the Q-criterion, computing the Jacobian $\frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}}$, Eigenvectors of the convective and diffusive fluxes, and a subroutine dedicated to tracking the interfaces as shown in Fig. 19, volume and other statistics for multi component flow applications.
2. ***implicit_fluxes.f90***: includes the operations for building the linear system $\left(\frac{\mathbf{I}}{dt} - \frac{\partial \mathbf{R}_i^n}{\partial \mathbf{U}}\right)$ as shown in Eq. (56) for the implicit-time stepping method including the diagonal and off-diagonal element matrices, or the through the matrix-free approach of [98,99].
3. ***flux_p.f90***: consists of the subroutines dedicated for computing the convective and diffusive fluxes in 2D and 3D across all the faces/edges and Gaussian quadrature points as defined in Eq. (20).
4. ***riemann.f90***: all the subroutines that each one of them corresponds to a particular 2D or 3D approximate Riemann solver including the HLLC, Rusanov, Roe, and other variants of them, are included in this module.
5. ***source.f90***: contains the subroutines for computing the source terms for the corresponding governing equations selected, and the associated turbulence model as shown in Eq. (7).

4.5.7. Generic operations

This category of modules, is related to several operations related to the operation of the code, including variable declarations, reading and writing files, MPI communications, partitioning the mesh, translating from one mesh input to another, as well as memory management.

1. ***communications.f90***: includes all the subroutines related to establishing and performing the communication for the boundary extrapolated values across inter-processor boundaries and the communication for the stencil cells across inter-processor boundaries.
2. ***declarations.f90***: contains the definition of all the global object data types and variables, and the OpenMP status for the private variables. The definition of the variables is detailed in their commented section.
3. ***io.f90***: consists of most of the I/O operations including reading the grid files, and writing the solution files in several formats such as Tecplot or VTK, and reading and writing the checkpoint files using MPI-IO functionality.
4. ***main.f90***: the driver program of UCNS3D.
5. ***memory.f90***: includes the majority of the subroutines associated with the dynamic memory allocation and deallocation of all the shared and thread private datatypes.
6. ***mpi_p.f90***: contains the declarations of the MPI variables used in UCNS3D.
7. ***parameters.f90***: consists of the subroutine that reads all of the user-defined parameters from the UCNS3D.DAT file, and additional sets values for all the other parameters that are not included in the UCNS3D.DAT file such as turbulence model constants, type of low-Mach number correction etc. This subroutine has predefined optimised code profiles for different operations of the code such as RANS, DDES, ILES etc. This subroutine also controls if the MRF, MOOD, MULTISPECIES functionality of UCNS3D is going to be activated based on the presence of the corresponding files in the run directory of the code.
8. ***parts.f90***: contains several subroutines for the partitioning the mesh using either serial Metis, and customised variants of that based on the cell-type and reconstruction type, or ParMetis [107].
9. ***translate.f90***: includes all the subroutines and functions for translating the input mesh and boundary conditions to the native UCNS3D format.

5. Selected examples

5.1. Vortex evolution

The 2D vortex evolution test problem introduced by Balsara and Shu [114] is used, involving an isentropic vortex propagating at supersonic Mach number at 45° across the domain modelled by the unsteady inviscid Euler equations. The computational domain is given by $[0, 10] \times [0, 10]$ with periodic boundary conditions applied on all sides. The unperturbed domain has an initial condition $(\rho, u, v, p) = (1, 1, 1, 1)$, where temperature and entropy are defined as $T = p/\rho$, and $S = p/\rho^{\gamma}$ and the vortex perturbations are given by:

$$\begin{aligned} \delta T &= -\frac{(\gamma - 1)\epsilon^2}{8\gamma\pi^2} e^{(1-r^2)}, \\ (\delta u, \delta v) &= \frac{\epsilon}{2\pi} e^{0.5(1-r^2)} \left(-(y-5), (x-5)\right). \end{aligned} \tag{62}$$

The vortex strength is $\epsilon = 5$ and adiabatic gas constant $\gamma = 1.4$. The e_{L^2} and error is computed as follows:

$$e_{L^2} = \sqrt{\frac{\sum_i \int_{\Omega_i} \left(\mathbf{U}_e\left(x, t_f\right) - \mathbf{U}_c\left(x, t_f\right)\right)^2 dV}{\sum_i |\Omega_i|}}, \tag{63}$$

where $\mathbf{U}_c\left(x, t_f\right)$ and $\mathbf{U}_e\left(x, t_f\right)$ are the computed and exact solutions at the end of the simulation t_f. The exact solution $\mathbf{U}_e\left(x, t_f\right)$ being given by the initial condition itself at t_0. A mixed-element unstructured mesh is considered for this test problem of 16, 32, 64, 128 and 256 edges per side resolution similar to the one depicted in Fig. 2, and the simulation is run for a time of $t_f = 10$. The main aim of this test problem is to assess the accuracy of the Central, WENO and CWENOZ schemes for a smooth flow problem, and determine if the schemes can achieve their designed order of accuracy. For a more comprehensive assessment of all the methods of UCNS3D towards smooth flow problems, the reader is referred to

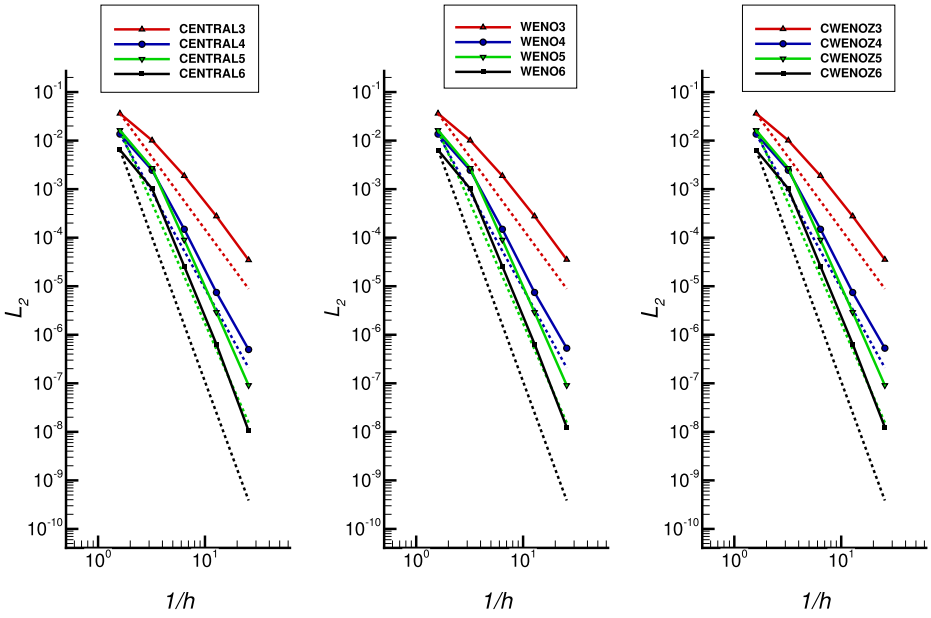


Fig. 5. e_{L^2} density error norm variation with $1/h$, h being the cells min edge, at $t_f = 10$ using several schemes and order of accuracy, for the vortex-evolution test problem on a 2D mixed-element unstructured mesh. It can be noticed that all the schemes achieve convergence rates close to their designed ones (indicated by dotted lines) as the grid resolution is increased.

[1,60–62]. It can be seen from Fig. 5, that all the schemes achieve convergence rates close to their designed order of accuracy as the grid resolution is increased

5.2. Shu-Osher problem

The well-established Shu-Osher [115] test problem is employed, which involves the interaction between a shock wave and an entropy wave. The computational domain is $[-4.5, 4.5] \times [0, 1]$, with periodic boundary conditions in y-axis, and the supersonic inflow and outflow condition on the left and right side of the domain, respectively. The initial profile consists of a shock wave $(\rho, u, v, p) = (3.857143, 2.629369, 0, 10.333333)$ on the left with $x < -4$ and an entropy wave $(\rho, u, v, p) = (1 + 0.2sin(5x), 0, 0, 1)$ in the rest of the domain. A 2D triangular mesh is utilised with a resolution of each edge being $h = 0.025$ with approximately $33,886$ elements. The reference solution is computed with a 1D solver of the Euler equations using $10,000$ grid points and employing a 5th order WENO scheme. The calculation is run until $t = 1.8$. From the density distribution plots shown in Fig. 6 it can be noticed that all the schemes achieve a good agreement with the reference solution, however the WENO 4th-order scheme exhibits an overprediction of the peaks and valleys at the shock-wave entropy interaction region while the CWENO and CWENOZ schemes do not present any oscillations. It needs to be stressed that CWENO or CWENOZ schemes are considerably cheaper in terms of computational resources compared to the WENO schemes ranging from 20%-150% for this 2D problem, as detailed in Tsoutsanis and Dumbser [1].

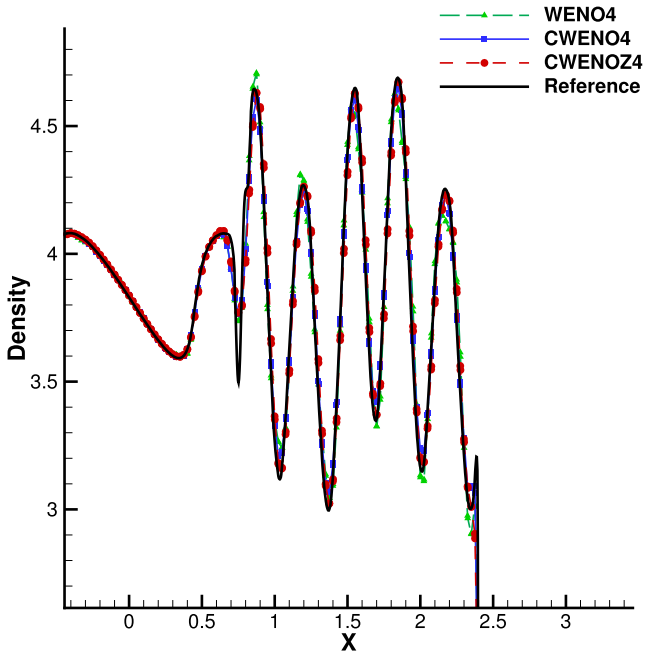


Fig. 6. Density distribution for the Shu-Osher [115] shock tube test problem at the final time $t = 1.8$ using various schemes and comparison with the reference solution obtained from the one-dimensional Euler equations on $10,000$ grid points using a WENO-5th order scheme. It can be noticed that the CWENO and CWENOZ are less oscillatory compared to the WENO schemes.

5.3. iLES of Taylor Green vortex $Re = 1600$

The iLES of the 3D viscous Taylor-Green vortex test problem at $Re = 1600$ is employed, for assessing the performance of several schemes. It is one of the most widely used test problem for the validation of numerical methods, and in particularly at relative coarse-"under-resolved" meshes within the LES context [20,50,60,116–121], since at these resolutions the dissipation and dispersion characteristics of non-linear methods are pronounced. The computational domain is defined as $\Omega = [0, 2\pi]^3$ with periodic boundary conditions. This formulation of the Taylor-Green vortex problem is initialised with the following velocity, density and pressure fields:

$$u(x, y, z, 0) = \sin(kx)\cos(ky)\cos(kz), \tag{64}$$

$$v(x, y, z, 0) = -\cos(kx)\sin(ky)\cos(kz), \tag{65}$$

$$w(x, y, z, 0) = 0, \tag{66}$$

$$\rho(x, y, z, 0) = 1, \tag{67}$$

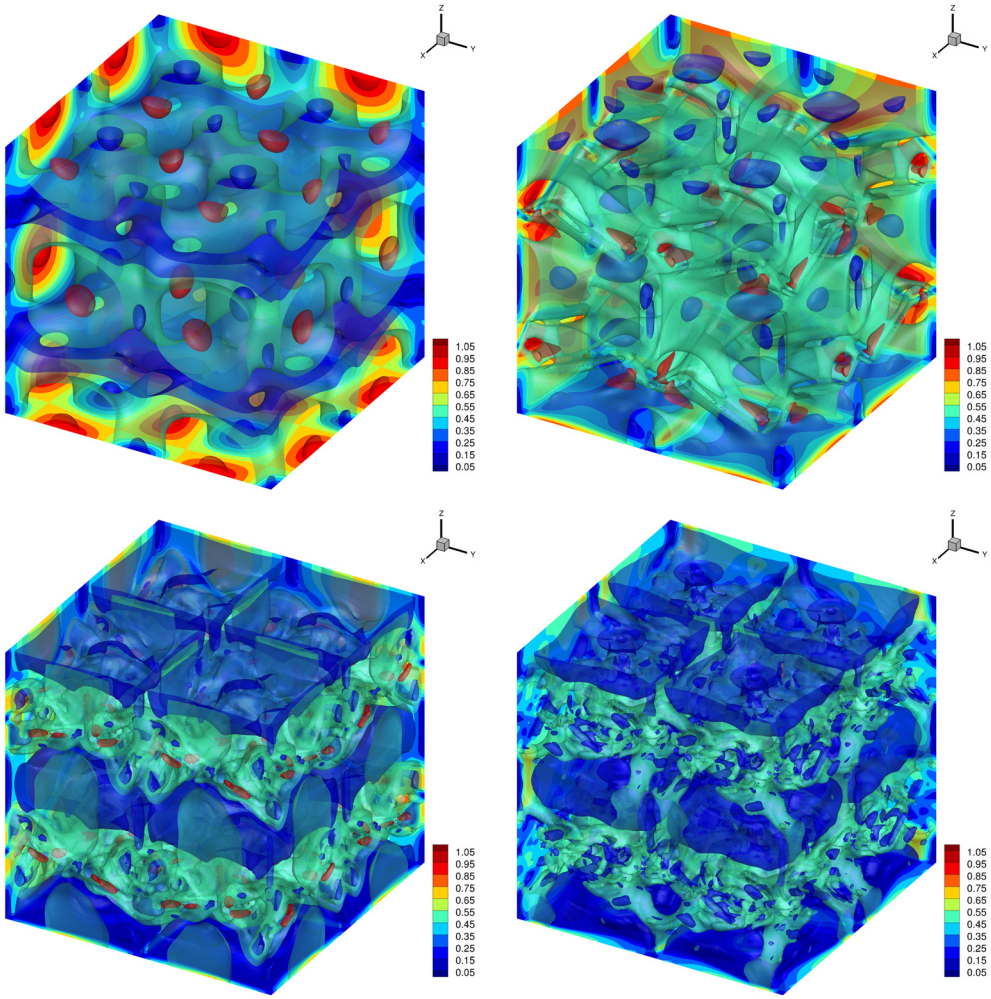


Fig. 7. Solution of the Taylor-Green vortex flow at Re=1600 computed with the CWENO5 scheme on a hexahedral mesh of 128^3. The isosurfaces of the velocity magnitude at three levels (0.1, 0.5, 1.0) are plotted at times $t = 1.3, 4.1, 8.2$ and 11 from top left to bottom right respectively.

$$p(x, y, z, 0) = 100 + \frac{\rho}{16}[\cos(2z) + 2] \cdot [\cos(2x) + \cos(2y)]. \quad (68)$$

The initial condition corresponds to an initial Mach number $M \approx 0.08$, with wavenumber $k = 2\pi/\lambda = 1$. Simulations were carried out on a hexahedral mesh 128^3 resolution with 5th-order spatial discretisation schemes (see Fig. 7 for solution of the Taylor-Green vortex flow at Re=1600 computed with the CWENO5 scheme on a hexahedral mesh of 128^3). The WENO, CWENO, and CWENOZ variants were used with a CFL number of 1.3 for the explicit Runge-Kutta 4th-order scheme, up to $t = 20$ for obtaining the dissipation statistics. The DNS results of Brachet et al. [122] are used for comparisons against the computed solutions.

From the obtained results as shown in Fig. 8 it can be noticed that the energy dissipation rate is in close agreement with the $Re = 1600$ DNS results. The main differences are that the CWENO and CWENOZ schemes exhibit a less dissipative behaviour compared to the equivalent WENO scheme at both grid-resolutions.

5.4. RANS of NASA CRM

This test case concerns the transonic flow over an aircraft configuration, the NASA Common Research Model (CRM) at cruise flight conditions. Drag prediction of the CRM model has been the main objective for the 4^{th}, 5^{th} and 6^{th} drag prediction workshops [123,124].

Case 1a from the 4^{th} workshop is considered for the current analysis, the CRM configuration includes wing, body and horizontal stabilizer. The flow is computed at a constant lift of, $C_l = 0.5$ with error of ± 0.001. The freestream conditions correspond to a Mach number of $M = 0.85$ with a Reynolds number of 5×10^6 based on the reference chord length. Two grids are employed labelled as coarse and medium consisting of 3.5 and 11.0 million elements respectively with a local refinement on the main wing and horizontal stabiliser as shown in Fig. 9. For a more detailed description of the setup of this problem the reader is referred to the work of Antoniadis et al. [125].

The main aim for employing the present transonic case is to assess the potential benefits of the WENO third and fifth order methods denoted with W3 and W4 respectively over a second order MUSCL scheme denoted as M2 as well with reference experimental and CFD data. The distribution of pressure coefficient on the CRM configuration is shown in Fig. 10 and at two representative spanwise stations on the main wing in Fig. 11. The third and fifth order profiles are plotted against the chord-wise length and compared with the NTF pressure measurements [123]. The presence of the shock on the wing's suction side is evident for both stations near the wing's mid-spanwise location. Grid refinement suggests smaller discrepancies with respect to the shock position ($\eta = 0.60$) compared with the experiment, regardless of the scheme adopted.

Table 4 summarises the force predictions, and the corresponding angle of attack for constant lift, the predicted drag coefficient and the error ΔC_d in terms of drag counts with respect to the experiment [123]. The medium resolution mesh provides a signif-

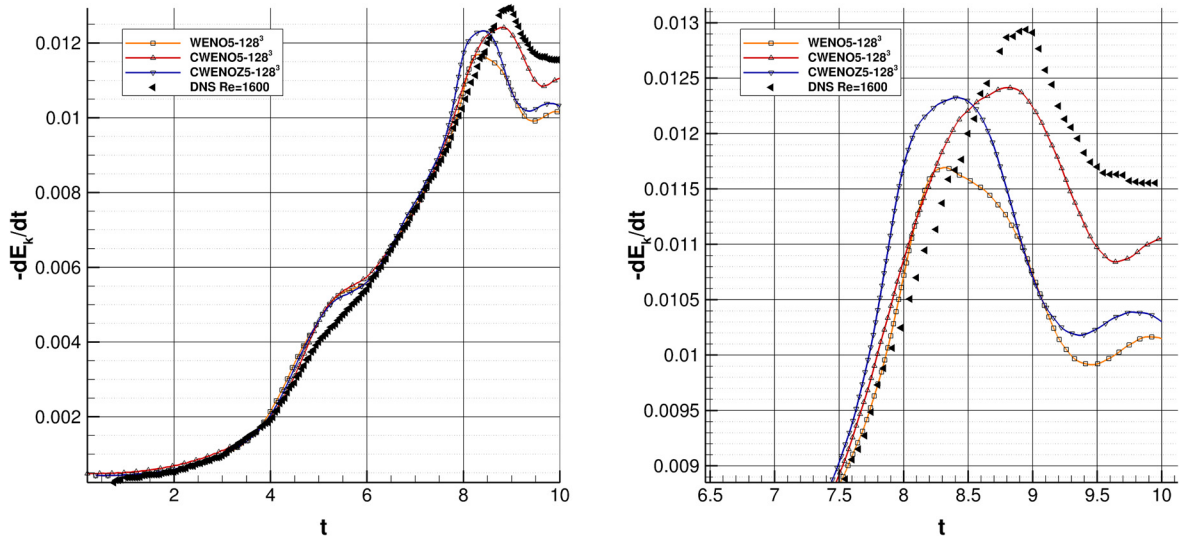


Fig. 8. Kinetic energy dissipation rate for the iLES of the Taylor-Green Vortex $Re = 1600$ obtained with various schemes on 128^3 hexahedral mesh, and comparison with the DNS results of Brachet et al. [122]. It can be noticed that all the schemes perform similarly, with CWENO and CWENOZ schemes being marginally less dissipative compared to WENO.

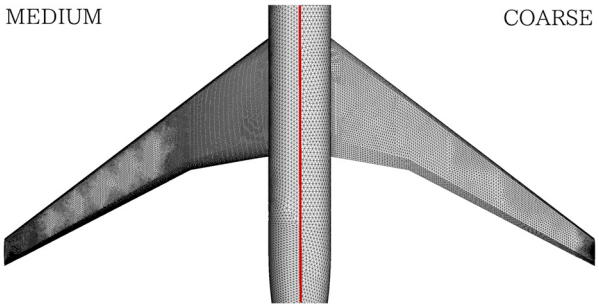


Fig. 9. Grid configurations for the transonic flow over the CRM model.

Table 4
Computed drag coefficient for all schemes and grids, compared with the average CFD and experimental values [123].

	Elements (10^6)	a	C_d	ΔC_d (counts)
C-M2	3.5	2.395	0.03227	47.6
C-W3	3.5	2.295	0.02962	21.11
C-W5	3.5	2.360	0.02942	19.06
M-M2	11.0	2.210	0.02810	5.89
M-W3	11.0	2.236	0.02742	0.93
M-W5	11.0	2.264	0.02716	3.49
DPW-CFD	-	2.340	0.02701	5
DPW-NTF	-	3.020	0.02751	-

Table 5
Normalised time taken per-iteration for the coarse mesh for the RANS of NASA CRM using UCNS3D and ANSYS Fluent.

	Normalised time taken per iteration
UCNS3D MUSCL2	1.4
UCNS3D WENO3	2.53
UCNS3D WENO5	3.87
ANSYS Fluent UPWIND2	1.0

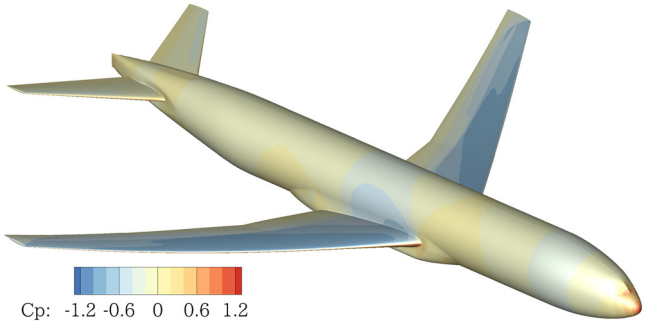
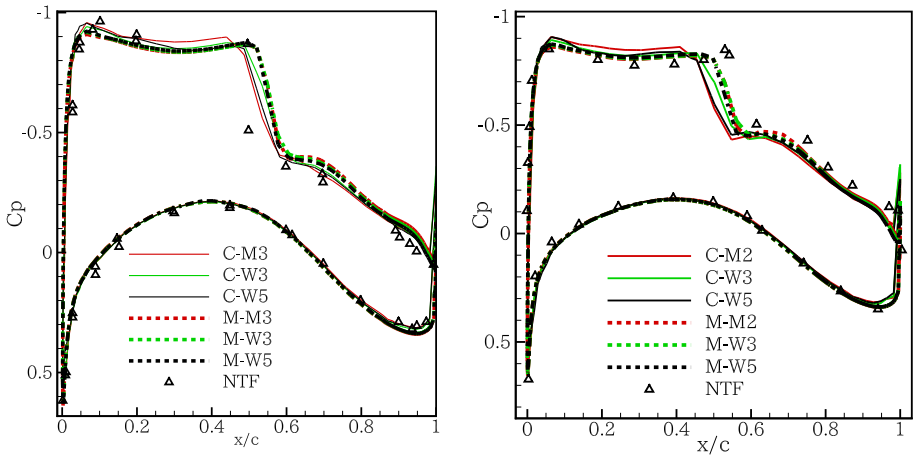


Fig. 10. Coefficient of pressure on the medium grid W3 solutions, CRM configuration ($M_\infty = 0.85$, $C_l = 0.5$ and $Re_c = 5.0 \times 10^6$).

icant improvement in terms of the error, while WENO schemes further improve the predictions at both grid resolutions. It has to be noted that for the present test problem the W3 scheme represents a more suitable method with a favourable balance between cost and accuracy for these types of aeronautical configurations under RANS, since the additional cost of the W5 scheme does not justify its use. For appreciation of the computational cost of the present simulations for this test problem, the time taken per iteration for the coarse mesh is presented and has been normalised with respect to the time taken per iteration for the same mesh with the commercial CFD software package ANSYS Fluent [126,127] as shown in Table 5.

Specifically Fluent version 19.2 was used, and the solver settings selected were as close as possible to the ones used by UCNS3D such as double precision, density based solver with Spalart-Allmaras turbulence model and ideal-gas law for density and Sutherlands viscosity, Roe-FDS scheme, 2nd-order accuracy for mean flow variables and for the turbulence model, and implicit time-stepping with a fixed (no ramping) CFL of 10 (which is the same as the one used by UCNS3D). All the simulation timing measurements were performed in a workstation consisting of a 32core AMD Threadripper Pro 3975WX CPU, 256 GB of DDR4 memory running a Centos 8 distribution. All the simulations were performed using 32 MPI processes.

Finally, the wall clock time taken for the simulation to converge was also measured, since this also provides a representative measure of the total computational cost. UCNS3D required 53 minutes to converge the lift coefficient within the tolerance of ± 0.001, while ANSYS Fluent required 45 minutes to converge with the same tolerance. It has to be noted that the wall clock times for WENO3 and WENO5 order schemes for UCNS3D are not provided since they were initialised with the converged results from the MUSCL2 simulations.



(a) $\eta = 0.50$ (b) $\eta = 0.60$

Fig. 11. Coefficient of pressure at two spanwise stations (η) on the main wing compared with the wind tunnel measurement [123], CRM configurations ($M_\infty = 0.85$, $C_l = 0.5$ and $Re_c = 5.0 \times 10^6$).

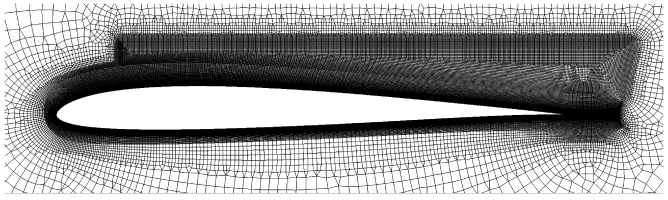


Fig. 12. Slice of the mesh used for the SD7003 test problem.

Table 6
Obtained results for the SD7003 test cases using WENO 4th-order schemes, and comparison with reference datasets.

Data set	C_L	C_D	x_{sep}/c	x_{rea}/c	Method
Present W4	0.903	0.058	0.038	0.391	WENO FV
Present W4-LM	0.928	0.051	0.029	0.329	WENO FV
Garmann et al. [128]	0.969	0.039	0.023	0.259	6th-order FD
Vermeire et al. [20]	0.941	0.049	0.045	0.315	5th-order FR
Beck et al. [129]	0.932	0.050	0.030	0.336	8th-order DG

5.5. iLES of SD7003 aerofoil

This test case concerns the turbulent flow over the SD7003 wing at Mach number of $M = 0.2$, angle of attack of $\alpha = 8°$ and Reynolds number $Re = 60,000$ based on the chord length. This flow is characterised by a laminar separation region, which reattaches further downstream forming a Laminar Separation Bubble (LSB), and along the separation bubble transition to turbulence occurs. This test case is widely used to assess the performance of various numerical schemes in the context of LES, for predicting separation, transition and turbulent flow [19,20,128,129]. The purpose of this simulation is to assess the potential benefits or not of employing the low-Mach (LM) number treatment previously introduced.

A hybrid unstructured mesh of approximately 5.6 million cells is generated, consisting of hexahedral and prismatic cells, as illustrated in Fig. 12. The domain extends $20c$ upstream and downstream, and $0.2c$ in the span-wise direction, where c is the chord length. The grid resolution at the boundary layer region gives a $y^+ \approx 1$ at the first cell off the surface, and 80 cells are used in the span-wise direction. Periodic boundary conditions are used in the span-wise direction, no-slip boundary conditions at the surface of the aerofoil and free-stream conditions at the farfield. The WENO 4^{th}-order scheme is employed, with a CFL number of 0.9, and the solution is advanced in time with the explicit Runge-Kutta 3rd-order scheme. The simulations were run for $t = 20t_c$ to develop the flow, and an additional $20t_c$ for time averaging, where $t_c = c/U_\infty$.

Contour plots of time- and span-averaged streamwise velocity are shown in Fig. 13, where it can be noticed that the predicted laminar separation bubble near the leading edge on the suction side of the aerofoil is smaller for the result with the LM treatment compared to that without the LM treatment. The LM treatment improves the agreement with the reference data sets [20,128,129] in terms of the separation and reattachment positions, as shown in Table 6.

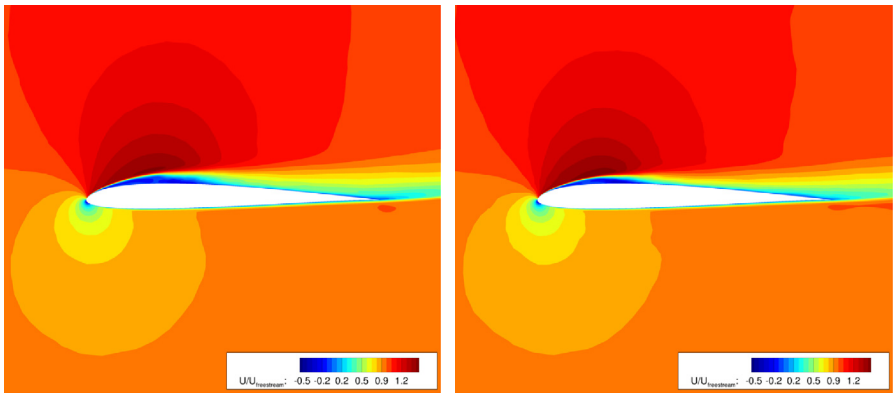
The instantaneous isosurfaces of Q-criterion coloured by the velocity magnitude are shown in Fig. 14 for the fully developed flow after $20t_c$. Smaller-scale flow structures seem to be captured by the numerical scheme with the LM fix, which is inline with previous findings from Simmonds et al. [60].

The span- and time-averaged pressure coefficient C_P distribution is plotted in Fig. 15, along with the reference data sets [20,128,129]. When the LM treatment is enabled, the result is in closer agreement with the reference data sets in particular with the ones of Beck et al. [129], including the location of the transition region, where without it the transition appears further downstream. Therefore, this LM treatment can prove beneficial in low-Mach number regions of compressible flows, unless very high-order methods are employed (5th-order and higher) where the benefits of this treatment are not easily realised.

5.6. Caradonna & Tung rotor

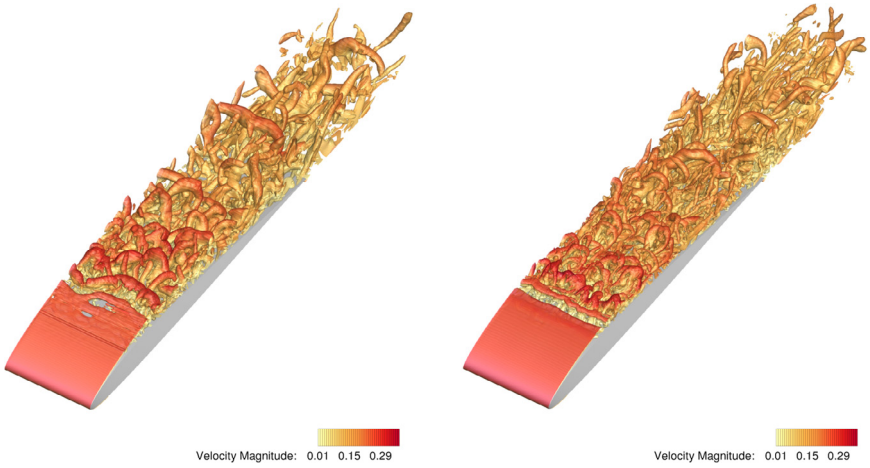
The accuracy and efficiency of the developed rotating framework is evaluated on the hovering two-bladed rotor Caradonna and Tung [130], where the readers are referred for the experimental setup. The blade is based on an untwisted and untapped NACA 0012 aerofoil with a radius of 1.143 m and an aspect ratio of 6. The case with a blade tip Mach number of $M_{tip} = 0.89$ and pitch angles of $\theta_c = 8°$ is considered for the computational assessment, using the MUSCL-MOGE limiter and the RANS equations with SA turbulence model with a 15 million elements mixed-element mesh, and the reader is referred to the work of Silva et al. [4] for a detailed description of the numerical parameters chosen and the setup of this test problem.

The operating conditions are considered fairly challenging both in terms of the experiment but also with respect to prediction abil-



(a) without LM treatment (b) with LM treatment

Fig. 13. Span-averaged and time-averaged stream-wise velocity contour for the SD7003 test problem using WENO 4^{th}-order schemes.



(a) without LM treatment (b) with LM treatment

Fig. 14. Instantaneous isosurfaces of Q criterion coloured by velocity magnitude for the SD7003 test problem using WENO 4^{th}-order schemes.

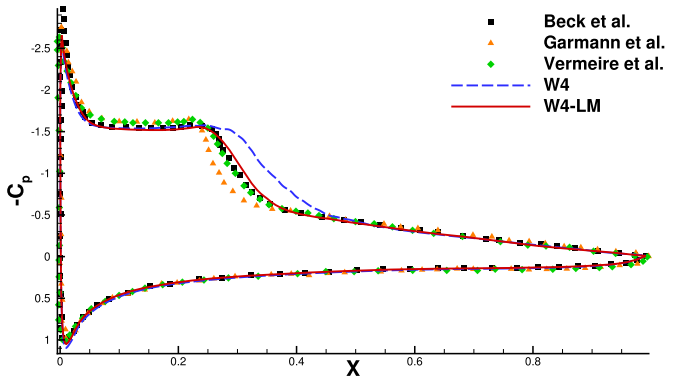


Fig. 15. Span-averaged and time-averaged pressure coefficient C_P distribution along the chord of the SD7003 aerofoil, using WENO 4th-order schemes and comparison with reference datasets from Garmann et al. [128], Vermeire et al. [20] and Beck et al. [129].

ities of CFD solvers, and the key objective is to assess the robustness and accuracy of the UCNS3D framework. This flow problem is characterised by the strong vortical structures which develop at the tip of the blade, and their interaction with the blade at each rotation, defined as the as Blade-Vortex Interaction (BVI). This interaction, is one of the most complex features of hovering flows and is highly sensitive to the numerical approach and spatial resolution [131]. It also significantly affects the rotor aerodynamic performance. Therefore, a numerical method that is able to capture the core of the vortex without being largely dissipated is a desired feature [132]. Fig. 16 shows the effect of order of the numerical-scheme on capturing the pressure profile at the tip blade, $r/R = 0.96$. This region is mostly influenced by the tip vortex interaction with a shock at $x/c \approx 0.25$, therefore an accurate and robust numerical method is necessary. The low-order predictions struggle to capture the sharp pressure drop and its recovery, presenting small spurious oscillations at $0.35 < x/c < 0.6$. While on, the higher-order solutions are in good agreement with the experiment through the entire span section.

Fig. 17 shows the predicted tip vortex trajectories of the computed solutions compared with the experiment [130] and prescribed wake models [133]. The vortex radial contraction shown in Fig. 17 (top), is well predicted by computations up to the first blade passage, where after this point the impact of a higher-order scheme is negligible, something that is attributed to the effect of the rotor hub which is neglected in the present computational model. Overall, the computed solutions of the vortex contraction agree with the work of Kocurek [133]. In Fig. 17 (b) it can be seen that the CFD results accurately predict the slow convection of the tip vortices seen in the vertical displacement (Z/R) up to a full cycle.

Finally, the helical wake structure of the tip vortex using the same level of iso-vorticity contours is shown in Fig. 18, where it

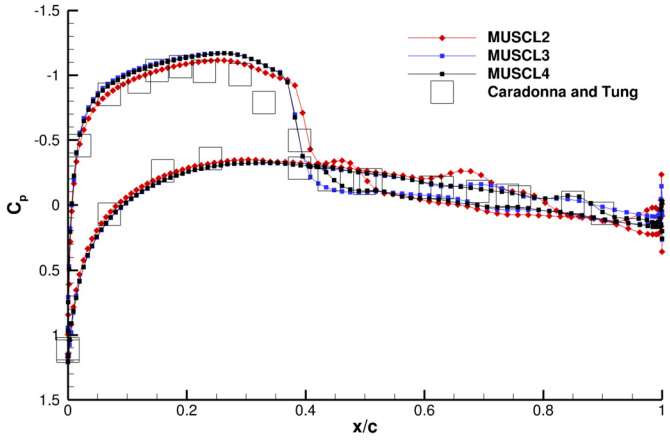


Fig. 16. Pressure Coefficient at $r/R = 0.97$ for all orders.

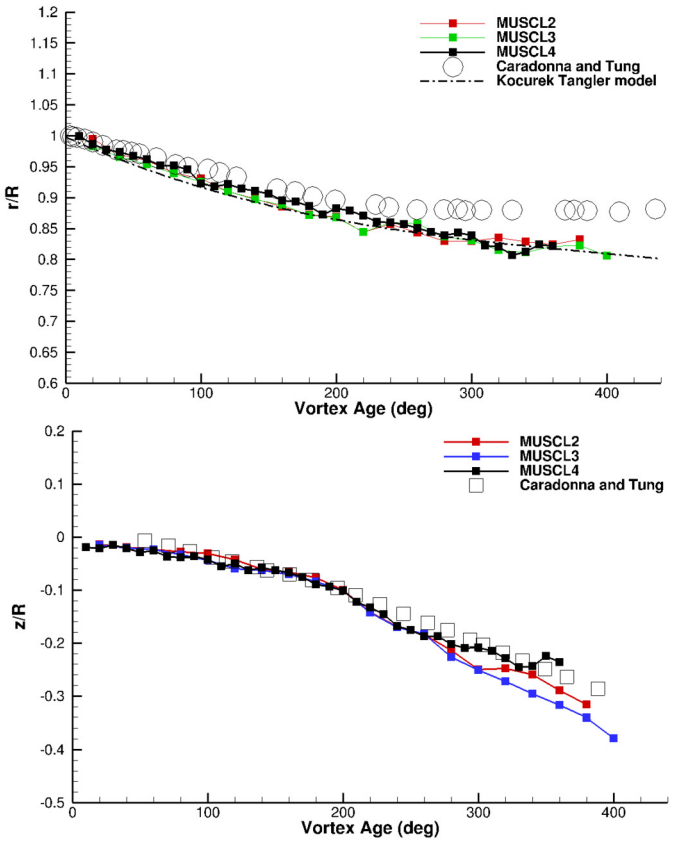


Fig. 17. Vortex age and trajectory of computed solutions in terms of vortex radial contraction (top) and downwash rate (bottom) against azimuth angle compared with experiment.

is clear that higher-order schemes improve the predictions of the wake sheet and vortex tip flow phenomena while preserving the helical vortex filaments up to several radii downstream the rotor. As the wake is transported downwards, the interaction of primary and secondary structures is noticed, which stretch between the tip vortices making an S-shaped path also seen experimentally [134]. The horseshoe vortices present on low-order scheme strain into the vortex tube as the numerical dissipation decreases, something also documented in [135]. It has to be stressed that until recently this secondary vortex production has been seen as overemphasised by high-order solutions and not believed to be an accurate physical phenomenon, but state-of-the-art volumetric flow measurement has confirmed its presence through the Lagrangian volumetric PIV

Table 7
Time averaged velocities of several flow features of the shock-bubble interaction problem obtained with the present simulation and compared against the experimental results of Haas and Sturtevant [138], and the computational results of Coralic and Colonius [136]. The time intervals that have been used to average the velocities are [10, 52] μs for the upstream interface, [140, 240] μs for the jet, and [140, 240] μs for the downstream interface. It can be noticed that all the obtained predictions are within the variations of the experiment, and in close agreement with the computational results of [136].

Grid size	u_{ui} (m/s)	u_{ji} (m/s)	u_{di} (m/s)
2D-fine	181.242	237.086	145.895
3D	178.402	228.375	144.197
Coralic & Colonius WENO5 [136]	173	230	145
Haas & Sturtevant exp. [138]	170 ± 17	230 ± 23	145 ± 15

variant technique [134]. The computational time taken per iteration to execute this case was compared with the commercial software ANSYS CFX 19.2 [127]. The solver settings were selected to reproduce the numerical specifications used by UCNS3D as close as possible, density based solver with SST turbulence model, ideal-gas law for density and 2nd-order accuracy with the same CFL. A single node consisting of two Intel E5-2620 v4 (Broadwell) CPUs giving 16 CPU cores and 128 GB of shared memory was used for the simulation measurements. After 11 hours wall-clock time of computation using 16 MPI processes, CFX and UCNS3D completed 1130 and 2506 iterations, respectively. The convergence of integrated force and torque on both solvers were achieved after approximately 800 iterations.

5.7. Helium bubble shock wave

The interaction of a weak shockwave (travelling in air) with a helium bubble has been modelled in 2D and 3D. Several variations of this test problem have been widely used [10,103,136,137] for assessing the performance of several techniques for multicomponent flow modelling and the present setup is based on the experimental setup by Haas and Sturtevant [138]. A bubble of diameter $D_b = 5$ cm, is placed within an air filled shock tube. The bubble consists of helium and air of 28% mass concentration. A shockwave moving from right to left impacts the bubble, contaminated by the surrounding air. The specific heats of 1.4 and 1.66 are used for air and helium, respectively, and the initial condition is given by:

$$(a_1\rho_1, a_2\rho_2, u, v, p, a_1) = \begin{cases} (0.0, 1.204, 0, 0, 101325, 0)\,, & \text{for Pre-shock,} \\ (0.0, 1.658, -114.49, 0, 159060, 0)\,, & \text{for Post-shock,} \\ (0.158, 0.061, 0, 0, 101325, 0.95)\,, & \text{for Bubble.} \end{cases} \quad (69)$$

The computation domain is discretised by a mixed-element unstructured mesh consisting of quadrilateral and triangular elements in 2D, and arbitrary hexahedrals and prisms in 3D with the shock-bubble interaction region being refined. A fine mesh is employed in 2D with an average element edge length in the shock bubble interaction zone of $e_f \approx D_b/400$, and an edge length of $e_c \approx D_b/50$ for the 3D mesh. A slip-wall boundary condition is used at the top and bottom boundaries of the domain, while inflow and outflow boundary conditions are prescribed at the right and left of the domain respectively. The reader is referred to [139] for more details on the parameters for the setup used. A CWENO5 scheme is employed, and the simulation is run until $t = 1000$ μs.

From the results obtained as shown in Fig. 19, Fig. 20 and Fig. 21, it can be seen that the time evolution of the bubble and shockwave interaction is correctly captured including the formation of a jet and a vortex ring at late times, and is in agreement

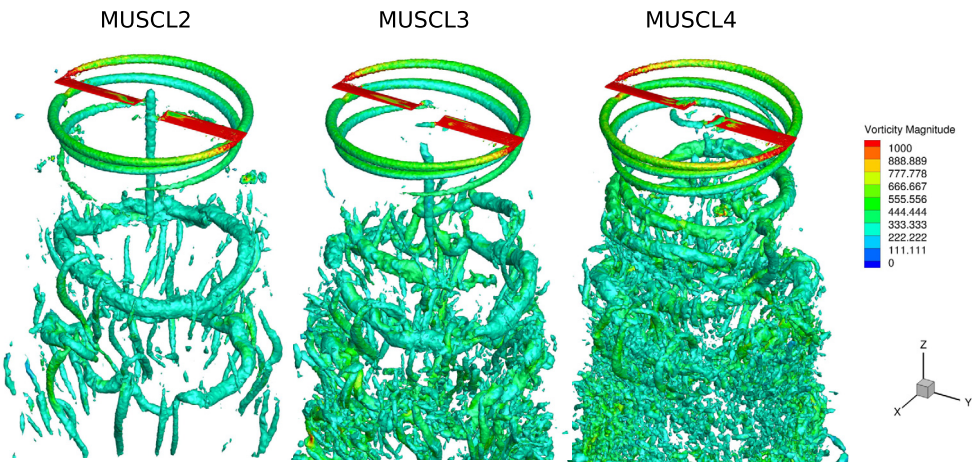


Fig. 18. Iso-vortex surfaces coloured by vorticity magnitude contours, comparing the effect of order resolution of vortex wake.

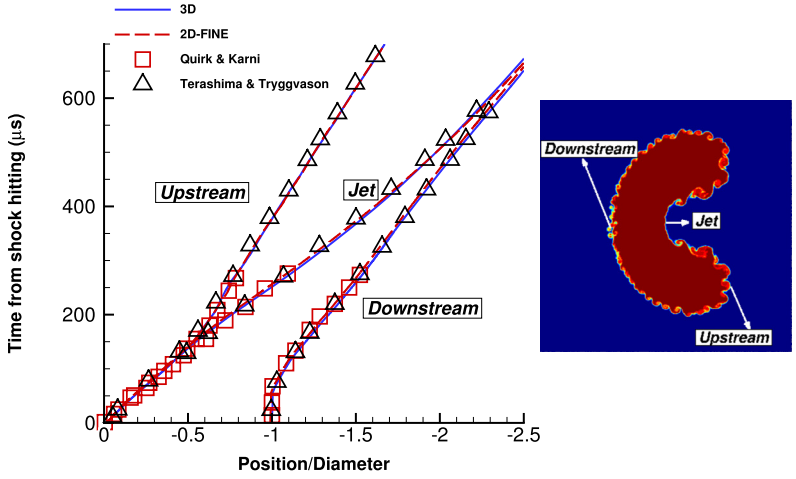


Fig. 19. Plot of the definition of the interfaces and of the evolution of the position of these interfaces using a CWENO5 scheme on the finest 2D mesh and 3D mesh. The evolution of the interface's position is in good agreement with the results of Terashima and Tryggvason [140] and Quirk and Karni [141].

with the results obtained by [10,103,136] qualitatively. As the bubble evolves, three distinct interfaces identified are the jet (ji), the upstream (ui) and the downstream (di) as shown in Fig. 19. From the space-time diagram of these distinct interface positions, the predicted locations are in good agreement with the reference results of Terashima and Tryggvason [140] and Quirk and Karni [141]. The results have been non-dimensionalised with respect to the diameter of the bubble at the time that the shock hits the bubble [10,140,141].

The results obtained in terms of the averaged velocities of the jet, upstream and downstream interface are in good agreement with the experimental results of Haas and Sturtevant [138], and the computational results of Coralic and Colonius [136] as shown in Table 7. Due to the resolution employed for the 3D setup of the problem, a slower merging of the jet and the downstream interface is seen compared to the 2D setup.

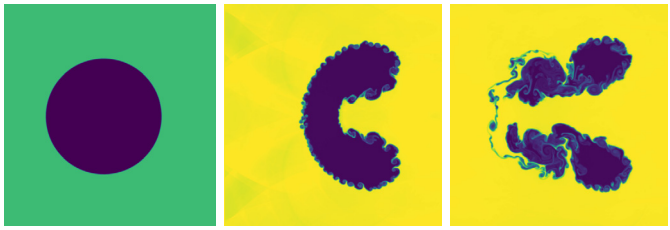
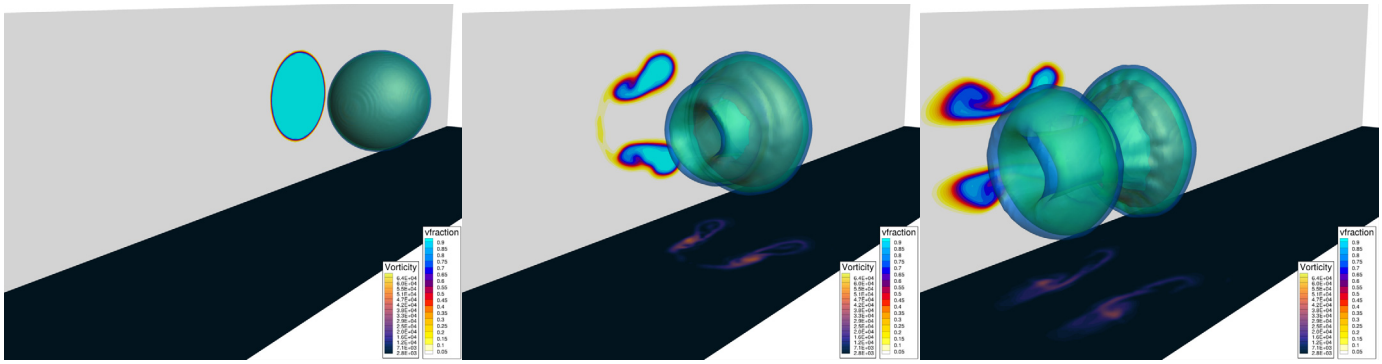


Fig. 20. Density contour plots (24 equally distanced levels between 0.19 to 1.74) of the computed solution of the shockwave helium bubble interaction test problem at various instants using the finest mesh. As the shock wave passes the helium bubble, Kelvin-Helmholtz instabilities develop at the material interface that later on break down while resulting in an asymmetric solution profiles.

6. Parallel performance

The parallel performance optimisation and benchmarking of `UCNS3D` across several HPC systems has been thoroughly documented in several studies performed under EU-PRACE projects [41,42,109,110]. The most recent representative benchmark concerns the iLES of the NASA High-Lift CRM model with 240 Million cells at Reynolds=5.49 Million and AoA=21.47° using the CWENO4 order scheme, as it shown in Fig. 22. A strong scalability test was performed, on the HLRS-HAWK system, based on the AMD Epyc 7742 Rome processors. Each node containing 2 processors, each one of them consisting of 64 cores running at 2.25 GHz. The benchmarks started from 16 nodes (2048 cores) which is the minimum number of nodes that the problem can fit in the memory, to 1024 nodes (131072 cores), using the MPI and OpenMP implementation with 8 MPI processes per node (4 in each processor), and 16 threads per process, so that all the physical cores are populated. This configuration has been found to be the optimum in this hardware architecture. From the results obtained as shown in Fig. 23



(a) $t = 00\mu s$ (b) $t = 629\mu s$ (c) $t = 980\mu s$

Fig. 21. Contour plots of volume fraction, vorticity magnitude at the centre of the computational domain and isosurfaces of three levels of volume fraction (0.25, 0.5, 0.9) of the computed solution of the shockwave 3D helium bubble interaction test problem at different instants.

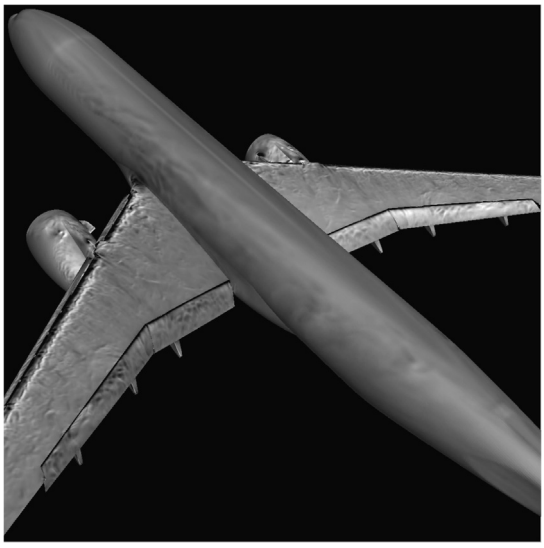


Fig. 22. Contour plot of skin friction coefficient on the surface of the NASA CRM High-Lift model used for the iLES simulation at Reynolds 5.49 Million, and AoA=21.47°.

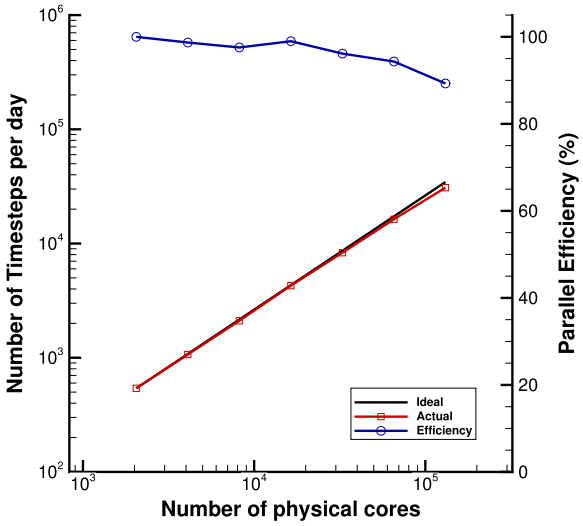


Fig. 23. Strong scalability testing on HLRS-HAWK, using a 240 Million elements mesh for the NASA CRM High-Lift model and a 4th-order accurate CWENO4 scheme for an iLES simulation. This run was obtained with the optimum 16 threads per task, 8 processes per node.

the performance is measured in terms of the number of physical timesteps that can be executed within a day (24 hours) and the actual performance is very close to the theoretical/ideal performance expected. It can also be noticed that parallel efficiency (defined as the ratio of true speedup over theoretical speedup) remains above 85% up to 131072 cores, which s is indicative of the good scalability of UCNS3D for iLES. For more detailed benchmarks and optimisation strategies of UCNS3D the reader is referred to [41,42,109,110].

7. Conclusions

The open-source CFD code UCNS3D was presented. The code can solve laminar, transitional, turbulent and multicomponent flows using mixed-element unstructured meshes. Multiple state-of-the-art high-order numerical methods incorporated into the software are useful considering the vastly different computational requirements of this wide range of challenging problems. We also describe the requirements for building UCNS3D, including the open-source software libraries that are available online and their design. A description of the files, and the code directories, is provided. We provided a representative set of examples, including advection of a smooth vortex, shock-entropy waves interaction, iLES of Taylor-Green vortex, RANS of transonic full aircraft configuration during the cruise, iLES of the SD7003 airfoil, RANS of the Caradonna & Tung rotor, and the multicomponent shockwave-interaction with a Helium bubble. The parallel performance of UCNS3D on a large-scale problem is also assessed, where it achieves a nearly-ideal parallel computational efficiency at $\approx 10^5$ processors.

Future developments pursued regarding UCNS3D include the overset mesh interface, addition of the DG framework, hybrid DG-FV, addition of the 6-equation and 7-equation multicomponent models, and inclusion of real-gas effects for high-Mach number flows.

Declaration of competing interest

Declaration of interest: none

Data availability

The datasets associated with the test problems presented are available at the Cranfield Online Research Data repository https://doi.org/10.17862/cranfield.rd.16447212.v1, https://doi.org/10.17862/cranfield.rd.11836164.v1, https://doi.org/10.17862/cranfield.rd.8983772.v1, https://doi.org/10.17862/cranfield.rd.19146182.v1.

Acknowledgements

UCNS3D has been developed over a number of years, and we would like to thank many people who have made significant contributions. In particular, we would like to thank Anastasiia Shamakina, Anna Mack, Venkata Ayyalasomayajula, and Thomas Ponweiser for developments associated with the parallel performance optimization, and big data handling. The development and application of UCNS3D have been supported by several funding bodies including EU funding (Grants 314139, 653838, 823767), Engineering and Physical Sciences Research Council (EPSRC) (grants EP/L000261/1, EP/P020259/1, EP/G069581/1, EP/T518104/1; award 13794), and UKRI Innovate UK (Grant 263261). Finally we would like to thank the reviewers for their recommendations.

References

[1] P. Tsoutsanis, M. Dumbser, Comput. Fluids 225 (2021), https://doi.org/10.1016/j.compfluid.2021.104961.
[2] P. Tsoutsanis, I. Kokkinakis, L. Konozsy, D. Drikakis, R. Williams, D. Youngs, Comput. Methods Appl. Mech. Eng. 293 (2015) 207–231, https://doi.org/10.1016/j.cma.2015.04.010.
[3] F. Ricci, P. Silva, P. Tsoutsanis, A. Antoniadis, Aerosp. Sci. Technol. 97 (2020), https://doi.org/10.1016/j.ast.2019.105648.
[4] P. Silva, P. Tsoutsanis, A. Antoniadis, Aerosp. Sci. Technol. 111 (2021), https://doi.org/10.1016/j.ast.2021.106518.
[5] M. Dumbser, O. Zanotti, R. Loubére, S. Diot, J. Comput. Phys. 278 (2014) 47–75.
[6] M. Dumbser, M. Kaser, V. Titarev, E. Toro, J. Comput. Phys. 226 (1) (2007) 204–243.
[7] V. Titarev, E. Toro, J. Comput. Phys. 201 (1) (2004) 238–260.
[8] M. Dumbser, W. Boscheri, M. Semplice, G. Russo, SIAM J. Sci. Comput. 39 (6) (2017) A2564–A2591, https://doi.org/10.1137/17M1111036.
[9] J. Zhu, J. Qiu, SIAM J. Sci. Comput. 40 (2018) A903–A928.
[10] Q. Wang, R. Deiterding, J. Pan, Y.-X. Ren, Comput. Fluids 202 (2020), https://doi.org/10.1016/j.compfluid.2020.104518.
[11] M. Wong, S. Lele, J. Comput. Phys. 339 (2017) 179–209, https://doi.org/10.1016/j.jcp.2017.03.008.
[12] G. Hu, R. Li, T. Tang, Commun. Comput. Phys. 9 (3) (2011) 627–648.
[13] L. Fu, Comput. Phys. Commun. 244 (2019) 117–131, https://doi.org/10.1016/j.cpc.2019.06.013.
[14] L. Fu, X. Hu, N. Adams, J. Comput. Phys. 349 (2017) 97–121, https://doi.org/10.1016/j.jcp.2017.07.054.
[15] L. Fu, X. Hu, N. Adams, J. Comput. Phys. 305 (2016) 333–359, https://doi.org/10.1016/j.jcp.2015.10.037.
[16] L. Fu, Comput. Phys. Commun. 235 (2019) 25–39, https://doi.org/10.1016/j.cpc.2018.10.009.
[17] S. Clain, S. Diot, R. Loubereé, J. Comput. Phys. 230 (10) (2011) 4028–4050, https://doi.org/10.1016/j.jcp.2011.02.026.
[18] M. Tavelli, M. Dumbser, J. Comput. Phys. 341 (2017) 341–376.
[19] B. Vermeire, P. Vincent, J. Comput. Phys. 327 (2016) 368–388.
[20] B. Vermeire, F. Witherden, P. Vincent, J. Comput. Phys. 334 (2017) 497–521.
[21] F. Witherden, A. Farrington, P. Vincent, Comput. Phys. Commun. 185 (11) (2014) 3028–3040, https://doi.org/10.1016/j.cpc.2014.07.011.
[22] P.-O. Persson, J. Peraire, SIAM J. Sci. Comput. 30 (6) (2008) 2709–2733.
[23] R. Moura, G. Mengaldo, J. Peiro, S. Sherwin, J. Comput. Phys. 330 (2017) 615–623, https://doi.org/10.1016/j.jcp.2016.10.056.
[24] D. De Grazia, G. Mengaldo, D. Moxey, P. Vincent, S. Sherwin, Int. J. Numer. Methods Fluids 75 (12) (2014) 860–877, https://doi.org/10.1002/fld.3915.
[25] N.K. Burgess, D.J. Mavriplis, in: 50th$ AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, January 2012, pp. 1–37.
[26] P. Castonguay, P. Vincent, A. Jameson, J. Sci. Comput. 51 (1) (2012) 224–256, https://doi.org/10.1007/s10915-011-9505-3.
[27] Z. Wang, J. Comput. Phys. 178 (1) (2002) 210–251.
[28] Z. Wang, L. Zhang, Y. Liu, J. Comput. Phys. 194 (2) (2004) 716–741.
[29] Z. Wang, Y. Liu, J. Comput. Phys. 179 (2) (2002) 665–697.
[30] Z. Xu, Y. Liu, C.-W. Shu, J. Comput. Phys. 228 (16) (2009) 5787–5802.
[31] C. Breviglieri, A. Maximiliano, E. Basso, J. Azevedo, in: 27th AIAA Applied Aerodynamics Conference, vol. 79128, 2009.
[32] T. Haga, K. Sawada, Z.J. Wang, Commun. Comput. Phys. 6 (5) (2009) 978–986.
[33] W. Boscheri, M. Semplice, M. Dumbser, Commun. Comput. Phys. 25 (2) (2019) 311–346, https://doi.org/10.4208/cicp.OA-2018-0069.
[34] D. Balsara, C. Altmann, C.-D. Munz, M. Dumbser, J. Comput. Phys. 226 (1) (2007) 586–620, https://doi.org/10.1016/j.jcp.2007.04.032.
[35] A. Bermúdez, S. Busto, M. Dumbser, J. Ferrín, L. Saavedra, M. Vázquez-Cendón, J. Comput. Phys. 421 (2020) 109743.
[36] M. Dumbser, D. Balsara, E. Toro, C.-D. Munz, J. Comput. Phys. 227 (18) (2008) 8209–8253.
[37] X. Liu, L. Xuan, Y. Xia, H. Luo, Comput. Fluids 152 (2017) 217–230, https://doi.org/10.1016/j.compfluid.2017.04.027.
[38] H. Jasak, Int. J. Nav. Archit. Ocean Eng. 1 (2) (2009) 89–94, https://doi.org/10.3744/JNAOE.2009.1.2.089.
[39] F. Palacios, M. Colonno, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, T. Taylor, J. Alonso, 2013.
[40] P. Farmakis, P. Tsoutsanis, X. Nogueira, Comput. Methods Appl. Mech. Eng. 363 (2020), https://doi.org/10.1016/j.cma.2020.112921.
[41] T. Ponweiser, P. Tsoutsanis, PRACE-HOVE 1 report, 2017, URL https://prace-ri.eu/wp-content/uploads/WP222.pdf.
[42] A. Shamakina, P. Tsoutsanis, PRACE-HOVE 2 report, 2019, URL https://prace-ri.eu/wp-content/uploads/WP288.pdf.
[43] P.R. Spalart, S.R. Allmaras, Rech. Aérosp. 1 (1994) 5–21.
[44] N. Nikitin, F. Nicoud, B. Wasistho, K. Squires, P. Spalart, Phys. Fluids 12 (7) (2000) 1629–1632, https://doi.org/10.1063/1.870414.
[45] P. Spalart, S. Deck, M. Shur, K. Squires, M. Strelets, A. Travin, Theor. Comput. Fluid Dyn. 20 (3) (2006) 181–195, https://doi.org/10.1007/s00162-006-0015-0.
[46] G. Allaire, S. Clerc, S. Kokh, J. Comput. Phys. 181 (2) (2002) 577–616, https://doi.org/10.1006/jcph.2002.7143.
[47] V. Maltsev, M. Skote, P. Tsoutsanis, Phys. Fluids 34 (2) (2022), https://doi.org/10.1063/5.0077314.
[48] A. Stroud, Math. Comput. 30 (134) (1976) 291–294, https://doi.org/10.1090/S0025-5718-1976-0391484-0.
[49] P. Tsoutsanis, V. Titarev, D. Drikakis, J. Comput. Phys. 230 (4) (2011) 1585–1601.
[50] P. Tsoutsanis, A. Antoniadis, D. Drikakis, J. Comput. Phys. 256 (2014) 254–276.
[51] V. Titarev, P. Tsoutsanis, D. Drikakis, Commun. Comput. Phys. 8 (3) (2010) 585–609.
[52] P. Tsoutsanis, D. Drikakis, J. Coupled Syst. Multiscale Dyn. 4 (2016) 170–186, https://doi.org/10.1166/jcsmd.2016.1104.
[53] A. Antoniadis, P. Tsoutsanis, D. Drikakis, in: 53rd AIAA Aerospace Sciences Meeting, vol. 0317, 2015.
[54] A. Antoniadis, P. Tsoutsanis, D. Drikakis, in: 42nd AIAA Fluid Dynamics Conference and Exhibit, vol. 2833, 2012.
[55] A. Antoniadis, P. Tsoutsanis, I. Kokkinakis, Z. Rana, D. Drikakis, in: 53rd AIAA Aerospace Sciences Meeting, vol. 0813, 2015.
[56] A. Antoniadis, D. Drikakis, I.W. Kokkinakis, P. Tsoutsanis, Z. Rana, in: 20th AIAA International Space Planes and Hypersonic Systems and Technologies Conference, vol. 3524, 2015.
[57] P. Tsoutsanis, H. Srinivasan, in: ECCOMAS Congress 2016, Crete, Greece, 2016.
[58] P. Tsoutsanis, N. Simmonds, A. Gaylard, in: ECCOMAS Congress 2016, Crete, Greece, 2016.
[59] P. Tsoutsanis, D. Drikakis, in: ECCOMAS Congress 2016, Crete, Greece, 2016.
[60] N. Simmonds, P. Tsoutsanis, A. Antoniadis, K. Jenkins, A. Gaylard, Appl. Math. Comput. 336 (2018) 368–393.
[61] P. Tsoutsanis, J. Comput. Phys. 362 (2018) 69–94.
[62] P. Tsoutsanis, J. Comput. Phys. X 4 (2019), https://doi.org/10.1016/j.jcpx.2019.100037.
[63] A. Jalali, C. Ollivier-Gooch, in: 21st AIAA Computational Fluid Dynamics Conference, 2013.
[64] S. Diot, S. Clain, R. Loubére, Comput. Fluids 64 (2012) 43–63, https://doi.org/10.1016/j.compfluid.2012.05.004.
[65] M. Dumbser, M. Castro, C. Pares, E. Toro, Comput. Fluids 38 (9) (2009) 1731–1748.
[66] X. Nogueira, L. Cueto-Felgueroso, I. Colominas, F. Navarrina, M. Casteleiro, Comput. Methods Appl. Mech. Eng. 199 (37–40) (2010) 2544–2558.
[67] G.W. Stewart, Society for Industrial and Applied Mathematics SIAM, 1998.
[68] T.J. Barth, D.C. Jespersen, 27th Aerospace Sciences Meeting, 1989.
[69] V. Venkatakrishnan, J. Comput. Phys. 118 (1) (1995) 120–130, https://doi.org/10.1006/jcph.1995.1084.
[70] C. Michalak, C. Ollivier-Gooch, J. Comput. Phys. 228 (23) (2009) 8693–8711.
[71] R. Borges, M. Carmona, B. Costa, W. Don, J. Comput. Phys. 227 (2008) 3191–3211.
[72] M. Castro, B. Costa, W. Don, J. Comput. Phys. 230 (2011) 1766–1792.
[73] J. Zhu, C.-W. Shu, J. Comput. Phys. 406 (2020) 109212.
[74] B. Thornber, A. Mosedale, D. Drikakis, D. Youngs, R. Williams, J. Comput. Phys. 227 (10) (2008) 4873–4894.
[75] C.-D. Munz, S. Roller, R. Klein, K. Geratz, Comput. Fluids 32 (2) (2003) 173–196, https://doi.org/10.1016/S0045-7930(02)00010-5.
[76] J. Park, C.-D. Munz, Int. J. Numer. Methods Fluids 49 (8) (2005) 905–931, https://doi.org/10.1002/fld.1032.
[77] P. Degond, M. Tang, Commun. Comput. Phys. 10 (1) (2011) 1–31, https://doi.org/10.4208/cicp.210709.210610a.
[78] S. Boscarino, G. Russo, L. Scandurra, J. Sci. Comput. 77 (2) (2018) 850–884, https://doi.org/10.1007/s10915-018-0731-9.
[79] E. Abbate, A. Iollo, G. Puppo, SIAM J. Sci. Comput. 41 (5) (2019) A2850–A2879, https://doi.org/10.1137/18M1232954.

[80] S. Busto, M. Tavelli, W. Boscheri, M. Dumbser, Comput. Fluids 198 (2020), https://doi.org/10.1016/j.compfluid.2019.104399.
[81] S. Busto, L. Rio-Martin, M. Vazquez-Cendon, M. Dumbser, Appl. Math. Comput. 402 (2021), https://doi.org/10.1016/j.amc.2021.126117.
[82] F. Rieper, G. Bader, J. Comput. Phys. 228 (8) (2009) 2918–2933.
[83] J. Fernández-Fidalgo, X. Nogueira, L. Ramírez, I. Colominas, Comput. Methods Appl. Mech. Eng. 335 (2018) 91–127.
[84] P.A. Silva, P. Tsoutsanis, A. Antoniadis, Aerosp. Sci. Technol. 122 (2) (2022), https://doi.org/10.1016/j.ast.2022.107401.
[85] E. Toro, M. Spruce, W. Speares, Shock Waves 4 (1) (1994) 25–34.
[86] E. Toro, Shock Waves 29 (2019) 1065–1082.
[87] P. Roe, J. Comput. Phys. 43 (2) (1981) 357–372, https://doi.org/10.1016/0021-9991(81)90128-5.
[88] V. Rusanov, USSR Comput. Math. Math. Phys. 1 (1961) 267–279.
[89] L. Ivan, C. Groth, J. Comput. Phys. 257 (PA) (2014) 830–862, https://doi.org/10.1016/j.jcp.2013.09.045.
[90] H. Nishikawa, Comput. Fluids 49 (2011) 62–86.
[91] A. Jalali, M. Sharbatdar, C. Ollivier-Gooch, Comput. Fluids 101 (2014) 220–232.
[92] G. Gassner, F. Lorcher, C. Munz, J. Comput. Phys. 224 (2) (2007) 1049–1063.
[93] S. Gottlieb, C.-W. Shu, Math. Comput. 67 (221) (1998) 73–85, https://doi.org/10.1090/S0025-5718-98-00913-2.
[94] S. Yoon, A. Jameson, AIAA J. 26 (9) (1988) 1025–1026.
[95] A. Jameson, S. Yoon, AIAA J. 25 (7) (1987) 929–935.
[96] I. Men'shov, Y. Nakamura, in: A Collection of Technical Papers of 6th Int. Symp. on CFD, vol. 2, Lake Tahoe, Nevada, 1995, p. 815.
[97] I. Men'shov, Y. Nakamura, Comput. Fluids 29 (6) (2000) 595–616.
[98] H. Luo, J. Baum, R. Lohner, J. Comput. Phys. 146 (2) (1998) 664–690.
[99] M. Petrov, A. Tambova, V. Titarev, S. Utyuzhnikov, A. Chikitkin, Comput. Math. Math. Phys. 58 (11) (2018) 1865–1886.
[100] P. Batten, M. Leschziner, U. Goldberg, J. Comput. Phys. 137 (1) (1997) 38–78.
[101] R.J. Spiteri, S.J. Ruuth, SIAM J. Numer. Anal. 40 (2) (2002) 469–491.
[102] A. Jameson, AIAA paper 6 (1991-1596), 1991, https://doi.org/10.2514/6.1991-1596.
[103] E. Johnsen, T. Colonius, J. Comput. Phys. 219 (2) (2006) 715–732, https://doi.org/10.1016/j.jcp.2006.04.018.
[104] MPI, https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf. (Accessed 10 October 2021).
[105] OpenMP, https://www.openmp.org/. (Accessed 10 October 2021).
[106] BLAS, http://www.netlib.org/blas/. (Accessed 10 October 2021).
[107] K. G., Encyclopedia of parallel computing, https://doi.org/10.1007/978-0-387-09766-4_500, 2011.
[108] TecIO, https://www.tecplot.com/products/tecio-library/. (Accessed 10 October 2021).
[109] P. Tsoutsanis, A. Antoniadis, K. Jenkins, Comput. Fluids 173 (2018) 157–170, https://doi.org/10.1016/j.compfluid.2018.03.012.
[110] P. Tsoutsanis, ARCHER performance report, 2017, pp. 157–170, URL www.archer.ac.uk/community/benchmarks/archer-knl/KNLperfUCNS3D.pdf.
[111] Tecplot, https://www.tecplot.com. (Accessed 10 October 2021).
[112] Paraview, https://www.paraview.org. (Accessed 10 October 2021).
[113] Visit, https://visit-dav.github.io/visit-website/. (Accessed 10 October 2021).
[114] D. Balsara, C.-W. Shu, J. Comput. Phys. 160 (2) (2000) 405–452.
[115] C.-W. Shu, S. Osher, J. Comput. Phys. 83 (1) (1989) 32–78.
[116] D. Drikakis, C. Fureby, F. Grinstein, D. Youngs, J. Turbul. 8 (2007) 1–12.
[117] J. Bull, A. Jameson, AIAA J. 53 (9) (2015) 2750–2761.
[118] M. Dumbser, I. Peshkov, E. Romenski, O. Zanotti, J. Comput. Phys. 314 (2016) 824–862.
[119] J.-B. Chapelier, M. de la Llave Plata, E. Lamballais, Comput. Methods Appl. Mech. Eng. 307 (2016) 275–299.
[120] A. Sifounakis, S. Lee, D. You, J. Comput. Phys. 326 (2016) 845–861.
[121] C.-W. Shu, W.-S. Don, D. Gottlieb, O. Schilling, L. Jameson, J. Sci. Comput. 24 (1) (2005) 569–595.
[122] M. Brachet, D. Meiron, B. Nickel, R. Morf, U. Frisch, S. Orszag, J. Fluid Mech. 130 (1983) 411–452, https://doi.org/10.1017/S0022112083001159.
[123] J. Vassberg, E. Tinoco, M. Mani, B. Rider, T. Zickuhr, D. Levy, O. Brodersen, B. Eisfeld, S. Crippa, R. Wahls, J. Morrison, D. Mavriplis, M. Murayama, J. Aircr. 51 (4) (2014) 1070–1089, https://doi.org/10.2514/1.C032418.
[124] D. Levy, K. Laflin, E. Tinoco, J. Vassberg, M. Mani, B. Rider, C. Rumsey, R. Wahls, J. Morrison, O. Brodersen, S. Crippa, D. Mavriplis, M. Murayama, 2013.
[125] A.F. Antoniadis, P. Tsoutsanis, D. Drikakis, Comput. Fluids 146 (2017) 86–104, https://doi.org/10.1016/j.compfluid.2017.01.002.
[126] ANSYS, URL www.ansys.com/en-gb/training-center, 2021.
[127] ANSYS, URL www.ansys.com/en-gb/training-center, 2021.
[128] D. Garmann, M. Visbal, P. Orkwis, Int. J. Numer. Methods Fluids 71 (12) (2013) 1546–1565.
[129] A. Beck, T. Bolemann, D. Flad, H. Frank, G. Gassner, F. Hindenlang, C.-D. Munz, Int. J. Numer. Methods Fluids 76 (8) (2014) 522–548.
[130] F.X. Caradonna, C. Tung, NASA Technical Memorandum 81232, 1981.
[131] M. Costes, T. Renaud, B. Rodriguez, Int. J. Comput. Fluid Dyn. 26 (6–8) (2012) 383–405.
[132] N. Hariharan, A. Wissink, M. Steffen, M. Potsdam, in: 49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2011.
[133] J.D. Kocurek, J.L. Tangler, J. Am. Helicopter Soc. 22 (1) (1977) 24–35.
[134] A. Gardner, C. Wolf, M. Raffel, Prog. Aerosp. Sci. (2019) 100566.
[135] S.L. Wood, J.G. Coder, N.S. Hariharan, in: 2018 AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018.
[136] V. Coralic, T. Colonius, J. Comput. Phys. 274 (2014) 95–121, https://doi.org/10.1016/j.jcp.2014.06.003.
[137] A. Bagabir, D. Drikakis, Shock Waves 11 (2001) 209–218.
[138] J. Haas, B. Sturtevant, J. Fluid Mech. 181 (1987) 41–76, https://doi.org/10.1017/S0022112087002003.
[139] P. Tsoutsanis, E.M. Adebayo, A. Carriba Merino, A. Perez Arjona, M. Skote, J. Sci. Comput. 89 (2021), https://doi.org/10.1007/s10915-021-01673-y.
[140] H. Terashima, G. Tryggvason, J. Comput. Phys. 228 (11) (2009) 4012–4037, https://doi.org/10.1016/j.jcp.2009.02.023.
[141] J. Quirk, S. Karni, J. Fluid Mech. 318 (1996) 129–163, https://doi.org/10.1017/S0022112096007069.



https://dspace.org/

UCNS3D: An open-source high-order finite-volume unstructured CFD solver

Antoniadis, Antonis F.; Drikakis, Dimitris; Farmakis, Pericles S.; Fu, Lin; Kokkinakis, Ioannis ...

2022-06-21
Elsevier
https://doi.org/10.1016/j.cpc.2022.108453
Antoniadis AF, Drikakis D, Farmakis PS, et al., (2022) UCNS3D: An open-source high-order finite-volume unstructured CFD solver, Computer Physics Communications, Volume 279, October 2022, Article number 108453

Downloaded from a DSpace repository. https://dspace.org/