Journal of Computational Physics 363 (2018) 399–427

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An efficient Adaptive Mesh Refinement (AMR) algorithm for the Discontinuous Galerkin method: Applications for the computation of compressible two-phase flows

Andreas Papoutsakis [a,*], Sergei S. Sazhin [a], Steven Begg [a], Ionut Danaila [b], Francky Luddens [b]

[a] Advanced Engineering Centre, School of Computing, Engineering and Mathematics, University of Brighton, Brighton, BN24GJ, UK
[b] Laboratoire de Mathématiques Raphaël Salem, Université de Rouen Normandie, F-76801, Saint-Étienne-du-Rouvray, France

ARTICLE INFO

Article history:
Received 27 July 2017
Received in revised form 15 February 2018
Accepted 23 February 2018
Available online 6 March 2018

Keywords:
Droplets
Sprays
Vortex rings
Discontinuous Galerkin
Adaptive Mesh Refinement

ABSTRACT

We present an Adaptive Mesh Refinement (AMR) method suitable for hybrid unstructured meshes that allows for local refinement and de-refinement of the computational grid during the evolution of the flow. The adaptive implementation of the Discontinuous Galerkin (DG) method introduced in this work (ForestDG) is based on a topological representation of the computational mesh by a hierarchical structure consisting of oct-quad- and binary trees. Adaptive mesh refinement (h-refinement) enables us to increase the spatial resolution of the computational mesh in the vicinity of the points of interest such as interfaces, geometrical features, or flow discontinuities. The local increase in the expansion order (p-refinement) at areas of high strain rates or vorticity magnitude results in an increase of the order of accuracy in the region of shear layers and vortices.

A graph of unitarian-trees, representing hexahedral, prismatic and tetrahedral elements is used for the representation of the initial domain. The ancestral elements of the mesh can be split into self-similar elements allowing each tree to grow branches to an arbitrary level of refinement. The connectivity of the elements, their genealogy and their partitioning are described by linked lists of pointers. An explicit calculation of these relations, presented in this paper, facilitates the on-the-fly splitting, merging and repartitioning of the computational mesh by rearranging the links of each node of the tree with a minimal computational overhead. The modal basis used in the DG implementation facilitates the mapping of the fluxes across the non conformal faces.

The AMR methodology is presented and assessed using a series of inviscid and viscous test cases. Also, the AMR methodology is used for the modelling of the interaction between droplets and the carrier phase in a two-phase flow. This approach is applied to the analysis of a spray injected into a chamber of quiescent air, using the Eulerian–Lagrangian approach. This enables us to refine the computational mesh in the vicinity of the droplet parcels and accurately resolve the coupling between the two phases.

* Corresponding author.
E-mail addresses: a.papoutsakis@brighton.ac.uk (A. Papoutsakis), s.sazhin@brighton.ac.uk (S.S. Sazhin).

https://doi.org/10.1016/j.jcp.2018.02.048
0021-9991/

1. Introduction

The variety of different spatial scales observed in compressible, dispersed, multiphase flows reflects the more general problem of the interaction between the micro- and the macro-scales in fluid mechanics [1]. There is a significant number of examples where the physical flow problem is the result of a closely-binded synergy of phenomena occurring at different scales [2–4]. In compressible, dispersed, multiphase flows we observe the interaction of the macroscopic flow in the following ways:

i) with refined structures due to compressibility (e.g. shock and acoustic waves [49,5], thermal effects and chemical reaction regions [6], [7], [8]),
ii) with vortical structures [9], and
iii) with dispersed micro-scale droplets and particles [10], moving walls and detailed solid structures [11] or solidification interfaces[12].

Due to the complexity of these interactions, we need to resolve all scales of the problem. Small scale features in a complex flow are not known *a-priori* and can vary in time. To fully resolve a complex flow, fine resolution is required throughout the computational domain. This fine resolution allows us to describe fine structures and features of the flow.

Adaptive Mesh Refinement (AMR) addresses the problem of resolving this wide range of scales by focusing the discretisation resolution on the proximity of the fine structures. The Finite Element (FEM) framework [13] for the solution of PDE's gives us two options to enhance the spatial resolution (and accuracy) of discretisation. Firstly, this can be achieved by increasing the number of basis functions used for the discretisation of the field variables, resulting in an increase of the degrees of freedom within the element and the order of accuracy of the discretisation (i.e. *p*-type refinement) [14,9]. Secondly, this can be achieved by local increase of mesh resolution (i.e. *h*-type refinement), a strategy which is also relevant to the Finite Volume (FV) [15,16] and Finite Differences [49] framework. Also, a combination of both approaches (i.e. *h*/*p*-type refinement) can also be applied [17,18].

By localising the discretisation resolution, the desirable resolution can be achieved with minimal increase of the size of the problem. *H*-type refinement results in the re-arrangement of the computational grid. This can be achieved either by relocating the nodes of the mesh [6,19–21] or by splitting existing elements in smaller ones [22,17,23,24]. The relocation of the mesh vertices allows us to retain the connectivity topology but eventually leads to highly anisotropic elements [21]. This approach allows us to focus on areas of interest without increasing the total number of degrees of freedom. An alternative approach is to start from the finest resolution and locally coarsen the mesh and/or decrease the original order of accuracy [18]. This is achieved by agglomeration based, adaptive implementations. One of the advantages of this approach is that the finest grid is accurately prescribed. Thus, it can deal with highly anisotropic meshes appearing at the highest refinement level in high Reynolds viscous simulations [25]. Furthermore, agglomeration techniques, being inherently related to multigrid methods, can lead to major convergence performance benefits that stem from multigrid solvers [26,27].

Splitting of cells may be achieved by retaining the existing faces of the mesh [23,28] and introducing vertices inside the elements, or by introducing new vertices leading to non-conformalities [29,17]. The second approach to cell splitting results in an increase in the total number of elements and the re-arrangement of the inter-element connectivity relationships. This method of cell-splitting in AMR requires a versatile data structure. According to [30], approaches to cell splitting in AMR can be classified into three distinct categories: block structured AMR (SAMR), unstructured AMR (UAMR) and tree AMR (TAMR). SAMR utilises the regular mapping of the structured grids at the expense of numerical complexity. UAMR, on the other hand, offers the flexibility of unstructured meshes through the use of an adjacency graph. The TAMR approach, introduced in [31] and [32,30,12] for the *4est* code, uses a forest of trees structure for the description of the locally refined mesh and thus creates an internal mapping [30] for the derivation of the connectivity relations of the refined grid. In the category of unstructured split cell FEM solvers, the *deal-II FEM* library offers an object oriented data structure [33] for AMR on hexahedral elements, and has been widely used for the solution of the Navier–Stokes equations [34]. The *Libmesh* [35] FEM library also provides a versatile data structure for the implementation of adaptive oct-tree refinement on triangular and prismatic elements for numerical simulations in hybrid unstructured meshes. We use a forest of trees structure to describe the topology of a split-cell refined hybrid unstructured mesh. The connectivity graph relations of the refined grid are calculated based on the varying relative relations of the ancestral elements and the predefined internal structured mapping in the tree. The versatility of pre-defined tree split structures (e.g. oct- quad- and binary trees) has made them applicable to a wide range of problems [4].

The Discontinuous Galerkin method [36–39] combines high order accuracy with the ability to handle complex geometries described by hybrid unstructured meshes. The computational efficiency of this method (alongside the spectral volume [38, 40] and spectral difference [38,41,42] methods), however, is generally believed to be inferior to more commonly used methods, such as the Finite Differences (FD) and Finite Volume (FV) [43,44] methods.

The cell splitting AMR strategy results in irregular meshes with hanging nodes. Also *p*-type refinement leads to polynomial approximations of different orders across the elements [36]. DG, however, is perfectly adapted to handle irregular meshes with hanging nodes stemming out of AMR on structured grids [45–47,17,11]. Furthermore, the basis expansion of the solution within the DG element provides an explicit description of the field, resulting in gradient preserving properties during splitting and merging operations. The latter do not interfere with neighbouring cells. The DG method provides a

compact, discretisation stencil, where the inviscid and viscous fluxes are calculated from the solution within the element and only the surface integrals, at the adjacent neighbouring faces, are taken into account. Oct-tree based refinement has been associated with the generation of spurious shocks, in Finite Difference AMR implementations [4,48,49]. However, the compact, discretisation stencil used in the DG method, as well as the treatment of the non-conformal faces as faces of a polyhedral element, results in the elimination of spurious shocks and oscillations [50,51]. Moreover, accurate integration of the inviscid fluxes in non-conformal faces is known to remove the occurrence of spurious shocks in compressible simulations using Finite Volume [52] and Finite Difference [53] approaches. The compatibility of the DG discretisation with adapted unstructured meshes and the improvement of the computational efficiency of the DG method due to AMR leads to a powerful combination for solving complex flows.

The identification of the fine structures which drives AMR is primarily based on the characteristics of the resolved flow field. The gradients of the flow field, quantified by the magnitude of the vorticity or the strain tensor, show the regions of potential flow instabilities [54,9]. Also, the local density variation [55], and the density gradients [7] can serve as criteria to identify shocks. The geometrical features of the flow field, e.g. the solid boundaries [12,11] or the location of inertia particles [15,10] can also be used as AMR criteria. Refinement criteria based on the estimation of expected spatial errors have been suggested in [56,57]. These are used to drive AMR adaptation for unsteady problems. *A posteriori* error estimate techniques [50,51] are based on the identification of the spatial error distribution. This approach results in a refinement strategy that optimises the efficiency of the solution and accuracy of the refined discretisation [58]. Validation of simple gradient based criteria, as reasonable choices for the indication of regions to be refined, can be based on the analyses, using error estimate techniques [59].

In this paper a new mesh adaptive method associated with the Discontinuous Galerkin methodology is suggested. This approach allows on the fly local *h*/*p*-type refinement and de-refinement of the computational mesh [7]. An efficient algorithm for the implementation of *h*/*p*-type refinement is described and it is applied to the design of a new computational code (ForestDG). ForestDG is based on modal, hierarchical basis, as described in [60,61] for the case of conforming hybrid meshes. The use of modal basis is best suited for *h*/*p*-type refinement [62,63]. In contrast to nodal basis, modal basis are evaluated in the computational space, thus allowing a simpler evaluation of the volume integrals and fluxes across non-conforming element faces. We show that a hierarchical representation of a forest of binary, quad- and oct-trees is highly efficient and we introduce a unified methodology for the efficient splitting, merging and repartitioning of hexahedral, prismatic and tetrahedral elements. Furthermore, the accuracy and performance of the new code are assessed. Finally, the ability to capture discontinuities, moving vortical structures and the dispersed phase in multiphase flows, is demonstrated for several examples.

In Section 2 we present the general formulation of the governing equations for the modelling of mass, momentum and energy conservation of a compressible flow. In Section 3 the discretisation of these equations in the DG framework is described. In Section 4 we introduce the forest structure for the representation of an adaptive unstructured mesh. In Section 5 we describe the splitting and merging techniques occurring during the adaptation of the computational grid to the flow field solution. In Section 6 we describe the algorithm introduced for the efficient assignment of the connectivity problem during the adaptation of the grid. In Section 7 we present a series of test cases for subsonic and supersonic configurations where we assess the accuracy of the algorithm, the efficiency of the proposed method and the computational performance of the code. In Section 8 we demonstrate the capabilities of the code to capture moving structures for the problem of an inviscid supersonic flow around a cylinder in a duct. In Section 9 we demonstrate the resolution of the flow discontinuities and reaction regions arising from the interaction of the oblique shocks in the case of a hypersonic flow around a double cone configuration. In Section 10 we use the solver introduced in this paper for the solution of a dispersed multiphase flow arising during spray injection of gasoline fuel.

2. Governing equations

The method presented here is developed for the general case of viscous compressible flows. The basic set of governing equations correspond to the flow field of a compressible viscous fluid described by the state vector $\mathbf{U}(\mathbf{x}, t)$ in Eulerian coordinates, which contains the values of density ρ, momentum $\rho\mathbf{u}$ and energy ρe at each position of the computational domain $\mathbf{x}$ at time t. Depending on the case modelled, the state vector $\mathbf{U}(\mathbf{x}, t)$ can be extended to include the vibrational energy e_v needed for the Park's model [64] for air dissociation simulations, the species specific densities ρY_s and finally the equation for the turbulent viscosity ν_t in the cases of turbulent simulations. For the compressible turbulent dispersed two phase flow simulations, the carrier fluid phase is modelled as an Eulerian flow field $\mathbf{U}(\mathbf{x}, t)$ and the droplets are suspended in the carrier gas phase and are modelled using the Lagrangian approach.

Sub-grid turbulent fluctuations are modelled using the one-equation Spalart–Allmaras (SA) model [65], leading to a Detached Eddy Simulation (DES) [66,67] approach. The SA model offers an alternative to the standard LES models, that is not dependent on the filter width Δ. For the AMR methodology presented in our paper, the cell size changes substantially in space thus making the standard LES models inapplicable. Furthermore, in the spray injection case investigated in our paper, the Spalart–Allmaras model accounts for the effect of the cylinder head wall where a strong recirculation region and detachment of the boundary layer is observed [68].

The Favre averaging operator $\widetilde{(\cdot)} = \overline{\rho(\cdot)}/\overline{\rho}$ is used for the separation of the small turbulent fluctuations from the large ones. The state vector for the Favre averaged velocity $\mathbf{u}$ and specific energy e is defined as $\widetilde{\mathbf{U}}(\mathbf{x}, t) = (\overline{\rho}, \overline{\rho}\widetilde{u}_1, \overline{\rho}\widetilde{u}_2, \overline{\rho}\widetilde{u}_3, \overline{\rho}\widetilde{e})$.

The conservation of mass, momentum and energy provides the set of the governing equations for the turbulent compressible flow of the carrier phase. The strong conservative form for $\widetilde{\mathbf{U}}$ can be presented as [69]:

$$\frac{\partial \widetilde{\mathbf{U}}}{dt} + \nabla \cdot \mathbf{f}_{inv}\left(\widetilde{\mathbf{U}}\right) - \frac{1}{\mathrm{Re}} \nabla \cdot \mathbf{f}_{vis}\left(\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Theta}}\right) = \mathbf{w}_d\left(\widetilde{\mathbf{U}}\right) , \tag{1}$$

$$\widetilde{\mathbf{\Theta}} = \nabla \mathbf{f}_{aux}\left(\widetilde{\mathbf{U}}\right) , \tag{2}$$

where $\mathbf{w}_d$ is the vector of the source terms stemming from the two way coupling for the momentum and energy transfer between the carrier and the discrete phase, $\mathbf{f}_{inv}$ is the 5×3 tensor of the inviscid fluxes and $\mathbf{f}_{vis}$ is the 5×3 tensor for the viscous fluxes, defined as:

$$\mathbf{f}_{inv} = \begin{bmatrix} \overline{\rho}\widetilde{u}_j \\ \overline{\rho}\widetilde{u}_i\widetilde{u}_j + \overline{p}\delta_{i,j} \\ \left(\overline{\rho}\widetilde{e} + \overline{p}\right)\widetilde{u}_j \end{bmatrix}, \ \mathbf{f}_{vis} = \begin{bmatrix} 0 \\ 2\left(\mu + \mu_t\right) S^*_{i,j} \\ 2\left(\mu + \mu_t\right)\widetilde{u}_i S^*_{i,j} - \widetilde{q} \end{bmatrix}, \ \mathbf{w}_d = - \begin{bmatrix} 0 \\ n_d f_{d_i} \\ n_d f_{d_j}\widetilde{u}_j \end{bmatrix}, \ \mathbf{f}_{aux} = \begin{bmatrix} \widetilde{u}_i \\ \widetilde{e} \end{bmatrix} , \tag{3}$$

$S^*_{ij} = \frac{1}{2}\left(\frac{\partial \widetilde{u}_j}{\partial x_i} + \frac{\partial \widetilde{u}_i}{\partial x_j}\right) - \frac{1}{3}\delta_{ij}\frac{\partial \widetilde{u}_k}{\partial x_k}$ is the traceless rate of strain tensor related to the viscous stress tensor $\tau_{ij} = 2\mu S^*_{ij}$, the non-dimensional local viscosity μ is normalised by the dynamic viscosity at the reference temperature. Flux $\widetilde{q}$ is the summation of the translational, rotational and vibrational heat fluxes which are assumed to be in equilibrium. In this case the heat flux is related to the temperature gradient $\widetilde{q} = -\kappa \nabla \cdot T$ where $\kappa = Pr(\mu + \mu_t)$ is the thermal conductivity of the fluid. For low speed supersonic and subsonic simulations the value of the Prandtl number for the air is taken equal to 0.76.

The auxiliary state vector $\widetilde{\mathbf{\Theta}}$ contains the spatial gradients of the 4×1 auxiliary flux $\mathbf{f}_{aux}$ for the diffusive components of the state vector $\widetilde{\mathbf{U}}$ as defined in Equation (2). The viscous fluxes $\mathbf{f}_{vis}$ are evaluated from $\widetilde{\mathbf{\Theta}}$. The auxiliary variables vector $\widetilde{\mathbf{\Theta}}$ is discretised separately resulting in three equations for each of the four diffusive components of the state vector. As a result, Equation (2) is solved at the same accuracy with the state variables. Equations (1) and (2) comprise the coupled formulation of the governing equations for $\widetilde{\mathbf{X}} = \left[\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Theta}}\right]$ for a turbulent compressible two phase flow.

The contribution of sub-grid turbulence scales, not accounted by the spatially filtered state vector, is taken into account by the turbulent dimensionless viscosity term μ_t in the definition of the viscous fluxes in Equation (3). In the Spalart–Allmaras model μ_t is estimated by the variable $\hat{\nu}_t$ as:

$$\mu_t = \overline{\rho}\hat{\nu}_t f_{u1} , \tag{4}$$

where $\hat{\nu}_t$ is calculated by the integration of the equation for the turbulent viscosity ν_t used in the Spalart–Allmaras model:

$$\frac{\partial \overline{\rho}\hat{\nu}_t}{\partial t} + \frac{\partial \overline{\rho}\widetilde{u}_j\hat{\nu}_t}{\partial x_j} = \overline{\rho}c_{b1}\left(1 - f_{t2}\right)\hat{S}\hat{\nu}_t - \overline{\rho}\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\hat{\nu}_t}{d}\right)^2 + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\rho c_{b2}\frac{\partial \hat{\nu}_t}{\partial x_i}\frac{\partial \hat{\nu}_t}{\partial x_i}\right] - \frac{1}{\sigma}\frac{\partial \overline{\rho}}{\partial x_i}\frac{\partial \hat{\nu}_t}{\partial x_i} . \tag{5}$$

The constants and parameters f_{u1}, c_{b1}, f_{t2}, c_{w1}, f_w, c_{b1}, κ, f_{t2}, σ and c_{b2} are provided by the model described in [65] and its modified version for the areas of negative viscosity as described in [70]. Parameter $\hat{S}$ is related to the rate of strain tensor $|\widetilde{S_{ij}}| = \sqrt{2\widetilde{S}_{ij}\widetilde{S}_{ij}}$ and the vorticity magnitude of the resolved field, acting as a source term for turbulent kinetic energy of the flow. d is the distance from the walls of the computational domain.

The transport equations of each species s is expressed in terms of the conservation of density $\rho_s = \rho Y_s$ of each species as:

$$\frac{\partial \rho Y_s}{\partial t} + \nabla \cdot \left(\rho Y_s \mathbf{u}\right) = -\nabla \cdot \left(\rho_s \mathbf{v}^d_s\right) + w_s , \tag{6}$$

where Y_s is the mixture fraction of a species s, $\mathbf{v}^d_s = (u^d_s, v^d_s, w^d_s)$ is the corresponding diffusion velocity, and w_s is the chemical source term [7]. The rates of forward and backward reactions are obtained using the Arrhenius approximation, and the vibration-dissociation model developed by Park [64].

The energy equation for a flow with N_s reactive species is modified taking into accounting the total energy e_T of the mixture

$$\frac{\partial \rho \widetilde{e}_T}{\partial t} + \nabla \cdot \left(\left(\rho \widetilde{e}_T + p\right)\mathbf{u}\right) = \nabla \cdot \left(\tau \cdot \mathbf{u}\right) - \nabla \cdot \left(\mathbf{q}^t + \mathbf{q}^r + \mathbf{q}^v\right) - \nabla \cdot \sum_{s=1}^{s=n_s} \rho_s h_s \mathbf{v}^d_s . \tag{7}$$

Fluxes $\mathbf{q}^t, \mathbf{q}^r, \mathbf{q}^v$ are the translational, rotational, and vibrational heat flux vectors, respectively; h_s is the total specific enthalpy of the species s. Translational and rotational energies refer to the translational and rotational motions of the molecules, whereas vibrational energy refers to the energy of vibrations of the chemical bonds in polyatomic molecules. In the current work, a simplified form of the energy equation is used. In this equation, the vibrational energy of diatomic species is described by a single vibrational temperature. This allows us to solve the equation for the total vibrational energy

of the mixture instead of separate equations for each of the polyatomic species. This equation for the vibrational energy per unit volume is presented as:

$$\frac{\partial \rho \widetilde{e}_v}{\partial t} + \nabla \cdot (\rho \widetilde{e}_v \mathbf{u}) = -\nabla \cdot \mathbf{q}^v - \nabla \cdot \sum_{s=1}^{s=N_d} \rho_s e_s^v \mathbf{v}_s^d \,, \tag{8}$$

where e_s^v is the vibrational energy per unit mass of species s, w_u is the vibrational energy source term, N_d is the number of diatomic species in the mixture.

The relation between vibrational energy and vibrational temperature T^v is inferred from the following expression:

$$e^v = \sum_{s=1}^{N_d} e_s^v = \sum_{s=1}^{N_d} \frac{R}{M_s} \frac{\theta_s^v}{e^{\frac{\theta_s^v}{T^v}} - 1} \,, \tag{9}$$

where θ_s^v is the characteristic temperature of vibration for each diatomic species, M_s is the molecular weight, and R is the ideal gas molar constant.

Although the above equation provides an explicit definition of the vibrational energy it is solved with an iterative method for the calculation of the vibrational temperature from a given vibrational energy. Equation (1) has been non-dimensionalised over the characteristic length of the flow, gas dynamic viscosity μ_g and gas density ρ_g at ambient conditions. The Reynolds number of the flow is estimated as $\mathrm{Re} = \rho_g c L/\mu_g$, where c is the velocity of sound at ambient conditions.

The discrete phase is modelled as parcels of droplets with diameters d_d, velocities $\mathbf{v_d}$ and density ρ_d. The effect of the dispersed phase on the energy and momentum of the carrier phase is modelled as the source term $\mathbf{w}_d$ in Equation (3). n_d is the droplet number density. The term f_{d_i} in Equation (3) is the force acting on each individual droplet in the parcel. Assuming a steady Stokes flow, the expression for the drag force can be presented as:

$$n_d \mathbf{f}_d = \frac{3 n_d c_D \pi d_d \mu}{Re} \left(\widetilde{\mathbf{u}} - \mathbf{v}_d\right) \,, \tag{10}$$

where, $c_D = \left(1 + 0.16667 Re_d^{2/3}\right)$ is the correction for the Stokes force for large droplet Reynolds numbers ($\mathrm{Re}_d > 1$). The trajectories of droplets are described by the following equations:

$$\frac{d\mathbf{x}_d}{dt} = \mathbf{v}_d, \quad d\mathbf{v}_d = \mathbf{f}_d dt \,. \tag{11}$$

3. Discretisation of the equations

We consider the discretisation of the computational domain Ω into N elements E_m ($\Omega = \cup E_m$). A weak formulation of the governing equations is derived by multiplying the conservative form of these equations with a test function $w(\mathbf{x})$ and integrating them over the element. In the Galerkin context, the test function is taken from the same set of polynomial basis functions as used for the interpolation of the state vector $\widetilde{\mathbf{U}}$ and the extended state vector $\widetilde{\mathbf{X}} = \left[\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Theta}}\right]$. The interpolated distribution $\mathbf{X}_h^m$ for $\widetilde{\mathbf{X}}$ is defined for each element E_m as the weighted sum of N_p polynomial basis functions:

$$\mathbf{X}_h^m = \sum_{i=1}^{N_p} \mathbf{c}_i^m(t) b_i(\mathbf{x}) \,, \text{for } m = 1, N_p \,, \tag{12}$$

where p is the maximum degree of the basis functions. A similar expansion is assumed for $\mathbf{U}_h^m$ and $\mathbf{\Theta}_h^m$. In this expansion, the solution coefficients $\mathbf{c}_i(t)$ are the degrees of freedom. $b_i(\mathbf{x})$ is the tensor product of the Legendre polynomial basis functions in the three spatial dimensions. The integral formulation of Equations (1) is expressed as:

$$\int_{E_m} b_i \frac{\partial \mathbf{U}_h^m}{\partial t} dE + \oint_{S_m} b_i \mathbf{f}\left(\mathbf{X}_h^m\right) \cdot \mathbf{n} dS - \int_{E_m} \nabla b_i \cdot \mathbf{f}\left(\mathbf{X}_h^m\right) dE = \int_{E_m} b_i \mathbf{w}_d^m dE \quad \text{for} \quad i = 1, N_p, \; m = 1, N \,, \tag{13}$$

$$\int_{E_m} b_i \mathbf{\Theta}_h^m dE = \oint_{S_m} b_i \mathbf{f}_{aux}\left(\mathbf{U}_h^m\right) \cdot \mathbf{n} dS - \int_{E_m} \nabla b_i \cdot \mathbf{f}_{aux}\left(\mathbf{U}_h^m\right) dE \quad \text{for} \quad i = 1, N_p, \; m = 1, N \,, \tag{14}$$

providing a set of $N_p \times N$ equations for $\mathbf{c}_i^m(t)$.

The surface integrals are defined on the surface of the element $S_m = \partial E_m$ with $\mathbf{n}$ defined as the outward normal unit vector. In the weak formulation presented in Equation (13), the flux $\mathbf{f}$ represents both the viscous and inviscid fluxes in Equation (1) as $\mathbf{f} = \mathbf{f}_{inv} - (1/\mathrm{Re})\mathbf{f}_{vis}$; the flux $\mathbf{f}_{aux}$ was defined in Equation (3). In the DG context we do not require the continuity of the interpolated variables across the element faces. Thus, the values of $\mathbf{U}_h^m$ and $\mathbf{\Theta}_h^m$ on the faces S_m are

defined twice. The conservation of fluxes at the boundaries of the elements infers from the approach to the evaluation of the surface integrals used in our analysis. In the Local Lax–Friedrichs (LLF) scheme for the evaluation of viscous fluxes, the signs of surface sides of the elements (referred to as minus (−) or plus (+)) are introduced.

Calculating the fluxes from one of the two sides for the adjoin elements guarantees the conservation properties of the scheme. Specifically, for the LLF scheme the surface integrals for $\mathbf{f}_{vis}$ and $\mathbf{f}_{aux}$ in Equation (1) are evaluated from the opposite sides as:

$$\mathbf{f}_{vis} = \mathbf{f}_{vis}\left(\mathbf{U}^{-m}_{h}, \mathbf{\Theta}^{-m}_{h}\right), \quad \mathbf{f}_{aux} = \mathbf{f}_{aux}\left(\mathbf{U}^{+m}_{h}\right) . \tag{15}$$

The inviscid flux $\mathbf{f}_{inv}$ is evaluated from the mean value of the variables on the two face sides $\left\{\mathbf{U}^{m}_{h}\right\}$, where an artificial diffusion term, proportional to the jump of the fluxes on the bounding surface $[[\mathbf{U}^{m}_{h}]]$, is introduced via the equation:

$$\mathbf{f}_{inv} = \mathbf{f}_{inv}\left(\left\{\mathbf{U}^{m}_{h}\right\}\right) + \frac{\lambda_i}{2}[[\mathbf{U}^{m}_{h}]] , \tag{16}$$

where, in the LLF scheme, $\lambda_i = \max\left(|f'_{inv}(U_i)|\right)$ is the maximum absolute eigenvalue of the inviscid flux at the specific position of the interface (Local).

The volume and surface integrals in Equations (13) and (14) are defined in the physical space. The integrals are evaluated in the transformed domain for the computational space elements using the Gauss–Legendre quadrature rule and the Jacobian of the transformation [13]. Using the transformations described by Equations (A.1) to (A.3) the physical coordinates $(x)^{E_m}$ for each element of the discretisation E_m are transformed to a computational space Ω_m with coordinates η_1, η_2 and η_3. This transformation maps any hexahedral prismatic or tetrahedral element E_m to a cube with $\eta_i \in [-1, 1]$ using collapsed coordinates [13]. Introducing the Jacobian of the transformation to the computational space, the volume integrals of a field, expanded as in (12), is evaluated as:

$$\int_{E_m} b_i(\mathbf{x}) \sum_{j=1}^{N_p} \mathbf{c}^{m}_{j}(t) b_j(\mathbf{x}) dE = \int_{\Omega_m} b_i(\mathbf{x}) \sum_{j=1}^{N_p} \mathbf{c}^{m}_{j}(t) b_j(\vec{\eta})\, J(\vec{\eta}) d\Omega . \tag{17}$$

The above integral is now defined in the standard computational space and is calculated by evaluating the integral function on N_{qp} selected quadrature points of Ω_m: $\vec{\eta}_{i,j,k} = \vec{\eta}_{iq}$. Finally, the integral on the right hand side of (17) is approximated by the Gauss–Legendre quadrature rule with weights $W(\vec{\eta})$ pre-calculated at $\vec{\eta}_{iq}$ as:

$$\int_{\Omega_m} b_i(\mathbf{x}) \sum_{j=1}^{N_p} \mathbf{c}^{m}_{j}(t) b_j(\vec{\eta})\, J(\vec{\eta}) d\Omega = \sum_{iq=0}^{N_{eq}} \sum_{j=1}^{N_p} b_i(\vec{\eta}_{iq}) \mathbf{c}^{m}_{j}(t) b_j(\vec{\eta})\, J(\vec{\eta}) W(\vec{\eta}) . \tag{18}$$

Similarly, surface integrals in Equations (13), (14) are calculated on quadrature points $\vec{\eta}_{iq,f}$ defined at each active face of the computational space element Ω. Although the AMR methodology introduced here presents shock capturing characteristics by enhancing the resolution of discontinuities arising in compressible flows, a unified limiting approach is needed to detect and smooth sharp gradients. The unified Total Variation Bounded (TVB) limiter proposed by Shu and Cockburn [71] for capturing discontinuities with high order (p1 and higher) DG discretisation, is used in our analysis. Although the AMR presents shock capturing characteristics by enhancing the resolution of discontinuities arising in compressible flows, a unified limiting approach is needed to detect and smooth sharp gradients. The unified TVB limiting approach [71], described in [60,61] for $3D$ solutions and its extension [7] to the set of equations provided by the Park's model, is applied in the space of characteristics, as described in [72]. In the AMR methodology introduced here, the detection of discontinuities accounts for the distribution of the solution of non-conforming adjoint faces.

4. Mesh representation

Unstructured, conforming grids are represented using a finite serial addressing of the cells with each cell determined by the addresses of its vertices. The connectivity of the cells is defined by appointing the numerical addresses of the neighbouring cells to each face. In our case, the numerical list of the cells is substituted by a hierarchical graph representation of the elements.

The computational domain is discretised into N ancestral elements E_m constituting the initial coarse unstructured grid. Any ancestral element can be split into a number of kids of the same type. Any of the resulting siblings can be further split to more kids up to an arbitrary level L. A kid of a parent P obtains a unique address A and is assigned an index $i^A_l = i$ that identifies it as the ith element of its parent tree P at level l. The address of A is related to the address of the parent P as $A = \{P, i^A_l\}$. Expanding the genealogy of the parent element P, the kid A is defined as $A = \{E_m, i^A_1, i^A_2, \ldots, i^A_l\}$. A parent shares the same indices with its kids $i^A_0, i^A_1, \ldots, i^A_{l-1}$ up to the level $l-1$. The index of any tree at the zero level is the index of the initial ancestral mesh $i^A_0 = m$. Thus, each element obtains a unique address that is traced back to the ancestral element. The topology created by the hierarchical relations of the resulting elements forms a forest of nodes.

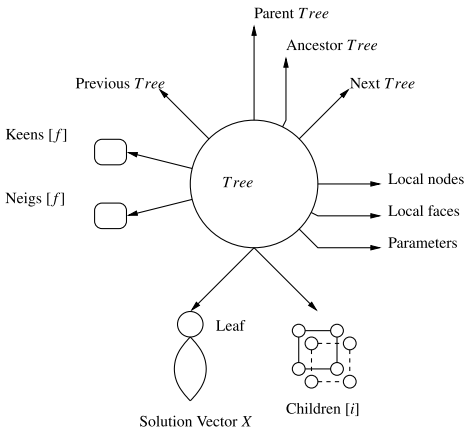

Fig. 1. The tree node data structure.

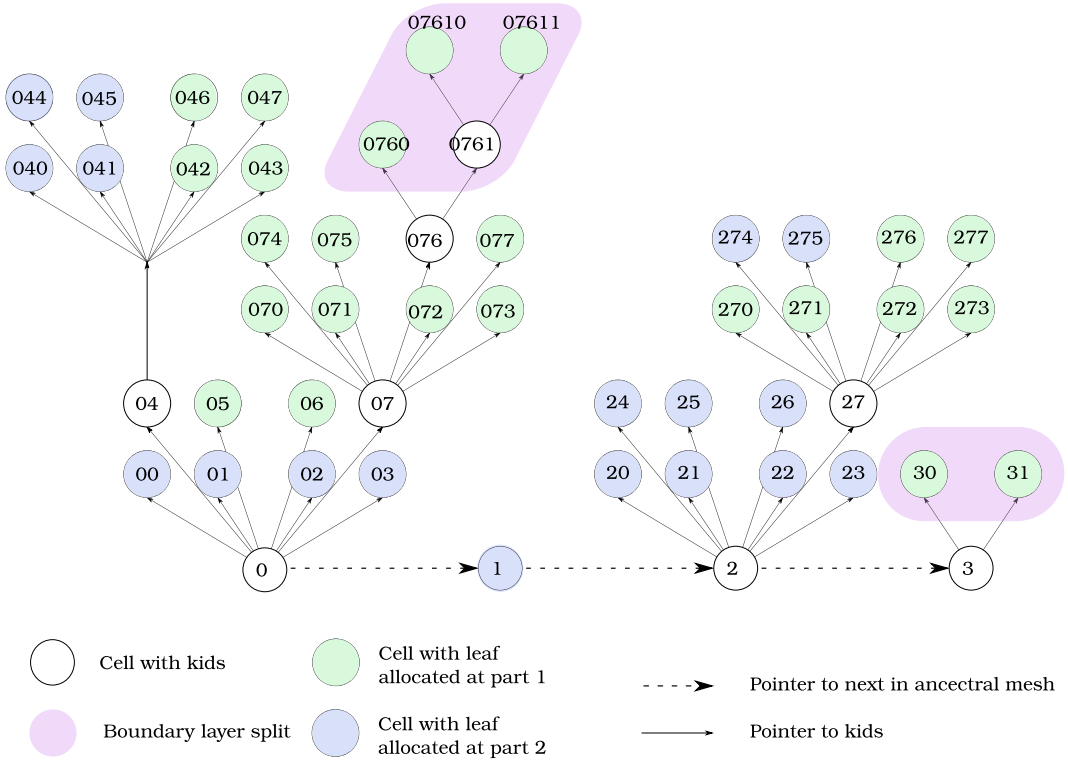

Fig. 2. An example of a forest of oct-trees representing a three dimensional adapted topology.

4.1. Forest and tree structures

The address A of each element is a pointer to the tree node data structure shown in Fig. 1. The pointers A for all the trees of the domain are stored in the forest data structure. An example of a developed forest is shown in Fig. 2. Each node, referred to as tree node, contains all the pointers needed to define the computational element. The most important pointers contained in the tree node structure are the pointers to the parent node P, to the lowest level ancestor E_m, to the previous tree node, to the next tree node and to the neighbours for each face of the element. Also, the tree node structure contains the arrays of the indices of the neighbouring faces and their relative angles. Finally, the pointers to the mesh vertices and the mesh faces, akin to the specific element, are also stored in the tree node structure.

Any node is accessed by a dynamic linked list. An arbitrary node is assigned as the “first” node of the mesh, shown in Fig. 1. By assigning a “next” node for every cell we can go through all cells of the grid. This is performed by advancing to the “next” node each time starting from the “first” one as shown in Fig. 3 (**Left**). The tree node in our implementation is a data structure that contains all necessary information needed for the definition of the relative relations of the cell

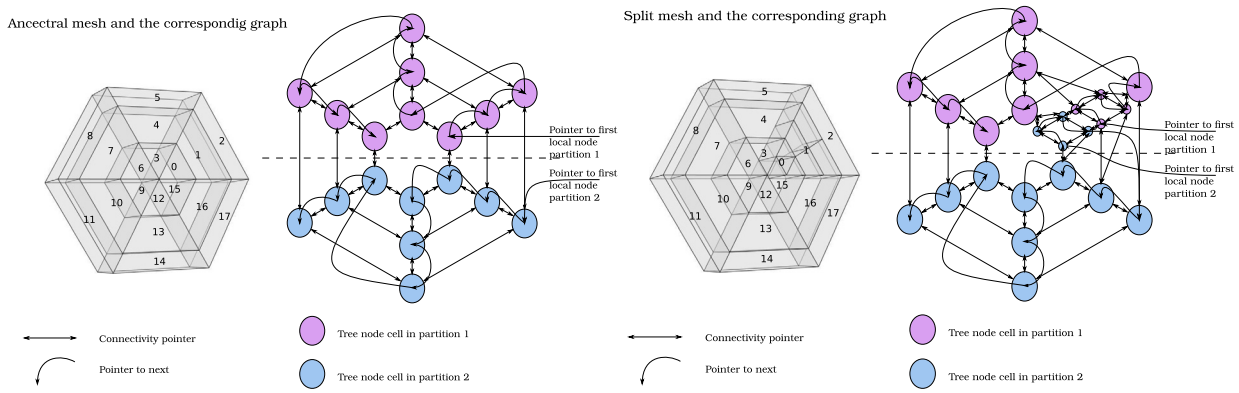

Fig. 3. Left: An example of hybrid unstructured mesh and the corresponding graph. **Right:** Prismatic and hexahedral cells are split into four cells; the nodes of the adapted mesh are repartitioned by introducing the new nodes to the local element lists and the connectivity pointers are re-defined.

and also its geometrical characteristics, i.e. nodes, edges and faces. A node of the graph can be split furnishing a tree of nodes while a single node is perceived as a unitary tree. This scheme provides the versatility of adding or removing nodes and manipulating the relations between the nodes without interfering with the addressing scheme of the remaining nodes. The connectivity pointers and the linked lists for accessing the nodes are cut and re-stitched to the new topology, without altering the addressing of the parts of the mesh that are not affected by the new topology, as shown in Fig. 3 (**Right**).

The actual solution vector **X** is stored at a special data structure of the tree node, named leaf in Fig. 1. The leaf contains the characteristic coefficients of the basis functions used for the description of the conservative variables, the Jacobian matrix, the mass matrix and the matrices used for the calculation of the derivative and basis values at the face and volume quadrature points. Eventually, the leaf data structure contains the memory consuming information that describes the actual field. When a tree node is split then the leaf is dropped. This means that the memory consuming data objects are de-allocated and are replaced by the pointers to the kids.

4.2. Domain decomposition

In ForestDG, the graph representation of the computational domain, shown in Fig. 3, is fed into the METIS [73] graph domain decomposition library which furnishes an optimised partitioning of the domain. For each partition, a node is assigned as the local first node of the graph defined by the pointer `forest->next`. The rest of the nodes are accessed iteratively by assigning a `crnt->next` pointer to the next node in the list as shown in Fig. 3 (**Left**). During a simulation, the graph is repartitioned resulting in a balanced computational load along the processes, as shown later in Fig. 6 for a case of three levels of refinement of an unstructured mesh consisting of tetrahedral elements.

5. Splitting and merging

In the event of splitting a node, a number of new nodes are created. A binary type splitting results in two children, a quad tree type of splitting results in four children, and an oct-tree splitting results in eight children. The parent node is removed from the linked list that controls the access to the cells and is replaced by the children nodes. The next pointers of the linked list are re-stitched in a such way that the parent's previous tree node now points to the first of the kids created and also the last kid points to the next of the parent, as shown in Fig. 3 (**Right**).

The geometry of the splitting for hexahedral, prismatic and tetrahedral elements is presented in Fig. 4. Hexahedral elements are split into eight elements which are self similar if the ancestral element has two parallel faces. Four vertices are positioned on the vertices of the higher level cell, three vertices are positioned at the midpoints of the adjacent edges, three vertices are positioned on the centroids of the adjacent faces and a final vertex is positioned at the centroid of the higher level hexahedral, as shown in Fig. 4(a).

The numbering of the children follows the numbering of the higher level cell vertices so that the 1st kid is adjacent to the 1st vertex of the cell and ith kid is adjacent to the ith vertex of the cell. The resulting kids can also be numbered by their positioning in relation to the coordinate system of the element faces adjacent to the kid. For each face f, the kids acquire index i^f which is a function of the face index f and the element kid number i. From the topology of Fig. 4 we can easily construct a simple operator E2F that provides the face index of a kid as $i^f = \text{E2F}(i_L^A, f)$. Naturally, E2F is not defined if i is not adjacent to f.

For prismatic elements, splitting leads to self similar elements (in the case when the ancestral element has two parallel faces) and the cell numbering follows the vertex numbering of the higher level cell. The 7th and 8th children are placed along the core of the prism with reversed orientations, as shown in Fig. 4(b).

Tetrahedral elements are split into four self-similar tetrahedra (only in the case when the ancestral element is a regular tetrahedron) located at the corresponding vertices of the higher level cell, and are numbered as shown in Fig. 4. The

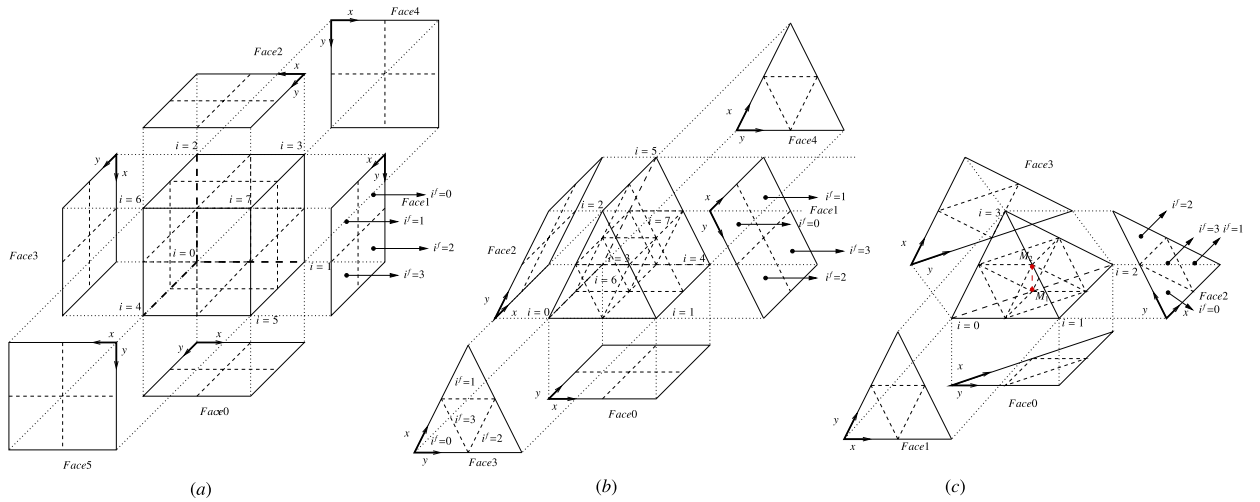

Fig. 4. Topology of oct-tree splitting **(a):** Hexahedral elements. **(b):** Prismatic elements. **(c):** Tetrahedral elements.

remaining space in the element is an octahedron which can be split in three different ways. In order to split the inner octahedron one must choose a splitting plane that lays on the edge connecting the midpoints of two opposite edges of the tetrahedron. One example is the edge M_1, M_2 shown in Fig. 4(c). Choosing the mode for which the distance of the midpoints M_1 and M_2 is minimal results in cells with the minimal skewness since this edge is akin to the parent element and remains un-split. The numbering of the resulting elements is shown in the same Fig. 4(c).

Given that the size of an element reduces to its half at each split, the level of splits L, needed to refine an element with a characteristic size $\mathcal{L}$ to a desirable size ℓ is given by the formula:

$$L = 1.0 + \frac{\log(\mathcal{L}/\ell)}{\log(2)} \ . \tag{19}$$

Several quantities can be used as a refinement criterion: density gradients serve as indicators of shocks and vorticity (or shear stress magnitude) identifies areas of vortical flow structures. Furthermore, the user can impose a predefined level of refinement based on specific geometrical properties that define certain regions of interest of the flow field.

6. Connectivity assignment

Once the cells are split to the required resolution, the connectivity of the cells needs to be remapped. For the ancestral grid $\cup E_m$ the connectivity is resolved by means of tracking the common mesh vertices at the onset of the simulation. The matching of the common element vertices gives the information on neighbouring cells, neighbouring faces and the orientation of these faces. For the element E_m and at each face f, the pointer $E_m->\text{neig}[f]$ to the neighbouring element E_n, the pointer $E_m->\text{face}[f]$ to the adjacent face f_a of the neighbour E_n and the relative orientation angle a of the adjacent faces $E_m->\text{angl}[f]$ are represented by the following relations:

$$N(E_m, f) = E_n \ , F(E_m, f) = f_n \ , A(E_m, f) = a \ . \tag{20}$$

The above connectivity relations are important for finding the relative position of the computational space of an element to the computational space of its neighbour needed for the evaluation of the surface integrals. In the event of splitting an element, the connectivity calculations cannot be based on tracking mesh vertices, not only because of non-conformalities (i.e. the neighbouring faces do not share common vertices anymore), but also because a search algorithm would impose a heavy computational load. Finally, the splitting and merging process is a dynamic procedure during which the neighbouring cells may not have been formed yet. Additionally, the creation of a new element raises the need to resolve the connectivity of the new element but also alters the connectivity of all surrounding elements which must be updated. Here, we present an explicit algorithm that solves the connectivity problem by providing an explicit expression for the three connectivity relations described by Equations (20).

6.1. Connectivity assignment of keen cells

On the first stage, the connectivity is evaluated among the siblings of a split element P. Given that the elements of the same type, split in the same way are also arranged in the same way, they present a global internal connectivity pattern. This is expressed by the array of keen elements K. We define a keen element as the neighbour of an element at a specific face if

this is a sibling, or the parent of the element if the kid shares the specific face with its parent. Thus, the keen is a neighbour assuming that all the siblings are incubated within the parent cell. This concept leads to an expression of the connectivity which is intrinsic to the specific branch that is being split. This split is neither affected nor affects the connectivity of the neighbouring branches.

For each type of splitting (oct-tree, quad-tree, binary-tree) on each type of tree (hexahedral, prismatic or tetrahedral) the connectivity among the siblings remains the same since the siblings are arranged in the same way within a parent. The following array provides the neighbours for the i_L^Ath kid $A=\{P,i_L^A\}$ of P within an hexahedral element for each face f:

$$K_A^{hex}[f,i_L^A]=\begin{bmatrix} \{P\} & \{P\} & \{P,0\} & \{P,1\} & \{P\} & \{P\} & \{P,4\} & \{P,5\} \\ \{P,1\} & \{P\} & \{P,3\} & \{P\} & \{P,5\} & \{P\} & \{P,7\} & \{P\} \\ \{P,2\} & \{P,3\} & \{P\} & \{P\} & \{P,6\} & \{P,7\} & \{P\} & \{P\} \\ \{P\} & \{P,0\} & \{P\} & \{P,2\} & \{P\} & \{P,4\} & \{P\} & \{P,6\} \\ \{P\} & \{P\} & \{P\} & \{P\} & \{P,0\} & \{P,1\} & \{P,2\} & \{P,3\} \\ \{P,4\} & \{P,5\} & \{P,6\} & \{P,7\} & \{P\} & \{P\} & \{P\} & \{P\} \end{bmatrix}. \tag{21}$$

As can be seen from the above expression, the keen of each kid $\{P,i_L^A\}$ is its neighbouring sibling $\{P,j\}$ if the face f is internal to the parent element. The keen is the parent element itself $\{P\}$ if f is adjacent to the border of the parent element.

Array K_A introduces a pre-solved internal connectivity among the siblings. The connectivity outside the parent will then be evaluated through the connectivity of the parent P and the relative orientations of the ancestral cell E_m, at the second stage. The neighbouring faces within a branch are evaluated by the arrays $F_A^{hex}[f,i_L^A]$ and $A_A^{hex}[f,i_L^A]$ which provide the neighbouring face index and the neighbouring face orientation angle, respectively, as:

$$F_A^{hex}[f,i_L^A]=\begin{bmatrix} 0 & 0 & 2 & 2 & 0 & 0 & 2 & 2 \\ 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \\ 0 & 0 & 2 & 2 & 0 & 0 & 2 & 2 \\ 3 & 1 & 3 & 1 & 3 & 1 & 3 & 1 \\ 4 & 4 & 4 & 4 & 5 & 5 & 5 & 5 \\ 4 & 4 & 4 & 4 & 5 & 5 & 5 & 5 \end{bmatrix}, \tag{22}$$

and

$$A_A^{hex}[f,i_L^A]=0\,. \tag{23}$$

The arrays of the keen elements K, the neighbouring faces F and the orientation angles A for prismatic and tetrahedral elements are presented in the Appendix B in Equations (B.1) to (B.6).

6.2. Connectivity assignment of neighbouring cells

In order to locate the direct neighbour of the element $A=\{E_m,i_1^A,i_2^A,\ldots,i_L^A\}$ at the face f, we evaluate the following recursive algorithm:

Algorithm 1 Location of neighbouring ancestor C for the kid A at the face f.

1: $l \leftarrow Level(A)$
2: $B \leftarrow A$
3: $C \leftarrow K_A[f,i_l^A]$
4: **while** $Level(B) \neq Level(C)$ **do**
5: $\quad B \leftarrow C$
6: $\quad C \leftarrow K_B[f,i_l^B]$
7: $\quad l \leftarrow Level(C)$ (or equivalent $l=l-1$)
8: **end while**

The above algorithm is a recursive evaluation of the keens of the element A which is stored in C while the new value is stored in B. At each iteration, C is a level lower than B. If C is at the same level with B, then B is a direct neighbour of $C=K_B[f,i_l^B]$. Hence, B and C are either two neighbouring siblings or two neighbouring ancestral elements E_m,E_n. If B and C have the same level from the first evaluation in line 2, it means that B is a sibling and a direct neighbour of A as described by the array (21) and the connectivity problem is solved. If B and C are at different levels, then C is the parent of B and the recursion continues as required by the condition in line 4. At the end of the algorithm, we obtain the array C which is a direct neighbour of B, and B which is an ancestor of A. At this point, we are certain that the neighbour of the initial array A is one of the descendants of C.

An example of connectivity tracking for two adjacent elements is shown in Fig. 5. The keen of A at the face $f=1$ is the element B. The evaluation of the connectivity of the keens for B will point to the prismatic element C, shown on the right

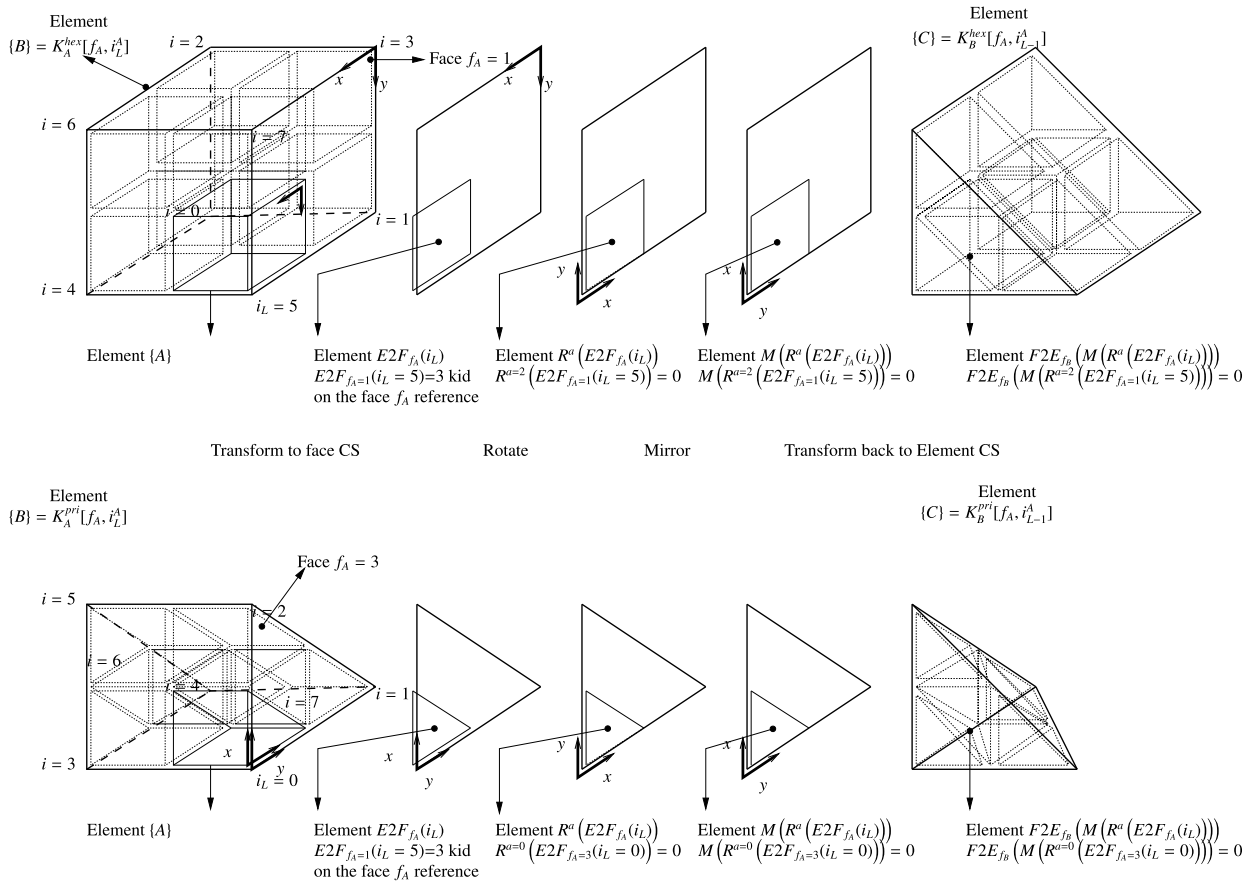

Fig. 5. Top: Identifying the neighbouring element of a split element at a hexahedral to prism of oct-tree splitting, incorporating a rectangular face. **Bottom:** Identifying the neighbouring element of a split element at a prism to tetrahedron of oct-tree splitting, incorporating a triangular face.

side of Fig. 5. The elements B and C can be two siblings of a parent at a lower level or two ancestral elements of the initial mesh. The element A is the 5th kid of B ($i_L^A = 5$) located at the bottom right corner of B. Using the operator $E2F_{f=1}(i_L^A)$, the index of A at the face f is $i^f = 1$. The face is then rotated, using operator R^a, by the orientation angle a for B and C inferred from (20) or (23). The rotation results in an index of the transformed face $R^a(E2F_{f=1}(i_L^A)) = 0$. Finally, given that the faces of B and C are opposite to each other, the face should be mirrored as $M(R^a(E2F_{f=1}(i_L^A))) = 0$. Thus, the neighbour of A at $f = 1$ is the kid $i_L^f = 0$ of C at face f_C. Introducing the reverse transformation $F2E_f(i^f)$, the neighbour of A is provided by the following expression as the jth kid of C:

$$j_L = \mathrm{F2E}_{f_B}\left(M\left(R^a\left(\mathrm{E2F}_{f_A}\left(i_L\right)\right)\right)\right) . \tag{24}$$

The mirror and rotation operators for rectangular and triangular faces are defined as:

$$M^{rec}(i^f) = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 3 \end{bmatrix} , \quad M^{tri}(i^f) = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 3 \end{bmatrix} , \quad R^a_{rec}(i^f) = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix} , \quad R^a_{tri}(i^f) = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 3 \end{bmatrix} . \tag{25}$$

In the general case, when the elements B and C at the intersection of two neighbouring branches are not just one level below, the neighbour of A is defined as:

$$N\left(\{E_m, i_1^A, i_2^A, \ldots, i_L^A\}, f\right) = \{C, \mathrm{F2E}_{f_B}\left(M\left(R^a\left(\mathrm{E2F}_{f_A}\left(i_l^A\right)\right)\right)\right), \ldots, \ \mathrm{F2E}_{f_B}\left(M\left(R^a\left(\mathrm{E2F}_{f_A}\left(i_L^A\right)\right)\right)\right)\} . \tag{26}$$

The important advantage of this method is that Equation (26) is explicit and does not involve the connectivity of the neighbouring branch. This equation is based on the connectivity at the closest intersection of the neighbouring branches between B and C. In cases of dynamic adaptation, it is never certain that the neighbouring cell actually exists or it is located in the same partition. Formula (26) gives the neighbouring cell address regardless if the topology of the neighbouring branches has changed or is going to change.

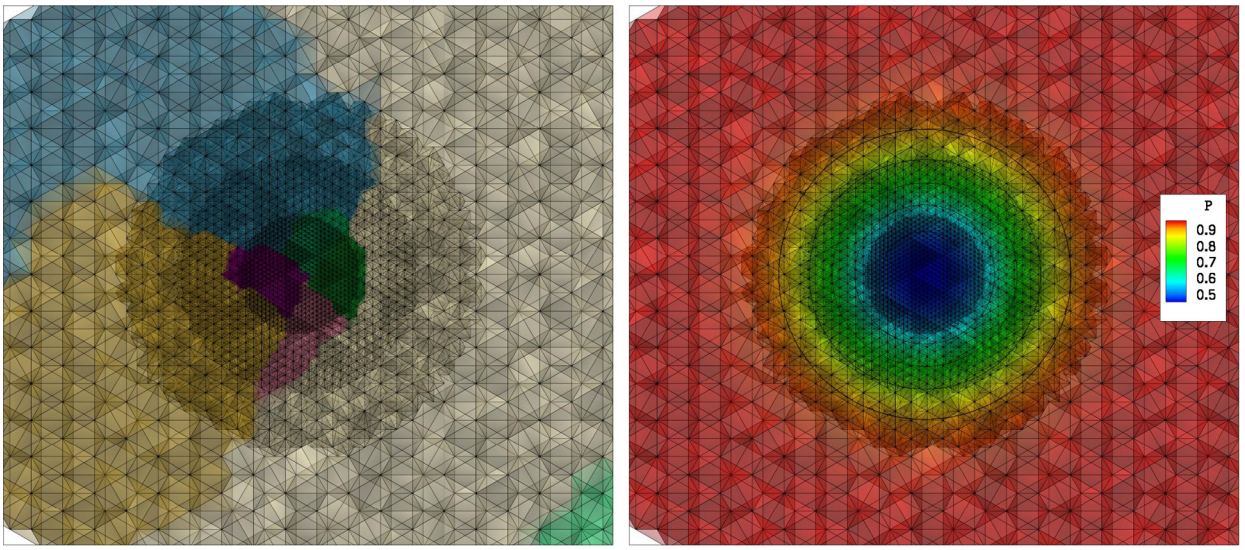

Fig. 6. Left: Domain decomposition for a tetrahedral mesh refined to three levels. Different colours represent the mapping of different partitions. **Right:** Pressure distribution for an isentropic vortex projected on a tetrahedral grid. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

In the case when cell B has not been refined up to the same level L of element A, Formula (26) becomes:

$$N(A) = \{C, \mathrm{F2E}_{f_B}\left(M\left(R^a\left(\mathrm{E2F}_{f_A}(i_l)\right)\right)\right), \ldots, \ \mathrm{F2E}_{f_B}\left(M\left(R^a\left(\mathrm{E2F}_{f_A}(i_{L-1})\right)\right)\right)\}, \tag{27}$$

where $L-1$ is the maximum level which can be reached at the branch of C.

Furthermore, the address obtained by Equation (26) can be a parent of a cell that has been refined further. In this case, the neighbour pointer of A at f points to the children of $N(A)$ located at the face f_B. Although the splitting/merging and connectivity algorithms have been designed to deal with an arbitrary level of non-conformalities, the numerical evaluation of the face fluxes has been designed only for non-conformalities 1:1 1:2 and 2:1. Thus, after the refinement of the mesh, a smoothing pass might need to be implemented. This pass reassures that if a neighbouring cell is refined to more than one level, then the current element is refined further.

6.3. Solution projection, face fluxes

When a cell is split into a number of kids or when a set of kids merge to their parent, the solution has to be projected from the parent to the kids or visa versa. This is achieved by the Galerkin projection of the solution [74] on the quadrature points of the new element to the basis of the new elements.

An example of Galerkin projection is shown in Fig. 6 (**Right**) where an isentropic vortex field is initially projected on a grid consisting of tetrahedral elements. In Fig. 6 (**Right**) we show the projection of an isentropic vortex field solution from an initial level equal to 2. The solution is projected to the merged tetrahedral elements of level 1 in the outer radius of the vortex and to the split tetrahedral elements with level 3 in the core of the vortex. The mapping of the quadrature points for the new elements in the computational space of the old element is described in Fig. 7 (**Left**).

In the right column of Fig. 7 (**Left**) the topology of the physical space and arrangement of the kids within the physical element are shown for three types of elements discussed above. The physical space is mapped to the computational space shown in the right column of the same figure. Although the arrangement of the kids in the physical space is straightforward and has been described in the previous section, the sub-domain of each kid is mapped to the computational domain of the parents through the transformations (A.1) to (B.3). Assuming that A and B are two overlapping elements with either A contained in B or B contained in A, the computational space coordinates of B are mapped to the computational coordinates of A as:

$$\{\eta_1^A, \eta_2^A, \eta_3^A\} : \mathbf{x}^A\left(\eta_1^A, \eta_2^A, \eta_3^A\right) = \mathbf{x}^B\left(\eta_1^B, \eta_2^B, \eta_3^B\right). \tag{28}$$

In the general case, which includes tetrahedral and prismatic elements, the transformation (28) is not linear. Equation (28) can be solved numerically by introducing the Jacobian $\partial\eta_i^A/\partial\eta_j^B$. All the descendants across the branch on an ancestral element are split in a similar manner. Thus, this transformation is valid for all the mappings from a parent to its kids and

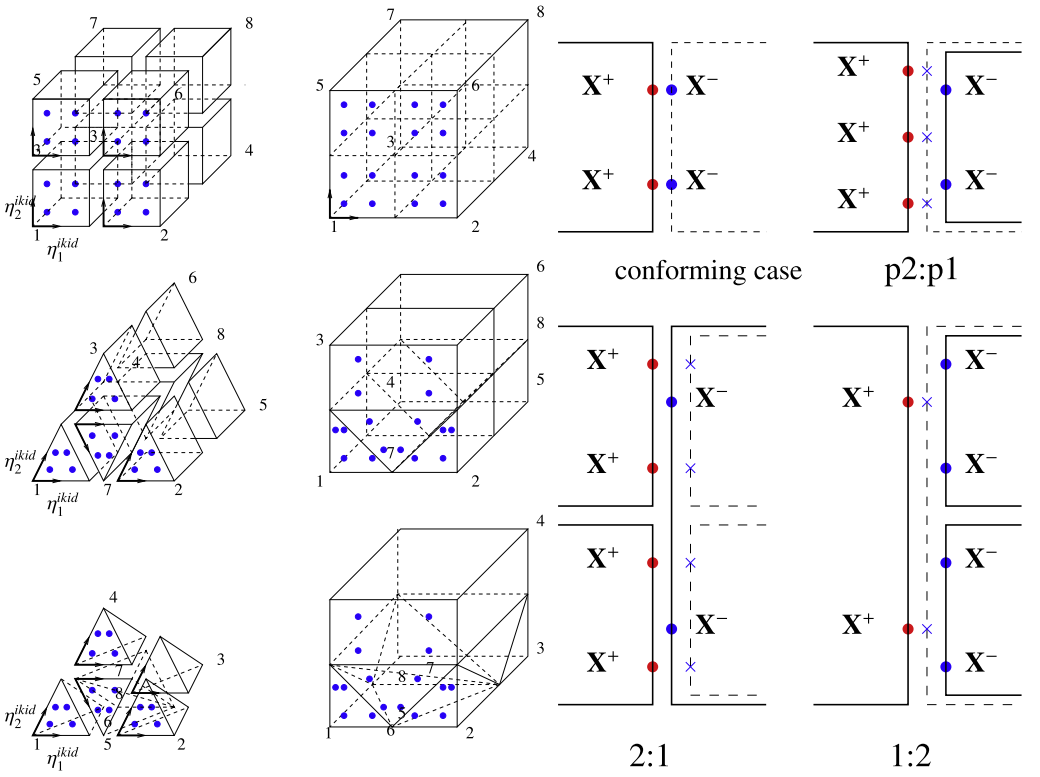

Fig. 7. Left: Transformation from physical to computational space for split hexahedral, prismatic and tetrahedral elements. Blue dots represent the quadrature points (assuming $p1$ expansion) on the computational space of the kids. These quadrature points are mapped taking into account the coordinate transformations for each type of elements to the computational space of their parent. The relative location of the quadrature points is shown in the right column for each type of element. **Right:** Types of non-conformalities encountered in h/p adaptive cases. Solid lines indicate the neighbouring cells, dashed lines indicate the ghost faces on which the solution is projected. In the first step (Scattering), in the case of 2:1 connectivity, for the evaluation of fluxes on the + side the conserved variables from the solution of the − side is projected to the ghost face of the same level. In the second step (Gathering), in the case of 1:2 connectivity, for the evaluation of the fluxes on the + side element, the fluxes as calculated on the − side are projected to the ghost face of the same level.

for all the kids to the parent within a branch. For the case of hexahedral elements, the above transformation is linear and the computational space of B can be mapped to the computational space of A as:

$$\eta_1^A = c_1^{1,B}\eta_1^B + d^{1,B},\ \ \eta_2^A = c_2^{2,B}\eta_2^B + d^{2,B},\ \ \eta_3^A = c_3^{3,B}\eta_3^B + d^{3,B}\,, \tag{29}$$

where coefficients c^B and d^B depend on the relative position between A and B in the computational space. This is shown for an example of the parent to kid mapping for a hexahedral element at the right column of Fig. 7 (**Left**). Evaluating the current solution on the quadrature points of the new elements, we can project the solution vector on the new elements.

The projection of the solution on kids or parents is also applied for the evaluation of the fluxes for non-conformal faces. Non-conformalities arise due to p-refinement, h-refinement or both as shown in Fig. 7 (**Right**). In [74,17], a gather operator and a scatter operator are used for the calculation of fluxes on non-conforming meshes. The gather operator $\mathbf{P}^{G\ \mathrm{iminor}}$ is used for evaluating the solution (fluxes) on the major element basis from the minor face basis iminor. The scatter operator $\mathbf{P}^{S\ \mathrm{iminor}}$ is used for evaluating the conserved variables from the major element face to the minor face basis iminor. The gather and scatter operations are described by the following equations:

$$\mathbf{f} = \sum_{\mathrm{iminor}=1}^{nmin} \mathbf{P}^{G\ \mathrm{iminor}}\mathbf{f}^{\mathrm{iminor}},\ \ \text{and}\ \ \mathbf{U}^{\mathrm{iminor}} = \mathbf{P}^{S\ \mathrm{iminor}}\mathbf{U}\,, \tag{30}$$

where *nminor* is the number of minor non-conforming faces. Operators $\mathbf{P}^{G\ \mathrm{iminor}}$ and $\mathbf{P}^{S\ \mathrm{iminor}}$ ensure that the flux from the one side of the non-conformal face is equal to the sum of the fluxes to the other side of the non-conformal face and vice versa. This is expressed by the following equations for an arbitrary field $\mathbf{U}$:

$$\oint_{S_{\mathrm{iminor}}} \left(\mathbf{U}^{\mathrm{iminor}} - \mathbf{U}\right) dS_{\mathrm{iminor}} = 0,\ \text{for each iminor},\ \ \text{and}\ \oint_{S} \left(\mathbf{U} - \tilde{\mathbf{U}}\right) dS = 0, \tag{31}$$

where $\tilde{\mathbf{U}}$ is the branch function that corresponds to the discontinuous distribution of the conserved variable vector of the children on the parent face. The calculation of the fluxes across non-conformal faces is performed by scattering the conserved variables, described by the neighbour major face solution, to the minor ghost faces. Thus, the problem is reduced to a conforming case where Equations (15) and (16) for the viscous and inviscid fluxes are applied. Following [74,17] in the case of a 2:1 connectivity, the calculated fluxes are evaluated on low level faces. In the case of a 1:2 connectivity the calculated fluxes are gathered using the $\mathbf{P}^{G\ \mathrm{iminor}}$ operator and evaluated on the higher level element.

For the cases of rectangular faces belonging to either hexahedral or prismatic faces the projection is based on the linear relation (29) which is solved explicitly. For triangular faces belonging either to prisms or tetrahedra the implicit relation (28) is used instead. In (28) one of the computational coordinates η_i^A and η_i^B are either -1 or 1 depending on the number of the adjacent faces of A and B. The ghost faces do not appear as a part of the computational domain, but they are linked to the tree nodes with which they overlap. As a result, the connectivity of an element with the ghost faces of the non-conforming neighbour reduces to a conforming case.

Since the fluxes are eventually projected to a different basis, their distribution across the faces is not identical [17]. The distribution of the fluxes across the minor faces is discontinuous while in the major face, the distribution of the fluxes is evaluated on a single element. On the major element, it is continuous and it can even be of a different order. Although, Equation (10) guarantees the conservation of the mass, the projection of the flux introduces an error at the level of the discretisation error of the scheme. According to [17], the level of this error in the total mass conservation increases with the number of elements. It was concluded, however, that this error remains at the levels of the discretisation error, as in the standard conforming cases.

Both in p- and in h-adaptivity the solution is either evaluated from a poor resolution or projected to a poor resolution. Thus, AMR can only locally improve the accuracy of the simulation and care must be taken so that important features of the flow should be resolved at a sufficiently high accuracy throughout the solution of the flow. If, however, the important flow structures are resolved adequately, the global accuracy of the simulation is determined by the refined resolution as highlighted in Section 7.1.

7. Test cases

In this section we present a series of test cases for which the accuracy and efficiency of the new code is evaluated. Firstly, we consider a viscous test case with a manufactured solution to investigate the accuracy of the code. Then the numerical efficiency and accuracy of the code for capturing oblique shock waves will be investigated.

7.1. Spatial discretisation

The method of manufactured solution [75,76] is used for the investigation of the order of accuracy of the DG discretisation. A steady state unidirectional flow field is considered:

$$\rho = 1.0; \quad p = 1.0; \quad u = 0,\ v = 0,\ w(x, y, z, t) = w_0 \left(1 - \cos(4\pi x/L)\right)\left(1 - \cos(4\pi y/L)\right)\ , \tag{32}$$

where w_0 is taken equal to $w_0 = 0.3c$. Outflow conditions are assumed for all boundary faces, i.e. $\mathbf{U}^+ = \mathbf{U}^-$ for all variables of the state vector.

Introducing this flow field into Equations (1) we obtain analytical expressions for the source term w_d. This source term balances the viscous forces and sustains the steady state manufactured solution (32). Due to the spatial and temporal discretisation errors, this solution is distorted. A small timestep and an implicit time marching scheme [9] are used to keep the temporal error low. The error of the numerical solution, compared with the exact solution (32), is described by the L_2 norm [77]:

$$L_2 = \left(\frac{1}{V_\Omega} \sum_{m=1}^{N} \int_{E_m} (\mathbf{u} - \mathbf{u}^{\mathrm{exact}})^2 dV \right)^{1/2} . \tag{33}$$

For the flow field described by (32), Equations (1) were integrated up to $t_{tot} = 0.1L/c$ on a computational domain of size L. Various discretisations were tested. The Reynolds number of the flow for all cases was $Re = \frac{\rho c L}{\mu} = 1000$. Each discretisation is characterised by a background uniform mesh with element size Δx. For each test case, h-refinement is uniformly imposed up to a level *L+1* and a refined resolution $\Delta x/2$ for all the elements within a circle of diameter 0.5 around the maximum velocity point of the solution (32) as shown in Fig. 8. The elements outside this area were refined to a lower level L and a nominal resolution Δx. The final solution for hexahedral and prismatic discretisations is shown in Fig. 8 (**Left**) and (**Right**). The order of the polynomial basis is uniform for all the elements. This test is introduced to assess the order of accuracy for the discretisation of our implementation, in the case of a non-conforming mesh, refinement. The maximum order of accuracy is $p + 1$ for basis functions which are polynomials of degree p. Thus, the L_2 norm of the error is expected to depend on the mesh resolution Δx as $L_2 \sim (\Delta x)^{p+1}$.

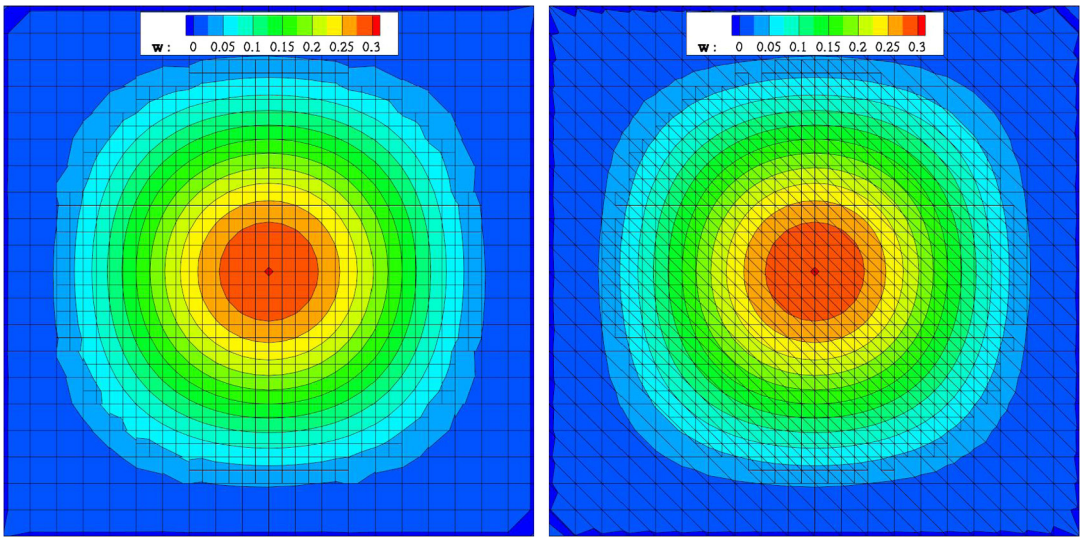

Fig. 8. The final distribution of the w velocity for a square subdomain of the manufactured solution field with size $L/2$ **Left**: For a spatially refined unstructured hexahedral mesh. **Right**: For a spatially refined unstructured prismatic mesh.

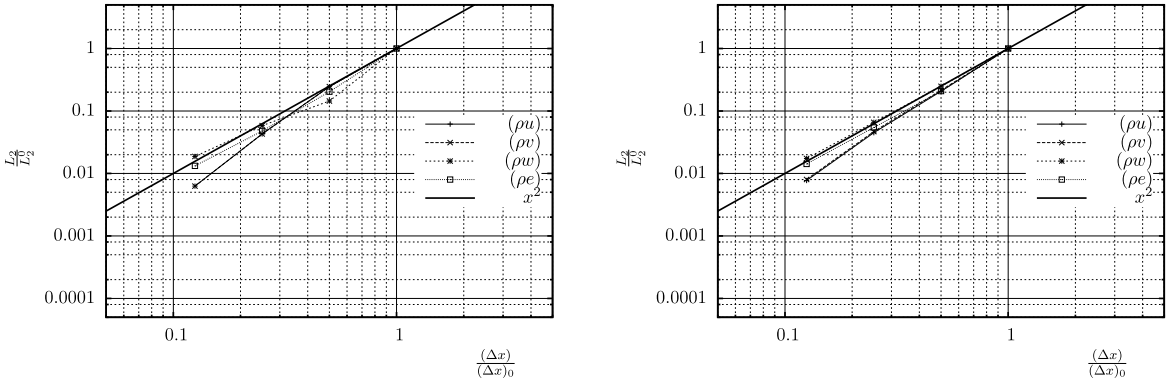

Fig. 9. Values of the L_2 error for the momentum components and energy, normalised by the value of the error for the coarse discretisation, versus mesh resolution for the second order discretisation ($p = 1$). **Left:** Hexahedral elements. **Right:** Prismatic elements.

The normalised value of the L_2 error is presented in Fig. 9 for a number of mesh sizes. Four mesh sizes were used, with $N = 10$, 20, 40 and 80 cells per direction at the edges of the domain, resulting in mesh sizes Δx equal to $\Delta x = L/N$. For each discretisation, the core of the domain is further refined to one more level. The L_2 error, defined in Equation (33), displays a second order decrease with the mesh size. Thus, the implementation of the adaptivity preserves the order of accuracy. Results similar to those shown in Figs. 9, but for the third and fourth orders of accuracy, with $p = 2$ and $p = 3$ polynomial orders, are presented in Fig. 10. As follows from this figure, the expected accuracy is achieved in these cases, as for $p = 1$.

7.2. Shock capturing

In this section, we assess the shock capturing capabilities of ForestDG for the advection dominated problem of an oblique shock generated by a supersonic flow at Mach number $M_1 = 2$ over an inclined ramp at $\theta = 10°$. This problem has an analytical solution and has been widely used as a testbed for compressible solvers [69]. The scope of this test is the assessment of the order of accuracy. In the next section this problem is used for the assessment of the computational efficiency of the AMR methodology. The shock angle β, Mach number M_2 and gas density ρ_2 at the downstream side of the shock were calculated from the analytical expressions [69]:

$$\tan\theta = 2\cot\beta\frac{M_1^2\sin^2\beta - 1}{M_1^2(\gamma + \cos 2\beta) + 2},\ \rho_2 = \rho_1\frac{(\gamma+1)M_1^2\sin^2\beta}{(\gamma-1)M_1^2\sin^2\beta + 2},\ M_2 = \frac{1}{\sin(\beta-\theta)}\sqrt{\frac{1 + \frac{\gamma-1}{2}M_1^2\sin^2\beta}{\gamma M_1^2\sin^2\beta - \frac{\gamma-1}{2}}}, \quad (34)$$

where, ρ_1 and M_1 are the density and Mach number of the gas at the upstream side, and γ is the heat capacity ratio.

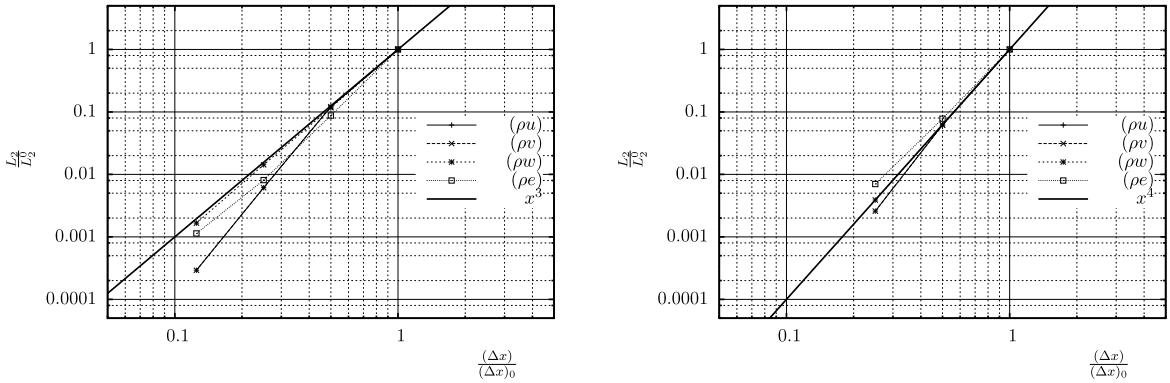

Fig. 10. Values of the L_2 error for the momentum components and energy, normalised by the value of the error for the coarse discretisation, versus mesh resolution for hexahedral elements. **Left:** Third order discretisation ($p = 2$). **Right:** Fourth order accurate discretisation ($p = 3$).

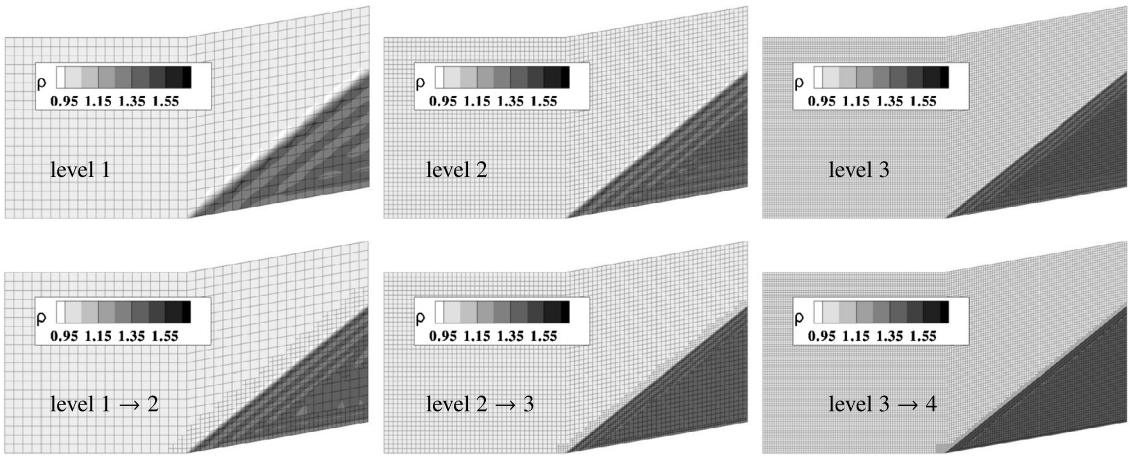

Fig. 11. Density distribution for the converged numerical solutions to the oblique shock problem. **Top:** Level 1, Level 2 and Level 3 solutions on a uniform hexahedral mesh. **Bottom:** Converged AMR solutions restarted from the corresponding solutions in the upper row.

The problem was solved numerically by integrating the inviscid form of Equations (1) on the geometry shown in Figs. 11. The computational domain consists of two blocks with unity sides forming a 10° half-wedge. The domain is discretised using 20×40 elements with base (coarse level) resolution $\Delta x_0 = 0.05$. For the solution presented in Fig. 12 (**Right**) the base mesh is dynamically refined by up to five levels to a resolution of $\Delta x = 0.003125$ based on the density gradient criterion. For this simulation, p1 polynomial basis was used. The shock capturing characteristics of the AMR approach resulted in the stable bounded solution without the use of the TVB limiter.

Neighbouring cells are also meshed to a gradually increasing resolution after the implementation of the smoothing pass. The three numerical solutions shown in the upper row of Fig. 11 were obtained using uniform grids starting from the base resolution and reaching up to four levels of refinement (not shown). The L_2 errors of the numerical solutions for the Mach number M_2 and density ρ_2 are shown in Fig. 12 (**Right**). As follows from this figure, the level of the L_2 error decreases as Δx^2, in agreement with the prediction of the second order discretisation ($p = 1$).

Each numerical solution shown in the top row of Fig. 11 was locally refined by one more level at areas where the density gradients indicate the location of the oblique shock. The solutions on the uniform meshes were integrated on the locally refined mesh until their residuals converged to new values. The values of the L_2 error for each refined solution are shown as the points connected with dashed lines in Fig. 12 (**Left**). In this figure, these errors are presented as functions of the spatial resolution of the refined area.

The local refinement of a solution in the area of the shock results in the reduction of the residuals of Level n solution to the values of Level $n + 1$ solution. Although the errors of the refined solutions do not reach exactly the L_2 levels, inferred from the corresponding uniform mesh solutions, the L_2 AMR errors decrease with the increasing resolution following the second order law. As follows from the intermediate solution shown in Fig. 12 (**Right**), the formation of the shock is a dynamic process. The local refinement has to continuously adapt to the changing flow geometry. For the oblique shock result, shown in Fig. 12 (**Right**), AMR provides an efficient way to capture flow discontinuities.

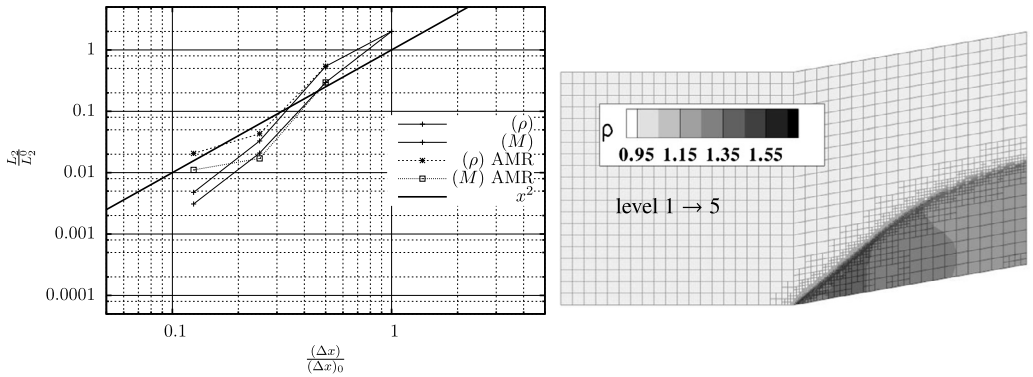

Fig. 12. Left: Values of L_2 errors for density ρ_2 and Mach number M_2, normalised with the error of the coarse discretisation at the upwind side of the shock, versus the mesh resolution. For the AMR cases Δx corresponds to the refined cell size. **Right:** Intermediate solution for density ρ_2 during the formation of the oblique shock for an AMR scheme from the base level 1 up to level 5.

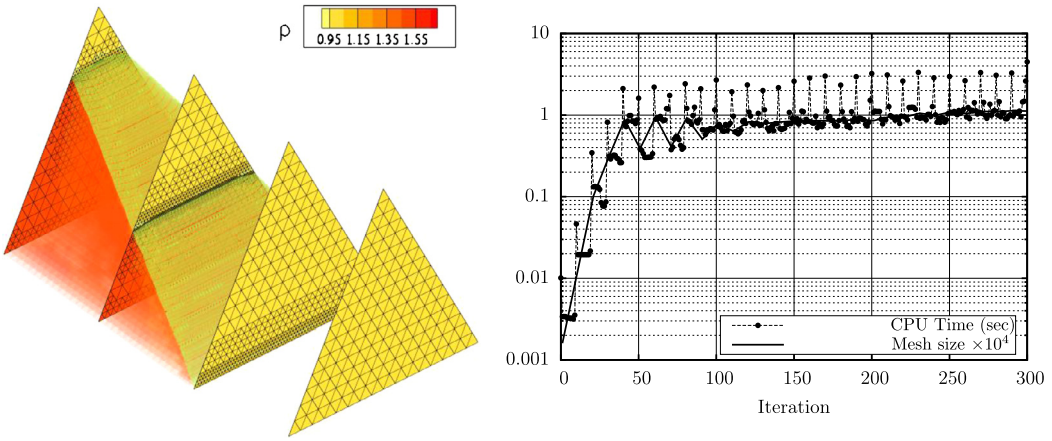

Fig. 13. Left: Density distribution for the oblique shock problem discretised with prismatic elements. Light (yellow) iso-surface indicates the $\rho = 1.05$ contour. **Right:** Performance of the AMR algorithm for the three dimensional oblique shock simulation using oct-trees. (Solid curve): CPU time for each timestep. (Dashed curve): Number of elements.

7.3. Computational efficiency

In this section we perform an assessment of the computational efficiency of the AMR implementation. The assessment is carried out for two problems. Firstly, the numerical solution of the three-dimensional problem of the oblique shock presented in the previous section. Computationally, this specific case represents problems where the complexity arising from the solution drives a gradual increase in the degrees of freedom of the simulation. In this test, we assess the scaling of the computational cost as the topological forest develops. The second problem is a typical viscous problem of the flow around a cylinder at Re $= 500.0$. In this case we assess the performance of the AMR methodology for a developed AMR forest which is periodically re-adapted to resolve the moving vortical structures.

In Fig. 13 we present the solution to the oblique shock problem on a domain consisting of two prisms joint on their triangular faces so that their rectangular bases form a 10° half-wedge. The initial mesh consists of only two prisms. Each of these prisms is uniformly refined up to five levels. The density gradient is used to identify the elements intersected by the oblique shock. These elements are further refined up to nine levels. This demonstrates the ability of the method to capture flow structures. In this specific case, the initial ancestral mesh with size $N = 2$, finally reaches $N \sim 30.000$ elements. These elements follow the evolution of the oblique shock to the steady state solution. For technical reasons, the domain decomposition is restricted to two computational domains, since the ancestral mesh consists of only two trees. As described in Section 4.2, the forest graph is dynamically repartitioned. Although most of the new trees reside under the second ancestral tree (the prism on the ramp), they are equally distributed among the two processors.

In Fig. 13 (**Right**) we present the CPU time per timestep, for the first 300 steps of simulation when the mesh starts from 16 elements (using oct-tree splitting, the two initial cells are split into 16 cells at the start of the simulation) and reaches $N = 11.685$ trees after the 300th iteration. The mesh is refined every 10 timesteps to account for the changes in the flow structure. As can be seen in Fig. 13 (**Right**) the computational cost of AMR is 2–3 times greater than the actual cost of the integration of the governing equations for the Euler explicit time advancement scheme. For the interval of 10

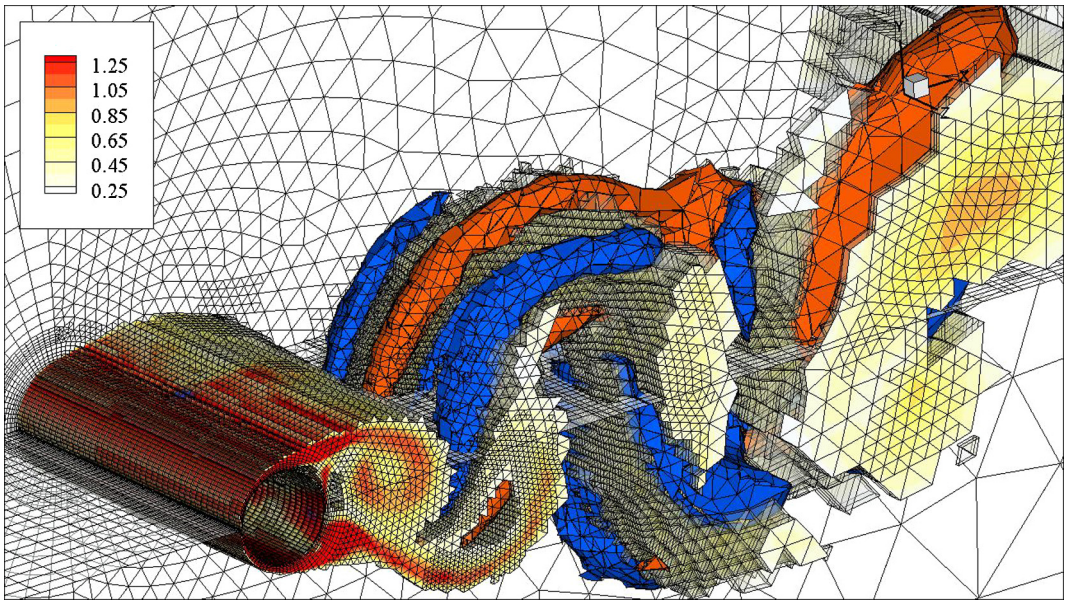

Fig. 14. Left: AMR simulation of the flow around a cylinder at $Re_D = 500$. The contour levels range from white to red and correspond to the vorticity magnitude. The red and blue iso-surfaces correspond to the lateral velocity levels $w = \pm 0.05M$. The positive and negative lateral velocities show the three dimensional structure at the location of the lambda vortices which connects the main trailing vortices of the wake (rollers). Only the elements with vorticity magnitude larger than 0.25 are shown, in order to reveal the structure of the oct-tree split mesh. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

timesteps, however, the computational overheads due to the use of the AMR sum up to between 20% and 30% of the total computational time. The benefits of using AMR are even more clearly seen if we consider the computational cost needed for reaching the required resolution in three dimensional cases for uniform meshes. In this case the number of cells for a 9 times finer mesh would have been of the order of 10^7. This is two to three orders of magnitude greater than the number of cells needed to reach the same resolution using AMR.

In order to assess the computational cost breakdown for the different steps of the AMR methodology, we carried out an AMR simulation for the case of the viscous subsonic flow around a cylinder with diameter D. The far field velocity is $U_0 = 0.3c$ and the macroscopic Reynolds number is $\mathrm{Re}_D = \frac{cMD}{\nu} = 500.0$, where c is the speed of sound. Under these specific conditions, a characteristic vortex path emerges in the wake of the cylinder [78]. In Fig. 14 we present the results of our simulation using the AMR methodology. The trailing vortices form cylindrically shaped rollers. The velocity field shows a three-dimensional structure with the formation of characteristic connecting vortices between the rollers, parallel to the streamwise direction. The initial mesh consists of $N = 39760$ prismatic and hexahedral elements and the simulation is discretised with $P1$ elements. The elements are refined at level 2 within a radius equal to the cylinder diameter D and at a level up to 3 based on the local vorticity magnitude. For our analysis, the total number of elements ranges from 250.000 to 300.000. Due to the size of the problem, the specific case was integrated in time with an explicit RK method using a time-step equal to $dt = 10^{-3}$. The mesh is refined every 100 steps. A characteristic displacement of the conveyed structures for this adaptation interval is $0.03D$, which corresponds to the smallest mesh size of the problem (The boundary layer elements size is 0.02D, allowing the resolution of the boundary layer structure at the wall with $y^+ \sim 1$). With this set-up, the vortical structures are always within the refined domain, allowing the potential for even greater adaptation intervals than the 100 steps at the far-field.

In Table 1 we present the computational time breakdown for each of the processes of the AMR methodology for four decompositions on 20, 40, 80 and 160 processors, clustered in nodes of 40 cores each. The times are measured using calls to a stopwatch subroutine based on the `MPI_WTIME()` function. The total time of the simulation is broken down into three parts: solution, mesh adaptation and re-partitioning which sum up to the total computational time. As can be seen from the table data, the most demanding process is the solution of the equations, which contains the costs of the calculation of the surface and volume integrals and the integration in time. In Table 1, the solution CPU time also contains the cost for communicating the solution across the partition boundaries and the calculation of the gradients (auxiliary variables $\boldsymbol{\Theta}$).

The mesh adaptation is the second most demanding part of the algorithm. The adaptation cost increases with the number of the processors involved in the solution. As can be seen from the table, the calculation of the connectivity process requires a small fraction of the adaptation part of the computational time. The connectivity process involves all the rearrangements of the linked lists needed for the updates of the mesh topology, the assignment of the keen neighbours, the calculation of the new connectivity and the smoothing of the mesh that guarantees $1:2$ connectivity. The main cost of the mesh adaptation part, in this case, is the allocation of the new leaves and the projection of the solution while merging or splitting.

Table 1
AMR computational cost breakdown.

Number of CPU's (nodes × cores)		1 × 20	1 × 40	2 × 40	4 × 40
Process	Units	Time breakdown			
Solution	(%)	84.82	74.97	64.43	53.25
Communication	(%)	9.51	8.99	10.88	8.46
Gradients	(%)	18.68	13.80	9.70	6.05
Adaptation	(%)	13.56	21.98	31.05	38.86
Connectivity	(%)	1.00	1.65	2.28	2.80
Partitioning	(%)	1.62	3.05	4.52	7.88
Exchange	(%)	1.21	2.14	2.56	3.68
Time	(s)	6331.0	5404.0	3449.0	2397.0

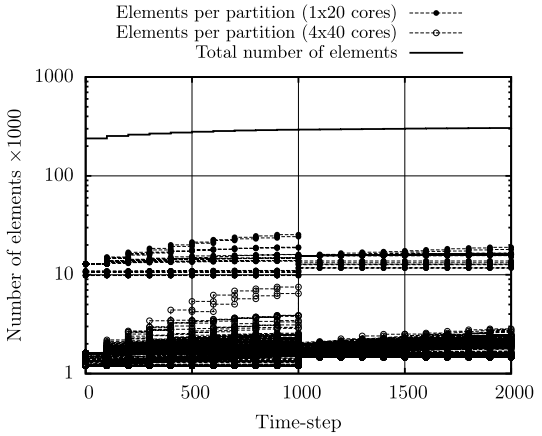

Fig. 15. Evolution of the total number of elements for the three dimensional simulation of the viscous flow around a cylinder with $L/D = 4$. (Solid curve): Total number of elements. (Dashed curve): Number of elements in each partition. Circles correspond to a 4×40 cores simulation and solid circles to a 1×20 cores simulation.

The last part of the algorithm is the re-partitioning of the computational domain, which results in a balanced workload among the processors. The most demanding part of this procedure is the exchange of the information across the processors. In Fig. 15 we present the evolution of the number of elements for the AMR simulation during the stages of the refinement to the highest level ($L = 3$) in the areas of high vorticity. During this stage new elements arise within small concentrated regions of the flow field shown in Fig. 14. These new elements result in an imbalanced increase in the number of elements in specific partitions, especially in the case where the domain is decomposed into the maximum number of partitions. Given that the re-partitioning of the domain is a computationally costly procedure, our strategy is to avoid re-partitioning at every adaptation step and re-partition only every 10 adaptation steps (i.e. to repartition every 1000 time steps only). This results in an increase in the imbalance of the decomposition, which is resolved after the re-partitioning, as shown in Fig. 14.

8. Application to the flow around a cylinder in a duct

In this section we present an inviscid simulation of the flow around a cylinder in a duct. The aim of this case is to demonstrate shock capturing and on the fly adaptation capabilities of the method. In this example, moving complex vortical structures, generated by the interaction of moving shocks, can be followed and resolved by applying AMR. Adequate resolution of these structures, often embedded in flow regions with discontinuities, is the key element for performing LES of high speed flows. For this simulation, the p1 polynomial basis was used and the unified TVB limiter was implemented for the capturing of the shocks.

In Fig. 16(**Left**) we present the density gradient magnitude distribution for the flow around a cylinder with diameter D after it has reached a quasi-steady state in time $t_1 = 10t^0$ ($t^0 = D/c$). In Fig. 16(**Right**) we present the pressure field for the same conditions at t_1. This cylinder is located between two parallel walls forming a duct with height $H = 8D$. The initial mesh consists of 3782 hexahedral elements, which are adapted up to five levels, depending on the density gradient. The bow shock formed upstream of the cylinder is reflected at the duct walls. The interaction of the initial bow shock with the reflected shock forms a stem at the vicinity of the wall. A shear layer is formed at the stems bifurcation which results in the formation of a Kelvin–Helmholtz instability as can be seen in Fig. 16(**Left**). The two reflected shocks from each side of the duct meet at the wake of the flow of the cylinder. In this area, the shocks interact with the vortical structures of the

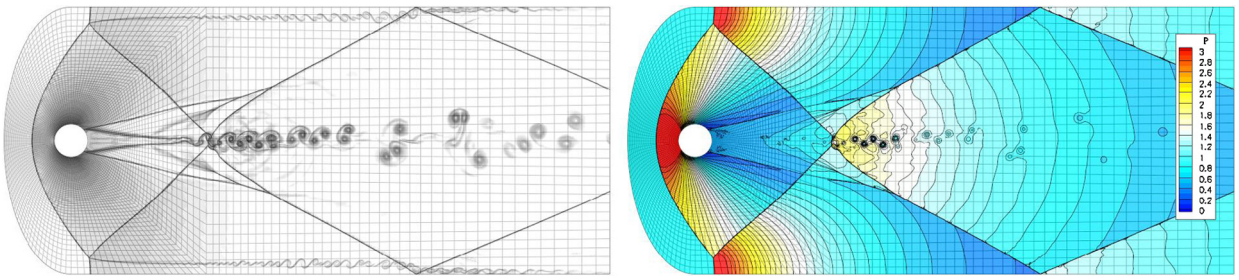

Fig. 16. Left: Distribution of the density gradient and **Right:** pressure distribution, for the ducted flow around a cylinder at $M = 2.0$, with the underlying ancestral mesh.

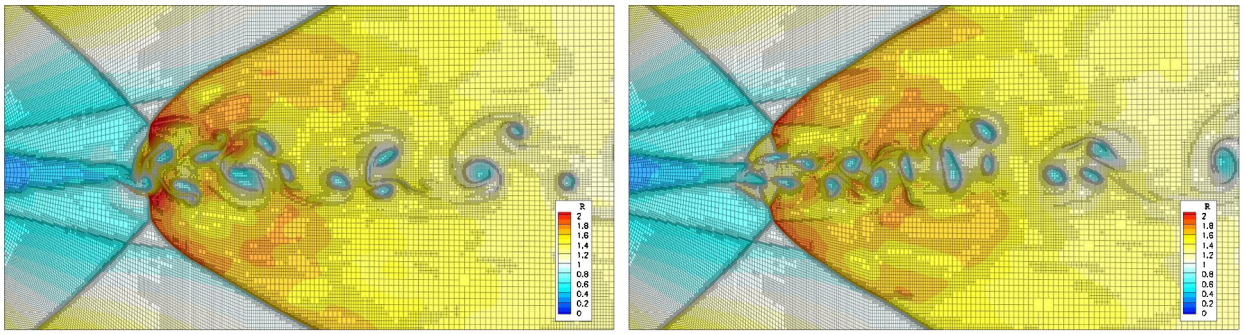

Fig. 17. Density distribution at the wake of the cylinder where the reflected shocks interact for two time instants: $t = t_1$ (**Left**) and $t = t_1 + \Delta t$ (**Right**), for $\Delta t = 1t^*$. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

wake. The AMR methodology provides a tool which allows us to follow flow structures in real time, focusing in the areas where high resolution is required, as shown in Fig. 17.

9. Application to the hypersonic flow around a double cone

Detailed experimental measurements for hypersonic chemically reacting flows are expensive to perform, and in some cases even impossible to obtain for very high re-entry speeds. As a result, basic mechanisms dominating high-speed, high enthalpy, chemically reacting flows are still poorly understood. It is therefore expected that numerical simulations will play a key-enabling role in the design and evolution of concepts for the next-generation space vehicles.

In this section we demonstrate the implementation of a chemistry mechanism in the Discontinuous Galerkin (DG) context. We include in the simulation the coupled flow-chemistry problem for high speed high enthalpy flow, which encompass non-equilibrium, chemical and energy relaxation processes triggered by high temperatures. These effects are accounted for by introducing finite-rate chemistry and energy relaxation into the governing equations. The objective is to validate a conservative, high-order accurate, shock-capturing method enhanced by the AMR, that would be applicable to complex three dimensional geometries for the simulation of high enthalpy, transitional and turbulent, chemically reacting flows.

In Fig. 18 (**Left**) we present the magnitude of the density gradient for the case of the hypersonic flow around a double cone at re-entry conditions ($M = 8.06$, $Re = 1.3 \cdot 10^5$) [7]. Under these conditions, the low density $\rho_\infty = 0.9422 \cdot 10^{-3}$ kg/m^3 atmospheric air dissociates. The chemistry mechanism employed to describe the dissociation of air as a mixture of oxygen and nitrogen incorporates five species. Following Nompelis [80], the widely used Park model [64] for the vibration-dissociation coupling is employed (see Section 2) assuming $T_\infty = 625K$ and $T^v_\infty = 712K$ for the translational and the rotational temperatures, respectively.

For this simulation, the p1 polynomial basis was used and the flow was initially solved as an inviscid flow. The extension of the unified TVB limiting approach [71,60,61] for the set of equations provided in the Park's model is applied in the space of characteristics [7], as described in [72]. The stability of the solution was greatly improved by the implementation of five levels of refinement in the vicinity of the shocks. A viscous solution is obtained by refining a layer close to the wall to accommodate for the creation of the boundary layer indicated by the vector field of the flow shown in the same Fig. 18 (**Left**). The dissociation of air is an endothermic reaction resulting in the reduction of the temperature at the surface of the cone shown in Fig. 19 (**Left**). In Fig. 19 (**Right**) we show the distribution of the atomic oxygen mixture fraction produced by the dissociation of the molecular oxygen. The results presented in these figures infer from the interaction between the oblique shock initiated at the tip of the cone with the shock generated by the shoulder of the double cone. These two primary shocks interact in the area above the cone shoulder creating a series of reflections. The detailed structure of the flow in this region is shown in the numerical Schlieren (density gradient distribution) presented in Fig. 18 (**Left**). The AMR method enables us to achieve detailed resolution of the complex flow structure.

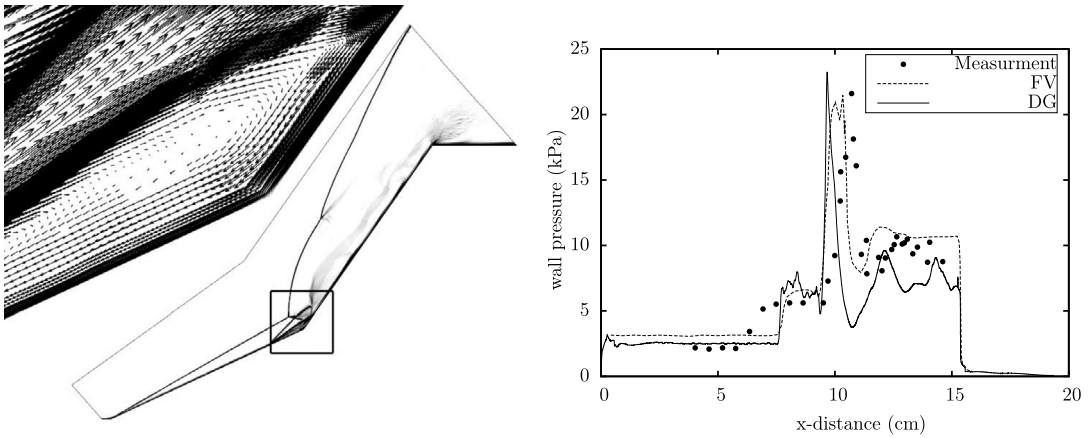

Fig. 18. Left: Magnitude of the density gradient for the double cone simulation. **Right:** Pressure distribution along the wall of the double cone. Solid curve: reactive simulation. Dashed curve: reactive simulation by Nompelis [79] using the finite volume (FV) method. Dots: experimental data taken from [79].

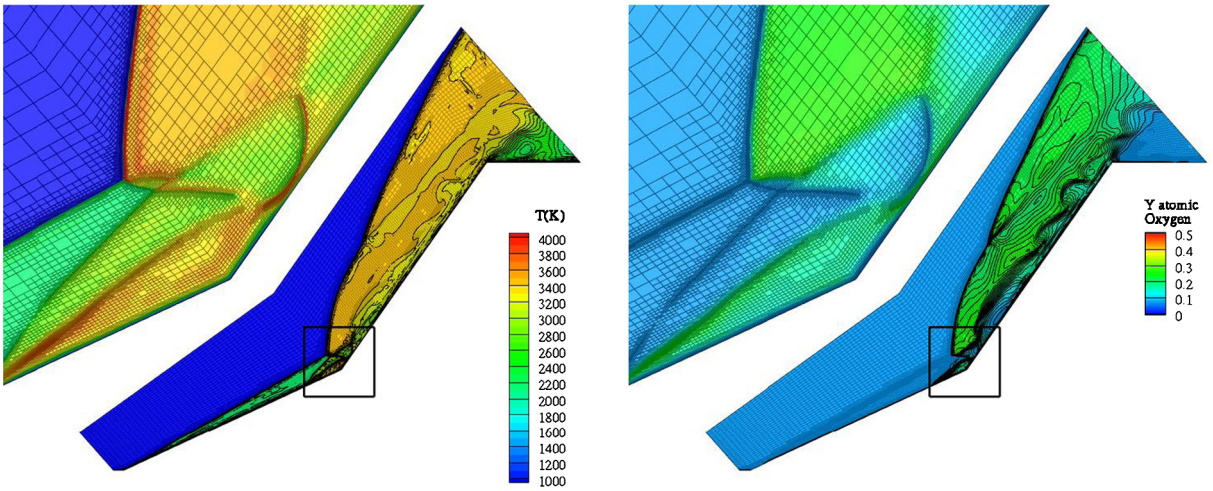

Fig. 19. Temperature field (**Left**) and mixture fraction of atomic oxygen (**Right**) for the double cone viscous simulation taking into account air dissociation. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

10. Application to a gasoline fuel spray

The modelling approach, described in the previous sections, have been applied to the simulation of a high-pressure, hollow-cone, gasoline fuel spray used in modern spray-guided gasoline engines. Results of this application are presented in this section.

The need to accurately model the interaction between droplets and carrier phase (coupling) in engineering applications is well known [81]. In sprays found in various engineering systems, including internal combustion engine injection systems [82] and particle deposition devices [83,84], the carrier phase flow induced by the dispersed phase and the momentum exchange between them is critical for the evolution of both phases. Momentum exchange in this case occurs at a microscopic scale which is usually well separated from the macroscopic scales of the problem. An Eulerian approach is typically used for modelling of such flows [82]. Hybrid Lagrangian–Eulerian approaches [81,85–88], however, offer more realistic modelling of the dispersed continuum which can be multi-valued.

The fuel used in our analysis is iso-octane, injected at a pressure of 100 bar for a duration of $T = 1$ ms. Injection mass flow rate $\dot{m}$ (measured experimentally using a rate tube) increased linearly from zero up to a maximum value of $\dot{m}_0 = 30$ g/s in 0.1 ms. During the following 0.8 ms the mass flow rate remains constant and decreases to zero during the last 0.1 ms. The temporal variation of the mass flow rate is described as:

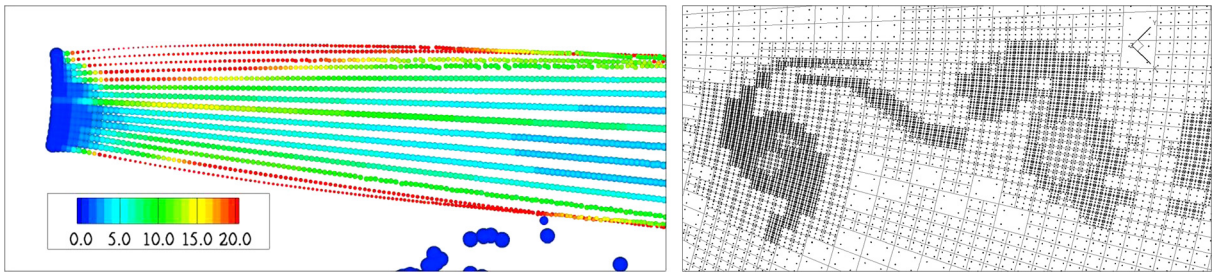

Fig. 20. (a), (**Left**) Injection of droplet parcels in the flow field. The size of the spheres shows droplet sizes (with maximal diameters equal to 30 μm and minimal diameters equal to 2 μm), the colour scale shown corresponds to the number of droplets per parcel p_d. (b), (**Right**) The computational grid in the wake of the spray at $t = 1.1$ ms. The smallest cell size corresponds to level 4 quad-tree splitting. The dots represent the quadrature points; four dots per cell correspond to a p1 polynomial basis and nine dots correspond to a p2 polynomial basis. h-adaptivity is imposed in high vorticity regions, p-adaptivity is imposed in the regions with high strain rate.

$$\dot{m} = \begin{cases} \dfrac{t}{0.1T}\dot{m}_0, & 0 \le t < 0.1T \\ \dot{m}_0, & 0.1T \le t < 0.9T \\ \dfrac{T-t}{0.1T}\dot{m}_0, & 0.9T \le t < T. \end{cases} \tag{35}$$

Every $i_{\text{inj}} = 25$ time steps (0.37 μs) of the simulation, $n_{\text{inj}} = 17$ droplet parcels are released at 45.0° relative to the axis of symmetry with a spread (divergence of wall thickness) of 5.0°. The injection point is located 1 mm off the middle of the injection cone edge. The magnitude of the droplet velocities was inferred from the mass flow rate of the injected fuel and Laser Droplet Anemometry (LDA) measurements and is equal to $v_d = 100$ m/s. The initial droplet diameter d_d is assumed equal to the injector slit lift $d_d = 27$ μm. The injection pattern described is shown in Fig. 20. The number of droplets p_d in each parcel injected in the domain during the simulation was calculated from the mass flow rate described by Equation (35) as:

$$p_d = (4.0/360.0)\frac{i_{\text{inj}}\dot{m}(t)\mathrm{d}t}{n_{\text{inj}}m_d}, \tag{36}$$

where m_d is the mass of each droplet ($m_d = \frac{\rho_f \pi d_d^3}{6}$). The number of droplets p_d in Equation (36) has been calculated for the computational domain used in our simulations, consisting of a 4^o sector, discretised by a hybrid unstructured mesh consisting of hexahedral and prismatic elements. Although the droplets are injected at a constant frequency any variations of the injection mass flow rate reflect on the number of droplets per parcel p_d. The droplet breakup is modelled using the WAVE model [81] as follows:

$$\frac{\mathrm{d}d_d}{\mathrm{d}t} = \frac{d_d - d_s}{\tau_{bu}}, \tag{37}$$

where

$$d_s = \begin{cases} 2B_0\Lambda, & 2B_0\Lambda \le d_d \\ \left(\dfrac{3}{2}d_d^2 \min\left(\pi U_m/2\Omega, \Lambda/4\right)\right)^{0.33}, & 2B_0\Lambda > d_d \end{cases}, \tag{38}$$

where U_m is the relative velocity between each parcel and the carrier phase. The calculation of the parameters Λ and Ω is discussed in detail in [89].

Experimental observations of the fuel spray were conducted in a quiescent chamber of fixed volume at 20 °C and 1 bar. The piezoelectric fuel injector was mounted in a vertical position at the top of the chamber. Measurements of the spray shape (geometry and thickness of plume) and droplet size and velocity distributions were carried out using high-speed photography and Phase Doppler Anemometry, respectively. The experimental set-up and measurement procedure are described in [88]. The computational mesh is refined 2 times in the vicinity of the spray, 3 to 4 times in areas of high vorticity and 4 times in the cells that contain droplets. In addition to h-adaptivity, p-adaptivity is imposed in the areas of high strain rate. This results in a combined h-p refined discretisation as shown in Fig. 20. The comparison between experimental result and the results of numerical simulation is shown in Figs. 21 to 24. The results presented in these figures agree with the results of experimental observations of these sprays [88]. As following from Fig. 21(b), an array of vortices is formed in the wake of the spray. Also, at the edges of the spray, two major counter-rotating, vortex rings are formed. The vortex rings induce a boundary layer at the cylinder head wall which detaches close to the injector needle tip. The boundary layer remains

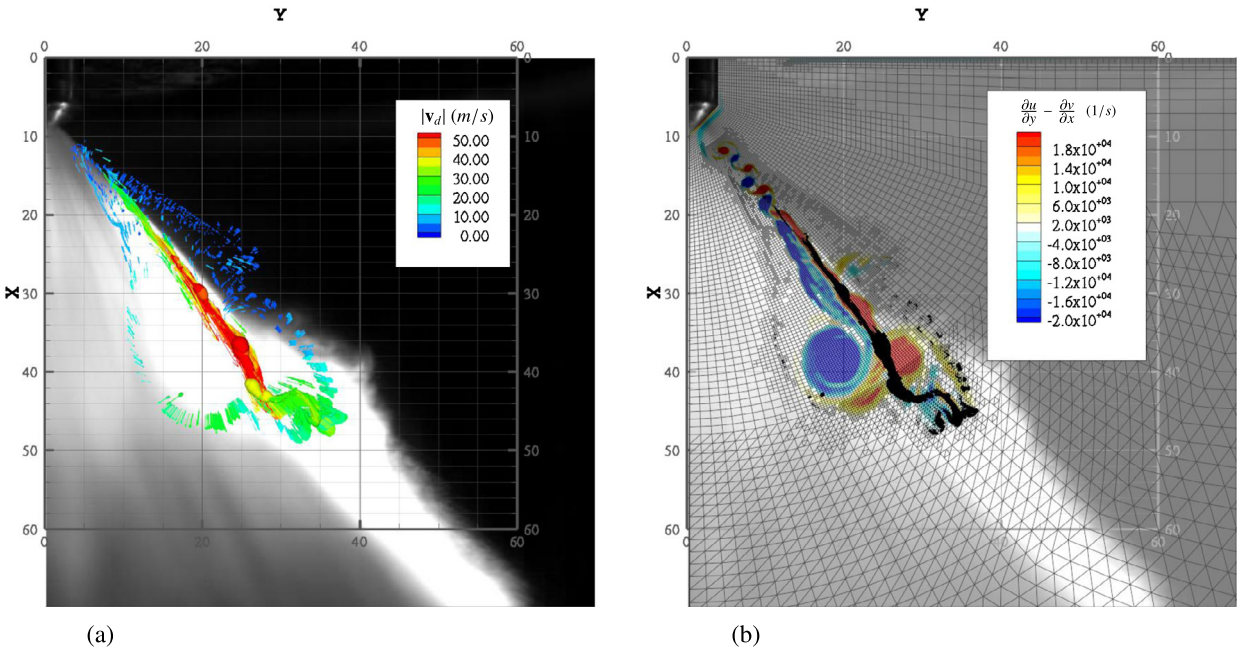

Fig. 21. Comparison of the experimental observations of droplet distribution from [88] (shadow graphs) with the numerical results. (a): Droplet distribution at $t = 1.1$ ms. Colour scale corresponds to velocity magnitude (m/s). Scatter size corresponds to the number of droplets per parcel. (b): Distribution of the z-component of the vorticity vector in (1/s) at $t = 1.1$ ms.

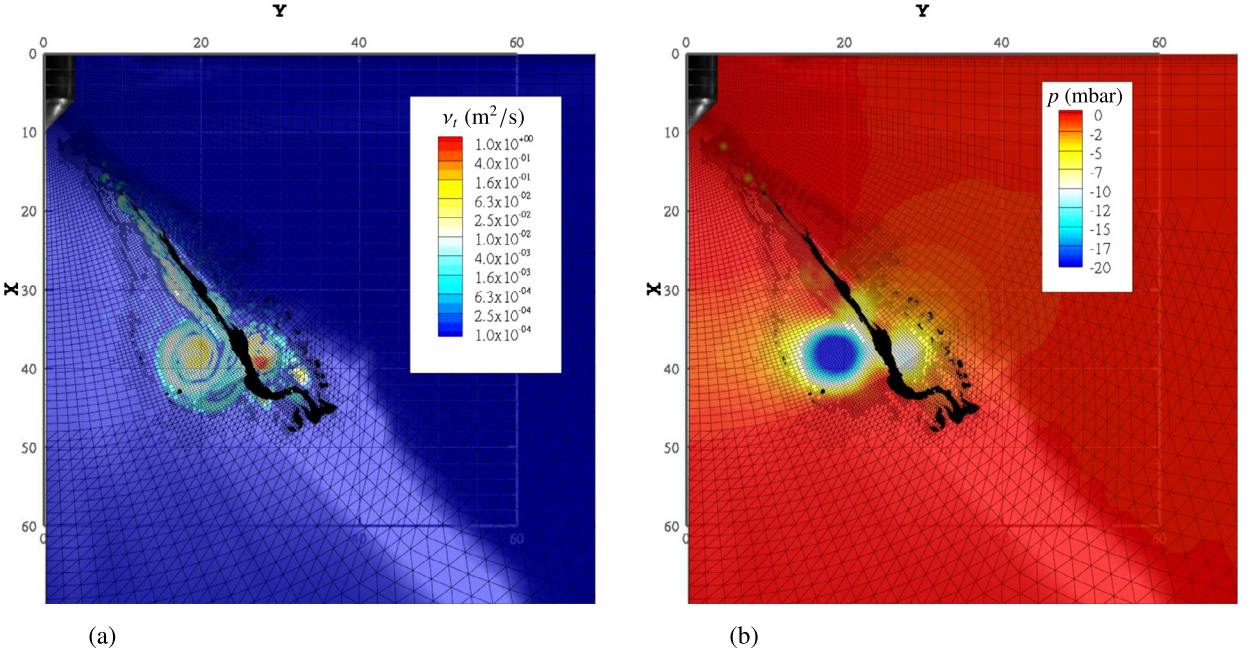

Fig. 22. Comparison of the experimental observations of droplet distribution from [88] (shadow graphs) with the numerical results. (a): Distribution of turbulent kinematic viscosity in (m^2/s) at $t = 1.1$ ms. (b): Distribution of pressure in (mbar) at $t = 1.1$ ms. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

attached from a distance $y = 12$ mm from the centreline. As can be seen in Fig. 21(b), the detached boundary layer is identified by the h-adaptivity criterion. The detachment of the boundary layer and the influence of the bounding walls (needle tip and cylinder head) are accounted for in the SA model in the DES framework.

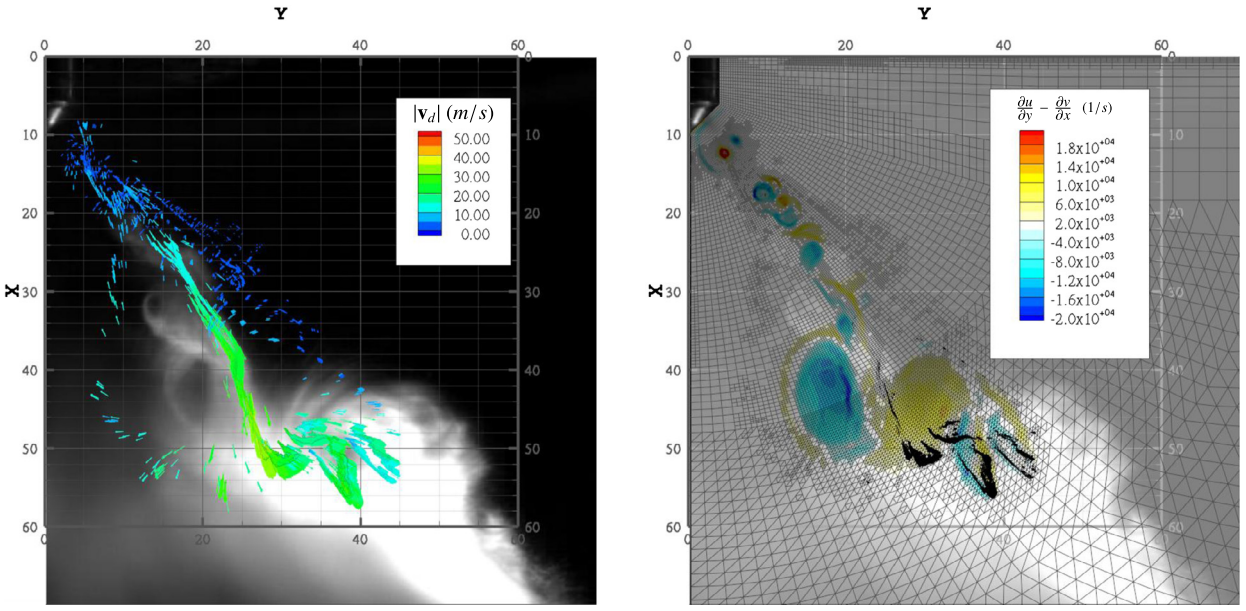

Fig. 23. Comparison of the experimental observations of droplet distribution from [88] (shadow graphs) with the numerical results. (a): Droplet distribution at $t = 1.67$ ms. Colour scale corresponds to velocity magnitude (m/s). Scatter size corresponds to the number of droplets per parcel. (b): Distribution of the z-component of the vorticity vector in (1/s) at $t = 1.67$ ms.

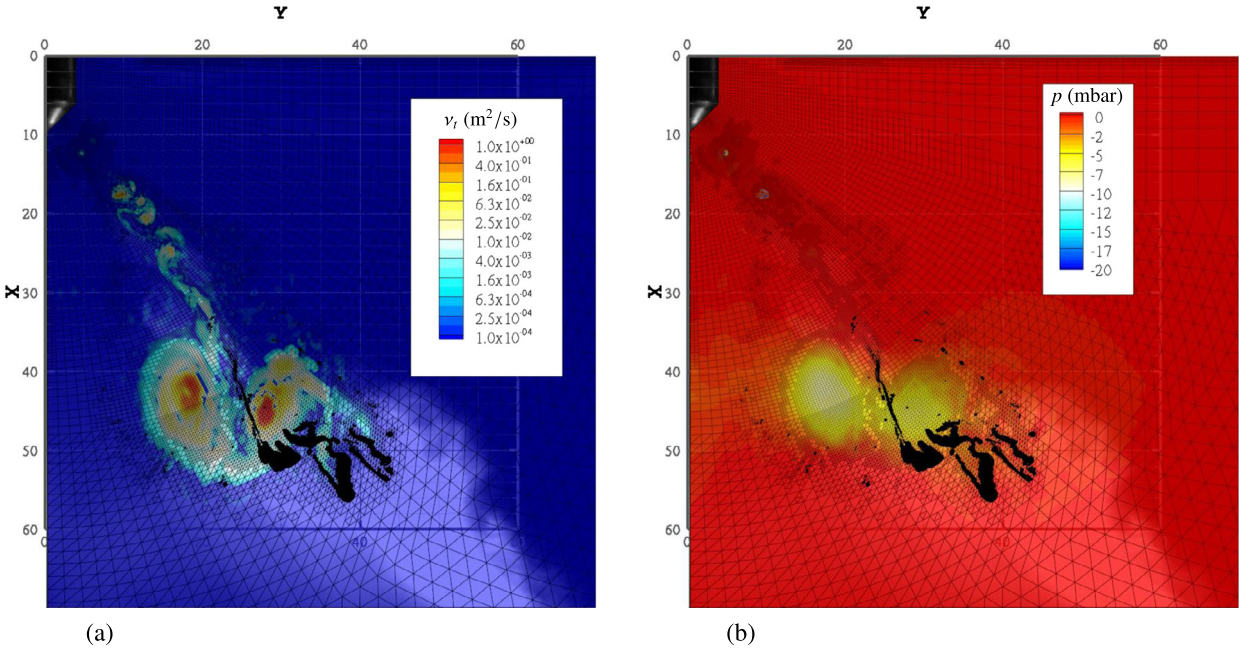

Fig. 24. Comparison of the experimental observations of droplet distribution from [88] (shadow graphs) with the numerical results. (a): Distribution of turbulent kinematic viscosity in (m^2/s) at $t = 1.67$ ms. (b): Distribution of pressure in (mbar) at $t = 1.67$ ms. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

11. Conclusion

The results of development of an Adaptive Mesh Refinement (AMR) approach on hybrid unstructured grids for the Discontinuous Galerkin (DG) method (ForestDG) has been presented. This approach is based on a topological representation of the computational mesh by a hierarchical structure consisting of oct- quad- and binary trees. The ancestral elements of the mesh are split into self-similar elements allowing each tree to grow branches to an arbitrary level of refinement. The

developed AMR (h-refinement) enables us to increase the spatial resolution for the computational mesh in the vicinity of the points of interest such as interfaces, geometrical features, or flow discontinuities. The local increase in the expansion order (p-refinement) in areas of high strain rates or vorticity magnitude results in an increase of the order of the accuracy. The connectivity of the elements, their genealogy and their partitioning have been described by linked lists of pointers. These pointers are attached to the tree data structure. This facilitates the on-the-fly splitting, merging, and repartitioning of the computational mesh by rearranging the links of each node of the tree.

This approach allows us to split or merge the computational mesh to an arbitrary level maintaining the gradients of the basis expansion within the element. This is performed by projecting the solution on the basis of the new elements. The calculated inviscid and viscous fluxes across the non-conformal faces retain the order of accuracy of the discretisation. The ability of the numerical code, based on this approach, to handle wide ranges of levels of adaptation and radically rearrange the grid to the new conditions makes it ideal for tracking moving flow structures and perform grid refinement to very high local resolution. The latter is required for the capturing moving shocks and tracking moving droplets. This approach allows us to perform continuous adaptation of the computational grid to the structures of the flow as they emerge from the temporal development of the solution. It has been shown that the forest of trees facilitates quick response to the changes in the grid, imposing minimal overhead on the computational cost. This is performed by defining the addressing, the decomposition and the connectivity relations in a such way that they are not affected by the local refinement of individual branches. Furthermore, the relations between the tree nodes are presented in an explicit form, avoiding search algorithms. Balanced computational load is maintained by repartitioning the grid. The decomposition boundaries can intersect trees and branches. Changes in the forest topology are expressed by re-stitching pointers and relations, rather than moving and rearranging data structures.

The suggested AMR methodology has been applied for the analysis of the interaction between droplets and the carrier phase. This has enabled us to refine the computational mesh in the vicinity of the droplet parcels, and perform accurate resolution of the coupling between the two phases. Results of the application of the new model for the analysis of the interaction between droplets and the carrier phase have been described. The accuracy of the new code has been assessed and results of its application to the analysis of a hollow-cone spray in gasoline engine-like conditions were presented.

Acknowledgement

The authors are grateful to EPSRC (grants EP/K005758/1 and EP/M002608/1) for financial support.

Appendix A

Transformation of coordinates from the computational space η_i to the physical space $x_i^{E_m}$ for a hexahedral element E_m.

$$\begin{aligned}\mathbf{x}_{hex}(\eta_1,\eta_2,\eta_3) = {} & ((1.0-\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_2^{E_m} + ((1.0-\eta_1)(1.0+\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_3^{E_m} \\ & + ((1.0+\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_0^{E_m} + ((1.0+\eta_1)(1.0+\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_1^{E_m} \\ & + ((1.0-\eta_1)(1.0-\eta_2)(1.0+\eta_3)/8.0)\,\mathbf{x}_6^{E_m} + ((1.0-\eta_1)(1.0+\eta_2)(1.0+\eta_3)/8.0)\,\mathbf{x}_7^{E_m} \\ & + ((1.0+\eta_1)(1.0-\eta_2)(1.0+\eta_3)/8.0)\,\mathbf{x}_4^{E_m} + ((1.0+\eta_1)(1.0+\eta_2)(1.0+\eta_3)/8.0)\,\mathbf{x}_5^{E_m}\end{aligned} \tag{A.1}$$

Transformation of coordinates from the computational space η_i to the physical space $x_i^{E_m}$ for a prism element E_m.

$$\begin{aligned}\mathbf{x}_{pri} = {} & ((1.0-\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_0^{E_m} + ((1.0-\eta_1)(1.0+\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_3^{E_m} \\ & + ((1.0+\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_2^{E_m} + ((1.0+\eta_1)(1.0+\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_5^{E_m} \\ & + ((1.0+\eta_3-\eta_2-\eta_2\eta_3)/4.0)\,\mathbf{x}_1^{E_m} + ((1.0+\eta_2+\eta_3+\eta_2\eta_3)/4.0)\,\mathbf{x}_4^{E_m}\end{aligned} \tag{A.2}$$

Transformation of coordinates from the computational space η_i to the physical space $x_i^{E_m}$ for a tetrahedral element E_m.

$$\begin{aligned}\mathbf{x}_{tet} = {} & ((1.0+\eta_3)/2.0)\,\mathbf{x}_3^{E_m} + ((1.0-\eta_3+\eta_2-\eta_2\eta_3)/4.0)\,\mathbf{x}_2^{E_m} \\ & + ((1.0-\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_0^{E_m} + ((1.0+\eta_1)(1.0-\eta_2)(1.0-\eta_3)/8.0)\,\mathbf{x}_1^{E_m}\end{aligned} \tag{A.3}$$

Appendix B

Keen, Face and Angle arrays for prisms are presented below

$$K_P[i,f] = \begin{bmatrix} \{P\} & \{P\} & \{P,6\} & \{P\} & \{P\} & \{P,7\} & \{P,2\} & \{P,5\} \\ \{P,6\} & \{P\} & \{P\} & \{P,7\} & \{P\} & \{P\} & \{P,0\} & \{P,3\} \\ \{P\} & \{P,6\} & \{P\} & \{P\} & \{P,7\} & \{P\} & \{P,1\} & \{P,4\} \\ \{P\} & \{P\} & \{P\} & \{P,0\} & \{P,1\} & \{P,2\} & \{P\} & \{P,6\} \\ \{P,3\} & \{P,4\} & \{P,5\} & \{P\} & \{P\} & \{P\} & \{P,7\} & \{P\} \end{bmatrix} \tag{B.1}$$

$$F[f,i] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 4 & 4 & 4 & 3 & 4 \\ 3 & 3 & 3 & 4 & 4 & 4 & 3 & 4 \end{bmatrix} \tag{B.2}$$

$$A[f,i] = \begin{bmatrix} 0 & 0 & 3 & 0 & 0 & 3 & 3 & 3 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.3}$$

Keen, Face and Angle arrays for tetrahedra are presented below

$$K_P[i,f] = \begin{bmatrix} \{P\} & \{P\} & \{P\} & \{P,7\} & \{P\} & \{P,7\} & \{P,7\} & \{P,3\} \\ \{P\} & \{P\} & \{P,6\} & \{P\} & \{P,6\} & \{P\} & \{P,2\} & \{P,6\} \\ \{P,5\} & \{P\} & \{P\} & \{P\} & \{P,5\} & \{P,0\} & \{P\} & \{P,5\} \\ \{P\} & \{P,4\} & \{P\} & \{P\} & \{P,1\} & \{P,4\} & \{P,4\} & \{P\} \end{bmatrix} \tag{B.4}$$

$$F[f,i] = \begin{bmatrix} 0 & 0 & 0 & 3 & 0 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 & 3 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 & 3 & 2 & 2 & 0 \\ 3 & 3 & 3 & 3 & 3 & 2 & 1 & 3 \end{bmatrix} \tag{B.5}$$

$$A[f,i] = \begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 & 2 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 0 & 2 & 0 & 2 \\ 0 & 2 & 0 & 0 & 2 & 0 & 2 & 0 \end{bmatrix} \tag{B.6}$$

Appendix C. Nomenclature

b	Basis function [–]
c	Speed of sound [$\mathrm{m\,s^{-1}}$]
$\mathbf{c}$	Basis expansion coefficient [–]
d_d	Droplet diameter [–]
e	Energy [–]
$\mathbf{f}$	Flux vector [–]
$\mathbf{f}_d$	Droplet drag force [–]
h	Total specific enthalpy [–]
L	Reference length [m], AMR Level [–]
M	Molecular weight [–]
R	Ideal gas constant [–]
N	Number of elements [–]
n_d	Droplet number density [–]
N_d	Number of diatomic species [–]
N_p	Number of polynomial basis [–]
p_d	Droplet number per parcel [–]
$\mathbf{q}$	Heat flux
Re	Reynolds number [–]
S_{ij}	Rate of strain tensor [–]
S^*_{ij}	Traceless rate of strain tensor [–]
t	Time [–]
u	Carrier phase velocity [–]
$\mathbf{U}$	State vector [–]
v	Droplet parcel velocity [–]
Y	Species mixture fraction [–]
$\mathbf{w}$	Source term [–]
$\mathbf{X}$	Coupled variables vector [–]

Greek symbols

η	Coordinates in the computational space [–]
$\mathbf{x}$	Coordinates in the physical space [–]
θ	Characteristic temperature [–]
$\mathbf{\Theta}$	Auxiliary variables vector [–]
μ_g	Dynamic viscosity of carrier phase [$\mathrm{kg\,m^{-2}\,s^{-1}}$]
μ_t	Turbulent dynamic viscosity [–]
ν_t	Turbulent kinematic viscosity [–]
ρ	Density [–]
ρ_g	Density of carrier phase [$\mathrm{kg\,m^{-3}}$]

Subscripts

aux	Auxiliary
d	Droplet
g	Gas phase
inv	Inviscid
r	Rotational
s	Species
t	Turbulent, Translational
T	Total
v	Vibrational
vis	Viscous

References

[1] C. Kleinstreuer, Modern Fluid Dynamics, Springer, Netherlands, 2010.
[2] C.C. Chu, I.G. Graham, T.Y. Hou, A new multiscale finite element method for high-contrast elliptic interface problems, Math. Comput. (MCOM) 79 (272) (2010) 1915–1955, https://doi.org/10.1090/S0025-5718-2010-02372-5.

[3] R.R. Millward, A New Adaptive Multiscale Finite Element Method with Applications to High Contrast Interface Problems, Ph.D. thesis, University of Bath, Bath, UK, 2011.
[4] F. Gibou, R. Fedkiw, S. Osher, A review of level-set methods and some recent applications, J. Comput. Phys. 353 (Supplement C) (2018) 82–109, https://doi.org/10.1016/j.jcp.2017.10.006.
[5] M. Carpenter, J.H. Casper, Accuracy of shock capturing in two spatial dimensions, AIAA J. 37 (9) (1999) 1072–1079.
[6] A.Y. Djaffar, G. Wagdi, W.G. Habashi, Finite element adaptive method for hypersonic thermochemical nonequilibrium flows, AIAA J. 35 (5) (1997) 1294–1302.
[7] A. Papoutsakis, K. Panourgias, J. Ekaterinaris, Discontinuous Galerkin discretization of chemically reacting flows, AIAA Paper, Aerospace Sciences Meeting, AIAA SciTech Forum (2014-0068), https://doi.org/10.2514/6.2014-0068.
[8] X. Zhao, A. Meganathan, S. Zhang, Computational approach for aeroheating with thermally coupled fields, J. Thermophys. Heat Transf. 31 (3) (2017) 489–499.
[9] K.T. Panourgias, A. Papoutsakis, J.A. Ekaterinaris, High-resolution p-adaptive DG simulations of flows with moving shocks, Int. J. Numer. Methods Fluids 75 (3) (2014) 205–230, https://doi.org/10.1002/fld.3893.
[10] A. Papoutsakis, S.S. Sazhin, S. Begg, I. Danaila, F. Luddens, A new approach to modelling the two way coupling for momentum transfer in a hollow-cone spray, in: R. Payri, X. Margot (Eds.), Proceedings of the 28th European Conference on Liquid Atomization and Spray Systems ILASS-Europe, Universitat Politecnica de Valencia, Valencia, Spain, 2017, pp. 448–455.
[11] A. Guittet, M. Theillard, F. Gibou, A stable projection method for the incompressible Navier–Stokes equations on arbitrary geometries and adaptive quad/octrees, J. Comput. Phys. 292 (C) (2015) 215–238, https://doi.org/10.1016/j.jcp.2015.03.024.
[12] M. Mirzadeh, A. Guittet, C. Burstedde, F. Gibou, Parallel level-set methods on adaptive tree-based grids, J. Comput. Phys. 322 (Supplement C) (2016) 345–364, https://doi.org/10.1016/j.jcp.2016.06.017.
[13] G. Karniadakis, S. Sherwin, Spectral/hp Element Methods for CFD, 2nd edition, Oxford University Press, 2003.
[14] E. Kubatko, S. Bunya, C. Dawson, J. Westerink, Dynamic p-adaptive Runge–Kutta Discontinuous Galerkin methods for the shallow water equations, Comput. Methods Appl. Mech. Eng. 198 (2009) 1766–1774.
[15] S. Tonini, M. Gavaises, A. Theodorakakos, Modelling of high-pressure dense diesel sprays with adaptive local grid refinement, Int. J. Heat Fluid Flow 29 (2) (2008) 427–448, https://doi.org/10.1016/j.ijheatfluidflow.2007.11.009.
[16] D. Isola, A. Guardone, G. Quaranta, Finite-volume solution of two-dimensional compressible flows over dynamic adaptive grids, J. Comput. Phys. 285 (2015) 1–23.
[17] M.A. Kopera, F.X. Giraldo, Analysis of adaptive mesh refinement for IMEX Discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations, J. Comput. Phys. 275 (2014) 92–117.
[18] C. Nastase, D. Mavriplis, High-order discontinuous Galerkin methods using an hp-multigrid approach, J. Comput. Phys. 213 (1) (2006) 330–357, https://doi.org/10.1016/j.jcp.2005.08.022.
[19] A. Fortin, T. Briffard, A. Garon, A more efficient anisotropic mesh adaptation for the computation of Lagrangian coherent structures, J. Comput. Phys. 285 (2015) 100–110.
[20] M. Woopen, G. May, J. Schutz, Adjoint-based error estimation and mesh adaptation for hybridized Discontinuous Galerkin methods, Int. J. Numer. Methods Fluids 76 (2014) 811–834.
[21] P. Frey, F. Alauzet, Anisotropic mesh adaptation for CFD computations, Comput. Methods Appl. Mech. Eng. 194 (48) (2005) 5068–5082, https://doi.org/10.1016/j.cma.2004.11.025.
[22] J. Flaherty, L. Krivodonova, J.F. Remachle, M.S. Shepard, Aspects of Discontinuous Galerkin methods for hyperbolic conservation laws, Finite Elem. Anal. Des. 38 (10) (2002) 889–908.
[23] L. Yelash, A. Müller, Lukáčová-Medvid'ová, F.M. Giraldo, V. Wirth, Adaptive Discontinuous evolution Galerkin method for dry atmospheric flow, J. Comput. Phys. 268 (1) (2014) 106–133.
[24] L. Tian, Y. Xu, J. Kuerten, J. van der Vegt, An h-adaptive local Discontinuous Galerkin method for the Navier–Stokes–Korteweg equations, J. Comput. Phys. 319 (C) (2016) 242–265, https://doi.org/10.1016/j.jcp.2016.05.027.
[25] D. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, J. Comput. Phys. 145 (1) (1998) 141–165, https://doi.org/10.1006/jcph.1998.6036.
[26] F. Bassi, L. Botti, A. Colombo, D.D. Pietro, P. Tesini, On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations, J. Comput. Phys. 231 (1) (2012) 45–65, https://doi.org/10.1016/j.jcp.2011.08.018.
[27] L. Botti, A. Colombo, F. Bassi, H-multigrid agglomeration based solution strategies for discontinuous Galerkin discretizations of incompressible flow problems, J. Comput. Phys. 347 (Supplement C) (2017) 382–415, https://doi.org/10.1016/j.jcp.2017.07.002.
[28] M. Kouhi, E. Oñate, D. Mavriplis, Adjoint-based adaptive finite element method for the compressible Euler equations using finite calculus, Aerosp. Sci. Technol. 46 (2015) 422–435, https://doi.org/10.1016/j.ast.2015.08.008.
[29] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Space–time Discontinuous Galerkin method for the compressible Navier–Stokes equations, J. Comput. Phys. 217 (2) (2006) 589–611, https://doi.org/10.1016/j.jcp.2006.01.018.
[30] C. Burstedde, J. Holke, Coarse mesh partitioning for tree-based AMR, SIAM J. Sci. Comput. 39 (5) (2017) C364–C392, https://doi.org/10.1137/16M1103518.
[31] C. Burstedde, L. Wilcox, O. Ghattas, `p4est`: scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM J. Sci. Comput. 33 (3) (2011) 1103–1133, https://doi.org/10.1137/100791634.
[32] T. Isaac, C. Burstedde, L. Wilcox, O. Ghattas, Recursive algorithms for distributed forests of octrees, SIAM J. Sci. Comput. 37 (5) (2015) C497–C531, https://doi.org/10.1137/140970963.
[33] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The `deal.II` library, version 8.5, J. Numer. Math. 25 (3) (2017) 137–146, https://doi.org/10.1515/jnma-2016-1045.
[34] W. Bangerth, R. Hartmann, G. Kanschat, deal.II – a general purpose object oriented finite element library, ACM Trans. Math. Softw. 33 (4) (2007) 24/1–24/27.
[35] B.S. Kirk, J.W. Peterson, R.H. Stogner, G.F. Carey, `libMesh`: a C++ library for parallel adaptive mesh refinement/coarsening simulations, Eng. Comput. 22 (3–4) (2006) 237–254, https://doi.org/10.1007/s00366-006-0049-3.
[36] B. Cockburn, G.E. Karniadakis, C.W. Shu, The development of Discontinuous Galerkin methods, in: Discontinuous Galerkin Methods, Theory, Computation and Applications, in: Lecture Notes in Computational Science and Engineering, vol. 11, 2000, pp. 3–50.
[37] J. Ekaterinaris, High-order accurate, low numerical diffusion methods for aerodynamics, Prog. Aerosp. Sci. 41 (3–4) (2005) 192–300.
[38] Z.J. Wang, High-order methods for the Euler and Navier–Stokes equations on unstructured grids, Prog. Aerosp. Sci. 43 (1–3) (2007) 1–41.
[39] B. Cockburn, C.-W. Shu, The Runge–Kutta Discontinuous Galerkin method for conservation laws V: multidimensional systems, J. Comput. Phys. 141 (2) (1998) 199–224.
[40] Y. Liu, M. Vinokur, Z. Wang, Spectral (finite) volume method for conservation laws on unstructured grids V: extension to three-dimensional systems, J. Comput. Phys. 212 (2) (2006) 454–472.
[41] Y. Liu, M. Vinokur, Z. Wang, Spectral difference method for unstructured grids I: basic formulation, J. Comput. Phys. 216 (2) (2006) 780–801.

[42] M. Yu, Z. Wang, Y. Liu, On the accuracy and efficiency of Discontinuous Galerkin, spectral difference and correction procedure via reconstruction methods, J. Comput. Phys. 259 (15) (2014) 70–95.
[43] J. Cheng, C.W. Shu, A high order ENO conservative Lagrangian type scheme for the compressible Euler equations, J. Comput. Phys. 227 (2) (2007) 567–1596.
[44] G.S. Jiang, C.W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[45] J.J.W. van der Vegta, S. Rhebergen, Hp-multigrid as smoother algorithm for higher order Discontinuous Galerkin discretizations of advection dominated flows. Part II: optimization of the Runge–Kutta smoother, J. Comput. Phys. 231 (2012) 7564–7583.
[46] S.M. Schnepp, T. Weiland, Efficient large scale electromagnetic simulations using dynamically adapted meshes with the Discontinuous Galerkin method, J. Comput. Appl. Math. 236 (2012) 4909–4924.
[47] S.M. Schnepp, Error-driven dynamical hp-meshes with the Discontinuous Galerkin method for three-dimensional wave propagation problems, J. Comput. Appl. Math. 270 (2014) 353–368.
[48] M. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (3) (1984) 484–512, https://doi.org/10.1016/0021-9991(84)90073-1.
[49] M. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1) (1989) 64–84, https://doi.org/10.1016/0021-9991(89)90035-1.
[50] R. Hartmann, P. Houston, Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations, J. Comput. Phys. 183 (2) (2002) 508–532, https://doi.org/10.1006/jcph.2002.7206.
[51] R. Hartmann, P. Houston, An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations, J. Comput. Phys. 227 (22) (2008) 9670–9685, https://doi.org/10.1016/j.jcp.2008.07.015.
[52] E. Rinaldi, P. Colonna, R. Pecnik, Flux-conserving treatment of non-conformal interfaces for finite-volume discretization of conservation laws, Comput. Fluids 120 (2015) 126–139.
[53] R. Teyssier, Cosmological hydrodynamics with adaptive mesh refinement – a new high resolution code called RAMSES, Astron. Astrophys. 385 (1) (2002) 337–364, https://doi.org/10.1051/0004-6361:20011817.
[54] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, P-multigrid solution of high-order Discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 207 (2005) 92–113.
[55] K. Mani, D. Mavriplis, Error estimation and adaptation for functional outputs in time-dependent flow problems, J. Comput. Phys. 229 (2) (2010) 415–440, https://doi.org/10.1016/j.jcp.2009.09.034.
[56] R. Verfürth, A posteriori error estimation and adaptive mesh-refinement techniques, J. Comput. Appl. Math. 50 (1) (1994) 67–83, https://doi.org/10.1016/0377-0427(94)90290-9.
[57] K. Fidkowski, Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints, J. Comput. Phys. 341 (Supplement C) (2017) 258–277, https://doi.org/10.1016/j.jcp.2017.04.005.
[58] K. Fidkowski, Y. Luo, Output-based space–time mesh adaptation for the compressible Navier–Stokes equations, J. Comput. Phys. 230 (14) (2011) 5753–5773, https://doi.org/10.1016/j.jcp.2011.03.059.
[59] D. Mavriplis, Unstructured Mesh Generation and Adaptivity, Tech. rep., Universities Space Association. ICASE Report No: 95–26, 1995.
[60] K. Kontzialis, J.A. Ekaterinaris, Discontinuous Galerkin discretizations for mixed type meshes with p-type adaptivity, AIAA Paper 297.
[61] K. Panourgias, K. Kontzialis, J.A. Ekaterinaris, A limiting approach for the three-dimensional dg discretisations in arbitrary type meshes, AIAA Paper 2011-3837, 20th CFD conference.
[62] I. Toulopoulos, J.A. Ekaterinaris, Discontinuous-Galerkin discretizations for viscous flow problems in complex domains, AIAA Paper 2005-1264 1264.
[63] I. Toulopoulos, J.A. Ekaterinaris, High-order discontinuous-Galerkin discretizations for computational aeroacoustics in complex domains, AIAA J. 44 (2006) 3.
[64] C. Park, Nonequilibrium Hypersonic Aerothermodynamics, John Wiley & Sons, NY, 1989.
[65] P.R. Spalart, S.R. Allmaras, A one-equation turbulence model for aerodynamic flows, Rech. Aérosp. 1 (1994) 5–21.
[66] P. Spalart, S. Deck, M. Shur, K. Squires, M. Strelets, A. Travin, A new version of detached-eddy simulation, resistant to ambiguous grid densities, Theor. Comput. Fluid Dyn. 20 (3) (2006) 181–195.
[67] M. Shur, P. Spalart, M. Strelets, A. Travin, A hybrid RANS-LES approach with delayed-DES and wall-modelled LES capabilities, Int. J. Heat Fluid Flow 29 (6) (2008) 1638–1649.
[68] Q. Dong, T. Ishima, H. Kawashima, W. Long, A study on the spray characteristics of a piezo pintle-type injector for di gasoline engines, J. Mech. Sci. Technol. 27 (7) (2013) 1981–1993, https://doi.org/10.1007/s12206-013-0510-3.
[69] J. Anderson, Computational Fluid Dynamics, Computational Fluid Dynamics: The Basics with Applications, McGraw-Hill Education, 1995, https://books.google.co.uk/books?id=dJceAQAAIAAJ.
[70] S.R. Allmaras, F.T. Johnson, P.R. Spalart, Modifications and clarifications for the implementation of the Spalart–Allmaras turbulence model, in: 7th International Conference on Computational Fluid Dynamics, Big Island, Hawaii, USA, 2012, Paper ICCFD7-1902.
[71] B. Cockburn, C. Shu, TVB Runge–Kutta Local Projection Discontinuous Galerkin Finite Element Method for conservation laws II: general framework, Math. Comput. 52 (1989) 411–435.
[72] Y. Liu, M. Vinokur, Nonequilibrium flow computations. I. An analysis of numerical formulations of conservation laws, J. Comput. Phys. 83 (1989) 373–397.
[73] G. Karypis, V. Kumar, MeTis: unstructured graph partitioning and sparse matrix ordering system, version 4.0, http://www.cs.umn.edu/~metis, 2009.
[74] D. Kopriva, A conservative staggered-grid Chebyshev multidomain method for compressible flows. II. A semi-structured method, J. Comput. Phys. 128 (2) (1996) 475–488, https://doi.org/10.1006/jcph.1996.0225.
[75] P.J. Roache, Verification and Validation in Computational Science and Engineering, Hermosa Publishers, 1998.
[76] W.L. Oberkampf, C.J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press, 2010.
[77] Z. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P. Persson, B. van Leer, M. Visbal, High-order CFD methods: current status and perspective, Int. J. Numer. Methods Fluids 72 (8) (2013) 811–845, https://doi.org/10.1002/fld.3767.
[78] C.H.K. Williamson, The existence of two stages in the transition to three-dimensionality of a cylinder wake, Phys. Fluids 31 (1988) 3165–3168, https://doi.org/10.1063/1.866925.
[79] I. Nompelis, G.V. Candler, M. MacLean, T.P. Wadhams, M.S. Holden, Numerical Investigation of Double-Cone Flow Experiments with High-Enthalpy Effects, AIAA Paper, 2010.
[80] I. Nompelis, Computational Study of Hypersonic Double Cone Experiments for Code Validation, Ph.D. thesis, University of Minesota, Minesota, US, 2004.
[81] S.S. Sazhin, Droplets and Sprays, Springer, London, 2014.
[82] M. Vujanović, Z. Petranović, W. Edelbauer, N. Duić, Modelling spray and combustion processes in diesel engine by using the coupled Eulerian–Eulerian and Eulerian–Lagrangian method, Energy Convers. Manag. 125 (2016) 15–25, https://doi.org/10.1016/j.enconman.2016.03.072, sustainable development of energy, water and environment systems for future energy technologies and concepts.

[83] J. Lee, B.N. Joshi, J. Lee, T. Kim, D. Kim, S. Al-Deyab, I. Seong, M. Swihart, W. Yoon, S. Yoon, Stable high-capacity lithium ion battery anodes produced by supersonic spray deposition of hematite nanoparticles and self-healing reduced graphene oxide, Electrochim. Acta 228 (2017) 604–610, https://doi.org/10.1016/j.electacta.2017.01.116.
[84] S. Kiselev, V. Kiselev, S. Klinkov, V. Kosarev, V. Zaikovskii, Study of the gas-particle radial supersonic jet in the cold spraying, Surf. Coat. Technol. 313 (2017) 24–30, https://doi.org/10.1016/j.surfcoat.2017.01.046.
[85] E.M. Sazhina, S.S. Sazhin, M. Heikal, V.I. Babushok, R.J.R. Johns, A detailed modelling of the spray ignition process in diesel engines, Combust. Sci. Technol. 160 (1) (2000) 317–344.
[86] O. Rybdylova, A. Osiptsov, S.S. Sazhin, S. Begg, M. Heikal, A combined viscous-vortex, thermal-blob and Lagrangian method for non-isothermal, two-phase flow modelling, Int. J. Heat Fluid Flow 58 (2016) 93–102, https://doi.org/10.1016/j.ijheatfluidflow.2015.12.003.
[87] O. Rybdylova, M.A. Qubeissi, M. Braun, C. Crua, J. Manin, L. Pickett, G. de Sercey, E. Sazhina, S.S. Sazhin, M. Heikal, A model for droplet heating and its implementation into ANSYS fluent, Int. Commun. Heat Mass Transf. 76 (2016) 265–270.
[88] T.S. Zaripov, A.K. Gilfanov, S.M. Begg, O. Rybdylova, S.S. Sazhin, M.R. Heikal, The fully Lagrangian approach to the analysis of particle/droplet dynamics: implementation into ANSYS fluent and application to gasoline sprays, At. Sprays 27 (6) (2017) 493–510.
[89] M. Turner, S.S. Sazhin, J. Healey, S. Martynov, A breakup model for transient diesel fuel sprays, Fuel 97 (2012) 288–305.