Computers and Fluids 173 (2018) 157–170

Contents lists available at ScienceDirect

Computers and Fluids

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

Improvement of the computational performance of a parallel unstructured WENO finite volume CFD code for Implicit Large Eddy Simulation

Panagiotis Tsoutsanis*, Antonis F. Antoniadis, Karl W. Jenkins

Centre for Computational Engineering Sciences, Cranfield University, Cranfield, MK43 0AL, United Kingdom

ARTICLE INFO

Article history:
Received 22 September 2017
Revised 29 January 2018
Accepted 1 March 2018
Available online 5 March 2018

ABSTRACT

In this paper the assessment and the enhancement of the computational performance of a high-order finite volume CFD code is presented. Weighted Essentially Non-Oscillatory (WENO) schemes are considered to be from the most computationally expensive numerical frameworks, in the context of high-resolution schemes particularly on hybrid unstructured grids. The focus of this study is to assess the computational bottlenecks of the solver for the WENO schemes for Implicit Large Eddy Simulation (ILES) and optimise the performance and efficiency through a series of code modifications e.g. formula rewriting, reduction of number operations, inclusion of linear systems libraries, non-blocking communications amongst others. The code is assessed on five different HPC systems; significant speed-up is achieved ranging from 1.5 to 8.5, with very high-order schemes benefiting the most. Good scalability is also obtained up to 10^4 number of cores, demonstrating viability and affordability of WENO type schemes for scale resolving simulations.

Crown Copyright © 2018 Published by Elsevier Ltd.
This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

Numerical methods of the high-order family were prohibited a decade ago in terms of great computational effort for large scale, moderate to high -Reynolds number cases; now these methods are being increasingly adopted for various industrial applications e.g. acoustics, combustion, rotating wings and turbo machinery and are available in commercial solver packages.

High-order numerical methods can provide improved accuracy at a reduced computational cost for transient CFD applications compared to second-order schemes. There is a wealth of high-order numerical methods for unstructured meshes developed across different frameworks including the finite volume (FV) [1–10], the Discontinuous Galerkin (DG) [11–16], the Spectral Finite Volume (SFV) methods [17–22] and the flux reconstruction scheme (FR) [23–28].

High-order methods such as WENO or DG with explicit time-stepping schemes for time advancement of time-dependent simulations have demonstrated to be attractive solutions for scale resolving simulations. The popularity of these solutions is mainly attributed to the favourable scalability of these solvers, where higher computational over communication load ratios can be obtained compared with lower-order spatial schemes.

WENO type schemes are considered to be computationally expensive particularly for unstructured meshes and for very high-order accuracy (>5th order). This is mainly due to their non-uniform stencils construction, cumbersome mesh partitioning, complex communication strategies; however, their enhanced accuracy, robustness, conservation properties, flexibility for multi-physics model extensions and ability to provide spurious-free solutions counterbalances the aforementioned drawback and makes them attractive to a wider set of applications. The method presented in this work is based on a series of works published over the past years; In references [7,29] the WENO framework is employed to discretise the Euler equations on hybrid unstructured grids; the extension of the method for time dependent laminar, transitional and turbulent flows are presented in [9] and for industrial scale applications and high-order discretisation of the RANS equations in [10]. The present higher-order WENO methods are producing significantly smaller errors for a series of test problems, compared to a standard 2nd-order Monotonic Upwind Scheme for Conservation Laws (MUSCL) scheme at approximately similar computing times [9,30]. Additionally they offer superior

* Corresponding author.
E-mail addresses: panagiotis.tsoutsanis@cranfield.ac.uk (P. Tsoutsanis), a.f.antoniadis@cranfield.ac.uk (A.F. Antoniadis), k.w.jenkins@cranfield.ac.uk (K.W. Jenkins).

https://doi.org/10.1016/j.compfluid.2018.03.012
0045-7930/Crown Copyright © 2018 Published by Elsevier Ltd. This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

scalability compared to lower-order schemes since their ratio of computation over communication time is larger than a MUSCL 2nd-order scheme [9].

Writing an efficient program code of WENO type method for mixed-element unstructured grids can be a daunting task. The complexity of the scheme and the explicit graph-type data format inherited by the unstructured framework results to several challenges: memory size limitation, random memory access patterns and computational cost amongst others. Major changes in the types of hardware utilised for scientific computing such as graphical processing units (GPUs), lower clock speeds, and many-core architecture processors and co-processors set additional challenges to the solver's efficiency. CFD codes will require radical changes in order to utilise the full potential of these new architectures. A possible and attractive direction would be the extension of current CFD solvers is to incorporate high-order and even very high-order methods which would be tuned to the extreme levels of parallelism of the new hardware. Additionally, that may imply that new implementation techniques of numerical methods need to be developed, taking into account the changes in hardware in terms of the different characteristics from the hardware architectures of the previous decades.

The prime objective of this current work is to assess the computational characteristics of WENO finite volume schemes for unstructured meshes for time dependent explicit Implicit Large Eddy Simulations (ILES). This is accomplished by identifying the code sections and subroutines that have the highest computational cost and memory footprint. The workflow pattern of unstructured finite volume CFD codes, includes operations characterised by non-coalescent memory accessing patterns, conditional branching for fluxes approximations, cell-based fine-grained adaptivity of the numerical methods. These type of operations prevents the CFD code from belonging to the embarrassingly parallel framework of applications.

As it is demonstrated in this work, the associated cost of these operations is insignificant compared to the cost of the numerous polynomial reconstructions that take place within the WENO framework. In this study we are interested in reducing the computational cost of WENO schemes, although other techniques for improving their compactness could be explored similarly to Dumbser et al. [31], we are pursuing this through a series of enhancements in the execution of the algorithms and operations rather altering the numerical methodology. These enhancements are tested across five different HPC architectures including a Intel Xeon Phi Knights-Landing manycore processor, and to the best of our knowledge this is the first time that very high-order finite volume WENO schemes for unstructured meshes, have been assessed, optimised and deployed for ILES of turbulent flows in this type of manycore architectures.

The rest of the paper is organized as follows. Section 2 is dedicated to the description of the general numerical framework, the implementation of these algorithms along with a brief description of the CFD solver code name UCNS3D (Unstructured Compressible Navier–Stokes 3D). In Section 3 we discuss the performance obtained with the original implementation of the code, identifying the most expensive processes, the optimisations explored for improving the computational performance, and the performance of the revised CFD code across five different HPC clusters for various schemes. Finally the conclusions of the present study are outlined in the last section.

2. Methodology

An overview of the governing equations, spatial and temporal discretisation schemes is presented; followed by the implementation of the schemes, where their computational and communication patterns are discussed in detail.

2.1. Governing equations

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

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

where $\mathbf{U}$ is the vector of the conserved mean flow variables, and $\underline{\underline{\mathbf{F}}}$ is the non-linear flux tensor given by:

$$\mathbf{U} = [\rho, \rho\vec{u}, E]^T, \tag{2}$$

$$\underline{\underline{\mathbf{F}}}(\mathbf{U}, \nabla \mathbf{U}) = \begin{pmatrix} \vec{u}^T \rho \\ \vec{u}^T \otimes \rho\vec{u} + \underline{\underline{\sigma}}(\mathbf{U}, \nabla \mathbf{U}) \\ \vec{u}^T (\mathbf{I}E + \underline{\underline{\sigma}}(\mathbf{U}, \overline{\overline{\nabla}}\mathbf{U})) - \kappa \nabla T \end{pmatrix}, \tag{3}$$

where the stress tensor $\underline{\underline{\sigma}}$ is given by:

$$\underline{\underline{\sigma}} = \left(p + \frac{2}{3}\mu \nabla \vec{u}\right)\mathbf{I} - \mu\left(\nabla \vec{u} + \nabla \vec{u}^T\right). \tag{4}$$

In the above equations, ρ is the density and ideal gas is assumed where the total energy is given 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 Sutherland law:

$$\frac{\mu_l}{\mu_0} = \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S}, \tag{5}$$

S is the Sutherland temperature and the subscript 0 denotes a reference state for the corresponding variables.

2.2. Numerical framework

The discretisation in a domain Ω is achieved by combining conforming arbitrary shaped elements of volume $|V_i|$. Integrating Eq. (1) over a mesh element using the finite volume formulation the ordinary differential equation is obtained as follows:

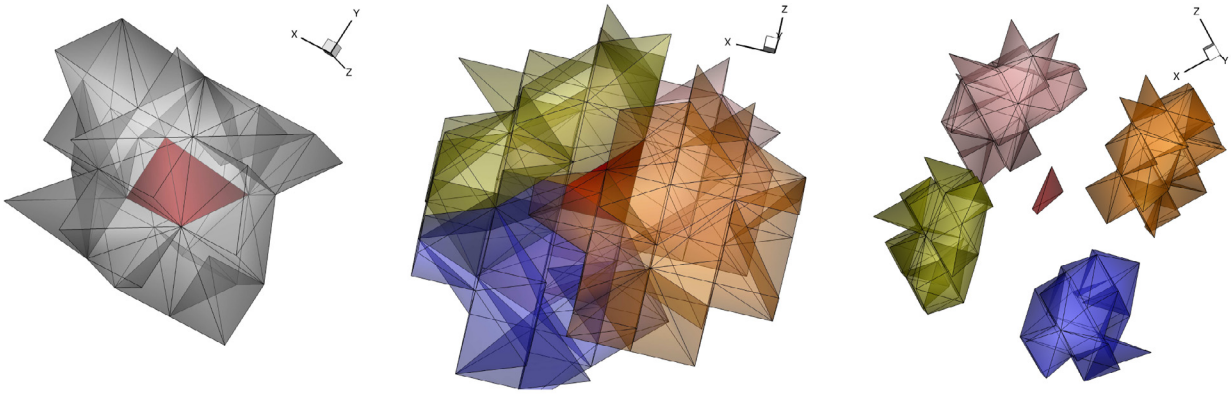
$$\frac{d\mathbf{U}_i}{dt} = \frac{1}{|V_i|} \sum_{l=1}^{N_f} \sum_{\alpha=1}^{N_{qp}} \underline{\underline{\mathbf{F}}}^{\mathbf{n},\mathbf{l}}(\mathbf{U}(\mathbf{x}_\alpha, t), \nabla \mathbf{U}(\mathbf{x}_\alpha, t))\omega_\alpha |A_l|, \tag{6}$$

where $\mathbf{U}_i$ is the volume averaged conserved variable vector, N_f is the number of faces per element, N_{qp} is the number of quadrature points used for approximating the surface integrals. $|A_l|$ is the surface area of the corresponding face, and α corresponds to different Gaussian integration points $\mathbf{x}_\alpha$ and weights ω_α over the face. The weight and distribution of the quadrature points depend upon the order of the Gaussian quadrature integration rule employed, and for the present study suitable rules for the employed polynomial order are used. The interface fluxes are computed based on the boundary extrapolated reconstructed values, which are obtained by a polynomial reconstruction from element-averaged data. The following section describes the methodology adopted for the spatial discretisation.

2.2.1. High-order finite volume k-exact least-square reconstruction

The main objective of the reconstruction process is to build a high-order polynomial $p_i(x, y, z)$ of arbitrary order r, for each considered element V_i that has the same average as a general quantity $\mathbf{U}_i$ this yields:

$$\mathbf{U}_i = \frac{1}{|V_i|} \int_{V_i} \mathbf{U}(x, y, z)\, dV = \frac{1}{|V_i|} \int_{V_i} p_i(x, y, z)\, dV. \tag{7}$$



(a) Central stencil (b) Directional stencils (c) Directional stencils,blown-up view

Fig. 1. Examples of central stencil and directional stencils for a 5th-order WENO scheme on a 3D tetrahedral mesh, with the considered element in red colour. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The present reconstruction algorithm is based upon the approaches of [7,29], that has been successful applied to smooth and discontinuous flow problems [9,30,32–34]. The reconstruction is carried out in a transformed system of coordinates in order to minimize scaling effects that appear in stencils consisting of elements of different size as shown in Fig. 1 and in order to improve the condition number of the system of equations. The reader is referred to [7,29,35] and references therein for more details of the approach adopted. The transformation is achieved by decomposing each element into tetrahedral elements. This transformation is only computed once at the initialisation phase of the simulation since the mesh does not change with time for the present study. Using an inverse mapping the considered element V_i can be transformed to the element V_i' in the reference co-ordinate system.

The spatial average of the conserved variable $\mathbf{U}_i$ does not change during coordinate transformation

$$\mathbf{U}_i = \frac{1}{|V_i|}\int_{V_i} \mathbf{U}(x,y,z)\ dV \equiv \frac{1}{|V_i'|}\int_{V_i'} \mathbf{U}(\xi,\eta)\ d\xi d\eta d\zeta . \tag{8}$$

To perform the reconstruction on the target element V_i, we construct the central reconstruction stencil $\mathcal{S}$ by recursively adding neighbouring elements, consisting of $M+1$ elements, including the target element V_i as shown in Fig. 1,

$$\mathcal{S} = \bigcup_{m=0}^{M} V_m, \tag{9}$$

where the index m refers to the local numbering of the elements in the stencil, with the element with index 0 being the considered element i. The rth order reconstruction polynomial at the transformed element V_0' is sought as an expansion over local polynomial basis functions $\phi_k(\xi,\ \eta,\ \zeta)$ given by:

$$p(\xi,\eta,\zeta) = \sum_{k=0}^{K} a_k\phi_k(\xi,\eta,\zeta) = \mathbf{U}_0 + \sum_{k=1}^{K} a_k\phi_k(\xi,\eta,\zeta), \tag{10}$$

where $\mathbf{U}_0$ corresponds to the general quantity at the considered element i, and $\xi,\ \eta,\ \zeta$ are the coordinates in the reference system. a_k are the degrees of freedom and the upper index in the summation of expansion K corresponds to the number of the degrees of freedom and is related to the order of the polynomial r by $K = \frac{1}{6}(r+1)(r+2)(r+3)-1$. For computing the degrees of freedom a_k, a minimum of K element are required within the stencil, in addition to the target element. For improving the robustness of the algorithm we use $M = 2\cdot K$ as described in [7,9,29,35].

To find the unknown degrees of freedom a_k for each element m from the stencil the cell average of the reconstruction polynomial $p(\xi,\ \eta,\ \zeta)$ must be equal to the element average of the solution $\mathbf{U}_m$:

$$\begin{aligned}\int_{V_m'} p(\xi,\eta,\zeta)\,d\xi d\eta d\zeta &= |V_m'|\mathbf{U}_0 + \sum_{k=1}^{K}\int_{V_m'} a_k\phi_k\,d\xi d\eta d\zeta \\ &= |V_m'|\mathbf{U}_m, \quad m = 1,\ldots,M.\end{aligned} \tag{11}$$

Where V_m' is the volume of the element m in the stencil, in the transformed coordinate system. Additionally the basis functions ϕ_k will satisfy the constraint of Eq. (7) irrespective of the values of degrees of freedom. The basis functions ϕ_k for all the elements in the stencil are defined as follows:

$$\phi_k(\xi,\eta,\zeta) \equiv \psi_k(\xi,\eta,\zeta) - \frac{1}{|V_0'|}\int_{V_0'} \psi_k\ d\xi d\eta d\zeta \quad k = 1,\ldots,K. \tag{12}$$

In the present study the basis functions are based on the Legendre type polynomial. Denoting the integrals of the basis function k over the considered element m in the stencil, and the vector of right-hand side by A_{mk} and $\mathbf{b}$ respectively as given by:

$$A_{mk} = \int_{V_m'} \phi_k\,d\xi d\eta d\zeta, \qquad b_m = |V_m'|(\mathbf{U}_m - \mathbf{U}_0), \tag{13}$$

we rewrite the equations for degrees of freedom a_k in a matrix form as

$$\sum_{k=1}^{K} A_{mk}a_k = b_m, \qquad m = 1,\ldots,M. \tag{14}$$

The complete linear system is given as:

$$\begin{bmatrix} \overline{\phi}_{1,1}(\xi,\eta,\zeta) & \overline{\phi}_{1,2}(\xi,\eta,\zeta) & \cdots & \overline{\phi}_{1,K}(\xi,\eta,\zeta) \\ \overline{\phi}_{2,1}(\xi,\eta,\zeta) & \overline{\phi}_{2,2}(\xi,\eta,\zeta) & \cdots & \overline{\phi}_{2,K}(\xi,\eta,\zeta) \\ \vdots & \vdots & \cdots & \vdots \\ \overline{\phi}_{M,1}(\xi,\eta,\zeta) & \overline{\phi}_{M,2}(\xi,\eta,\zeta) & \cdots & \overline{\phi}_{M,K}(\xi,\eta,\zeta) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_K \end{bmatrix}$$

$$= \begin{bmatrix} |V_1'|(\mathbf{U}_1 - \mathbf{U}_0) \\ |V_2'|(\mathbf{U}_2 - \mathbf{U}_0) \\ \vdots \\ |V_M'|(\mathbf{U}_M - \mathbf{U}_0) \end{bmatrix} \tag{15}$$

Where $\overline{\phi}_{m,k}$ corresponds to the integral of the basis function ϕ_k for element m. The matrix A_{mk} is purely based on the geometry of the stencil's elements, which remains unchanged during the simulation. On the other hand the information on the right-hand vector

b_m is dependent upon the solution, and the volume of each of the element in the stencil in the transformed coordinate system. The matrix $A^T_{km}A_{mk}$, is invertible and the Eq. (14), can be rewritten in the following form:

$$a_k = \left(A^T_{km}A_{mk}\right)^{-1}A^T_{km}b_m = A^{\dagger}_{km}b_m, \tag{16}$$

where the matrix $A^{\dagger}_{km}$ corresponds to the Moore–Penrose pseudo-inverse of A_{mk}. The polynomial coefficients are therefore obtained through a matrix vector multiplication of $A^{\dagger}_{km}$ with b_m. A QR decomposition based on Householder transformation is employed where the pseudo-inverse is decomposed in an orthogonal matrix Q and an upper triangular one R, and the inverse of R^TR is obtained by a forward-substitution followed by a backward-substitution, this yields:

$$A^{\dagger}_{km} = \left((QR)^T(QR)\right)^{-1}A^T_{km} = \left(R^TR\right)^{-1}A^T_{km}. \tag{17}$$

The main memory requirements of the least-square reconstruction are associated with the storage of the pseudo-inverse matrix $A^{\dagger}_{km}$, that is only computed once at the initialisation stage of the simulation.

2.2.2. WENO scheme

The WENO scheme employed in this study, utilises a non-linear combination of various reconstruction polynomials from the central stencil and the directional stencils; which are shown in Fig. 1, where each polynomial is weighted according to the smoothness of its solution, and it is based on the approaches of [7,35]. The polynomials are given as:

$$p^{\text{weno}}_i = \sum_{s=1}^{s_t}\omega_m p_s(\xi,\eta,\zeta), \tag{18}$$

where s_t is the total number of WENO stencils. Substituting back to Eq. (10) for $p_s(\xi,\eta,\zeta)$, we obtain the following expression:

$$p_s(\xi,\eta,\zeta) = \sum_{k=0}^{K}a^{(s)}_k\phi_k(\xi,\eta,\zeta). \tag{19}$$

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

$$\begin{aligned} p^{\text{weno}}_i &= \mathbf{U}_0 + \sum_{k=1}^{K}\left(\sum_{s=0}^{s_t}\omega_s a^s_k\right)\phi_k(\xi,\eta,\zeta) \\ &\equiv \mathbf{U}_0 + \sum_{k=1}^{K}\tilde{a}_k\phi_k(\xi,\eta,\zeta), \end{aligned} \tag{20}$$

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

$$\omega_s = \frac{\tilde{\omega}_s}{\sum_{s=1}^{s_t}\tilde{\omega}_s} \quad \text{where} \quad \tilde{\omega}_s = \frac{\lambda_m}{(\epsilon + \mathcal{SI}_s)^b}. \tag{21}$$

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

$$\mathcal{SI}_s = \sum_{1\le|\beta|\le r}\int_{V_0'}\left(\mathcal{D}^{\beta}p_s(\xi,\eta,\zeta)\right)^2(d\xi,d\eta,d\zeta), \tag{22}$$

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 of $\lambda_1 = 1000$ and a value to prevent division by zero of $\epsilon = 10^{-6}$ is used and D is the derivative operator. The reader is referred to [7,29] for the definition of geometrical sectors and a detailed explanation of the different set of geometrical conditions. The smoothness indicator is a quadratic function of the degrees of freedom (a^s_k) and Eq. (22) can be rewritten as:

$$\mathcal{SI}_s = \sum_{k=1}^{K}a^s_k\left(\sum_{q=1}^{K}\mathcal{OI}_{kq}a^s_q\right), \tag{23}$$

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

$$\mathcal{OI}_{kq} = \sum_{1\le|\beta|\le r}\int_{V_0'}\left(\mathcal{D}^{\beta}\phi_k(\xi,\eta,\zeta)\right)\left(\mathcal{D}^{\beta}\phi_q(\xi,\eta,\zeta)\right)(d\xi,d\eta,d\zeta), \tag{24}$$

and can be easily precomputed and stored at the initialisation stage of the simulation. Additionally the WENO reconstruction is carried out with respect to the characteristic variables for the present study. An average value in the direction normal to each element face is defined defined as $\mathbf{U}'_n$, the conserved vector $\mathbf{U}_{L,\,n}$, and the $\mathbf{U}_{R,\,n}$ correspond to the left and right states at the cell interface l in the normal direction. Denoting the right and left eigenvectors of the convective Jacobian matrix on the normal direction to the cell interface as $\mathbf{R}_l$ and $\mathbf{L}_l$ respectively, both of which are calculated based on the average cell state $\mathbf{U}'_n$. Then the projections of the degrees of freedom for each stencil s, a^s_k, are projected to the characteristic variables as:

$$\mathbf{B}^s_{ikl} = \mathbf{L}_l a^s_k, \qquad s = 1,\cdots,s_t, \qquad k = 0,\cdots,K, \tag{25}$$

where the WENO reconstruction is applied to each characteristic variable. Then the WENO modified degrees of freedom $\tilde{\mathbf{B}}_{ikl}$ are transformed back to the conserved formulation by the right eigenvector $\mathbf{R}_l$ as follows:

$$\tilde{a}_{k,l} = \mathbf{R}_l\tilde{\mathbf{B}}_{ikl}, \qquad k = 0,\cdots,K, \tag{26}$$

with the final WENO reconstruction polynomial for face l being given by:

$$\mathbf{P}_{il}(\xi,\eta,\zeta) = \mathbf{U}_0 + \sum_{k=1}^{K}\tilde{a}_{k,l}\phi_k(\xi,\eta,\zeta). \tag{27}$$

The WENO reconstruction with respect to characteristic variables is more computationally expensive as opposed to the conservative variables, since the WENO weights are different for each element's interface, and due to the forward and backward projection of the degrees of freedom to characteristic variables.

2.2.3. Gradients for viscous terms

For the evaluation of the viscous fluxes the boundary extrapolated values for velocity $\nabla\mathbf{u}_{i,\,l,\,a}$ and temperature $\nabla T_{i,\,l,\,a}$ for element i, for face l and Gaussian quadrature point a are required. They are computed by multiplying the transpose of the inverse Jacobian $(J^{-1})^T_i$ with the first order derivatives of the polynomial of the central stencil. For the gradient of u-velocity $\nabla u_{i,\,l,\,a}$ this is computed as follows:

$$\begin{pmatrix}\frac{\partial u}{\partial x}\\ \frac{\partial u}{\partial y}\\ \frac{\partial u}{\partial z}\end{pmatrix}_{i,l,a} = \left(J^{-1}\right)^T_i\begin{pmatrix}\frac{\partial p(\xi,\eta,\zeta)}{\partial\xi}\\ \frac{\partial p(\xi,\eta,\zeta)}{\partial\eta}\\ \frac{\partial p(\xi,\eta,\zeta)}{\partial\zeta}\end{pmatrix}_{i,l,a} \tag{28}$$

For the evaluation of the velocity and temperature gradients at the no-slip wall boundaries, the Dirichlet and Neumann boundary conditions are satisfied respectively through the least-square reconstruction similarly to [6,9] (Fig. 3).

2.2.4. Numerical fluxes

For the evaluation of the convective fluxes the approximate Harten-Lax-van Leer-Contact (HLLC) Riemann solver [36] is implemented for the inter-cell numerical flux. For the evaluation of the viscous fluxes the extrapolated interface variables $U^{\pm}_{i,a}$ and their unlimited gradients $\nabla U^{\pm}_{i,a}$ from the k-exact least-square reconstruction are averaged from two discontinuous states as detailed in [9,37]. For the gradients however additionally penalty terms are

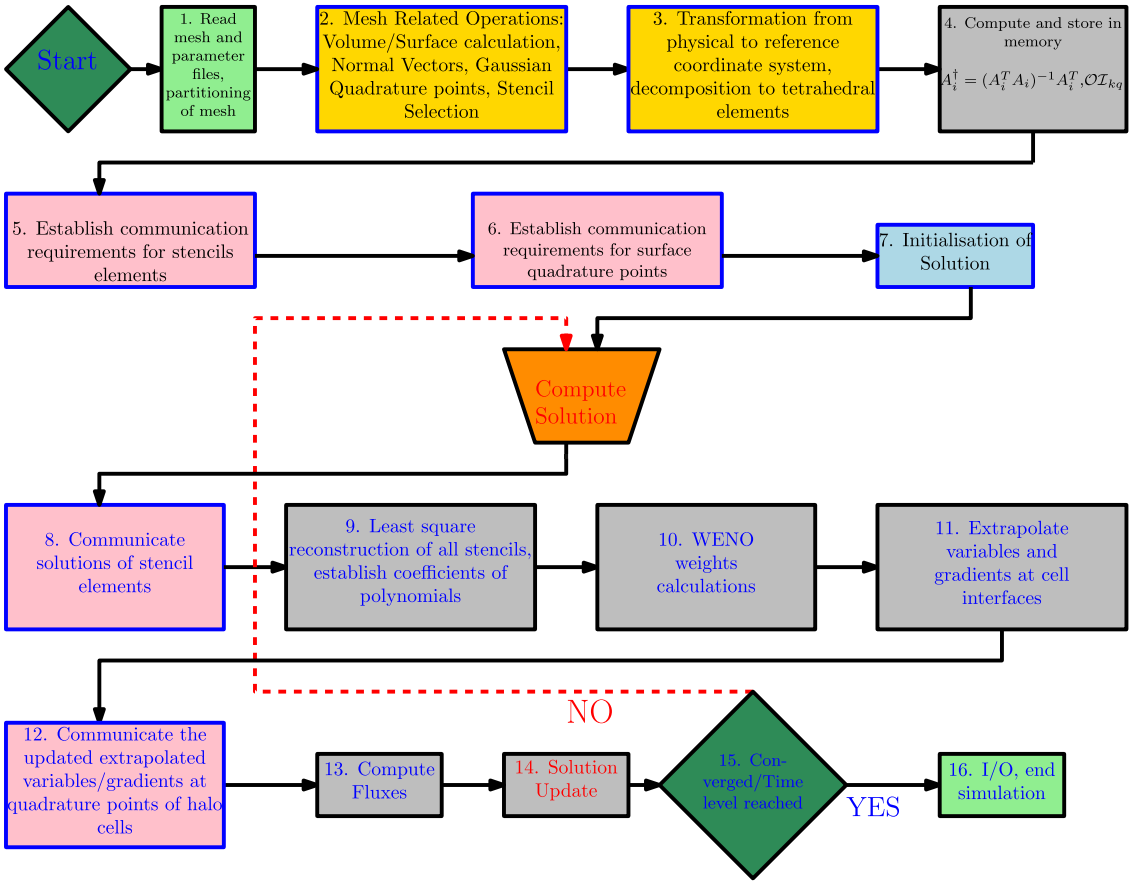


Fig. 2. Flow chart diagram of the UCNS3D code.

included following the formulation of Gassner et al. [38] for suppressing odd-even decoupling modes in the numerical solutions [39], in the following manner:

$$\nabla \mathbf{U} = \frac{1}{2}(\nabla \mathbf{U}_L + \nabla \mathbf{U}_R) + \frac{\alpha}{L_{int}}(\mathbf{U}_R - \mathbf{U}_L)\vec{n}, \tag{29}$$

where L_{int} is the distance between the cell-centres of the adjacent cells, and $\alpha = 4/3$ similarly to [39,40].

2.2.5. Time stepping algorithms

Having constructed the numerical fluxes $\underline{\underline{\mathbf{F}}}^{\mathbf{n,l}}$ as expressed in the semi-discrete conservative formulation, the next step involves the advancement of the solution in time. The explicit Strong Stability Preserving (SSP) Runge–Kutta 3rd-order method [41] has been employed for the time integration, and a CFL number of 0.9 for unsplit finite volume schemes [36] is used for all the test-cases (Fig. 4).

3. Code profiling and performance

The UCNS3D solver, is written in Fortran 2003 programming language, making use of object oriented programming, including abstract data types. It employs the message passing interface (MPI), and the Open Multi-Processing (OpenMP) application programming interface (API). The Metis partitioner [42] is used to decompose the mesh to numerous partitions; the total number of partitions is equal to the number of MPI processes. The work flow of the solver is composed into two main phases: the initialisation and the run-time as shown in Fig. 2.

Processes 1–7 correspond to the initialisation phase and the processes 8–15 correspond to the time advancement stage. In the

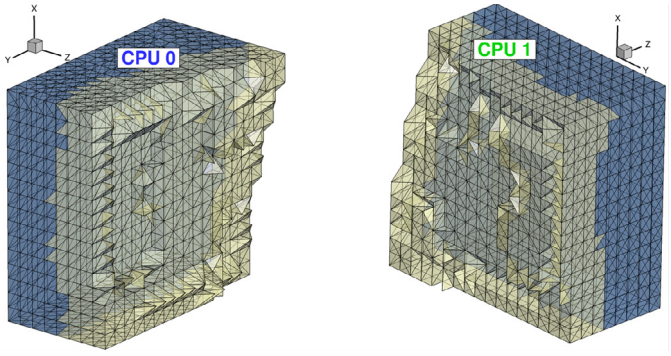


Fig. 3. Schematic of the communication requirement of stencil element variables between two partitioned domains.

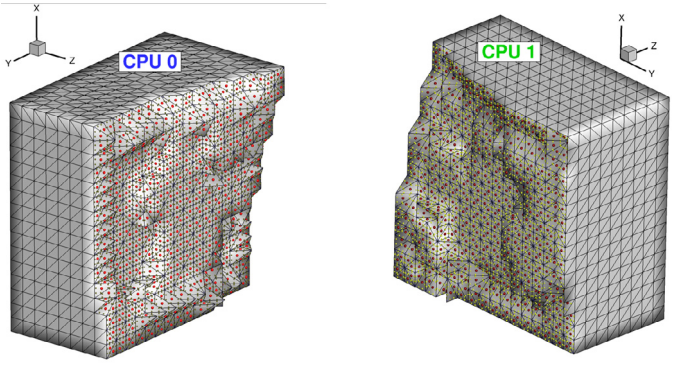


Fig. 4. Schematic of the communication requirement of boundary extrapolated variables at Gaussian quadrature points between two partitioned domains.

Table 1
Least-square system statistics.

Polynomial order r	Number of elements m per stencil	Number of degrees of freedom k per stencil
1	8	4
2	20	10
3	40	20
4	70	35
5	128	56
6	168	84

present study the main focus would be to improve the performance of the code for the run-time phase, therefore the I/O operations and the initialisation phases of the simulations are not accessed. As it can be seen in Fig. 2, the grey coloured boxes correspond to processes that are associated with floating point operations, the pink coloured boxes correspond to processes associated with communication between CPUs. In the following subsection details regarding the most expensive sections both in terms of computational and communication are discussed.

3.1. Least-squares reconstruction

For the least-squares reconstruction as shown in Eq. (16) the Moore–Penrose pseudo-inverse of A_{mk} matrix, $A^{\dagger}_{mk}$ is stored in the computer memory at the initialisation phase and the polynomial coefficients are therefore obtained through a matrix vector multiplication of $A^{\dagger}_{mk}$ with b_m as

$$a_k = A^{\dagger}_{mk} b_m. \tag{30}$$

This operation is repeated for all the conserved variables, and for all the admissible stencils. As the order of the polynomial increases the size of these matrices grows rapidly as shown in Table 1. This operation is implemented by making use of the intrinsic Fortran function **MATMUL**, and is performed as shown in the Algorithm 1.

Algorithm 1 Pseudocode for least-square reconstruction procedure for cell i.

```
procedure LSQ RECON(a_{k,j,s})
    for each admissible stencil s do
        for each variable j do
            a_{k,j,s} =MATMUL(A^†_{mk,s}, b_{m,j,s})
        end for
    end for
end procedure
```

3.2. WENO weights computation

The weights of Eq. (21) are computed for all element's face of the considered element, and for all the admissible stencils. As the order of the polynomial increases the size of these matrices grows fast as shown in Table 1. This operation is implemented by making use of the intrinsic Fortran function **MATMUL**, and **DOTPRODUCT** and is performed as shown in the Algorithm 2.

3.3. Extrapolation of variables and gradients

The WENO reconstructed degrees of freedom are computed according to Eq. (20), the next step involves the extrapolation of the reconstructed variables, along with the extrapolation of the unlimited gradients that are required by the viscous fluxes at the Gaussian quadrature points at element's interface. This operation is implemented by making use of the intrinsic Fortran function **MATMUL**, and is performed as shown in the Algorithm 3.

Algorithm 2 Pseudocode for WENO weights computation for cell i.

```
procedure WENO WEIGHTS(a_{k,j,s})
    for each face l do
        ! Averaging
        Compute U'_n = 1/2 (U_{L,n} + U_{R,n})
        Compute eigenvectors R_l and L_l
        for each admissible stencil s do
            for each degree of freedom k do
                ! Project to characteristic variables
                Compute B^s_{j,ikl}=MATMUL(L_{j,j}, a^s_{j,k})
            end for
            for each conserved variable j do
                SI_s =MATMUL(C I_{kq}, B^s_{j,ikl})
                SI_s =DOTPRODUCT(SI_s, B^s_{j,ikl})
            end for
        end for
        ! Compute Weights
        for each conserved variable j do
            Set ω̃_s = 0, ω_s = 0
            for each admissible stencil s do
                Compute ω̃_s = λ_s / (ε + SI_s)^4
            end for
            Compute sum Σ_{s=1}^{s_t} ω̃_s
            for each admissible stencil s do
                Compute ω_s = ω̃_s / (Σ_{s=1}^{s_t} ω̃_s)
            end for
        end for
        for each degree of freedom k do
            B̃_{j,ikl} = ( Σ_{s=0}^{s_t} ω_s B^s_{j,ikl} )
        end for
        ! Project to conserved variables
        for each degree of freedom k do
            Compute ã_{j,ikl}=MATMUL(R_{j,j}, B̃_{j,ikl})
        end for
    end for
end procedure
```

3.3.1. Communication of stencil elements variables

The first communication requirement between different processors, is to exhange the mean variables of the elements in the stencils $b_{m,j,s}$ that belong to other processors, in order to solve the least-square reconstruction problem. It must be noted that due to the nature of the WENO directional stencils, and the geometrical sectors defined by the considered cells, there are instances where due to the high-aspect ratio cells the directional stencils might extend farther away from the central stencil. This implies that there might be cases where information might be needed from some processors, that they might not require any information from the current processor. Therefore resulting in cases where it is only needed to receive rather than send data from some processors. The `MPI_SENDRECV` function is used as shown in Algorithm 4.

3.4. Communication of variables/gradients at quadrature points of halo cells

The second requirement between different processors, is to exchange the reconstructed variables and their gradients at each element's interface between processor blocks for each Gaussian quadrature point. The operation is performed by using the **MPI SENDRECV** function as shown in Algorithm 5.

Algorithm 3 Pseudocode for extrapolation of variables/gradients at cell faces *i*.

procedure EXTRAPOL($\mathbf{U}(\mathbf{x}_\alpha, t), \nabla\mathbf{U}(\mathbf{x}_\alpha, t)$)
  for each face l **do**
    for each Gaussian quadrature point α **do**
      for each degree of freedom k **do**
        Compute the basis functions $\phi_k(\xi, \eta, \zeta)$
      end for
      ! Convective part
      for each variable j **do**
        Compute the sum $\sum_{k=1}^{K} \tilde{a}_{j,ikl}\phi_k(\xi, \eta, \zeta)$
        Compute the reconstructed solution as
        $\mathbf{U}_{i,j,l,\alpha} = \mathbf{U}_{i,j} + \sum_{k=1}^{K} \tilde{a}_{j,ikl}\phi_k(\xi, \eta, \zeta)$
      end for
      ! Diffusive part
      Using only the central stencil $s = 1$
      for each variable j **do**
        The variables now are (u, v, w, T)
        for each degree of freedom k **do**
          Obtain $\frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\xi}, \frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\eta}, \frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\zeta}$
        end for
        Compute $\frac{\partial U_{i,j,l,\alpha}}{\partial\xi} = \sum_{k=1}^{K} \frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\xi} a_{k,j}$
        Compute $\frac{\partial U_{i,j,l,\alpha}}{\partial\eta} = \sum_{k=1}^{K} \frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\eta} a_{k,j}$
        Compute $\frac{\partial U_{i,j,l,\alpha}}{\partial\zeta} = \sum_{k=1}^{K} \frac{\partial\phi_k(\xi,\eta,\zeta)}{\partial\zeta} a_{k,j}$
        $\nabla U_{i,j,l,\alpha}(\mathbf{x}) =$
        $= \mathbf{MATMUL}((J^{-1})_i^T, \nabla U_{i,j,l,\alpha}(\xi))$
      end for
    end for
  end for
end procedure

Algorithm 4 Pseudocode for stencil elements communication.

procedure COMMSTENCIL($\mathbf{U}_m$)
  COMMUNICATION 1
  for each CPU J that only receives information from the current CPU N **do**
    Create 1D array to store the cell average values $\mathbf{U}_m$ to be sent from CPU N
  end for
  for each CPU J that only receives information from the current CPU N **do**
    CALL `MPI_SENDRECV`
  end for
  COMMUNICATION 2
  for each CPU L that only sends information to the current CPU N **do**
    Create 1D array to store the cell average values $\mathbf{U}_m$ to be received from CPU L
  end for
  for each CPU L that only sends information to the current CPU N **do**
    CALL `MPI_SENDRECV`
  end for
  COMMUNICATION 3
  for each CPU R that receives/sends information from/to the current CPU N **do**
    Create 1D array to store the cell average values $\mathbf{U}_m$ to be received from CPU R
    Create 1D array to store the cell average values $\mathbf{U}_m$ of all the elements to be sent to CPU R
  end for
  for each CPU R that receives/sends information from/to the current CPU N **do**
    CALL `MPI_SENDRECV`
  end for
end procedure

Algorithm 5 Pseudocode for inter-processor quadrature points communications.

procedure COMMBOUND($\mathbf{U}(\mathbf{x}_\alpha, t), \nabla\mathbf{U}(\mathbf{x}_\alpha, t)$)
  for each CPU P that receives/sends information from/to the current CPU N **do**
    Create 1D array to store the boundary extrapolated values and gradients to be received from CPU P
    Create 1D array to store the boundary extrapolated values and gradients to be sent to CPU P
  end for
  for each CPU P that receives/sends information from/to the current CPU N **do**
    CALL `MPI_SENDRECV`
  end for
end procedure

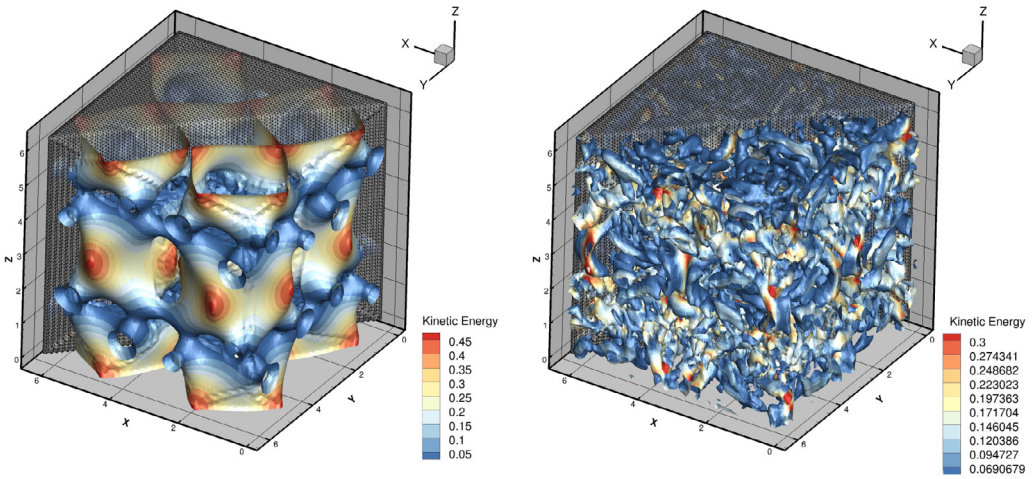
4. Numerical setup

This section presents the results obtained for various three dimensional test cases. The performance of the original implementation of CFD code is assessed in terms of scalability and efficiency as well as identifying the most time-consuming procedures. Then, a justification of the improvements that were pursued and detailed and finally the performance of the revised implementation of the CFD code is discussed. For all test cases the statistics in terms of maximum computational time, and maximum communication time are collected for 1000 iterations and the process is repeated at least three times to ensure reproducibility of the statistics and finally the average of those is used.

4.1. Data sets

For the numerical experiments, two test-problems are assessed of different computational and communication requirements. The first test problem is the ILES of the viscous Taylor–Green Vortex flow at a Reynolds number of $Re = 1,600$ which is used to study vortex stretching and dissipation characteristic of numerical schemes. It has been widely employed by many authors to quantify the dissipation behaviour of various high-order methods [43–48]. For the Taylor-Green Vortex flow case, two grids are utilised consisting of 0.2 and 2.1 million tetrahedral elements and two numerical schemes a WENO 2nd-order and a WENO 6th-order since they have vastly different computational requirements and hence will provide a representative indication of the relative performance improvements. Indicative results obtained on the fine mesh with a WENO 6th-order scheme are shown in Fig. 5 where the turbulent structures are shown with the q-criterion isosurfaces for two instances, one at the start and one at the end of the simulation.

The second test case is an ILES of the transitional turbulent subsonic flow past the SD7003 wing at an angle of attack of 8°, a Mach number of 0.2, and a Reynolds number of $Re = 60,000$. This test case is widely employed for assessing the behaviour of various numerical schemes for external aerodynamics by Visbal [49], Uranga et al. [50], Rizzetta and Visbal [51], Garmann et al. [52], Beck et al. [53], Bassi et al. [54], and Vermeire and Vincent [23].



(a) $t = 1.0$ (b) $t = 9.8$

Fig. 5. Isosurfaces of Q criterion at $Q = 0.5$, coloured by kinetic energy at different instants for the 3D Taylor Green Vortex test problem using a WENO 5th-order scheme on the tetrahedral mesh.

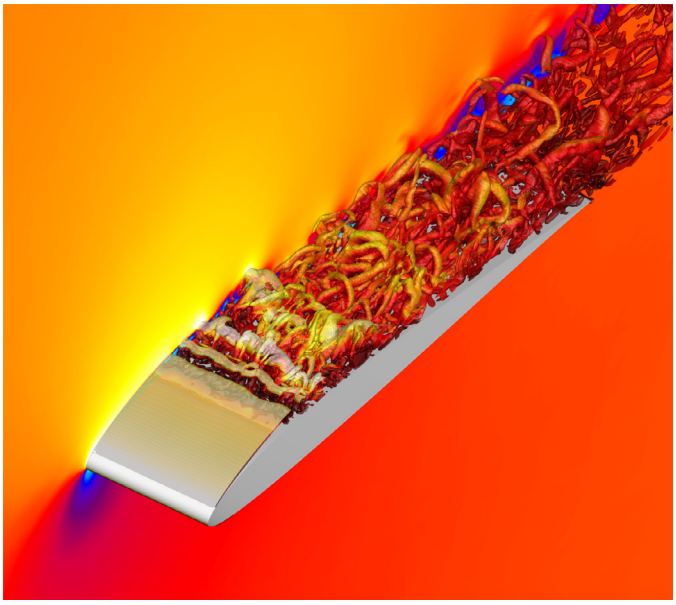


Fig. 6. Isosurfaces of Q criterion at $Q = 100$, coloured by Mach number for the SD7003 wing test problem using a WENO 4th-order scheme.

One hybrid grid is generated consisting of approximately 5 million hexahedral and 6 million prismatic elements with periodic boundary conditions on z-axis and farfield boundary conditions elsewhere apart from the surface of the wing where no-slip boundary conditions are enforced. For this test problem the WENO 4th-order scheme is employed since it was considered to have a good balance between accuracy and cost, given the computational budget available. The flow over the wing is shown in Fig. 6 where the separation possible transition near the leading edge is shown with the help of isosurfaces of q-criterion coloured by the Mach number.

4.2. HPC platforms

Five HPC platforms are utilised two assess the solver performance on both test-cases; each platform has different specifications in terms of interconnect, processors and memory, the information is listed in Table 2. For all the HPC platforms apart from ARCHER (KNL) platform, the **MPI** implementation is used with Intel MPI 5.1.3 and Intel Fortran compiler version 16.0.3. For all the runs apart from ARCHER (KNL) platform the exact compilation flags are set `-i4 -r8 -O3 -ipo -xHost -fp model precise`. For the ARCHER (KNL) platform the **MPI+OpenMP** implementation is employed with Intel MPI 2017, and Intel Fortran compiler version 17.0.0. For all the runs in the ARCHER (KNL) platform the compilation flags are set as `-i4 -r8 -O3 -xmic-avx512 -ipo -fp-model precise -qopenmp -qopenmp-link=static` and a guided schedule is used for all the **OpenMP** regions, since it was found to be fastest. Enabling the inter procedural optimisation has shown to have a significant effect on the performance, approximately 30% speed up and it is therefore used for all the tests. The computational times presented hereafter correspond to the maximum time across all CPUs, in other word the slowest CPU was taken into account for all the simulations, since it was found to be the most representative of the time that the simulation will take.

4.3. Original code performance

Assessing the performance of the original code implementation the Taylor Green Vortex on the finest tetrahedral mesh test case was utilised using the WENO 6th-order scheme. The purpose of this initial test is to identify the time-consuming processes of the simulation, and proceed with optimisation of only the most compute intensive operations. The test was performed on the Hazelhen HPC platform using 384, 3072 and 12,288 cores. The breakdown of the percentage of the total time taken by various processes is illustrated in Fig. 7 top three pie charts. It can be noticed that the WENO weights calculation is the most expensive operation followed by the least-square reconstruction and the extrapolation of the flow variables and gradients at the cell interfaces. In terms of communication cost the most expensive part is the communication of the reconstructed solutions and their gradients for the Gaussian quadrature points between inter-processor boundaries rather than the communication of the solutions for the stencil elements across inter-processor boundaries. Therefore only three processes were identified as having the potential to significantly improve the performance of the CFD code namely the WENO weights calculation, the least-squares reconstruction and the communication for the Gaussian quadrature points between inter-processor boundaries and the optimisation of those will be pursued.

Table 2
Description of HPC resources used for the numerical tests.

Specifications	Hazelhen (HW)	SuperMUC (SB)	SuperMUC (HW)	ARCHER (IB)	ARCHER (KNL)
Computing Site	High Performance Computing Center Stuttgart (HLRS)	Leibniz Supercomputing Centre (LRZ)	Leibniz Supercomputing Centre (LRZ)	UK National Supercomuting Centre (EPCC)	UK National Supercomuting Centre (EPCC)
Location	Stuttgart, Germany	Garching, Germany	Garching, Germany	Edinburgh, UK	Edinburgh, UK
Processor Type	Intel Haswell E5-2680 v3	Intel Sandy Bridge E5-2680	Intel Haswell E5-2697 v3	Intel Ivy Bridge E5-2697 v2	Intel Knights Landing Xeon Phi (7210)
Processor Frequency	2.5 GHz	2.7 GHz	2.6 GHz	2.7 GHz	1.3 GHz
Cores per node	24	16	28	24	64 (4 hardware threads)
Memory per node	128 GB	32 GB	64 GB	64/128 GB	96 GB and 16 GB on chip (MCDRAM)
Number of nodes	7712	9216	3072	4544 (64 GB), 376 (128 GB)	12
Interconnect	Cray Aries	Infiniband FDR10	Infiniband FDR14	Cray Aries	Cray Aries

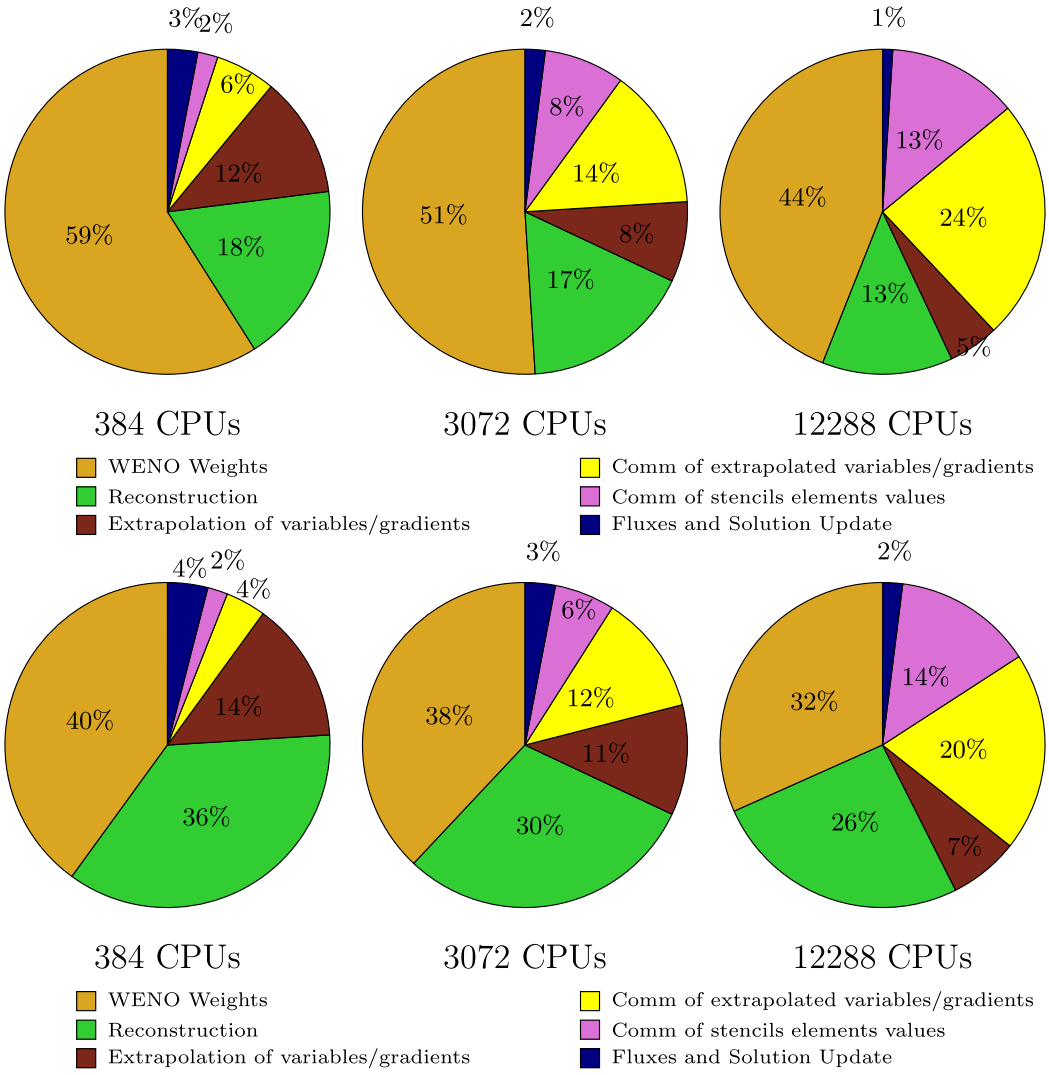


Fig. 7. Top: baseline code performance profile, **bottom**: optimised code performance profile. 3D Taylor Green Vortex test problem using a WENO 6th-order scheme on the finest tetrahedral mesh, at Hazelhen HPC platform.

4.4. Reconstruction optimisation performance

The two processes associated with the reconstruction method including the WENO weights computation and the least-square reconstruction are further assessed using the Intel Vtune profiler. It was confirmed what already has been seen from manual time measurement and log output, the main computational hotspot is the WENO weights calculation algorithm. Within this routine the most costly part was the projection to characteristic variables for each degree of freedom and the weights calculation, since this is a process repeated for all the stencils, and all the faces of the considered cell. For both processes most of the time was spent in dense linear algebra operations including matrix-vector products and dot products using the intrinsic Fortran **MATMUL** functions. The key

Table 3
CPU times and speed ups for the WENO reconstruction routine and two of its sub-routines, for the Taylor–Green vortex test problem, using a WENO 6th-order scheme on Hazelhen at 512 processes for the coarse tetrahedral mesh.

Routine	Original code time	Preliminary code (GEMV) Time | Speed up	Final code (GEMM) Time | Total speed up
WENO Reconstruction	3073 s	1466 s|2 ×	478 s|6.5 ×
WENO weights	2135 s	429 s|5 ×	197 s|10 ×
LSQ Reconstruction	780 s	367 s|2 ×	157 s|5 ×

ingredients to improve the performance of these routines are listed below:

1. Modify the storage of multidimensional arrays to optimise memory access patterns.
2. Rewrite loops of matrix-vector multiplications, into matrix-matrix multiplications when possible.
3. Utilise **BLAS** library for matrix multiplications instead of intrinsic Fortran functions.

4.4.1. Least-square reconstruction optimisation

For the least-square reconstruction, outlined in Algorithm 1, we improve the memory access patterns by storing all matrices with the fastest varying index stored first following the Fortran column major order. Moreover, the loops are organised in such a way that the largest index is looped last in order to minimize loop overheads. The matrix-vector product of **MATMUL** in line 4 of Algorithm 1 is replaced with the equivalent call to **GEMV** provided by Fortran 95 **BLAS** binding from Intel MKL, version 11.3. This can be seen in Table 3, in the columns "Original Code" and "Preliminary Code (GEMV)", in that way the computation of the least-square reconstruction is speeded up by a factor of 2. Pursuing additional optimisation within the loop where all the conserved variables are computed in line 3 of Algorithm 1, by removing the **MATMUL** Fortran function to perform the matrix-vector multiplication of the pseudo inverse matrix and the vector of each of the conserved variables for all the stencil elements. This function is replaced with a matrix-matrix multiplication of the pseudo-inverse matrix with the matrix containing the conserved variables $\mathbf{b}_{m,j,s}$ for all the stencil elements using the **GEMM** library function as shown in Algorithm 6. This modification resulted in the least-square reconstruction process to take 5 times less time as shown in Table 3.

Algorithm 6 Pseudocode of optimised Least-square reconstruction procedure for cell i.

1: **procedure** LSQ RECON($a_{k,j,s}$)
2: **for** each admissible stencil s **do**
3: $\mathbf{a}_{k,j,s}$ =**GEMM**($A^{\dagger}_{mk,s}$, $\mathbf{b}_{m,j,s}$)
4: **end for**
5: **end procedure**

4.4.2. WENO weights computation optimisation

For the WENO weights computation a similar optimisation strategy as the least-square reconstruction is adopted. **MATMUL** functions are replaced by calls to **GEMV BLAS** library functions. This optimisation resulted in a speed up of a factor of 5 compared to the original code as it can be seen in Table 3. By eliminating the loops performed at lines 7, 11 and 31 of Algorithm 2, and replacing the **MATMUL** function of the matrix-vector product with the **GEMM BLAS** library functions of the equivalent matrix-matrix products, and the **DOTPRODUCT** by the **DOT BLAS** library functions as seen in Algorithm 7 a speed-up of 10 is achieved compared to the original demonstrated in Table 3.

Algorithm 7 Pseudocode of optimised WENO weights computation for cell i.

1: **procedure** WENO WEIGHTS($a_{k,j,s}$)
2: **for** each face l **do**
3: **! Averaging**
4: Compute $\mathbf{U}'_n = \frac{1}{2}(\mathbf{U}_{L,n} + \mathbf{U}_{R,n})$
5: Compute eigenvectors $\mathbf{R}_l$ and $\mathbf{L}_l$
6: **for** each admissible stencil s **do**
7: **! Project to characteristic variables**
8: Compute $\mathbf{B}^s_{j,ikl}$=**GEMM**($\mathbf{L}_{j,j}$, $\mathbf{a}_{k,j,s}$)
9: SI_s =**GEMM**($\mathcal{OI}_{kq}$, $\mathbf{B}^s_{j,ikl}$)
10: **for** each conserved variable j **do**
11: $\mathcal{SI}_s$ =**DOT**(SI_s, $\mathbf{B}^s_{j,ikl}$)
12: **end for**
13: **end for**
14: **! Compute Weights**
15: **for** each conserved variable j **do**
16: Set $\tilde{\omega}_s = 0$, $\omega_s = 0$
17: **for** each admissible stencil s **do**
18: Compute $\tilde{\omega}_s = \frac{\lambda_s}{(\epsilon+\mathcal{SI}_s)^4}$
19: **end for**
20: Compute sum $\sum_{s=1}^{s_t} \tilde{\omega}_s$
21: **for** each admissible stencil s **do**
22: Compute $\omega_s = \frac{\tilde{\omega}_s}{\sum_{s=1}^{s_t} \tilde{\omega}_s}$
23: **end for**
24: **end for**
25: **for** each degree of freedom k **do**
26: $\tilde{\mathbf{B}}_{j,ikl} = \left(\sum_{s=0}^{s_t} \omega_s \mathbf{B}^s_{j,ikl}\right)$
27: **end for**
28: **! Project to conserved variables**
29: Compute $\tilde{a}_{j,ikl}$=**GEMM**($\mathbf{R}_{j,j}$, $\tilde{\mathbf{B}}_{j,ikl}$)
30: **end for**
31: **end procedure**

4.5. Communication optimisation performance

The Communication of stencil elements variables has a much smaller footprint in terms of time taken compared to the communication of the boundary extrapolated variables and gradients for all the test runs performed. Therefore the optimisation efforts were focused on the communication of the boundary extrapolated variables and gradients. The MPI-3 sparse collective operations is considered for the optimisation of the boundary extrapolated variables and gradients communication exchange at each Gaussian quadrature point at each element/block interface. The collective operations with MPI Types such as `MPI_Dist_graph_create_adjacent` and `MPI_Neighbour_alltoall` for avoiding the overhead of manual send and receive buffer packing and unpacking are considered. However, it got apparent that the time required for copying data to and from send and receive buffers was negligible

Table 4
Runtime for the communication routine of the boundary extrapolated variables at Gaussian quadrature points and two of its sub-routines, for the SD7003 test problem, using a WENO 4th-order scheme on Hazelhen at 1024 processes.

Routine	Original code time	Non-blocking communication time | Total speed up
Reconstructed solution exchange	3084 s	2642 s|1.16 ×
⊢ MPI_SENDRECV	3059 s	–
⊢MPI_WAITALL	–	2616 s

compared to the communication of the messages and therefore this approach was not implemented. On the other hand the blocking MPI_SENDRECV calls are replaced by non-blocking communications of MPI_ISEND, MPI_IRECV and MPI_WAITALL as shown in Algorithm 8. This resulted in an improvement of 16% with respect to the original code as shown in Table 4.

Algorithm 8 Pseudocode for optimised inter-processor quadrature points communications.

```
procedure COMMBOUND(U(x_α, t), ∇U(x_α, t))
    for each CPU P that receives/sends information from/to the current CPU N do
        Create 1D array to store the boundary extrapolated values and gradients to be received from CPU P
        Create 1D array to store the boundary extrapolated values and gradients to be sent to CPU P
    end for
    for each CPU P that receives/sends information from/to the current CPU N do
        CALL MPI_ISEND
        CALL MPI_IRECV
    end for
    CALL MPI_WAITALL
end procedure
```

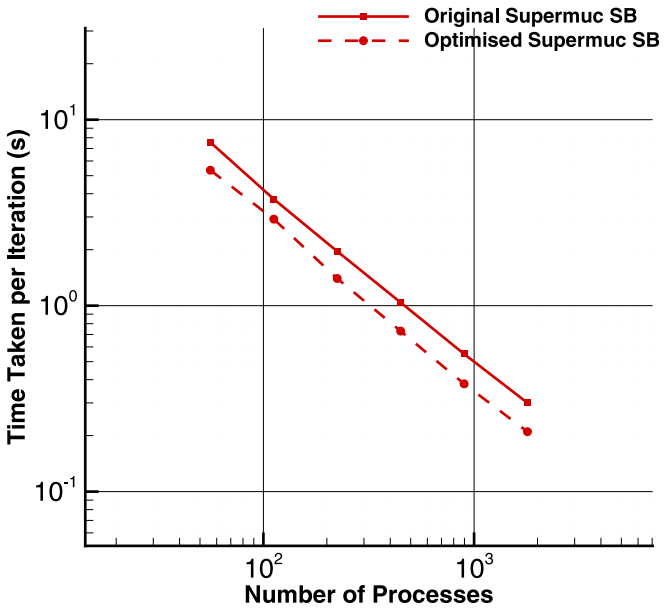


Fig. 8. Strong scalability results for the ILES of the Taylor-Green vortex using the WENO 2nd-order scheme on the finest tetrahedral mesh.

4.6. Final performance gains

The performance gains of the revised code are assessed on the same tests and the same number of processors for the Taylor Green Vortex on the finest tetrahedral mesh test case using a WENO 6th-order scheme. The purpose of this test is to identify significant differences in the breakdown of computational time similarly to the Fig. 7. The tests are performed on the Hazelhen HPC platform using 384, 3072 and 12,288 cores. The breakdown of the percentage of the total time taken by processes is illustrated in Fig. 7. WENO weights process have equivalent cost to the least-square reconstruction, also noticed in Table 3. The percentage of total time taken by the communication of the boundary extrapolated values and gradients is reduced from 24% to 20%.

Since the computational load increases as the polynomial order gets higher, we are expecting different gains for different orders of accuracy for the same test problem. This is reflected by the performance improvements of the greater size of the matrix operations while employing the **GEMM** library calls. At the same time there are overheads associated with making calls to the external library. However, these overheads become increasingly insignificant with higher order discretisation.

We perform simulations employing the WENO 2nd-order scheme on the finest tetrahedral mesh for the Taylor–Green vortex on the SuperMUC (SB) cluster by performing a strong scalability test of the original code and the optimised code. The simulations are performed from 48 to 1024 CPUs as seen in Fig. 8; there isn't significant drop in performance in terms of scalability. It apparent that optimised code is approximately 50% faster per iteration than the original code, even with the overheads associated with **GEMM** calls, since the cumulative improvements. It has to be noted that on 1024 CPUs, approximately 2050 elements are assigned to each CPU.

The second test is composed of simulation employing the WENO 6th-order scheme on the finest tetrahedral mesh for the Taylor-Green vortex, on the SuperMUC (SB) and (HW) cluster, and on the Hazelhen (HW) cluster. The goal is to perform strong scalability test of both codes and assess the improvement across different machines. The simulations are performed from 384 to 12,288 CPUs as seen in Fig. 9. The speed up for the WENO 6th-order scheme spans from 5 to 6.5, across all CPU counts. Similar performance is obtained for the simulation running on 384 CPUs on the revised code with the same case running on 2400 CPUs on original code. It needs to be stressed that at the highest core counts each CPU holds is approximately 160 elements, highlighting that the ratio of computational to communication load is still high even for low number of elements assigned to each CPU block. Furthermore, the scalability performance hold at a acceptable level for the number of cpu tested, suggesting that higher core counts could be explored with lower number of elements per CPU. For very high-order of accuracy it is expected that the improvements in performance will be even greater, since the performance of the GEMM library calls are proportional to the size of the matrices.

The flow past the SD7003 wing is considered in the context of ILES; the discretisation scheme employed is the WENO 4th-order scheme and the case is run on the SuperMUC (HW) cluster. The simulations are performed from 864 to 7200 CPUs as seen in Fig. 10. Speed up for the WENO 4th-order scheme ranges between 1.8 and 2.2, across all CPU counts. The WENO 4th-order scheme

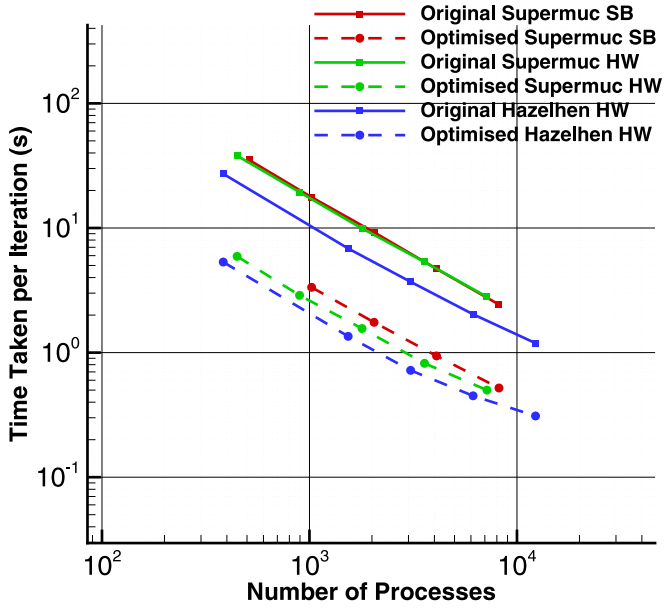


Fig. 9. Strong scalability results for the ILES simulation of the Taylor–Green vortex using the WENO 6th-order scheme on the finest tetrahedral mesh.

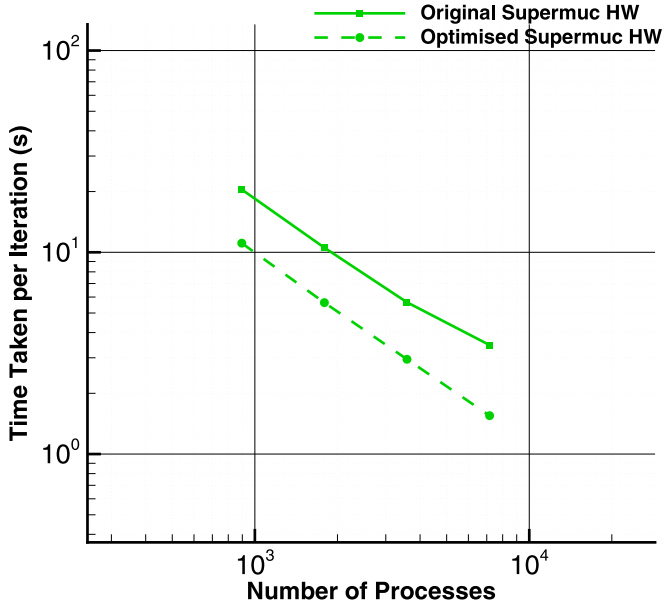
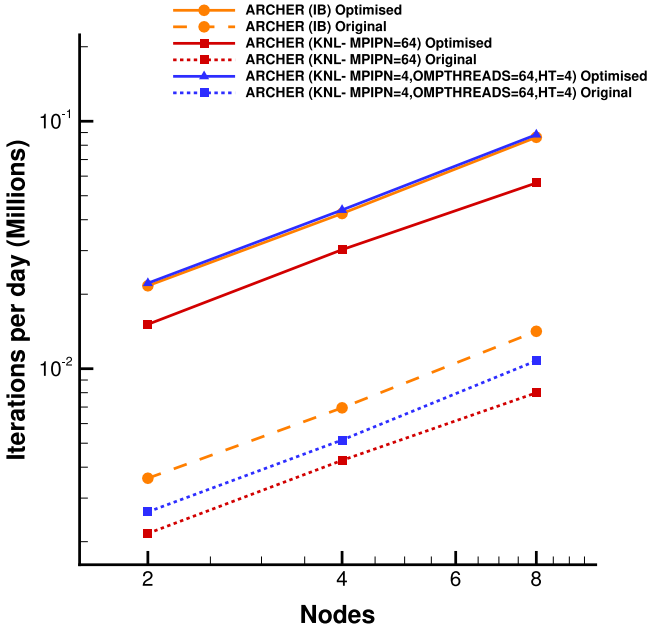
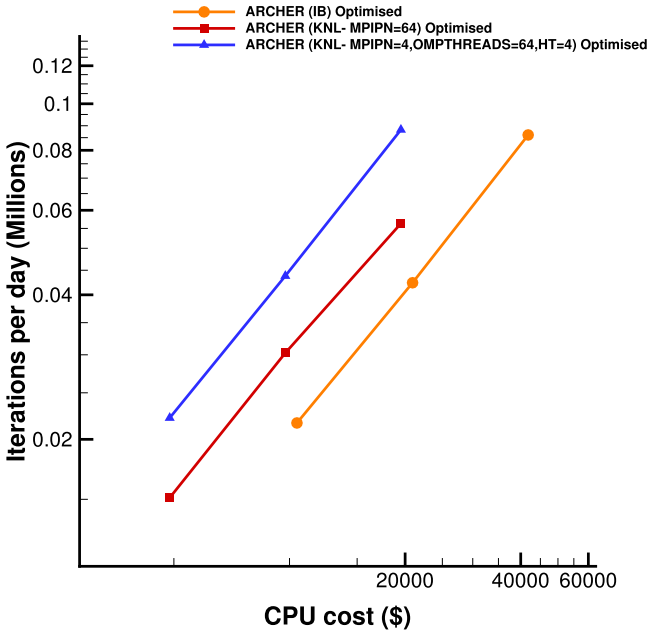


Fig. 10. Strong scalability results for the ILES of the SD7003 wing using the WENO 4th-order scheme.



(a) Iterations vs. Number of Nodes



(b) Iterations vs. Purchasing cost of CPU

Fig. 11. Strong scalability results for the ILES simulation of the Taylor-Green vortex using the WENO 6th-order scheme on the coarsest tetrahedral mesh using the ARCHER(IB) and ARCHER(KNL) platform.

is not experiencing the same performance increase as the WENO 6th-order scheme. This can be attributed to the fact that the polynomial order is smaller and the fact that mixed-element grids introduces additional overheads in terms of element imbalance between partitioned blocks.

We assess the performance and scalability of the optimised solver on the ARCHER Knights Landing-(KNL) platform cluster composed of Xeon Phi processors. The test case is the Taylor-Green vortex flow with the WENO 6th-order on a coarse resolution grid composed of 0.2 million elements. The performance gains of using the **GEMM** library calls in an architecture with a much wider 512-bit vector units, and faster memory bandwidth.

The simulations are performed on the (KNL) from 1 to 8 nodes as seen in Fig. 11. A speed up from 7 to 8.5 is achieved across all CPU counts with the optimised code compared with the original one. The performance increase in this architecture is even more pronounced since the data for the simulation fits within the fast memory of the (KNL) processor. Due to the nature of the unstructured CFD solver, where non-coalescent memory access patterns are present, the increased memory bandwidth in combination with the much wider 512-bit vector units in this architecture results in significant performance improvements. Initially 64 MPI processes per Knights Landing chip were used, but this was found not to be the optimum configuration. With the utilisation of hybrid **MPI+OPENMP** implementation of the code, it was found that

the ideal combination for running the simulations was with 4 MPI processes per Knights Landing chip, and 64 threads spawned from each one of them, resulting a total of 256 threads per chip. By using the hybrid **MPI+OPENMP** a speed up of approximately 1.4 was observed compared to the pure MPI implementation of the revised code. Finally the utilisation of the ARCHER (KNL) platform results in higher computational performance per purchasing cost of the CPUs in each computational node, as compared to the ARCHER (IB) nodes as seen in Fig. 11. This is a good indication on the potential of these architectures for pushing the boundaries of efficiency and cost, for large scale scientific computing applications.

5. Conclusions

In this work the performance of high-order WENO schemes of the UCNS3D CFD solver is assessed under the ILES framework. The code is profiled, optimised and tested across five different HPC clusters. Optimisation of the most compute-hungry processes is performed i.e. WENO weights computation, least-square reconstruction and communication of inter-processor boundary extrapolated variables and gradients. Significant performance benefits are achieved by making use of **BLAS** library functions and replacing matrix-vector operations with matrix-matrix. Blocking MPI communication functions are replaced with non-blocking ones for the exchange of inter-block variables. Across five different HPC platforms the speed-up observed between the revised and the original code, spanned from 1.5 to 8.5. Finally, the use of hybrid MPI+OpenMP capability of the code resulted in an additional performance gains of 1.4 on the ARCHER Knights Landing cluster. The results obtained are encouraging as one of the major challenges of high-order schemes is the inherited cost. Future-proofing high-order solvers by incorporating enabling technologies for new and future hardware architectures will ensure that the high-order movement will remain relevant and better bridge the gap with the CFD workhorse of lower-order methods.

Acknowledgements

The authors would like to acknowledge the computing time at HAZELHEN at the High-Performance Computing Center Stuttgart (HLRS), Germany and SuperMUC at the Leibniz Supercomputing Centre (LRZ) in Garching, Germany in the framework of the PRACE project funded in part by the EU Horizon 2020 research and innovation programme (2014–2020) under grant agreement 653838. The authors would also like to acknowledge the computing on the UK national high-performance computing service ARCHER, that was provided through the UK Turbulence Consortium in the framework of the EPSRC grant EP/L000261/1.

References

[1] Dumbser M, Kaser M, Titarev V, Toro E. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. J Comput Phys 2007;226(1):204–43.
[2] Haselbacher A. A WENO reconstruction algorithm for unstructured grids based on explicit stencil construction. 43rd AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings; 2005. p. 3369–78. doi:102514/62005-879.
[3] Li W, Ren Y. Quadrature-free non-oscillation finite volume scheme for solving Navier-Stokes equations on unstructured grids. AIP Conference Proceedings, 1376; 2011. p. 639–41. doi:101063/13652000.
[4] Nogueira X, Cueto-Felgueroso L, Colominas I, Navarrina F, Casteleiro M. A new shock-capturing technique based on moving least squares for higher-order numerical schemes on unstructured grids. Comput Methods Appl Mech Eng 2010;199(37–40):2544–58.
[5] Ollivier-Gooch C. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. J Comput Phys 1997;133(1):6–17.
[6] Gooch CO, Altena MV. A high-order-accurate unstructured mesh finite-volume scheme for the advection-diffusion equation. J Comput Phys 2002;181(2):729–52.
[7] Tsoutsanis P, Titarev V, Drikakis D. WENO schemes on arbitrary mixed-element unstructured meshes in three space dimensions. J Comput Phys 2011;230(4):1585–601.
[8] Wolf W, Azevedo J. High-order ENO and WENO schemes for unstructured grids. Int J Numer Methods Fluids 2007;55(10):917–43.
[9] Tsoutsanis P, Antoniadis A, Drikakis D. WENO schemes on arbitrary unstructured meshes for laminar, transitional and turbulent flows. J Comput Phys 2014;256:254–76.
[10] Antoniadis A, Tsoutsanis P, Drikakis D. Assessment of high-order finite volume methods on unstructured meshes for RANS solutions of aeronautical configurations. Comput Fluids 2017;146:86–104.
[11] Cockburn B, Shu C-W. Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. J Sci Comput 2001;16(3):173–261.
[12] Dennis J, Nair R, Tufo H, Levy M, Voran T. Towards an efficient and scalable discontinuous Galerkin atmospheric model. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium; 2005. https://doi.org/10.1109/IPDPS.2005.438.
[13] Dumbser M, Balsara D, Toro E, Munz C-D. A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. J Comput Phys 2008;227(18):8209–53.
[14] Uranga A, Persson P-O, Drela M, Peraire J. Implicit large eddy simulation of transition to turbulence at low Reynolds numbers using a discontinuous Galerkin method. Int J Numer Methods Eng 2011;87(1–5):232–61.
[15] Xu Z, Liu Y, Shu C-W. Hierarchical reconstruction for discontinuous Galerkin methods on unstructured grids with a WENO-type linear reconstruction and partial neighboring cells. J Comput Phys 2009;228(6):2194–212.
[16] Zhu J, Qiu J, Shu C-W, Dumbser M. Runge-Kutta discontinuous Galerkin method using WENO limiters ii: Unstructured meshes. J Comput Phys 2008;227(9):4330–53.
[17] Wang Z. Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation. J Comput Phys 2002;178(1):210–51.
[18] Wang Z, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids. II. Extension to two-dimensional scalar equation. J Comput Phys 2002;179(2):665–97.
[19] Wang Z, Zhang L, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids IV: extension to two-dimensional systems. J Comput Phys 2004;194(2):716–41.
[20] Xu Z, Liu Y, Shu C-W. Hierarchical reconstruction for spectral volume method on unstructured grids. J Comput Phys 2009;228(16):5787–802.
[21] Zhou Y, Wang Z. Implicit large eddy simulation of transitional flow over a SD7003 wing using high-order spectral difference method. 40th AIAA Fluid Dynamics Conference and Exhibit; 2010. doi:10.2514/6.2010-4442.
[22] Breviglieri C, Maximiliano A, Basso E, Azevedo J. Improved high-order spectral finite volume method implementation for aerodynamic flows. 27th AIAA Applied Aerodynamics Conference; 2009. doi:10.2514/6.2009-4119.
[23] Vermeire B, Vincent P. On the properties of energy stable flux reconstruction schemes for implicit large eddy simulation. J Comput Phys 2016;327:368–88.
[24] Vermeire B, Vincent P. On the behaviour of fully-discrete flux reconstruction schemes. Comput Methods Appl Mech Eng 2017;315:1053–79.
[25] Witherden F, Farrington A, Vincent P. PyFR: an open source framework for solving advection-diffusion type problems on streaming architectures using the flux reconstruction approach. Comput Phys Commun 2014;185(11):3028–40.
[26] Jameson A, Vincent P, Castonguay P. On the non-linear stability of flux reconstruction schemes. J Sci Comput 2012;50(2):434–45.
[27] Vincent P, Castonguay P, Jameson A. A new class of high-order energy stable flux reconstruction schemes. J Sci Comput 2011;47(1):50–72.
[28] Huynh H. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. In: Collection of technical papers - 18th AIAA computational fluid dynamics conference, 1; 2007. p. 698–739.
[29] Titarev V, Tsoutsanis P, Drikakis D. WENO schemes for mixed-element unstructured meshes. Commun Comput Phys 2010;8(3):585–609.
[30] Tsoutsanis P, Kokkinakis I, Konozsy L, Drikakis D, Williams R, Youngs D. An investigation of the accuracy and efficiency of structured and unstructured, compressible and incompressible methods for the vortex pairing problem. Comput Methods Appl Mech Eng 2015;293:207–31.
[31] Dumbser M, Boscheri W, Semplice M, Russo G. Central WENO schemes for hyperbolic conservation laws on fixed and moving unstructured meshes. SIAM J Sci Comput 2017;39(6):A2564–91. doi:10.1137/17M1111036.
[32] Antoniadis AF, Tsoutsanis P, Drikakis D. Numerical accuracy in RANS computations of high-lift multi-element airfoil and aircraft configurations. 53rd AIAA aerospace sciences meeting, AIAA 2015-0317, Kissimmee, Florida. AIAA, editor; 2015.
[33] Drikakis D, Antoniadis AF, Tsoutsanis P, Kokkinakis I, Rana Z. Azure: an advanced CFD software suite based on high-resolution and high-order methods. 53rd AIAA aerospace sciences meeting ,AIAA 2015-0813, Kissimmee, Florida. AIAA, editor; 2015.
[34] Antoniadis AF, Drikakis D, Kokkinakis I, Tsoutsanis P, Rana Z. High-order methods for hypersonic shock wave turbulent boundary layer interaction flow. 20th AIAA international space planes and hypersonic systems and technologies conference, AIAA 2015-3524, Glasgow, Scotland. AIAA, editor; 2015.
[35] Dumbser M, Käser M, Titarev VA, Toro EF. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. J Comput Phys 2007;226(1):204–43.
[36] Toro EF. Riemann solvers and numerical methods for fluid dynamics - A practical introduction. Berlin: Springer-Verlag; 1997.

[37] Ivan L, Groth C. High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. J Comput Phys 2014;257(PA):830–62. doi:10.1016/j.jcp.2013.09.045.
[38] Gassner G, Lorcher F, Munz C-D. A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. J Comput Phys 2007;224(2):1049–63.
[39] Jalali A, Sharbatdar M, Ollivier-Gooch C. Accuracy analysis of unstructured finite volume discretization schemes for diffusive fluxes. Comput Fluids 2014;101:220–32.
[40] Nishikawa H. Robust and accurate viscous discretisation via upwind scheme-I: basic principle. Comput Fluids 2011;49:62–86.
[41] Gottlieb S. On high order strong stability preserving Runge-Kutta and multi step discretizations. J Sci Comput 2005;25(112):105–28.
[42] Karypis G, Kumar V. Multilevel k-way partitioning scheme for irregular graphs. J Parallel Distrib Comput 1998;48:96–129.
[43] Drikakis D, Fureby C, Grinstein F, Youngs D. Simulation of transition and turbulence decay in the Taylor-Green vortex. J Turbul 2007;8:1–12.
[44] Bull J, Jameson A. Simulation of the Taylor-Green vortex using high-order flux reconstruction schemes. AIAA J 2015;53(9):2750–61.
[45] Dumbser M, Peshkov I, Romenski E, Zanotti O. High order ADER schemes for a unified first order hyperbolic formulation of continuum mechanics: viscous heat-conducting fluids and elastic solids. J Comput Phys 2016;314:824–62.
[46] Chapelier J-B, de la Llave Plata M, Lamballais E. Development of a multiscale LES model in the context of a modal discontinuous Galerkin method. Comput Methods Appl Mech Eng 2016;307:275–99.
[47] Sifounakis A, Lee S, You D. A conservative finite volume method for incompressible Navier-Stokes equations on locally refined nested cartesian grids. J Comput Phys 2016;326:845–61.
[48] Shu C-W, Don W-S, Gottlieb D, Schilling O, Jameson L. Numerical convergence study of nearly incompressible, inviscid Taylor-Green vortex flow. J Sci Comput 2005;24(1):569–95.
[49] Visbal M. High-fidelity simulation of transitional flows past a plunging airfoil. AIAA J 2009;47(11):2685–97.
[50] Uranga A, Persson P-O, Drela M, Peraire J. Implicit large eddy simulation of transition to turbulence at low reynolds numbers using a discontinuous Galerkin method. Int J Numer Methods Eng 2011;87(1–5):232–61.
[51] Rizzetta D, Visbal M. Numerical investigation of plasma-based control for low-Reynolds-number airfoil flows. AIAA J 2011;49(2):411–25.
[52] Garmann D, Visbal M, Orkwis P. Comparative study of implicit and subgrid-scale model large-eddy simulation techniques for low-Reynolds number airfoil applications. Int J Numer Methods Fluids 2013;71(12):1546–65.
[53] Beck A, Bolemann T, Flad D, Frank H, Gassner G, Hindenlang F, et al. High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. Int J Numer Methods Fluids 2014;76(8):522–48. doi:10.1002/fld.3943.
[54] Bassi F, Botti L, Colombo A, Ghidoni A, Massa F. Linearly implicit Rosenbrock-type Runge-Kutta schemes applied to the discontinuous Galerkin solution of compressible and incompressible unsteady flows. Comput Fluids 2015;118:305–20.



https://dspace.org/

Improvement of the computational performance of a parallel unstructured WENO finite volume CFD code for Implicit Large Eddy Simulation

Tsoutsanis, Panagiotis; Antoniadis, Antonis F.; Jenkins, Karl W.

2018-03-05
Elsevier
https://doi.org/10.1016/j.compfluid.2018.03.012
Tsoutsanis P, Antoniadis AF, Jenkins KW. (2018) Improvement of the computational performance of a parallel unstructured WENO finite volume CFD code for Implicit Large Eddy Simulation. Computers and Fluids, Volume 173, September 2018, pp. 157-170

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